

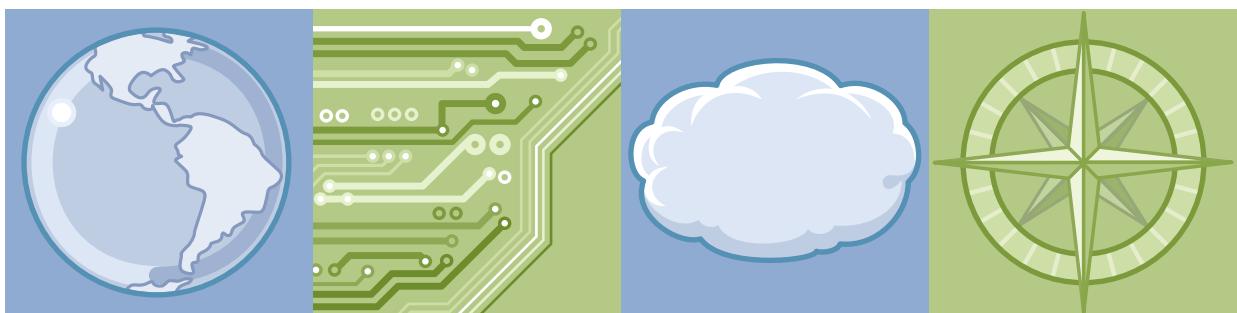


IBM Training

Student Exercises

Developing Rule Solutions in IBM Operational Decision Manager V8.7

Course code WB392 / ZB392 ERC 2.0



WebSphere Education

Trademarks

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

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

| | | |
|------------|--------------|-----------------|
| CICS® | DB2® | developerWorks® |
| Express® | ILOG® | Insight™ |
| Insights™ | Orchestrate® | PartnerWorld® |
| Redbooks® | SPSS® | Tivoli® |
| WebSphere® | Worklight® | 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.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

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

July 2015 edition

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

Contents

| | |
|--|-------------|
| Trademarks | xi |
| Exercises description | xiii |
| General exercise information..... | xv |
| Exercise 1. Operational Decision Manager in action | 1-1 |
| Section 1. Running the Miniloan web application | 1-3 |
| 1.1. Setting up your environment | 1-3 |
| 1.2. Running the application | 1-4 |
| Section 2. Managing business rules in Decision Center | 1-7 |
| 2.1. Editing rules in Decision Center Business console | 1-8 |
| 2.2. Reviewing the rule history | 1-13 |
| 2.3. Changing the credit score requirements | 1-15 |
| Section 3. Validating changes with Decision Validation Services | 1-20 |
| 3.1. Opening the Decision Center Enterprise console | 1-21 |
| 3.2. Creating and running a test suite | 1-22 |
| Section 4. Deploying updated rules from Decision Center to Rule Execution Server | 1-26 |
| 4.1. Creating the RuleApp | 1-27 |
| 4.2. Deploying the RuleApp | 1-30 |
| Section 5. Monitoring the deployed rules in Rule Execution Server | 1-32 |
| Section 6. Viewing the effects in the Miniloan application | 1-35 |
| Section 7. Synchronizing environments between Rule Designer and Decision Center | 1-36 |
| 7.1. Opening Rule Designer | 1-37 |
| 7.2. Taking a quick tour of Rule Designer | 1-39 |
| 7.3. Synchronizing Rule Designer and Decision Center environments | 1-41 |
| Section 8. Shutting down the modules | 1-47 |
| Exercise 2. Setting up rule projects | 2-1 |
| Section 1. Creating the rule project | 2-2 |
| 1.1. Setting up your environment for this exercise | 2-2 |
| 1.2. Creating the rule project | 2-3 |
| 1.3. Importing the execution object model (XOM) | 2-7 |
| 1.4. Creating a reference to the XOM in the rule project | 2-10 |
| Section 2. Setting up the BOM | 2-12 |
| 2.1. Creating the BOM | 2-12 |
| 2.2. Exploring the BOM | 2-14 |
| 2.3. Identifying the next tasks in rule project development | 2-16 |
| Section 3. Creating the ruleset parameters | 2-18 |
| Section 4. Creating a ruleset variable | 2-20 |
| Section 5. Creating rule packages | 2-22 |
| Section 6. Publishing from Rule Designer to Decision Center | 2-24 |
| 6.1. Publishing the rule project to Decision Center | 2-24 |
| 6.2. Opening the project in Decision Center Business console | 2-26 |
| 6.3. Opening the project in the Decision Center Enterprise console | 2-27 |
| 6.4. Modifying the ruleset variable in Decision Center | 2-30 |
| 6.5. Synchronizing Rule Designer with Decision Center | 2-31 |
| Section 7. Deleting a project from Decision Center | 2-33 |

| | |
|---|------------|
| Section 8. Creating a rule project from Decision Center | 2-34 |
| 8.1. Creating a project in Rule Designer from Decision Center | 2-34 |
| 8.2. Finalizing the rule project in Rule Designer | 2-36 |
| Exercise 3. Working with the BOM | 3-1 |
| Section 1. Finalizing the XOM | 3-2 |
| 1.1. Setting up your environment for this exercise | 3-2 |
| 1.2. Understanding the problem | 3-4 |
| 1.3. Finish implementing the Borrower class | 3-5 |
| 1.4. Finish implementing the Test class | 3-10 |
| Section 2. Creating a rule project and its BOM | 3-12 |
| Section 3. Working with the vocabulary that is required to author rules | 3-14 |
| 3.1. Creating the default vocabulary | 3-14 |
| 3.2. Examining the verbalization for members of the Borrower class | 3-16 |
| 3.3. Editing the default verbalization | 3-19 |
| Exercise 4. Refactoring | 4-1 |
| Section 1. Refactoring rule artifacts after vocabulary changes | 4-2 |
| 1.1. Setting up your environment for this exercise | 4-2 |
| 1.2. Exploring the default verbalization | 4-3 |
| 1.3. Changing the default verbalization of the getBankruptcyAge method | 4-5 |
| 1.4. Changing the default verbalization of the duration member | 4-6 |
| Section 2. Maintaining consistency between the BOM and the XOM | 4-8 |
| 2.1. Adding a XOM class that is missing in the BOM | 4-8 |
| 2.2. Adding a XOM method that is missing in the BOM | 4-10 |
| 2.3. Creating a rule to use this new attribute | 4-11 |
| 2.4. Deprecating BOM members | 4-12 |
| Exercise 5. Working with ruleflows | 5-1 |
| Section 1. Exploring a ruleflow diagram | 5-2 |
| 1.1. Setting up your environment for this exercise | 5-2 |
| 1.2. Exploring the ruleflow | 5-2 |
| 1.3. Exploring action tasks and transitions | 5-7 |
| 1.4. Using ruleset parameters and variables in rules | 5-9 |
| Section 2. Creating a ruleflow | 5-11 |
| Exercise 6. Exploring action rules | 6-1 |
| Section 1. Exploring rule structure | 6-2 |
| 1.1. Setting up your environment for this exercise | 6-2 |
| 1.2. Exploring the rules | 6-2 |
| Section 2. Exploring rule behavior | 6-4 |
| Section 3. Working with automatic variables | 6-6 |
| Exercise 7. Authoring action rules | 7-1 |
| Section 1. Authoring action rules in the Intellirule editor | 7-2 |
| 1.1. Setting up your environment for this exercise | 7-2 |
| 1.2. Authoring rules with the Intellirule editor | 7-3 |
| Section 2. Authoring actions rules in the Guided editor | 7-8 |
| Section 3. Authoring rules with ruleset parameters | 7-10 |
| Section 4. Creating an action rule template | 7-13 |
| Section 5. Authoring action rules from your action rule template | 7-15 |

| | |
|---|-------------|
| 5.1. Publishing the BOM and the rules to Decision Center | 7-15 |
| 5.2. Authoring the rule in Decision Center | 7-16 |
| 5.3. Deleting projects | 7-18 |
| Exercise 8. Authoring decision tables and decision trees | 8-1 |
| Section 1. Authoring a decision table | 8-2 |
| 1.1. Setting up your environment for this exercise | 8-2 |
| 1.2. Creating the table | 8-3 |
| 1.3. Completing the table cells with values | 8-8 |
| Section 2. Authoring a decision tree | 8-15 |
| 2.1. Creating the tree | 8-15 |
| Exercise 9. Working with static domains | 9-1 |
| Section 1. Exploring a collection domain | 9-2 |
| 1.1. Setting up your environment for this exercise | 9-2 |
| 1.2. Exploring the domain | 9-3 |
| 1.3. Testing the rule | 9-5 |
| Section 2. Working with an enumeration of literals | 9-8 |
| 2.1. Creating the domain | 9-8 |
| 2.2. Testing the domain in a rule | 9-11 |
| Section 3. Defining an enumeration of static references | 9-13 |
| 3.1. Creating the static references Java class | 9-13 |
| 3.2. Authoring an action rule that uses a domain of static references | 9-19 |
| Exercise 10. Working with dynamic domains | 10-1 |
| Section 1. Creating a dynamic domain in Rule Designer | 10-3 |
| 1.1. Setting up your environment for this exercise | 10-3 |
| 1.2. Creating the domain | 10-4 |
| 1.3. Examining the dynamic domain in Rule Designer | 10-12 |
| Section 2. Using the dynamic domain in a rule | 10-14 |
| Section 3. Updating the dynamic domain | 10-19 |
| Section 4. Updating the XOM | 10-23 |
| Section 5. Publishing the BOM and rule projects to Decision Center | 10-24 |
| 5.1. Publishing the BOM and the rules | 10-24 |
| Section 6. Examining rules in Decision Center | 10-26 |
| 6.1. Opening the published projects in Decision Center | 10-26 |
| 6.2. Creating a Resources smart folder | 10-28 |
| Section 7. Modifying the dynamic domain in Decision Center | 10-30 |
| 7.1. Modifying the values of the dynamic domain | 10-30 |
| 7.2. Modifying the domain value in the rule | 10-31 |
| Section 8. Updating Rule Designer from Decision Center | 10-34 |
| 8.1. Synchronizing the rule project | 10-34 |
| Exercise 11. Queries and ruleset extraction | 11-1 |
| Section 1. Searching for rule artifacts | 11-2 |
| 1.1. Setting up your environment for this exercise | 11-2 |
| 1.2. Searching for specific criteria across the rules | 11-2 |
| 1.3. Searching for dependencies between rules | 11-4 |
| Section 2. Querying rule projects | 11-7 |
| 2.1. Searching for rules that may become applicable | 11-7 |
| 2.2. Searching for rules that may lead to a state | 11-9 |

| | |
|--|-------|
| 2.3. Searching for rules that use a BOM member | 11-10 |
| 2.4. Applying actions with queries | 11-11 |
| Section 3. Extracting a ruleset archive | 11-14 |
| 3.1. Defining a query-based ruleset extractor | 11-14 |
| 3.2. Using a ruleset extractor to extract a ruleset archive | 11-15 |
| 3.3. Exploring the content of the ruleset archive | 11-17 |
| 3.4. Exporting a ruleset archive without a ruleset extractor | 11-17 |

Exercise 12. Executing rules locally **12-1**

| | |
|--|-------|
| Section 1. Executing rules locally by using a launch configuration | 12-2 |
| 1.1. Setting up your environment for this exercise | 12-2 |
| 1.2. Creating a launch configuration | 12-2 |
| 1.3. Running the launch configuration | 12-5 |
| 1.4. Debugging with a launch configuration | 12-6 |
| Section 2. Executing rules locally by using a runner application | 12-8 |
| 2.1. Creating a Java project for rules | 12-8 |
| 2.2. Exploring the runner application | 12-9 |
| 2.3. Defining the objects that are passed to the rules | 12-10 |
| 2.4. Executing the rules locally by using the runner application | 12-13 |

Exercise 13. Debugging a ruleset **13-1**

| | |
|---|-------|
| Section 1. Running the debug launch configuration | 13-2 |
| 1.1. Setting up your environment for this exercise | 13-2 |
| 1.2. Running the debug configuration | 13-2 |
| 1.3. Setting breakpoints in the rules | 13-4 |
| Section 2. Debugging with breakpoints | 13-6 |
| 2.1. Inspecting the agenda | 13-6 |
| Section 3. Debugging with breakpoints in the ruleflow | 13-8 |
| Section 4. Resolving the problem | 13-10 |

Exercise 14. Enabling tests and simulations **14-1**

| | |
|---|-------|
| Section 1. Validating the rule project | 14-2 |
| 1.1. Setting up your environment for this exercise | 14-2 |
| 1.2. Validating the rule project | 14-2 |
| 1.3. Choosing a constructor | 14-3 |
| 1.4. Editing argument names for column headings in the template | 14-5 |
| Section 2. Generating the scenario file template | 14-7 |
| Section 3. Populating the template and testing the scenario | 14-10 |
| 3.1. Viewing the prepopulated scenario file | 14-10 |
| 3.2. Running the tests with the populated scenario file | 14-10 |
| Section 4. Customizing input for the scenario file template | 14-12 |
| 4.1. Removing columns from the template | 14-12 |
| 4.2. Creating a virtual attribute to test complex objects | 14-13 |
| 4.3. Regenerating the scenario file template | 14-15 |
| 4.4. Testing the customized scenario file | 14-16 |
| Section 5. Running tests remotely by deploying the XOM | 14-18 |
| Section 6. Creating a DVS project to enable remote tests | 14-20 |
| Section 7. Running tests on the remote server | 14-23 |
| Section 8. Repackaging the SSP in a DVS Project | 14-25 |
| 8.1. Repackage the SSP | 14-25 |
| 8.2. Deploying the SSP to the application server | 14-25 |

| | |
|--|-------------|
| Section 9. Testing the redeployed SSP | 14-29 |
| Exercise 15. Managing deployment | 15-1 |
| Section 1. Creating a RuleApp in Rule Designer | 15-2 |
| 1.1. Setting up your environment for this exercise | 15-2 |
| 1.2. Creating a RuleApp | 15-2 |
| 1.3. Adding a RuleApp property to a RuleApp | 15-5 |
| 1.4. Adding a ruleset property to a ruleset | 15-6 |
| Section 2. Creating deployment configurations | 15-9 |
| 2.1. Creating a Java SE-based Rule Execution Server configuration | 15-9 |
| 2.2. Updating a Rule Execution Server configuration | 15-10 |
| 2.3. Creating a WebSphere Application Server-based Rule Execution Server configuration | 15-11 |
| Section 3. Deploying a RuleApp from Rule Designer | 15-13 |
| 3.1. Deploying the RuleApp from the RuleApp itself | 15-13 |
| 3.2. Deploying a RuleApp from the Rule Execution Server configuration | 15-14 |
| Section 4. Deploying the managed XOM from Rule Designer | 15-17 |
| 4.1. Deploying the managed XOM from the rule project | 15-17 |
| 4.2. Deploying a RuleApp associated with a managed XOM | 15-19 |
| Exercise 16. Exploring the Rule Execution Server console | 16-1 |
| Section 1. Exploring the Rule Execution Server console | 16-2 |
| 1.1. Setting up your environment for this exercise | 16-2 |
| 1.2. Exploring the Rule Execution Server console pages | 16-2 |
| Section 2. Exploring the deployed RuleApps | 16-4 |
| Section 3. Working with deployed resources | 16-5 |
| Section 4. Exploring the Diagnostics and Server Info tabs | 16-6 |
| Section 5. Exploring the REST API tab | 16-7 |
| Section 6. Managing RuleApps and rulesets | 16-8 |
| 6.1. Creating a RuleApp | 16-8 |
| 6.2. Adding a ruleset to your RuleApp | 16-8 |
| 6.3. Adding a managed XOM to your ruleset | 16-11 |
| 6.4. Deleting the RuleApp | 16-11 |
| Section 7. Testing a deployed ruleset | 16-12 |
| 7.1. Testing the ruleset | 16-12 |
| Section 8. Testing a ruleset with no associated managed XOM | 16-15 |
| Exercise 17. Executing rules with Rule Execution Server in Java SE | 17-1 |
| Section 1. Enabling ruleset decision traces | 17-2 |
| 1.1. Setting up your environment for this exercise | 17-2 |
| 1.2. Enabling the tracing ruleset property | 17-2 |
| Section 2. Creating the client project for a RuleApp | 17-4 |
| Section 3. Exploring the generated client application project | 17-7 |
| Section 4. Completing and running the client project for a RuleApp | 17-10 |
| Exercise 18. Executing rules with Rule Execution Server in Java EE | 18-1 |
| Section 1. Building the web application | 18-3 |
| 1.1. Setting up your environment for this exercise | 18-3 |
| 1.2. Finalizing the client application code to call your rules for execution | 18-4 |
| 1.3. Building the application for deployment | 18-5 |
| Section 2. Deploying the web application to WebSphere Application Server | 18-7 |

| | |
|---|-------------|
| 2.1. Deploying the web application | 18-7 |
| 2.2. Verifying application deployment to the application server | 18-7 |
| Section 3. Executing the web application with the RuleApp deployed | 18-9 |
| Exercise 19. Executing rules as a hosted transparent decision service (HTDS) | 19-1 |
| Section 1. Deploying the RuleApp to Rule Execution Server | 19-2 |
| 1.1. Setting up your environment for this exercise | 19-2 |
| 1.2. Deploying the RuleApp | 19-2 |
| Section 2. Getting the HTDS WSDL file from the deployed ruleset | 19-4 |
| Section 3. Creating the client project | 19-7 |
| Section 4. Finalizing and executing the web service client application | 19-12 |
| 4.1. Exploring the web service client application | 19-12 |
| 4.2. Writing the code that executes your ruleset as a web service | 19-12 |
| 4.3. Running the Main class | 19-15 |
| Section 5. Viewing decision service statistics | 19-17 |
| Exercise 20. Auditing ruleset execution through Decision Warehouse | 20-1 |
| Section 1. Enabling ruleset monitoring | 20-3 |
| 1.1. Setting up your environment for this exercise | 20-3 |
| 1.2. Enabling monitoring | 20-3 |
| 1.3. Deploying the updated RuleApp from Rule Designer | 20-5 |
| 1.4. Running the client application | 20-5 |
| Section 2. Retrieving decision traces in Rule Execution Server console | 20-7 |
| Section 3. Optimizing Decision Warehouse | 20-11 |
| 3.1. Working with monitoring options | 20-11 |
| 3.2. Specifying filters on the trace data | 20-12 |
| 3.3. Removing BOM serialization | 20-13 |
| Section 4. Deleting trace information from the database | 20-15 |
| Exercise 21. Working with the REST API | 21-1 |
| Section 1. Using REST services to test ruleset execution | 21-2 |
| Section 2. Using REST services to deploy and execute rules | 21-5 |
| 2.1. Setting up your environment for this exercise | 21-5 |
| 2.2. Deploying the RuleApp resources | 21-6 |
| 2.3. Testing ruleset execution | 21-7 |
| 2.4. Viewing execution traces in Decision Warehouse | 21-8 |
| 2.5. Cleaning the scenario | 21-8 |
| Exercise 22. Working with decision services | 22-1 |
| Section 1. Creating decision service rule projects | 22-2 |
| 1.1. Setting up your environment for this exercise | 22-2 |
| 1.2. Creating a decision operation | 22-3 |
| 1.3. Creating a run configuration for the decision operation | 22-5 |
| Section 2. Creating a main decision service rule project | 22-8 |
| Section 3. Creating decision operations for the decision service | 22-11 |
| 3.1. Creating a decision operation | 22-11 |
| Section 4. Creating and using a deployment configuration | 22-14 |
| 4.1. Creating a deployment configuration | 22-14 |
| 4.2. Setting the target server | 22-15 |
| 4.3. Deploying the decision service | 22-17 |
| 4.4. Deploying the XOM for testing | 22-18 |

| | |
|--|-------|
| 4.5. Verifying deployment in Rule Execution Server console | 22-18 |
| Section 5. Publishing from Rule Designer to Decision Center | 22-20 |
| Section 6. Using the Decision Governance Framework in Business console | 22-22 |
| 6.1. Project management: Paul | 22-22 |
| 6.2. Creating a release: Paul | 22-23 |
| 6.3. Creating a Change Activity: Paul | 22-23 |
| 6.4. Creating a Validation Activity: Paul | 22-24 |
| 6.5. Validation: Abu | 22-25 |
| 6.6. Generating your own scenario file | 22-25 |
| 6.7. Creating the test suite: Abu | 22-27 |
| 6.8. Rule authoring: Bea | 22-29 |
| 6.9. Approving the change activity: Paul | 22-30 |
| 6.10. Validation: Abu | 22-31 |
| 6.11. Approving the activity and deploy: Paul | 22-31 |
| 6.12. Completing the release: Paul | 22-32 |
| 6.13. Deploying the Training Release: Paul | 22-32 |

Trademarks

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

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

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

| | | |
|------------|--------------|-----------------|
| CICS® | DB2® | developerWorks® |
| Express® | ILOG® | Insight™ |
| Insights™ | Orchestrate® | PartnerWorld® |
| Redbooks® | SPSS® | Tivoli® |
| WebSphere® | Worklight® | 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.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

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

Exercises description

This course includes the following exercises:

- Exercise 1, "Operational Decision Manager in action"
- Exercise 2, "Setting up rule projects"
- Exercise 3, "Working with the BOM"
- Exercise 4, "Refactoring"
- Exercise 5, "Working with ruleflows"
- Exercise 6, "Exploring action rules"
- Exercise 7, "Authoring action rules"
- Exercise 8, "Authoring decision tables and decision trees"
- Exercise 9, "Working with static domains"
- Exercise 10, "Working with dynamic domains"
- Exercise 11, "Queries and ruleset extraction"
- Exercise 12, "Executing rules locally"
- Exercise 13, "Debugging a ruleset"
- Exercise 14, "Enabling tests and simulations"
- Exercise 15, "Managing deployment"
- Exercise 16, "Exploring the Rule Execution Server console"
- Exercise 17, "Executing rules with Rule Execution Server in Java SE"
- Exercise 18, "Executing rules with Rule Execution Server in Java EE"
- Exercise 19, "Executing rules as a hosted transparent decision service (HTDS)"
- Exercise 20, "Auditing ruleset execution through Decision Warehouse"
- Exercise 21, "Working with the REST API"
- Exercise 22, "Working with decision services"

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

Most of the exercises are independent and do not require you to complete a previous exercise. Exercises that you do in Rule Designer have "start" files and "answer" (or "solution") files. You can use the "answer" files when you run out of time or are unable to complete the exercise correctly. If necessary,

the start and answer files make it possible to skip exercises or do them out of sequence. However, you are encouraged to complete them in the order in which they are presented. In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

Most exercises include required sections that should always be completed. It might be necessary to complete these sections before you can start later exercises. Some exercises might also include optional sections that you might want to complete if you have sufficient time and want an extra challenge.

General exercise information

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

User IDs and passwords

Here is a list of user ID and password information for this course.

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

How to follow the exercise instructions

Exercise structure

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

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



Information

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

Here is an example from Exercise 1 of this course.



Example

Excerpt from Exercise 1

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

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

Text highlighting in exercises

Different text styles indicate various elements in the exercises.

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

- Menu items
- Field names
- Icons

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

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

Tracking your progress

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

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

Required exercise sections

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

Dependencies between exercises are listed in the exercise introduction.

Optional exercise sections

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

File references for exercises

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

- <*InstallDir*>: This directory is the IBM Operational Decision Manager installation directory, which includes two subdirectories:
 - ODM: Contains Operational Decision Manager
 - WAS: Contains WebSphere Application Server
- <*LabfilesDir*>: This directory contains the files that are required during demonstrations, exercises, and the workshop steps, such as samples of code that you can copy and paste.

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

- <*InstallDir*> is: C:\Program Files\IBM\ODM87
This folder is the default IBM Operational Decision Manager installation directory on Windows.
- <*LabfilesDir*> is: C:\labfiles

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



Stop

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

Projects for exercises

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

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

To open Rule Designer, you click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Rule Designer** (or you can use the shortcut on the desktop).



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

C:\labfiles\workspaces\myWorkspace

When you type a path, an empty workspace is created.

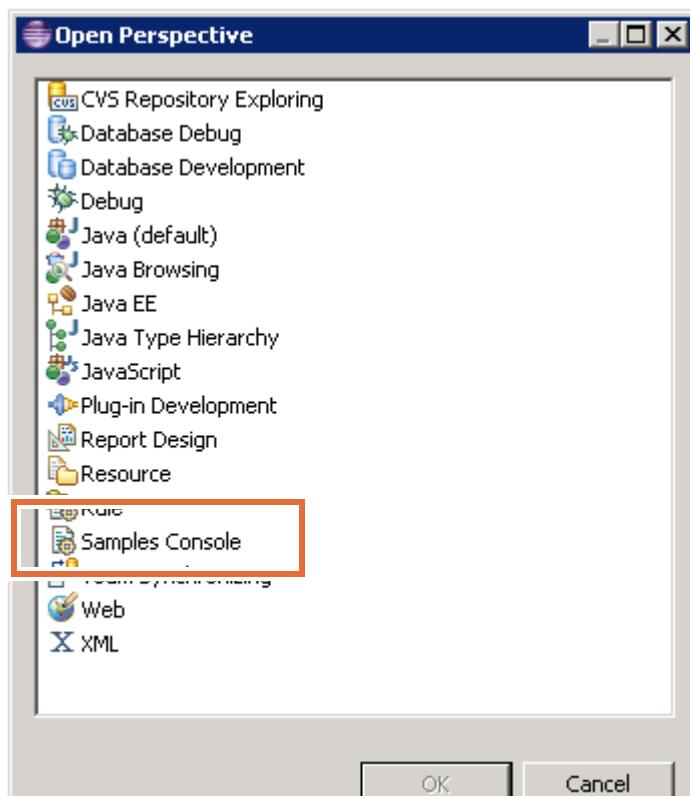


Troubleshooting

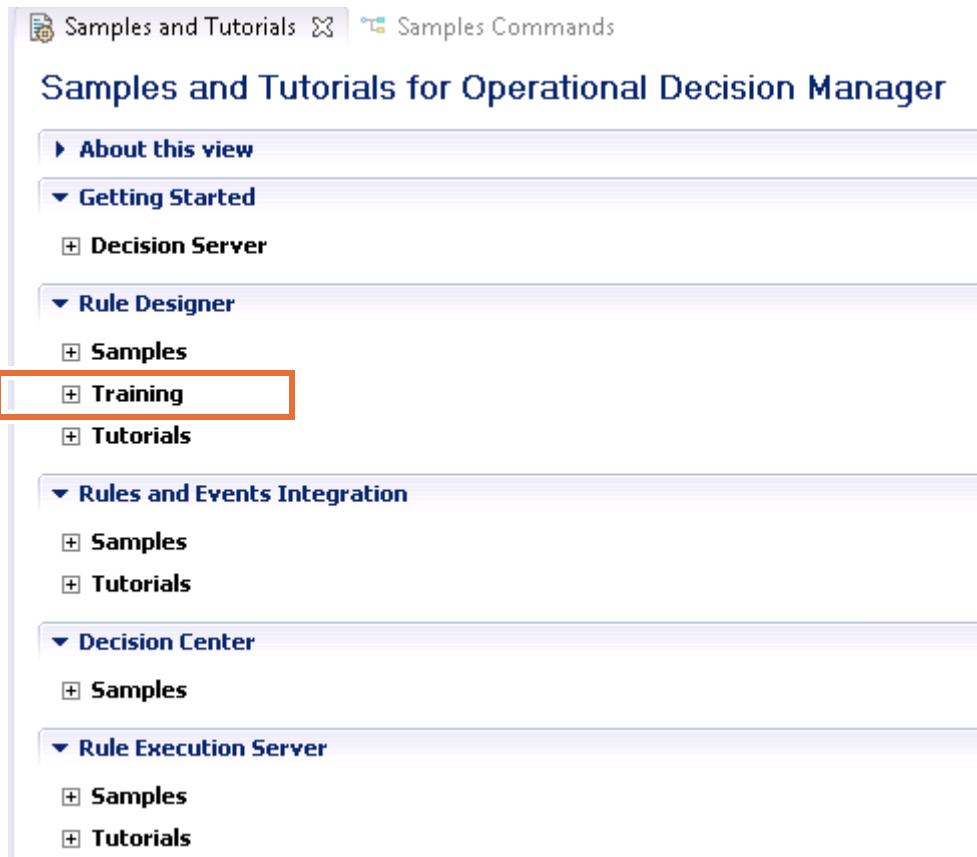
With Eclipse 4.2.2, the Rule perspective does not open directly, so you must close the Welcome view and manually switch perspectives by clicking the **Open Perspective** icon in the upper-right corner of the Eclipse window.



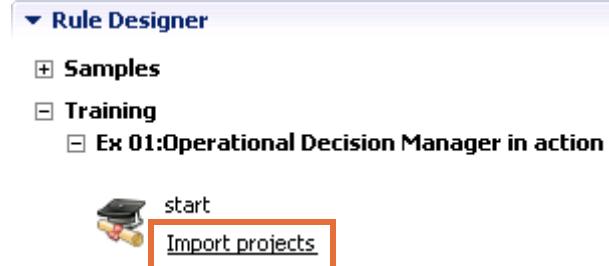
You can choose which perspective to use from the Open Perspective dialog box. For most exercises, you start from the Samples Console perspective.



The Samples and Tutorials page opens. In the **Rule Designer** section, notice the additional **Training** node. When you expand this node, you see all the exercises for this course.



For example, expand **Training > Ex 01: Operational Decision Manager in action**, and under **start**, click **Import projects**.



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

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



Sample server port

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

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

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

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

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

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

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

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

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

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

`C:\Program Files\IBM\ODM87\WAS\AppServer\profiles\ODMSample8700\logs`

- b. Open the file `AboutThisProfile.txt` with any text editor.

- c. Find the value of `PORT` at the end of the line that starts with:

`HTTP transport port:`



Stop

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

Using the product documentation

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

To access the local documentation while working in Rule Designer, you must first start it by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Information Center for Operational Decision Manager (local)**.

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

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.0



Information

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

Exercise 1. Operational Decision Manager in action

What this exercise is about

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

What you should be able to do

After completing this exercise, you should be able to:

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

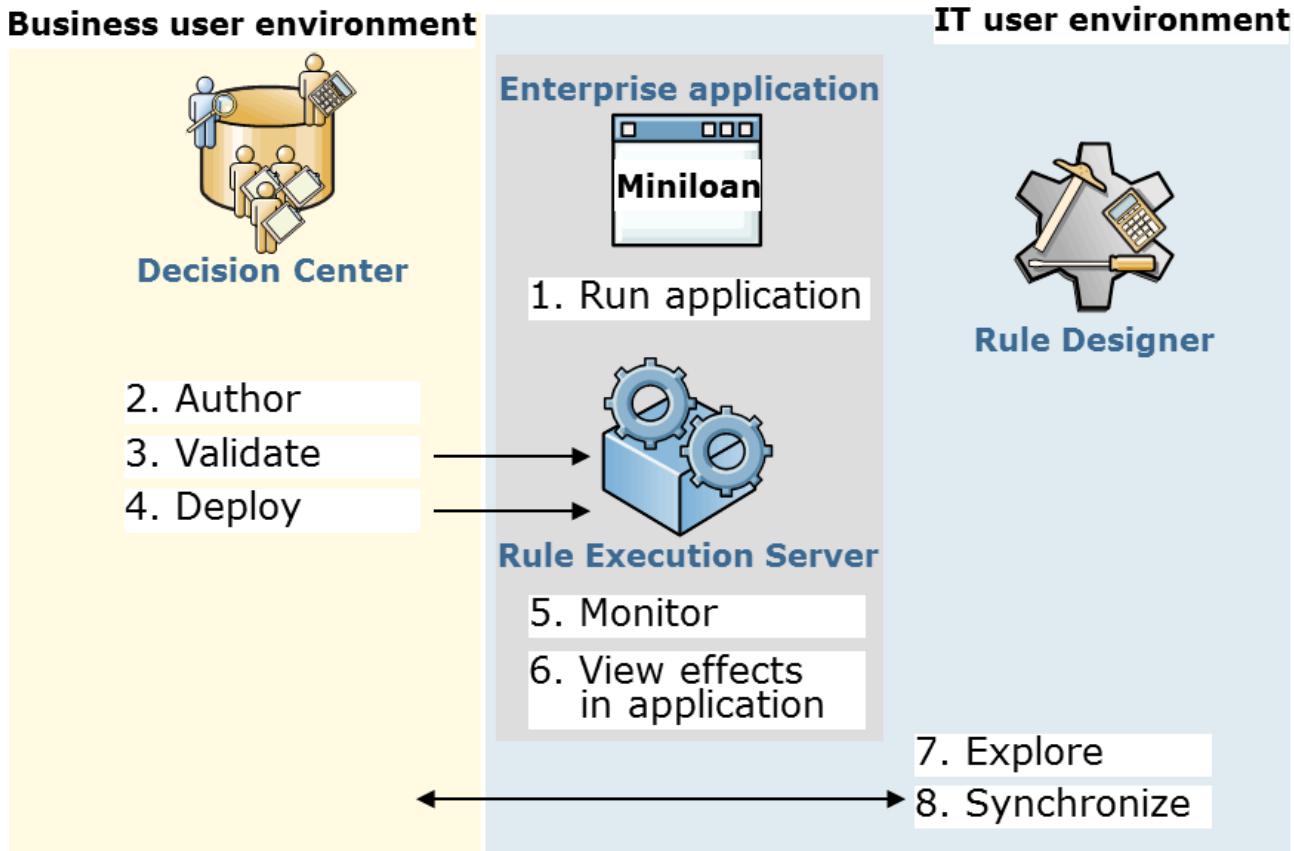
Introduction

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

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

- "Running the Miniloan web application"
- "Managing business rules in Decision Center"
- "Validating changes with Decision Validation Services"
- "Deploying updated rules from Decision Center to Rule Execution Server"
- "Monitoring the deployed rules in Rule Execution Server"
- "Viewing the effects in the Miniloan application"
- "Synchronizing environments between Rule Designer and Decision Center"

The exercise workflow is shown here.



Requirements

There are no specific requirements for this exercise.



Important

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

<InstallDir>\ODM\studio\training

The default directory for <InstallDir> is C:\Program Files\IBM\ODM87. If you are not using the provided lab environment for this course, make sure that you know the location of <InstallDir>.

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

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

Section 1. Running the Miniloan web application

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

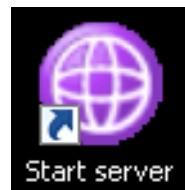
Scenario

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

1.1. Setting up your environment

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

- 1. Start the sample server.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server** (or you can use the “Start server” shortcut on the desktop).



- b. Wait until the server is started.

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

```
Start server
[exec] checkIfServerExistsAndIsRunning:
[exec]
[exec] startServerIfNotAlreadyRunning:
[exec]
[exec] wsStartServer:
[exec] [startServer] profileName=ODMSample8700 registry=C:\Program Files\I
M\ODM87\WAS\AppServer\properties\profileRegistry.xml
[exec] [startServer] profileHome=C:\Program Files\IBM\ODM87\WAS\AppServer\p
rofiles\ODMSample8700
[exec]
[exec] BUILD SUCCESSFUL
[exec] Total time: 1 minute 24 seconds

checkIfIsInstallingDC:

update.decisioncenter.db:

delete.new.installation.files:
[samples.echo] Dec 17, 2014 12:42:17 PM com.ibm.rules.sampleserver.Log info
[samples.echo] INFO: GBRPS0029I: start.server is completed.
[samples.echo] GBRPS0029I: start.server is completed.

BUILD SUCCESSFUL
Total time: 1 minute 45 seconds
Press any key to continue . . .
```

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

1.2. Running the application

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

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

The Miniloan application opens.

The screenshot shows the 'Miniloan Validation' application. At the top, there are two sections: 'Borrower Information' and 'Loan Information'. Under 'Borrower Information', the 'Name' field contains 'Joe'. Under 'Loan Information', the 'Amount' field contains '500000'. Below these sections is a large button labeled 'Validate Loan'. At the bottom of the page, there is a section titled 'Rule set Information' with a 'Version' dropdown menu showing 'Latest'.

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

Validate Loan

Rule set Information

Version Latest

Scenario

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

Is Joe eligible for this loan?

- ___ 2. Click **Validate Loan**.

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

| Miniloan Validation | |
|--|---|
| Borrower Information | Loan Information |
| Name: <input type="text" value="Joe"/> | Amount: <input type="text" value="500000"/> |
| Yearly Income: <input type="text" value="80000"/> | Duration (months): <input type="text" value="240"/> |
| Credit Score: <input type="text" value="600"/> | Yearly Interest Rate : <input type="text" value="0.05"/> |
| Validate Loan | |
| The loan is rejected. | |
| Messages: | |
| Too big Debt-To-Income ratio | |
| Ruleset Information | |
| Version <input type="text" value="Latest"/> | |
| Rules Execution Summary [1 rule(s) executed.] | |
| Ruleset version | Latest |
| Rule(s) executed. | Rule eligibility.minimum income was executed in rule task miniloan#eligibility. |

Scenario

Joe's loan request is rejected. Miniloan responds by reporting that the ratio of debt to income is too high. Operational Decision Manager provides detailed traceability that allows Miniloan to explain to Joe why this loan was rejected.

In the Miniloan application interface, you can see exactly which rules were involved in the decision.

- 3. Look at the **Rules Execution Summary** section of the application interface, and notice that the `eligibility.minimum income` rule was executed to produce this decision.



Information

A *ruleset* is a set of rules that can be deployed to an application and return a business decision.

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

As you can see, decisions that are made by using your business rules can 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.

Operational Decision Manager 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

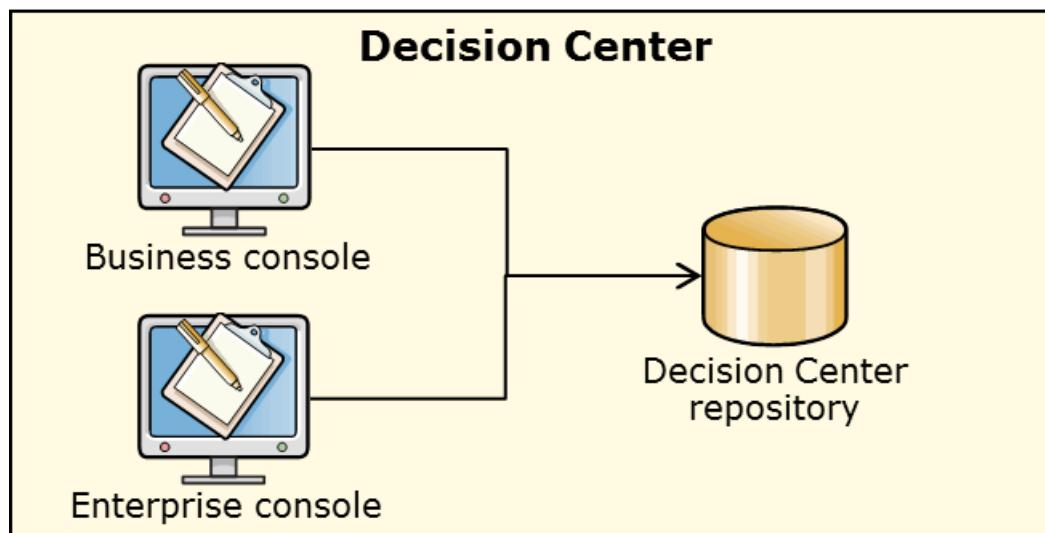
Several customers, including Joe, complained about having their loan requests rejected. You do not change the rules merely so that Joe can get a house, but a significant number of loan rejections might be an indication of errors in the rules. Some types of errors can be corrected early on, while in other 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 a business user profile 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.

In later sections of this exercise, you use Decision Center Enterprise console, which is for advanced business users who manage and administer rules, to run validation tests and deploy rules.



2.1. Editing rules in Decision Center Business console

- 1. Open the Business console from the Start menu by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Decision Center Business console**. You can also double-click the “Decision Center Business console” shortcut on the desktop.



Information

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

<http://localhost:9080/decisioncenter>

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

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

Privacy

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

Agree and Proceed



Troubleshooting

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

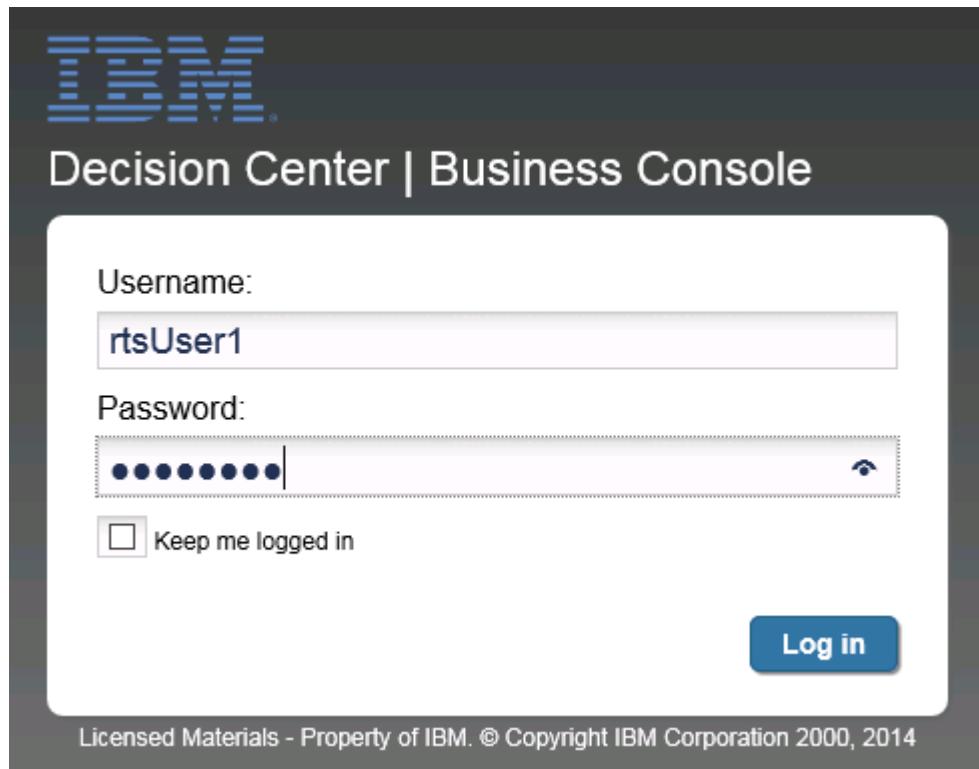
- 3. Sign in as a business user with the following values for the **Username** and **Password** fields.

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



Note

These values are case-sensitive.



- ___ 4. Click **Log in**.



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

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



- ___ 6. Click **Classic Rule Projects**.

A screenshot of the "Classic Rule Projects" interface. It shows a header with "Decision Services" and "Classic Rule Projects" (which is highlighted with a red box). Below the header are filters for "Date" and "Name". A project card for "Loan Validation Service" is displayed, showing it was created by "rtsAdmin" on "Nov 13, 2014". A green "NEW" badge is visible in the top right corner of the card.

___ 7. Scroll down to find **miniloan-rules** and click it.



The **eligibility** folder is open and lists the following rules:

- minimum credit score
- minimum income
- repayment and score

A screenshot of the IBM ODM interface showing the "Rules" tab selected. The top navigation bar shows "All Projects > miniloan-rules > main". Below the navigation, there are filters for "Project: miniloan-rules" and search fields for "Name" and "Folder". The main content area shows a list of rules under the "eligibility" folder. Each rule is represented by a small icon and its name: "minimum credit score", "minimum income", and "repayment and score". There is also a section labeled "validation".

- __ d. 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' ;

```

```

miniloan-rules > main >
minimum income (v1.0) ★ ⓘ

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.

- \_\_ 8. Edit the rule and change the debt-to-income ratio from 0.3 to 0.5.

- \_\_ a. Click **Edit** to open the rule editor.



- \_\_ b. In the **if** part of the rule, change 0.3 to: 0 . 5

**if**

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

- \_\_\_ 9. 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 new 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**

- \_\_\_ 10. 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**.

A new version of this element will be created.

You can add a comment to associate with this version which can be viewed in the timeline.

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

**Create New Version**

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

miniloan-rules > main >

 **minimum income (v1.1)** ★ ⓘ

**if**

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

**then**

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

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

## 2.2. Reviewing the rule history

- \_\_\_ 1. Compare versions 1.0 and 1.1 of the minimum income rule.  
\_\_\_ a. Click **Compare**.

miniloan-rules > main >

 **minimum income (v1.1)** ★ ⓘ

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

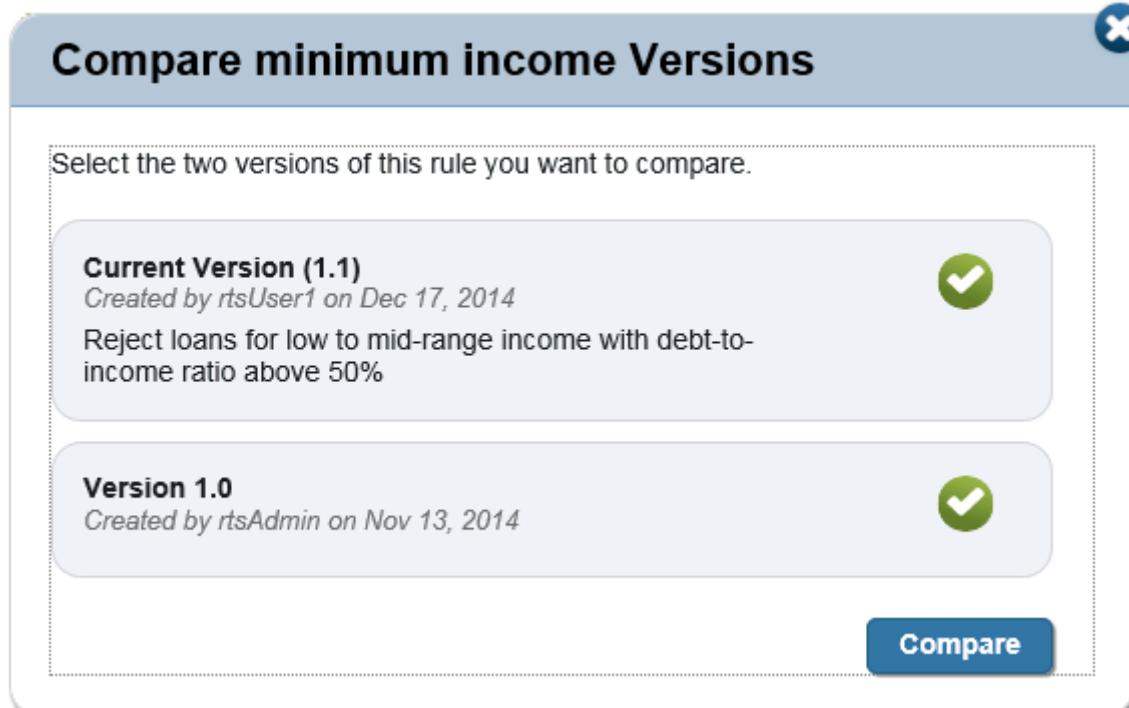
**Compare** 

**Properties** **Stream**

Updated by rtsL Dec 17, 2014

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

- 2. Click **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.

| ▼ Hide summary | Content (2)                                                                                                                                                               | Properties (0) |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
|                | <ul style="list-style-type: none"> <li>★ <i>0.3 was changed to 0.5</i></li> <li>✚ <i>and the yearly income of 'the borrower' is less than 500000</i> was added</li> </ul> |                |

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 Dec 17, 2014

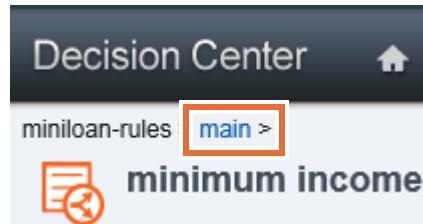
**if**  
the yearly repayment of '**the loan**' is more than the yearly income of '**the borrower**' \* **0.5**  
**and the yearly income of 'the borrower'** is less than **500000**

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

- \_\_\_ 3. Close the Compare window and return to the miniloan-rules project.
- \_\_\_ a. Click X in the upper-right corner of the Compare window to return to the minimum income rule main view.



- \_\_\_ b. Click the **main** breadcrumb to return to the miniloan-rules project.



## 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. Click **repayment and score** to open the repayment and score decision table.

The screenshot shows the IBM ODM interface. At the top, it says "All Projects > miniloan-rules > main". There is a "Take Snapshot" button. Below that, there are two tabs: "Rules" and "Snapshots", with "Rules" selected. A search bar says "Project: miniloan-rules". Under the "Rules" tab, there is a section titled "eligibility" with three items: "minimum credit score", "minimum income", and "repayment and score". The "repayment and score" item is highlighted with a red rectangle.

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

The screenshot shows a dialog box titled "repayment and score". It contains the following text:  
By default, the row order of this decision table is set to Automatic. The table opens in an editor that automatically organizes the rows based on their content.  
To set the row order manually, select Manual.  
Choose how you want the row order to be set  
A dropdown menu shows "Automatic" with a checked checkbox next to it. To its right is a blue button labeled "Help me decide".  
The text continues: The decision table applies your change when you click OK. You can change this property anytime in the Details dialog of this table.  
At the bottom right is a blue "OK" button with a white arrow pointing to it.

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

miniloan-rules > main >

### repayment and score (v1.0)

|   | debt to income | credit score | message                                                                             |
|---|----------------|--------------|-------------------------------------------------------------------------------------|
| 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; 600[     | debt-to-income too high compared to credit score                                    |
| 6 | [45 %; 50 %[   | [600; 800[   |  |
| 7 | ≥ 50 %         | [0; 800[     | debt-to-income too high compared to credit score                                    |

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

miniloan-rules > main >

### repayment and score (v1.0)

|   | debt to income | credit score |
|---|----------------|--------------|
| 1 | [0 %; 30 %[    | [0; 200[     |
| 2 | [0 %; 30 %[    | [200; 800[   |
| 3 | [30 %; 45 %[   | [0; 400[     |
| 4 | [30 %; 45 %[   | [400; 800[   |
| 5 | [45 %; 50 %[   | [ 0 ; 500 [  |
| 6 | [45 %; 50 %[   | [600; 800[   |
| 7 | ≥ 50 %         | [0; 800[     |

- \_\_\_ f. Click another area of the table to make sure the value changed.

- \_\_\_ g. Click **Save**.

- \_\_\_ h. 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

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



## Troubleshooting

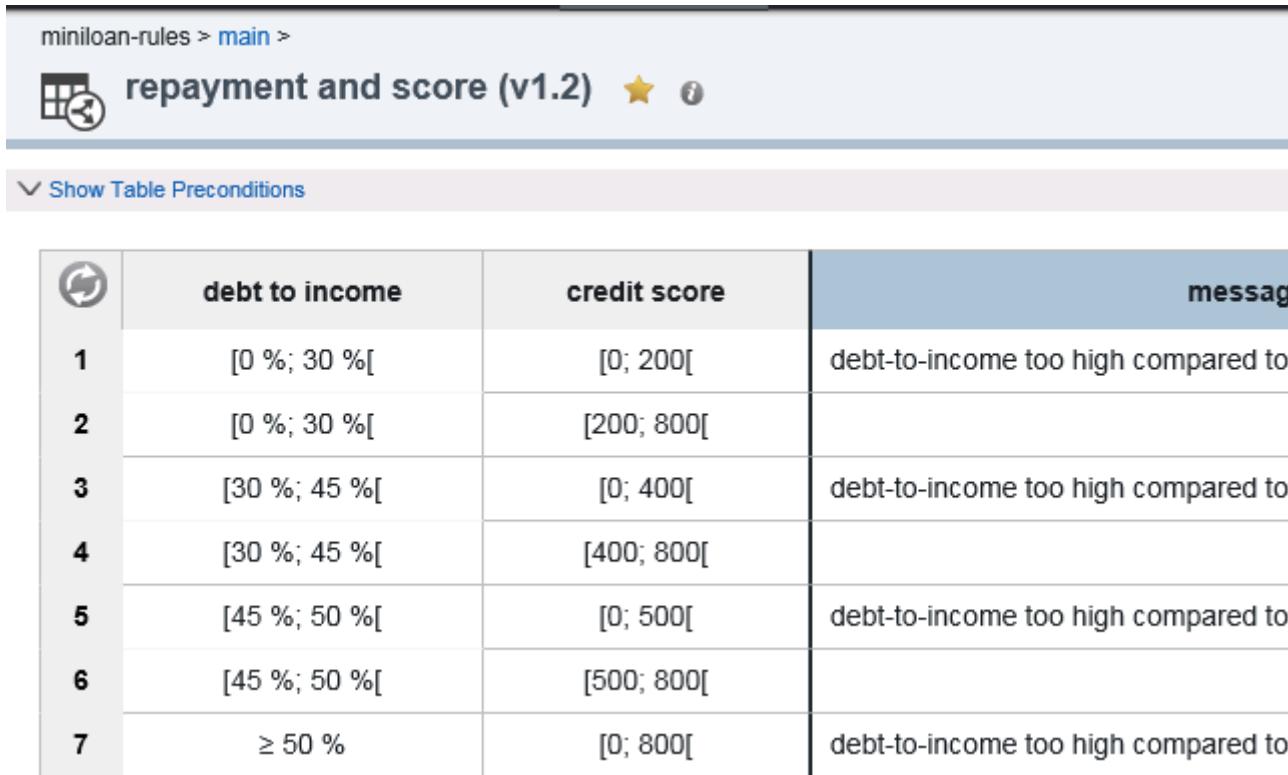
If your change is not saved, go back to the rule editor by clicking **Edit**, change the value again, and then click a different cell before clicking **Save**.

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.

| credit score | message                                          |
|--------------|--------------------------------------------------|
| [0; 200[     | debt-to-income too high compared to credit score |
| [200; 800[   |                                                  |
| [0; 400[     | debt-to-income too high compared to credit score |
| [400; 800[   |                                                  |
| [0; 500[     | debt-to-income too high compared to credit score |
| [600; 800[   | Error:<br>Rows have gap(s).                      |
| [0; 800[     | debt-to-income too high compared to credit score |

- \_\_ 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.



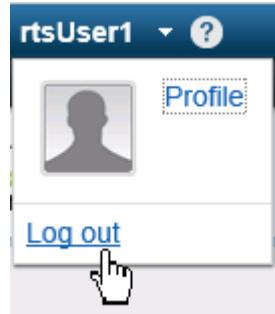
miniloan-rules > main >

**repayment and score (v1.2)** ★ ⓘ

▼ Show Table Preconditions

|   | debt to income | credit score | message                             |
|---|----------------|--------------|-------------------------------------|
| 1 | [0 %; 30 %[    | [0; 200[     | debt-to-income too high compared to |
| 2 | [0 %; 30 %[    | [200; 800[   |                                     |
| 3 | [30 %; 45 %[   | [0; 400[     | debt-to-income too high compared to |
| 4 | [30 %; 45 %[   | [400; 800[   |                                     |
| 5 | [45 %; 50 %[   | [0; 500[     | debt-to-income too high compared to |
| 6 | [45 %; 50 %[   | [500; 800[   |                                     |
| 7 | ≥ 50 %         | [0; 800[     | debt-to-income too high compared to |

- \_\_\_ 3. Log out of 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 3. Validating changes with Decision Validation Services

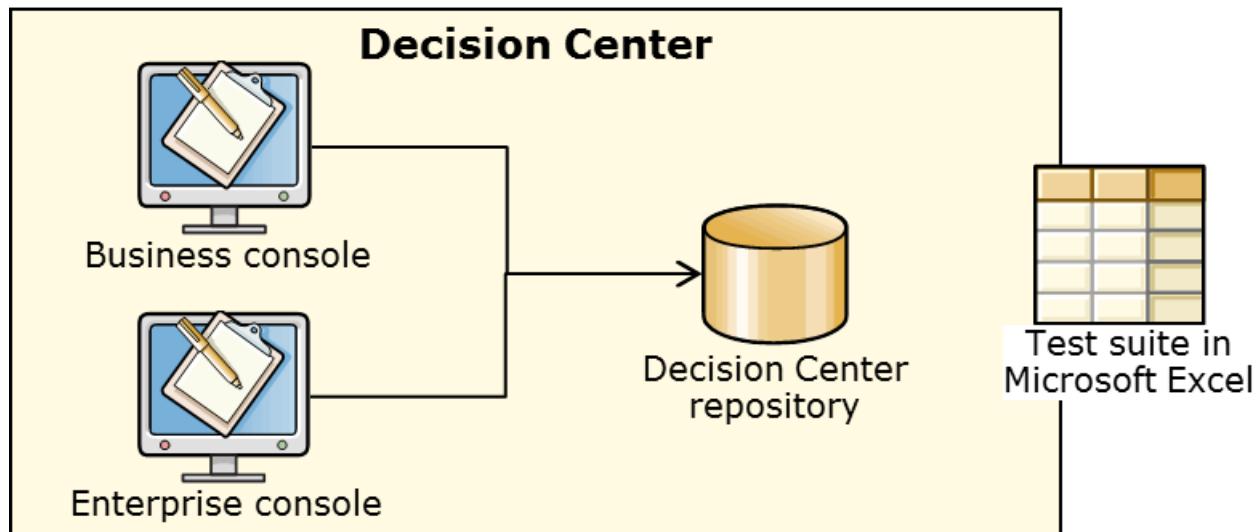
### Scenario

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 Decision Validation Services, the rule authors can run tests and simulations directly from the Decision Center Enterprise console. As of this writing, Decision Center Business console does not offer the ability to use Decision Validation Services.

In this section, you run tests and simulations from the Decision Center Enterprise 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>\ODM\gettingstarted\EnterpriseConsole
```

The scenario file is available in Microsoft Excel 2003 format. You can open it and click the **Scenarios** tab to view the scenarios that you are about to test. The scenario file contains two scenarios with their corresponding expected results. The scenarios are listed in the **Scenarios** tab, and the anticipated results in the **Expected Results** tab.

### 3.1. Opening the Decision Center Enterprise console

- \_\_ 1. Sign in to Decision Center Enterprise console as `rtsUser1` to access the **miniloan-rules** project.
  - \_\_ a. Open the Enterprise console from the Start menu by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Decision Center Enterprise console**. You can also use the shortcut on the desktop.



#### Information

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

`http://localhost:9080/teamserver`

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

- \_\_ b. Enter `rtsUser1` in the **Username** and **Password** fields and click **Log In**.

- \_\_ c. On the **Home** tab of the Enterprise console, select **Work on a project**, and then select **miniloan-rules** from the **Project in use** list.

The screenshot shows the IBM Decision Center interface. At the top, there is a navigation bar with tabs: Home, Explore, Compose, Query, Analyze, and Project. The 'Explore' tab is currently selected. Below the navigation bar, the page title is "Welcome to the Decision Center Home Page". Under this title, there are two main sections: "Work on a project" and "Work on a Decision Service".

**Work on a project:**

- Work on a project
- Work on a Decision Service

Project in use:  **loanvalidation-rules** (highlighted with a red box)

Branch in use:  **miniloan-rules** (highlighted with a red box)

Current action:  **Work on branch** (highlighted with a red box)

**Work on a Decision Service:**

Decision Service in use:  **Loan Validation Service** (highlighted with a red box)

Branch in use:  **<none>** (highlighted with a red box)

Current action:  **Work on branch** (highlighted with a red box)

- \_\_ 2. Click the **Explore** tab and click the **eligibility** folder.

Notice the folders and rules that you see here match the rules in the Business console.

### 3.2. Creating and running a test suite

- \_\_ 1. On the **Compose** tab, create a test suite.  
\_\_ a. Click the **Compose** tab.

- \_\_\_ b. Select **Test Suite**, and click **OK**.



- \_\_\_ 2. In Step 1: Properties, enter **Miniloan Test** as name for the test suite, and click **Next**.

### Properties

|                      |                                                                         |
|----------------------|-------------------------------------------------------------------------|
| <b>Name*</b>         | <input type="text" value="Miniloan Test"/>                              |
| <b>Folder</b>        | <input type="text" value="/"/>                                          |
| <b>Group</b>         | <input type="text" value="&lt;none&gt;"/>                               |
| <b>Documentation</b> | <div style="border: 1px solid #ccc; height: 150px; width: 100%;"></div> |

Move on to next step

- \_\_\_ 3. In Step 2: Rules Tested, click **Next** to test all the rules in the project.

### Rules tested

By default, all rules of the current branch will be tested.

Or you can test:

rules from a baseline of this branch: <none> ▾

rules selected from: <none> ▾

For this ruleset, the entry point is: <default> (miniloan) 

|                                       |                                         |                                                                                                                                                                                                                                                                                                                                                |                                       |                                               |
|---------------------------------------|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|-----------------------------------------------|
| <input type="button" value="Cancel"/> | <input type="button" value="Previous"/> | <input style="background-color: #cccccc; cursor: pointer; border: none; font-weight: bold; color: black; padding: 0 5px; margin-left: 5px; margin-right: 5px; border-radius: 5px; font-size: 10pt; height: 25px; width: 150px; vertical-align: middle; text-decoration: none; text-align: center; outline: none;" type="button" value="Next"/> | <input type="button" value="Finish"/> | <input type="button" value="Finish and Run"/> |
| Move on to next step                  |                                         |                                                                                                                                                                                                                                                                                                                                                |                                       |                                               |

- \_\_\_ 4. In Step 3: Scenarios, select **Excel 2003** as the format for the scenario file.

- \_\_\_ 5. To upload the scenario file, click **Browse**, and select:

<InstallDir>\ODM\gettingstarted\EnterpriseConsole\miniloan-test.xls

Where, <InstallDir> refers to the installation directory, for example:

C:\Program Files\IBM\ODM87

### Scenarios

**Format:**

**Generate template:** Generate a scenario file template for this test suite that you will edit or

**File:**

|                                       |                                         |                                     |                                       |                                               |
|---------------------------------------|-----------------------------------------|-------------------------------------|---------------------------------------|-----------------------------------------------|
| <input type="button" value="Cancel"/> | <input type="button" value="Previous"/> | <input type="button" value="Next"/> | <input type="button" value="Finish"/> | <input type="button" value="Finish and Run"/> |
|---------------------------------------|-----------------------------------------|-------------------------------------|---------------------------------------|-----------------------------------------------|

- \_\_\_ 6. Click **Finish and Run**.

- \_\_\_ 7. On the Run page, make sure that the sample server is selected, and click **Run**.

Decision Center might take a few seconds to complete this action. When testing is complete, the report opens, and you can expand the details for each scenario to see the results.

### Summary

|                         |                                                                                        |
|-------------------------|----------------------------------------------------------------------------------------|
| Number of scenarios     | 2                                                                                      |
| Scenarios with failures | 1     |
| Success Rate            | 50%  |

### Details

| Name                                                                                         | Status                                                                                                                                                                                                                                                                                                                |                                                                     |      |         |                                                                                   |                             |                                                                     |
|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|------|---------|-----------------------------------------------------------------------------------|-----------------------------|---------------------------------------------------------------------|
|  Scenario 1 |                                                                                                                                                                                                                                     |                                                                     |      |         |                                                                                   |                             |                                                                     |
|                                                                                              | <table><thead><tr><th>Status</th><th>Test</th><th>Message</th></tr></thead><tbody><tr><td></td><td>the loan is approved equals</td><td>The observed value "true" does not equal the expected value "false"</td></tr></tbody></table> | Status                                                              | Test | Message |  | the loan is approved equals | The observed value "true" does not equal the expected value "false" |
| Status                                                                                       | Test                                                                                                                                                                                                                                                                                                                  | Message                                                             |      |         |                                                                                   |                             |                                                                     |
|             | the loan is approved equals                                                                                                                                                                                                                                                                                           | The observed value "true" does not equal the expected value "false" |      |         |                                                                                   |                             |                                                                     |
|  Scenario 2 |                                                                                                                                                                                                                                     |                                                                     |      |         |                                                                                   |                             |                                                                     |

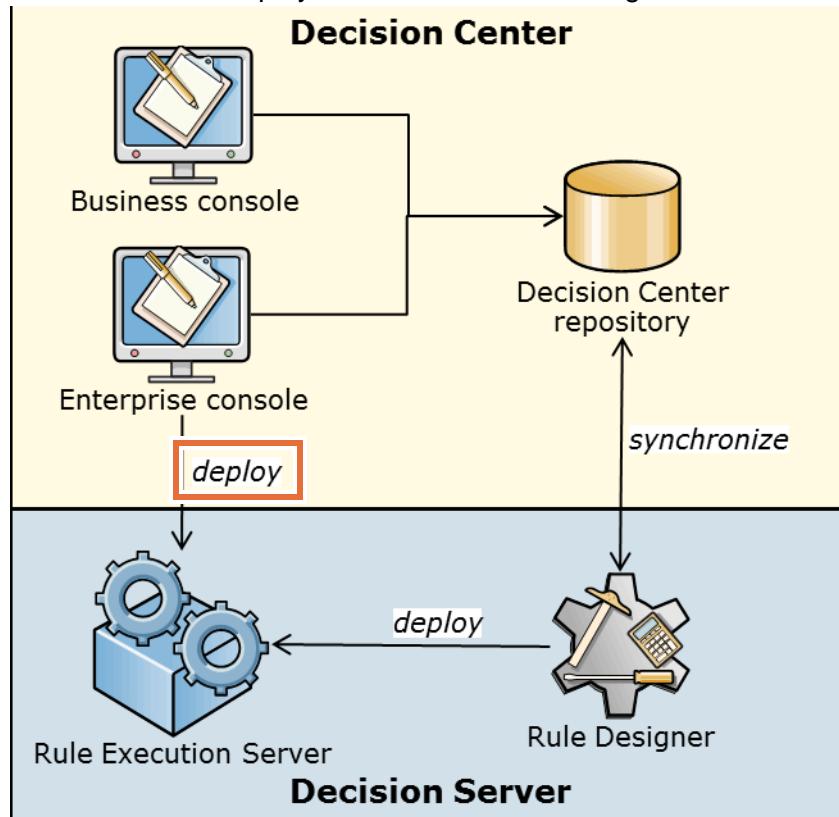
Notice that Scenario 1 failed because the expected result was that the loan would be rejected. However, now that you increased the allowable debt-to-income ratio in the minimum income rule, the loan is approved.

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

### Scenario

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

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



In this task, you take the role of configuration manager in Decision Center Enterprise console to create a RuleApp, which serves as a container for the `miniloan-rules` project.



### Information

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

## 4.1. Creating the RuleApp

- 1. Sign out of the Decision Center Enterprise console.



- 2. Sign in again, but this time as a user with configuration privileges:
- **User name:** rtsConfig
  - **Password:** rtsConfig
- 3. On the Decision Center **Home** tab, make sure that **miniloan-rules** is selected in the **Project in use** list.
- 4. Click the **Configure** tab.
- 5. Click **Manage RuleApps**.



### Configure

#### Deployment

##### [Edit Ruleset Extractors](#)

View and edit the list of extractors used to generate rulesets

##### [Manage RuleApps](#)

Manage RuleApps, generate a RuleApp archive, deploy a RuleApp on a Rule Execution Server

##### [Manage Servers](#)

Create, delete, and edit the servers on which you deploy your projects

- 6. Under **Available RuleApps**, click **New**.  
 — 7. In the **Name** field, type: miniloanruleapp



#### Hint

Make sure that you type the names exactly as described in these instructions.

- \_\_\_ 8. In the **Rulesets** section, click **New** to add a ruleset to the RuleApp.

The screenshot shows a 'New RuleApp' dialog box. At the top is a blue cube icon labeled 'New RuleApp'. Below it are two buttons: 'Save' (blue) and 'Cancel' (red). The main area is titled 'Properties' and contains the following fields:

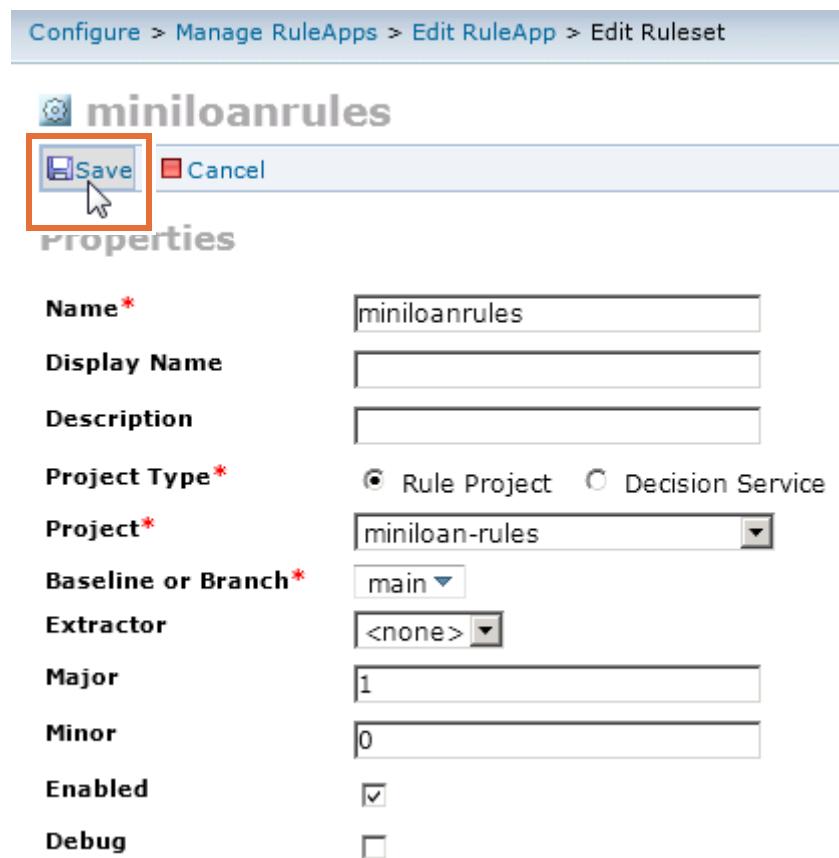
|              |                   |
|--------------|-------------------|
| Name*        | miniloanruleapp   |
| Display Name | (empty)           |
| Major        | 0                 |
| Minor        | 0                 |
| Description  | (empty text area) |

Below this is a section titled 'Rulesets' with the sub-section 'No ruleset found'. At the bottom are three buttons: 'New' (highlighted with a red box), 'Edit', and 'Delete'.

The New Ruleset page opens.

- \_\_\_ 9. In the **Name** field, type: miniloanrules  
\_\_\_ 10. For **Project Type**, keep the default selection: **Rule Project**  
\_\_\_ 11. In the **Project** field, select **miniloan-rules** to specify that this ruleset comes from the miniloan-rules project.  
\_\_\_ 12. Keep **main** for the **Baseline or Branch** field.

13. At the top of the page, click **Save** to save the ruleset.

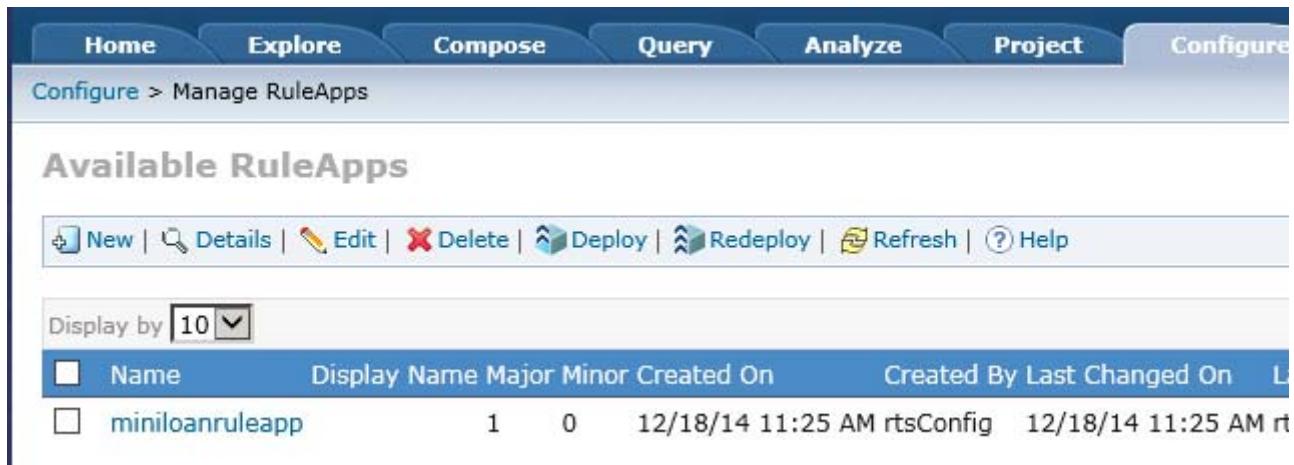


The screenshot shows the 'miniloanrules' properties dialog box. The 'Save' button at the top left is highlighted with a red box. Other buttons include 'Cancel'. The form contains the following fields:

|                            |                                                                                      |
|----------------------------|--------------------------------------------------------------------------------------|
| <b>Name*</b>               | miniloanrules                                                                        |
| <b>Display Name</b>        |                                                                                      |
| <b>Description</b>         |                                                                                      |
| <b>Project Type*</b>       | <input checked="" type="radio"/> Rule Project <input type="radio"/> Decision Service |
| <b>Project*</b>            | miniloan-rules                                                                       |
| <b>Baseline or Branch*</b> | main                                                                                 |
| <b>Extractor</b>           | <none>                                                                               |
| <b>Major</b>               | 1                                                                                    |
| <b>Minor</b>               | 0                                                                                    |
| <b>Enabled</b>             | <input checked="" type="checkbox"/>                                                  |
| <b>Debug</b>               | <input type="checkbox"/>                                                             |

- c. When prompted to save the RuleApp with the added ruleset, click **Save**.

Your RuleApp is ready to be deployed.



The screenshot shows the 'Available RuleApps' list view. The 'miniloanruleapp' entry is selected, indicated by a blue selection bar. The list includes:

| Name            | Display Name    | Major | Minor | Created On        | Created By | Last Changed On   | Last Changed By |
|-----------------|-----------------|-------|-------|-------------------|------------|-------------------|-----------------|
| miniloanruleapp | miniloanruleapp | 1     | 0     | 12/18/14 11:25 AM | rtsConfig  | 12/18/14 11:25 AM | rtsConfig       |

Next, you deploy the RuleApp directly to the Rule Execution Server.

## 4.2. Deploying the RuleApp

- Under Available RuleApps, select miniloanruleapp, and on the toolbar, click Deploy.

### Available RuleApps

The screenshot shows a software interface titled "Available RuleApps". At the top is a toolbar with icons for New, Details, Edit, Delete, Deploy (which has a mouse cursor pointing at it), Redeploy, Refresh, and Help. Below the toolbar is a dropdown menu labeled "Display by" with the value "10". The main area is a table with columns: Name, Display Name, Major, Minor, Created On, Created By, Last Changed On, and Last. One row is selected, showing "miniloanruleapp" in the Name column and "V1.0" in the Major column.

- On the Deployment Baseline page, in the **Create a baseline for this deployment** field, type V1.0, and click **Next**.

**Deployment Baseline**

Create a baseline for this deployment  
V1.0

Cancel Previous Next

This step creates a deployment baseline that is called V1.0, that is, a snapshot in Decision Center of the version of each element about to be deployed.

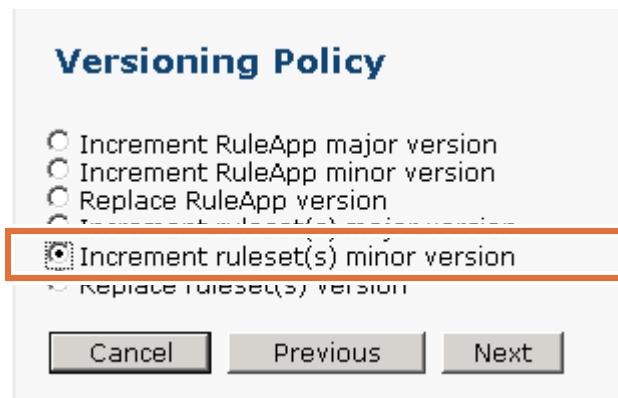
- On the “RuleApp target” page, keep **Deploy on a Rule Execution Server** selected, and click **Next**.

**RuleApp target**

Deploy on a Rule Execution Server  
 Generate a RuleApp archive

Cancel Previous Next

- \_\_\_ 4. On the Versioning Policy page, select **Increment ruleset(s) minor version**, which leaves the RuleApp version as 1.0, but increases the ruleset version to 1.1.



- \_\_\_ 5. Click **Next**.  
\_\_\_ 6. On the Server List page, keep **Sample** selected and click **Deploy**.



When processing is complete, you see the Deployment Succeeded message.

| Archive content      | Operation                           | Result                     |
|----------------------|-------------------------------------|----------------------------|
| /miniloanruleapp/1.0 | Element updated                     | /miniloanruleapp/1.0       |
| /miniloanrules/1.0   | + Version changed and element added | /miniloanrules/ <b>1.1</b> |

### List of deployment baselines created

| Baseline Name | Project        | RuleApp         | Ruleset       |
|---------------|----------------|-----------------|---------------|
| V1.0          | miniloan-rules | miniloanruleapp | miniloanrules |

**Back**

- \_\_\_ 7. Click **Back**, log out of the Enterprise console, and close the Enterprise console window.

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

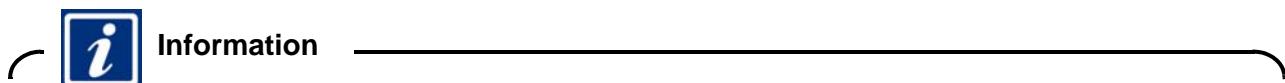
You can now go to the Rule Execution Server console and see whether the RuleApp was properly deployed.

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

### To view the deployed RuleApp:

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

`http://localhost:9080/res`



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

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

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

The screenshot shows the Rule Execution Server console interface. The title bar reads "IBM Rule Execution Server console". The top navigation bar has tabs: Home, Explorer (which is selected), Decision Warehouse, Diagnostics, and Servers. Below the tabs, the breadcrumb navigation shows "Explorer > RuleApps".

The main area is titled "RuleApps View". It contains three buttons: "Add RuleApp", "Deploy RuleApp Archive", and "Update RuleApps".

The "Navigator" pane on the left has a tree view. A red box highlights the "RuleApps" node, which has two children: "/miniloanruleapp/1.0" (with two sub-items: "/miniloanrules/1.0" and "/miniloanrules/1.1"). Other nodes in the tree are "Resources (4)", "Libraries", and "Service Information".

The "RuleApps" section displays the total number of RuleApps: "Total Number of RuleApps 1". It shows a summary: "1 RuleApp(s)" and a "Name:" input field. A table lists the single RuleApp:

| Select All               | Name            | Version | Creation Date                     |
|--------------------------|-----------------|---------|-----------------------------------|
| <input type="checkbox"/> | miniloanruleapp | 1.0     | Nov 13, 2014 3:45:36 PM GMT-08:00 |

There are now two versions of the ruleset in the RuleApp, including the original version that was used when you first ran the Miniloan application, and the one you deployed now.

4. Click **/miniloanrules/1.1** to see the details.

**Ruleset View**

[Test Ruleset](#) [View Statistics](#) [View Execution Units](#) [Upload Ruleset Archive](#) [Add Managed URI](#)

**/miniloanruleapp/1.0/miniloanrules/1.1**

|                      |                                    |
|----------------------|------------------------------------|
| <b>Name</b>          | miniloanrules                      |
| <b>Version</b>       | 1.1                                |
| <b>Creation Date</b> | Dec 18, 2014 11:33:59 AM GMT-08:00 |
| <b>Display Name</b>  |                                    |
| <b>Description</b>   |                                    |
| <b>Rule engine</b>   | Classic Rule Engine                |
| <b>Status</b>        | enabled                            |
| <b>Debug</b>         | disabled                           |

[Permanent link](#)

---

**Ruleset Parameters**

| Direction | Name     | Kind   |
|-----------|----------|--------|
|           | borrower | native |
|           | loan     | native |

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

---

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

---

[Show Properties \(4\)](#)

---

[Show Monitoring Options \(tracing currently disabled\)](#)

---

[Show HTDS Options](#)

---

[Show Archive Content](#)

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.

5. Scroll down and click **Show Properties** and check the value of the `ilog.rules.teamserver.baseline` ruleset property.

The screenshot shows the 'Properties' view in the IBM ODM Rule Execution Server. At the top, there is a 'Hide Properties' link and a '4 properties' summary. Below is a table with columns 'Name' and 'Value'. The 'ilog.rules.teamserver.baseline' row has its value 'V1.0' highlighted with a red box. Other rows show 'ilog.rules.teamserver.permalink.project' with value 'http://localhost:9080/teamserver/faces/home.jsp?project=mi...rules&baseline=V1.0&locale=en\_US&datasource=jdbc%2Filo...', 'ilog.rules.teamserver.permalink.report' with value 'http://localhost:9080/teamserver/faces/servlet/ReportingServ...rules&baseline=V1.0&locale=en\_US&datasource=jdbc%2Filo...', and 'ruleset.bom.enabled' with value 'true'. Below the table, it says 'properties 1 - 4 of 4'. At the bottom, there is an 'Upload properties from file' section with a 'Choose file:' input field, a 'Browse...' button, and 'Proceed to update', 'Preview update', and 'Override' checkboxes.

| Select All               | Name                                    | Value                                                                                                                    |
|--------------------------|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | ilog.rules.teamserver.baseline          | V1.0                                                                                                                     |
| <input type="checkbox"/> | ilog.rules.teamserver.permalink.project | http://localhost:9080/teamserver/faces/home.jsp?project=mi...rules&baseline=V1.0&locale=en_US&datasource=jdbc%2Filo...   |
| <input type="checkbox"/> | ilog.rules.teamserver.permalink.report  | http://localhost:9080/teamserver/faces/servlet/ReportingServ...rules&baseline=V1.0&locale=en_US&datasource=jdbc%2Filo... |
| <input type="checkbox"/> | ruleset.bom.enabled                     | true                                                                                                                     |

properties 1 - 4 of 4

Upload properties from file

Choose file:  Browse... Proceed to update Preview update  Override

This property corresponds to the baseline information that you defined when you deployed the RuleApp from Decision Center.

6. In the Navigator pane, expand **Resources**, click **miniloan-xom.zip**, and click **Show references from Rulesets**.

The screenshot shows the 'Resource View' pane in the IBM ODM Rule Execution Server. On the left is a 'Navigator' pane with sections for 'RuleApps', 'Resources', 'Libraries', and 'Service Information'. In the 'Resources' section, 'miniloan-xom.zip/1.0' is selected. The main area shows details for 'miniloan-xom.zip/1.0' with fields: Name (miniloan-xom.zip), Version (1.0), Checksum (eb8e48a30e47ac49ad8aaaa45af9be7748a25322), Creation Date (Nov 13, 2014 3:45:38 PM GMT-08:00), and URI (resuri://miniloan-xom.zip/1.0). Below this, under '2 Reference(s) from Rulesets', there is a table with columns 'Name', 'Version', and 'Ruleset Path'. It lists two entries: 'miniloanrules' (version 1.0, path /miniloanrule...) and 'miniloanrules' (version 1.1, path /miniloanrule...). A note at the bottom says 'References from Rulesets 1 - 2 of 2'.

| Name          | Version | Ruleset Path     |
|---------------|---------|------------------|
| miniloanrules | 1.0     | /miniloanrule... |
| miniloanrules | 1.1     | /miniloanrule... |

References from Rulesets 1 - 2 of 2

Notice that the `miniloanrules` ruleset that you just deployed is listed as using this `miniloan-xom` resource. You learn more about these resources later.

7. Close the Rule Execution Server console.

## Section 6. Viewing the effects in the Miniloan application

### Scenario

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

### To run the Miniloan application with the updated ruleset:

- 1. Reopen the Miniloan application in a web browser window.

`http://localhost:9080/miniloan-center`

- 2. Click **Validate Loan** with the default values.

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

| Borrower Information            |       | Loan Information       |        |
|---------------------------------|-------|------------------------|--------|
| Name:                           | Joe   | Amount:                | 500000 |
| Yearly Income:                  | 80000 | Duration (months):     | 240    |
| Credit Score:                   | 600   | Yearly Interest Rate : | 0.05   |
| <b>Validate Loan</b>            |       |                        |        |
| The loan is approved.           |       |                        |        |
| The yearly repayment is 39,597. |       |                        |        |
| <b>Messages:</b>                |       |                        |        |

### Rule set Information

Version `Latest`

### Rules Execution Summary [0 rule(s) executed.]

|                   |                     |
|-------------------|---------------------|
| Ruleset version   | <code>Latest</code> |
| Rule(s) executed. |                     |

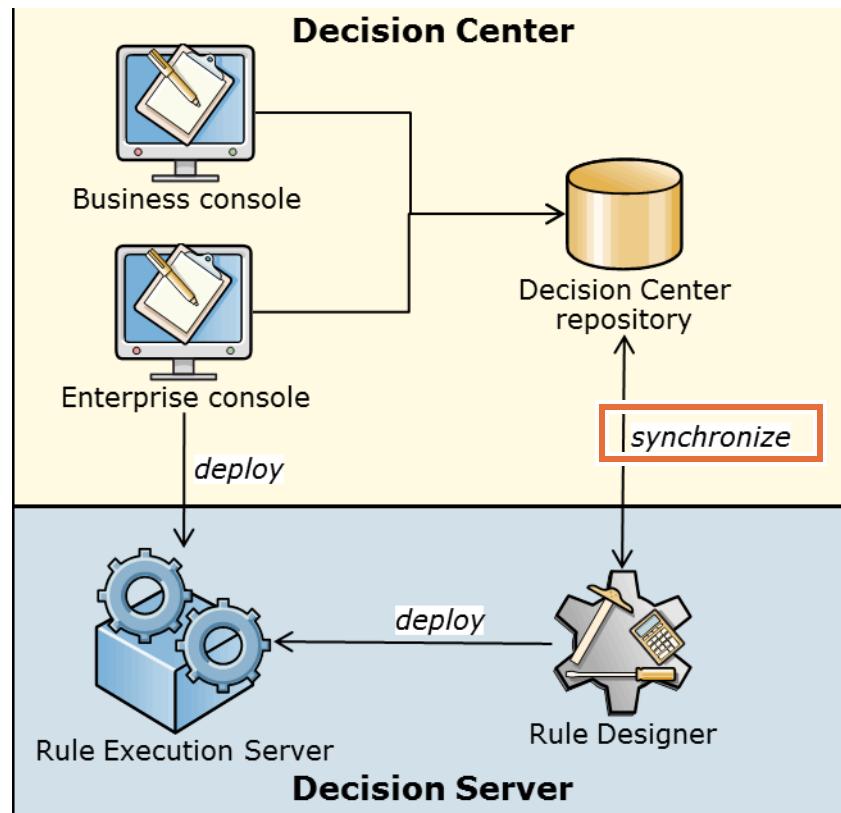
- 3. When you are finished, close the browser.

## Section 7. Synchronizing environments between Rule Designer and Decision Center

### Scenario

The IT team is preparing to upgrade the Miniloan application infrastructure, and they want to run some tests with the business rules. Before starting, they connect to Decision Center from Rule Designer. By using the synchronization tools in Rule Designer, they can verify that the rules that are stored in their file system match the latest versions that are stored in the Decision Center repository.

For this step, you take on the role of developer. You use Rule Designer to synchronize the rules that are stored in the development environment with the updates stored in the Decision Center repository.



## 7.1. Opening Rule Designer

- 1. Click Start > All Programs > IBM > Operational Decision Manager V8.7 > Rule Designer (or you can use the shortcut on the desktop).



Rule Designer

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

C:\labfiles\workspaces\intro



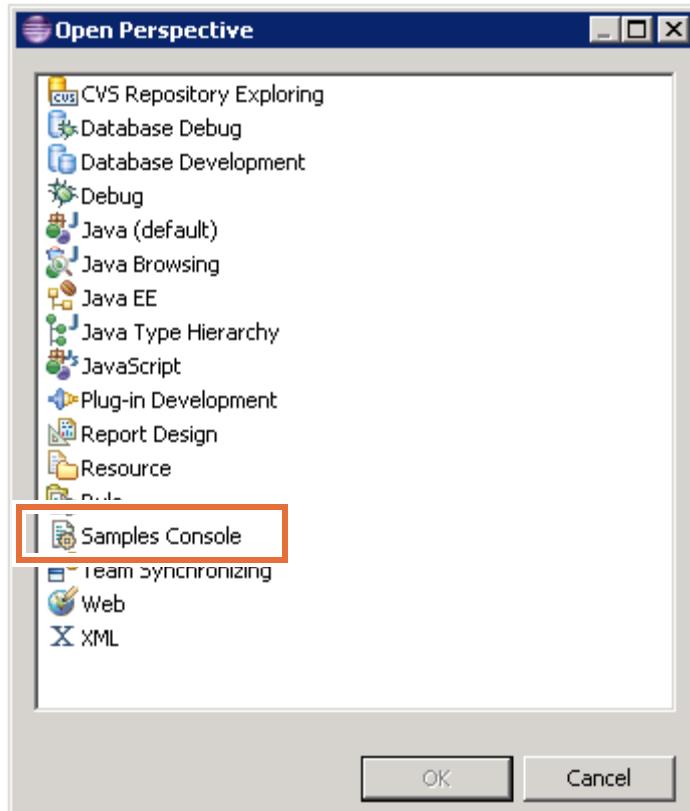
### Troubleshooting

The default perspective for Rule Designer is the Java perspective. You must use the **Open Perspective** icon to switch to a different view, such as the Rule perspective or the Samples Console.

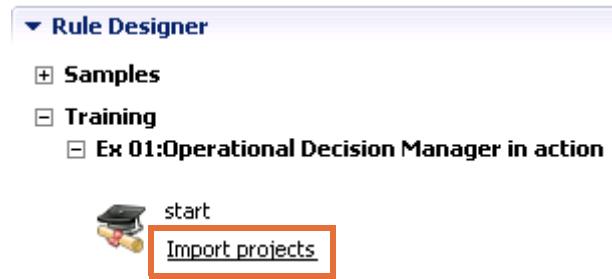
- 3. Close the Welcome view.
- 4. Switch to the Samples Console and import the start project for this exercise.
  - a. In the upper-right corner, click the **Open Perspective** icon to switch from the Java perspective.



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

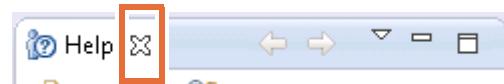


- \_\_\_ c. In the Rule Designer section, expand **Training > Ex 01: Operational Decision Manager in action**, and under **start**, click **Import projects**.



The Eclipse perspective automatically switches to the Rule perspective.

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

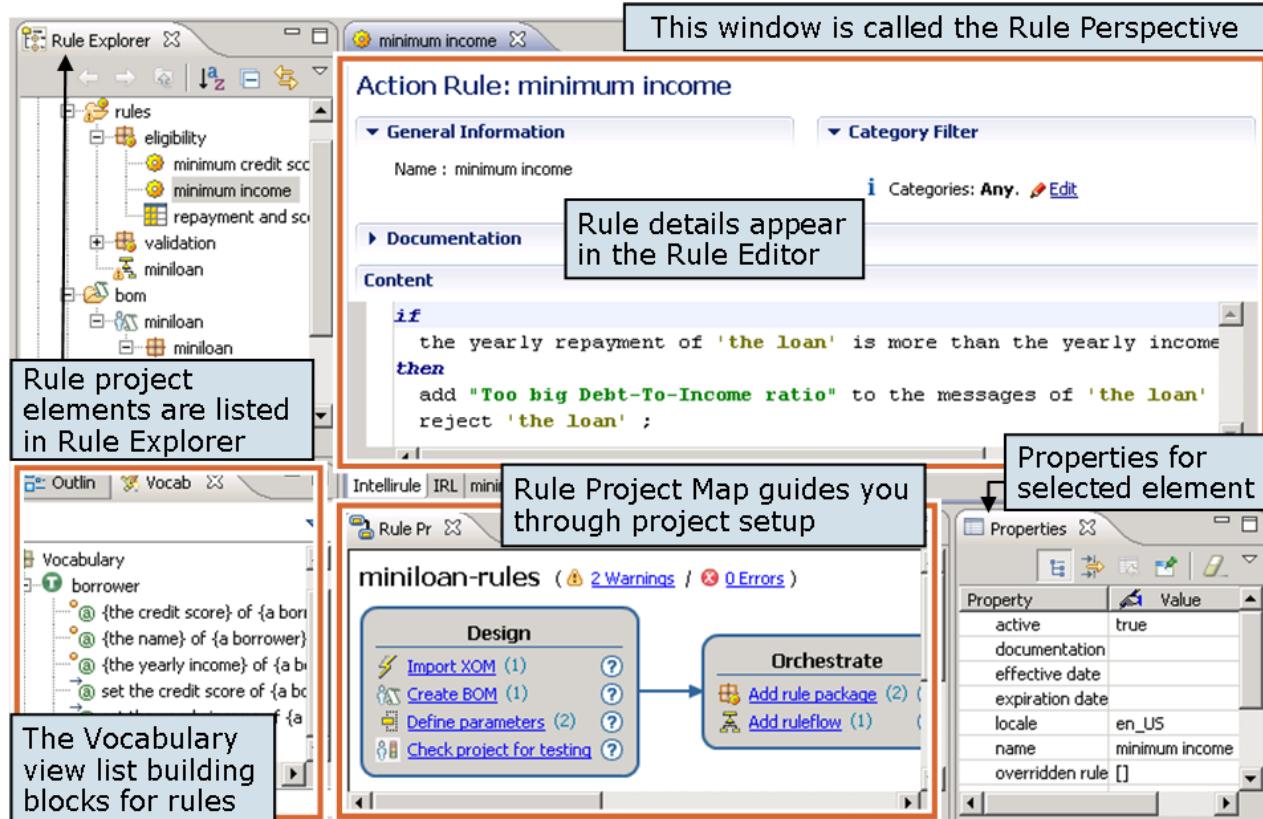


To give yourself more working area, you can close the panes (such as Help or Welcome panes) that open by default in Eclipse.

## 7.2. Taking a quick tour of Rule Designer

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

As you work through the steps in this section, you open and work in the various windows and views that you see in this graphic.



1. Look at the various windows and panes that are open in the Rule perspective.

Most of the views are empty, except Rule Explorer.



The Rule Explorer contains these projects:

- **miniloan-rules**: The 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-rules** to see the **rules** and **bom** folders.
3. Expand the **rules** folder to see the **eligibility** and **validation** folders.
- a. Expand the **eligibility** and **validation** folders to see the rules that they contain.

- \_\_\_ b. Notice that these folders are the same folders that you saw in Decision Center.
- \_\_\_ 4. 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.
- \_\_\_ 5. 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.
- \_\_\_ 6. In Rule Explorer, go back to the **rules** folder to see how the rules use the vocabulary.
  - \_\_\_ a. Expand 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?

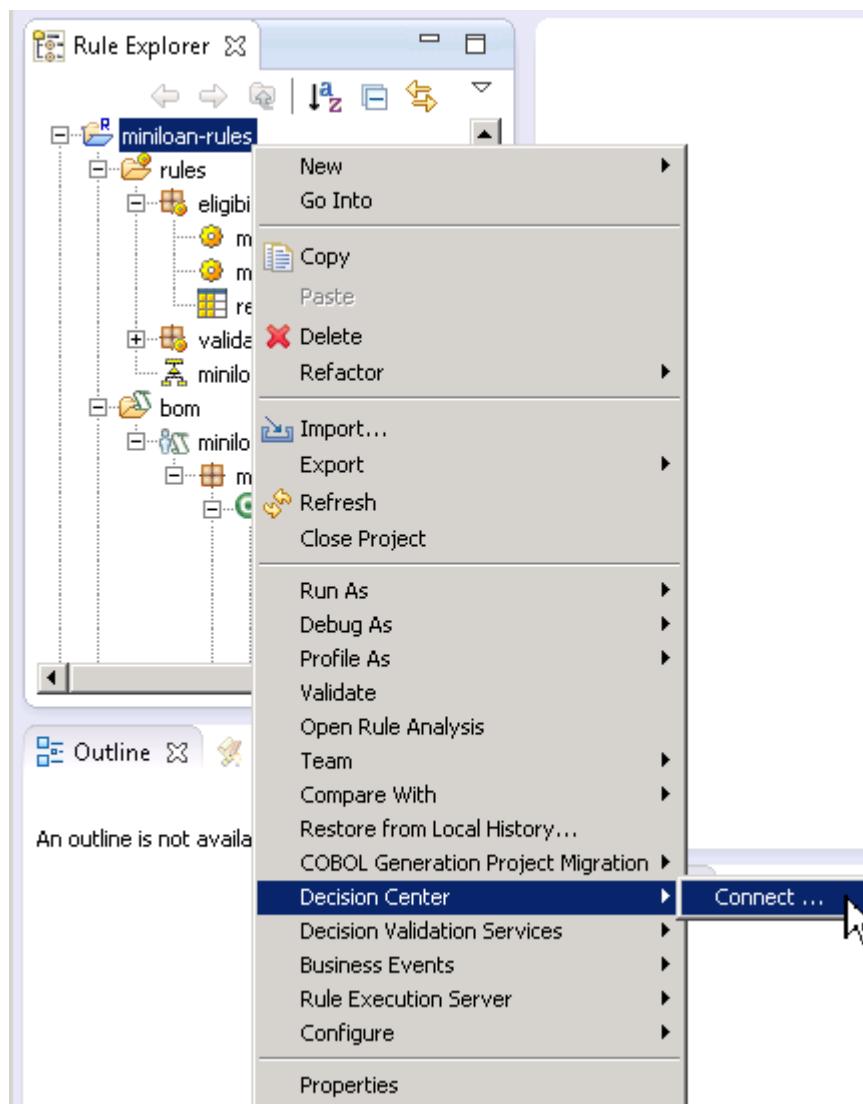
*Remember this comparison as a first look at the connection between rules and vocabulary.*

---

- \_\_\_ b. Notice the rule statement. This rule does not include your edits from Decision Center.  
To get the latest version of the rules from Decision Center, you now use the Rule Designer synchronization tools.

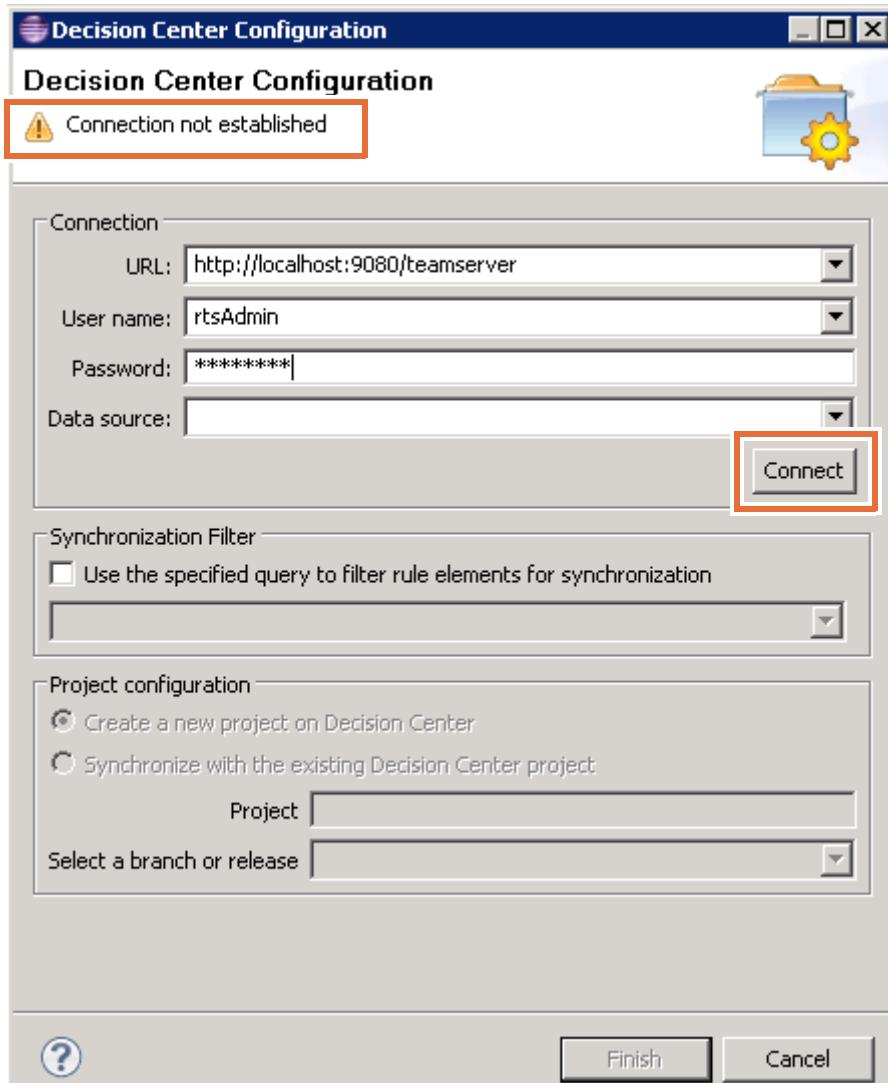
### 7.3. Synchronizing Rule Designer and Decision Center environments

1. In Rule Explorer, right-click the **miniloan-rules** rule project, and click **Decision Center > Connect**.



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

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

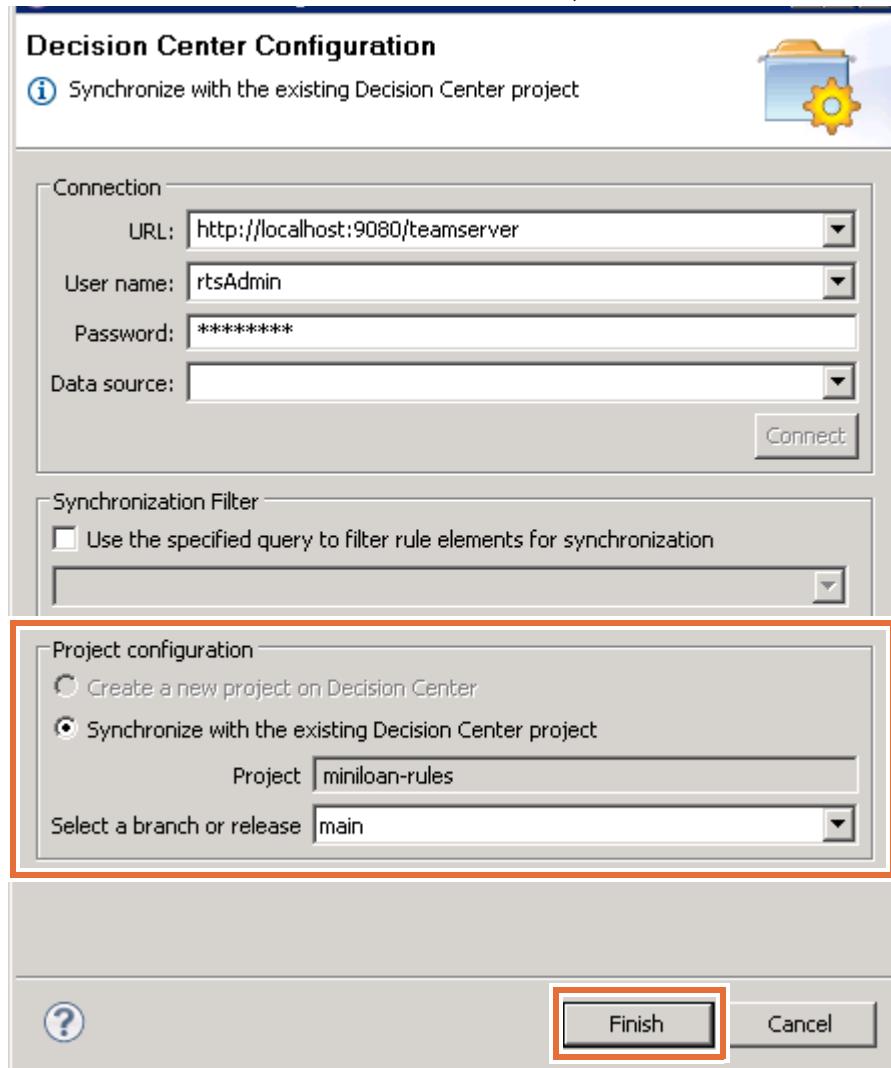


The warning message, Connection not established, is shown until you establish the connection.

3. Click **Connect**.

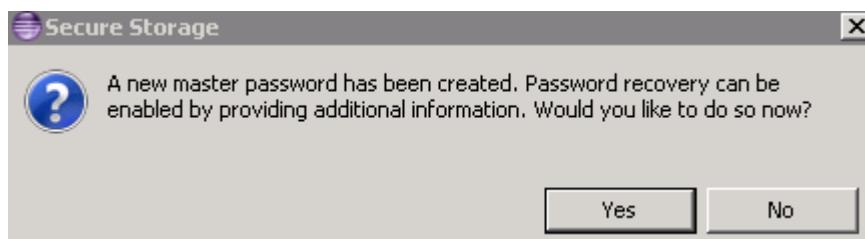
When the connection is established, the warning message disappears and the Project configuration area becomes active.

4. Select **main** from the menu next to **miniloan-rules**, and click **Finish**.

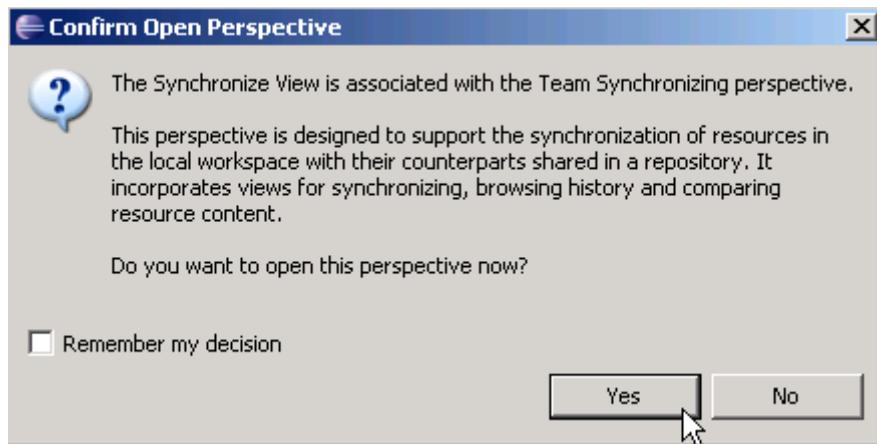


### Troubleshooting

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

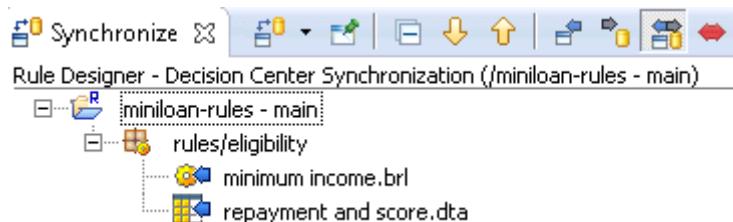


- \_\_\_ 5. When the window prompts you to change to Team Synchronizing perspective, click **Yes**.



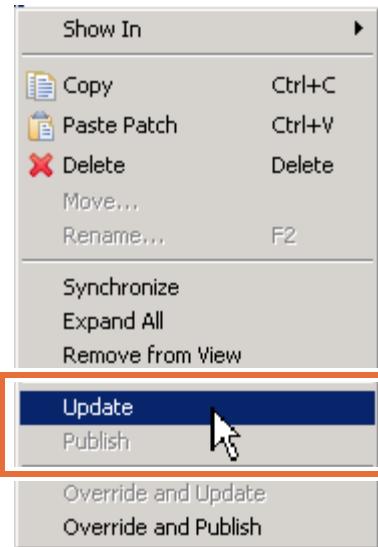
After synchronization is complete, the Synchronize view shows the `miniloan-rules` rule project, and indicates that changes were made to rules in that rule project.

- \_\_\_ 6. In the Synchronize view, expand the `miniloan-rules - main` rule project to see which resources and rules were changed.



- \_\_\_ a. In particular, expand the **rules/eligibility** folder to see the rules that you updated in Decision Center.
  - `minimum income.brl`
  - `repayment and score.dta`
- \_\_\_ b. Double-click either of these rules to open it in the Text Compare view and see the differences that now require synchronization.
- \_\_\_ c. Close the Text Compare view for the rule that you selected.

- \_\_\_ d. Right-click **miniloan-rules - main** and click **Update**.



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



### Important

During synchronization, not all artifacts require synchronization. Some artifacts are generated, such as artifacts in the **output** folder, and should not be synchronized and stored.

- \_\_\_ 7. Return to the Rule perspective.



### Hint

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



- \_\_\_ 8. If the minimum income rule is closed, reopen it by expanding the **miniloan-rules > rules > eligibility** folder in Rule Explorer, and double-clicking **minimum income**.

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

if

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

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

- \_\_\_ 9. Close Rule Designer, and confirm closing when prompted to confirm.

You successfully synchronized the technical and business environments.

## Section 8. Shutting down the modules

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

- \_\_\_ 1. Close all browsers for the Miniloan application, Rule Execution Server, and Decision Center consoles.
- \_\_\_ 2. Close Rule Designer and click **OK** when prompted to confirm your exit from Eclipse.
- \_\_\_ 3. Stop the sample server.
  - \_\_\_ a. Click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Stop server.**
  - \_\_\_ b. After the sample server is stopped, close the terminal window.

### End of exercise

## Exercise review and wrap-up

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

# Exercise 2. Setting up rule projects

## What this exercise is about

This exercise shows you how to set up rule applications in Rule Designer.

## What you should be able to do

After completing this exercise, you should be able to:

- Create a rule project
- Set up the rule project to reference the execution object model (XOM)
- Generate a business object model (BOM) and a default vocabulary
- Create ruleset parameters
- Create rule packages
- Synchronize rule projects with Decision Center

## Introduction

In this exercise, you work through these tasks to start developing a business rule application in Rule Designer. You set up the rule project structure, define the business and execution object models, and define the ruleset parameters, which are used to pass objects between the external application and the rule engine.

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

- Section 1, "Creating the rule project"
- Section 2, "Setting up the BOM"
- Section 3, "Creating the ruleset parameters"
- Section 4, "Creating a ruleset variable"
- Section 5, "Creating rule packages"
- Section 6, "Publishing from Rule Designer to Decision Center"
- Section 7, "Deleting a project from Decision Center"
- Section 8, "Creating a rule project from Decision Center"

## Requirements

There are no specific requirements for this exercise.

## Section 1. Creating the rule project

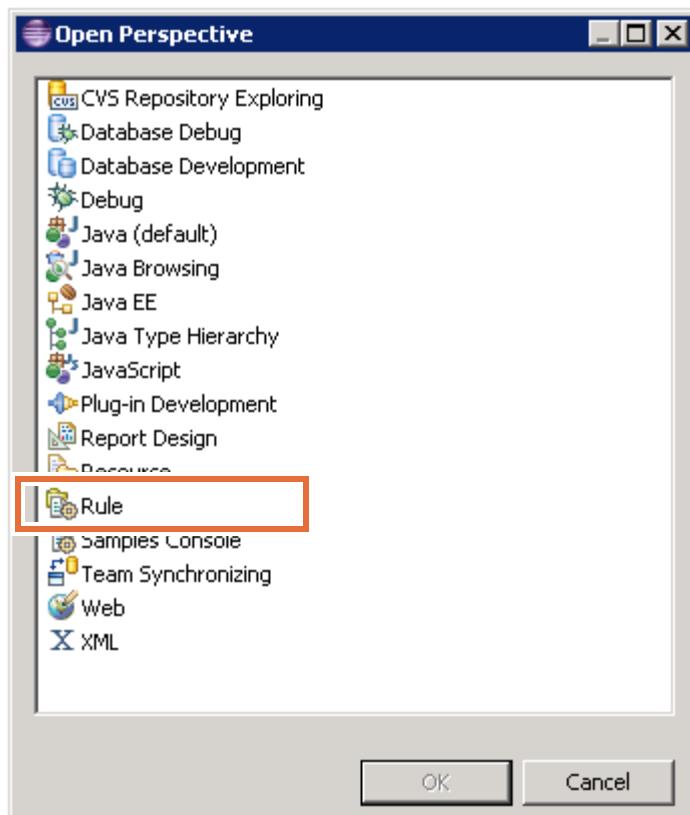
In this exercise, you create the initial elements that are required to set up a business rule application, including a rule project, the associated XOM and BOM, and the vocabulary.

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

- 1. Open Rule Designer by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Rule Designer**.
- 2. In the **Workspace** field of the Workspace Launcher window, create a workspace.
  - a. Type a workspace name:  
C:\labfiles\workspaces\rule\_project
  - b. Make sure that the **Use this as the default and do not ask again** check box is not selected.
  - c. Click **OK**.
- The `<LabfilesDir>\workspaces\rule_project` workspace is created.
- 3. Close the Welcome view.
- 4. Switch to the Rule perspective.
  - a. Click the **Open Perspective** icon in the upper-right part of the Eclipse window.



- \_\_\_ b. Select **Rule**, and click **OK**.



Rule Designer opens the workspace in the Rule perspective.

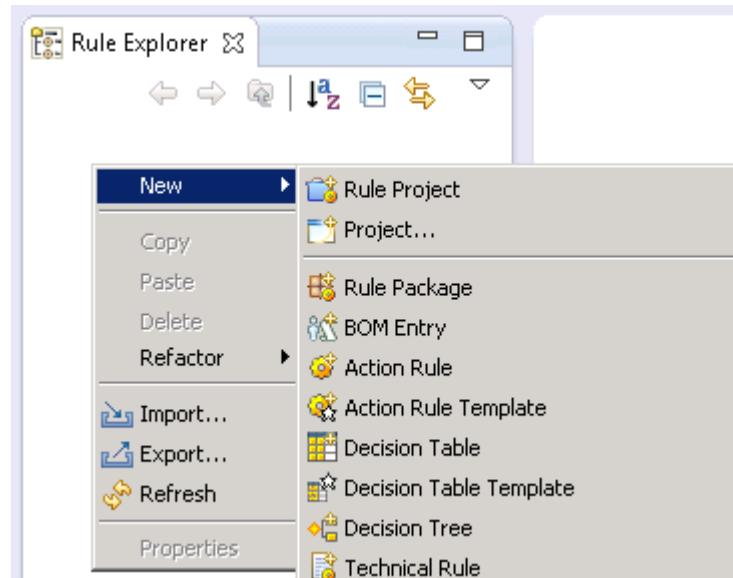
- \_\_\_ 5. If needed, close the Help view.

## 1.2. Creating the rule project

- \_\_\_ 1. From the **File** menu, click **New > Rule Project**.

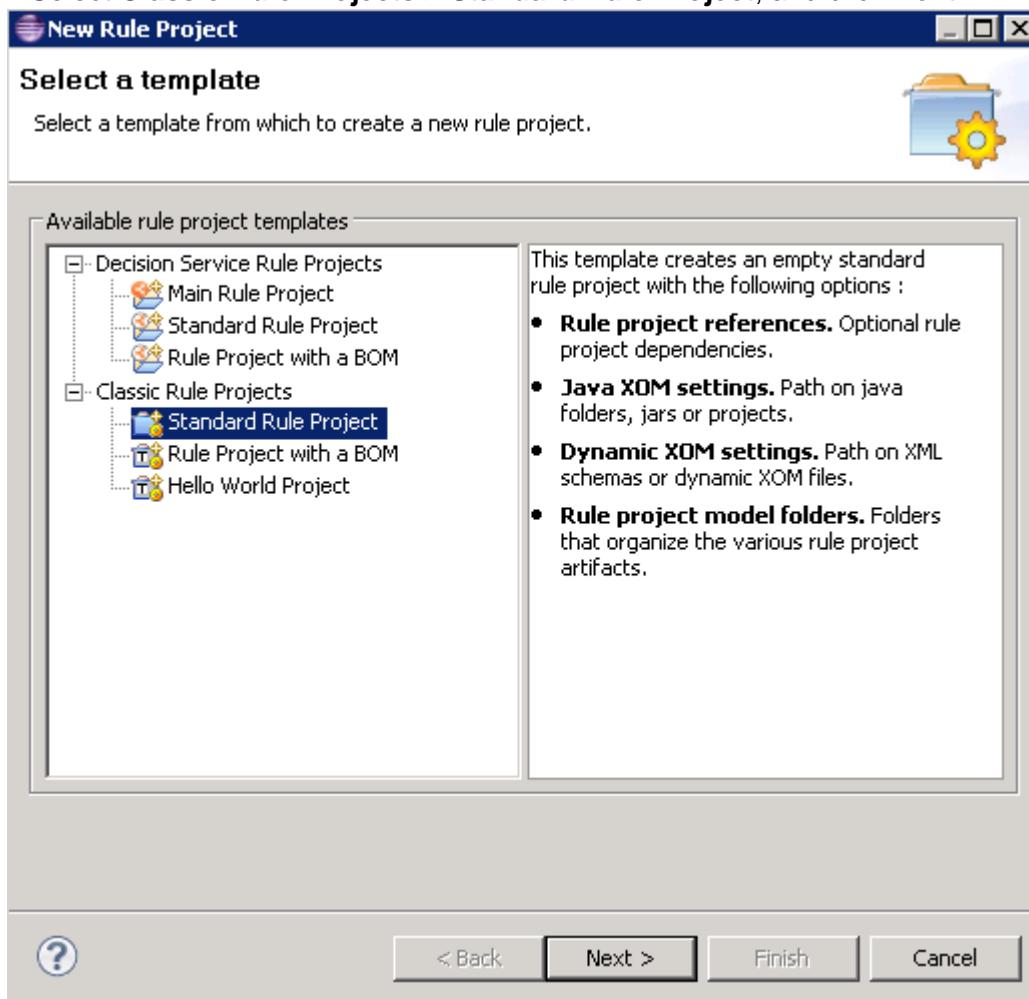
**Hint**

You can also right-click anywhere in the Rule Explorer to access the menu and click **New > Rule Project**.



2. In the New Rule Project window, create a standard rule project:

- a. Select **Classic Rule Projects > Standard Rule Project**, and click **Next**.



- b. In the **Project name** field, type: `myloan-rules`  
 c. Make sure that the **Use default location** check box is selected and click **Next**.  
 d. Click **Next** to skip the **Rule Project References** page because this exercise does not ask for any such references.  
 e. Click **Next** to skip the **Rule Project XOM Settings** page for Java Execution Object Model (XOM).

The **Rule Project XOM Settings** page for Dynamic Execution Object Model (XOM) opens.



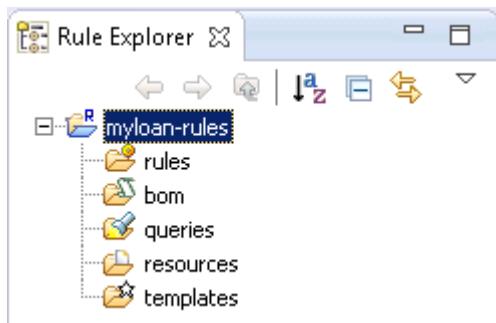
#### Note

At this step of the exercise, there are no XOM projects available in your workspace that your rule project can reference. You import such a XOM project later in this exercise.

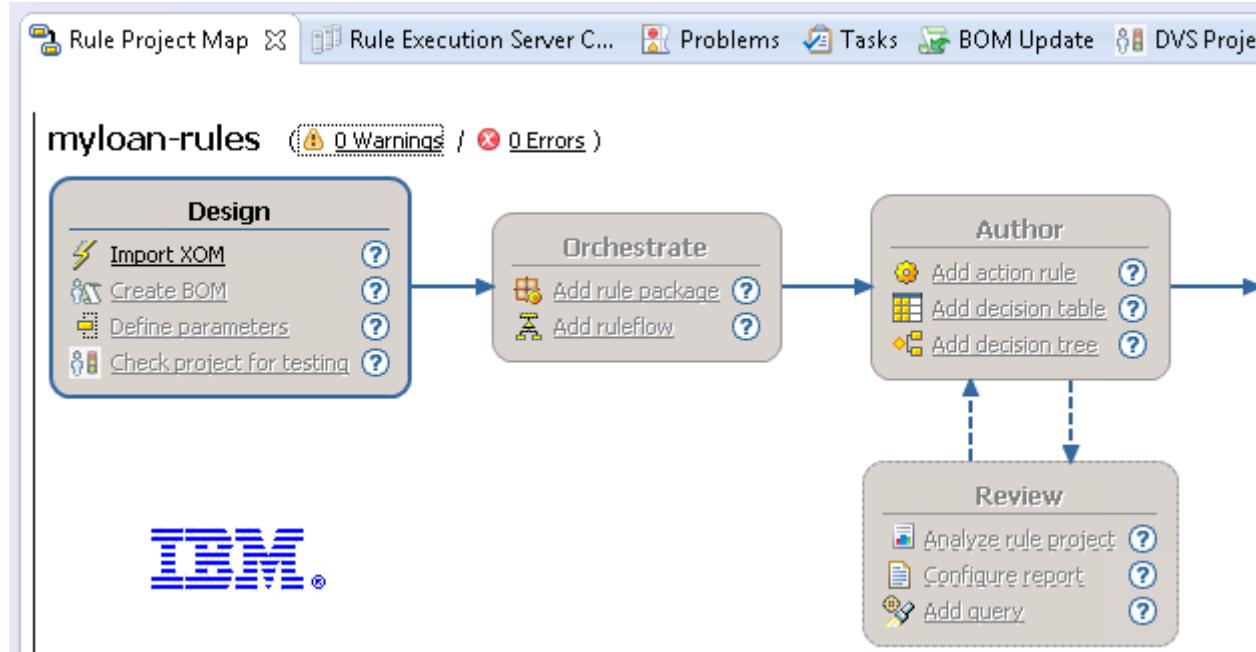
- \_\_\_ f. Click **Next** to skip the **Rule Project XOM Settings** page for Dynamic Execution Object Model (XOM).
- \_\_\_ g. In the Rule Project Folders page, notice the default rule project folders and their purpose:
- **BOM Folder**: To store the project business object model (BOM)
  - **Query Folder**: To store any queries that you create to manage rules
  - **Resource Folder**: To store all your project resource files
  - **Source Folder**: To store all your project rules
  - **Template Folder**: To store any templates that you set up to help create multiple rules with the same structure

- \_\_\_ h. Click **Finish**.

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



- \_\_\_ 3. Make sure the `myloan-rules` project is selected and notice the Rule Project Map in the lower part of the perspective.



**Hint**

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

### 1.3. Importing the execution object model (XOM)

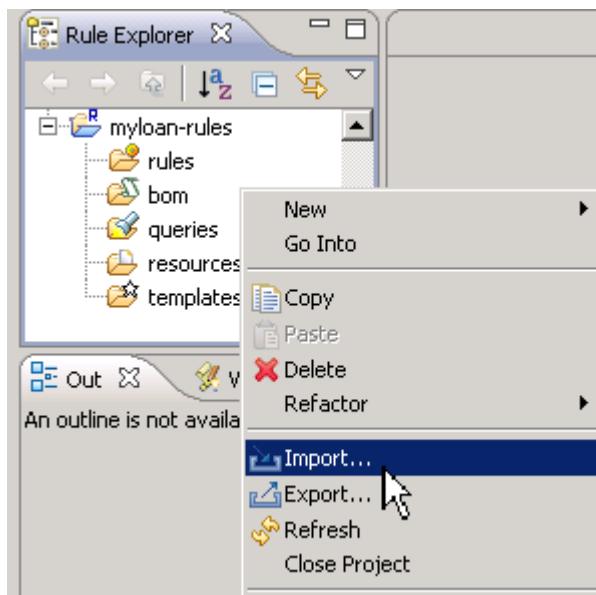
In the **Design** part of the Rule Project Map, notice that only the **Import XOM** link is enabled. This link indicates that the first task in designing your ruleset is to import a XOM into your project. By importing a XOM, you create a reference between your rule project and a XOM project in your workspace. To be able to create such a reference, you must first have the XOM in your Rule Designer workspace.

**Information**

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

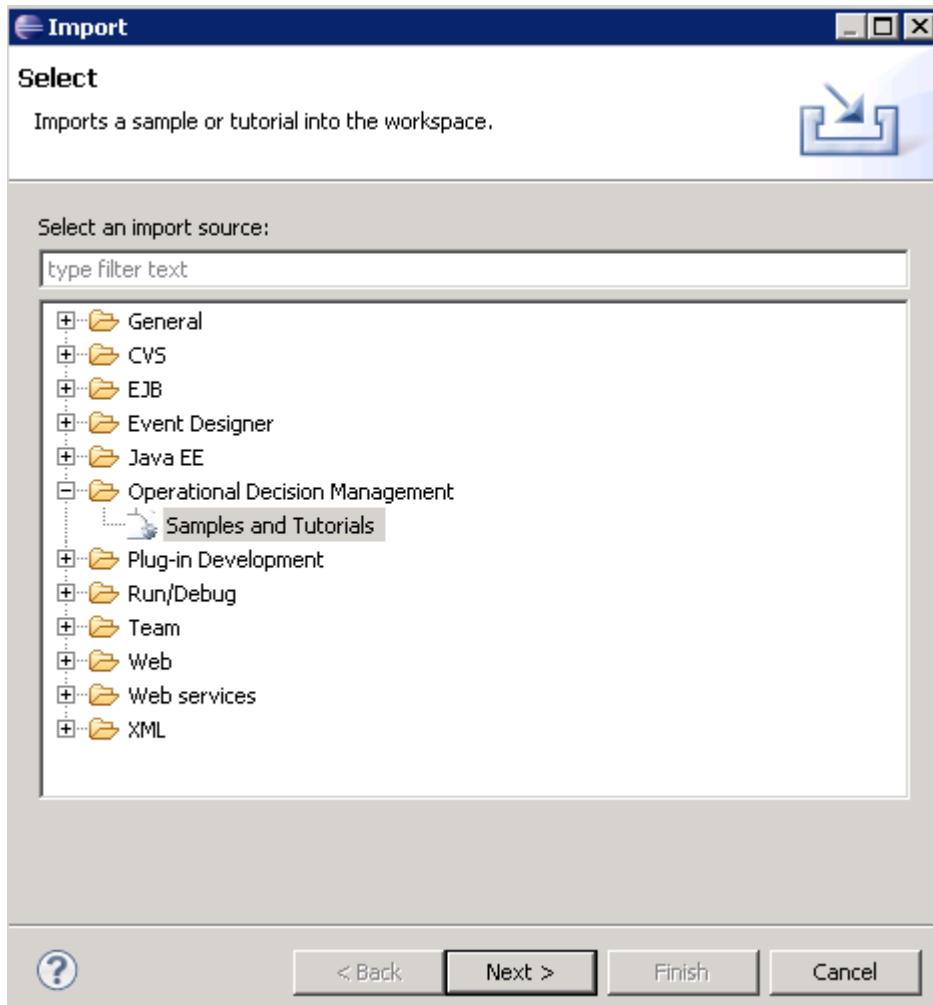
For this exercise, the required XOM is the `loan-xom` Java project and is provided for you. You import this XOM by importing the project **Ex 02: Setting up rule projects > 01-start** by following the next procedure.

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



The Select page of the Import wizard opens.

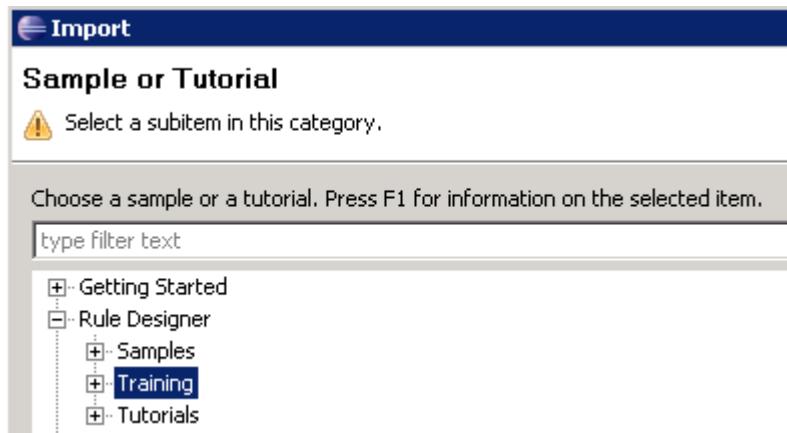
- 2. In the Select page, select **Operational Decision Manager > Samples and Tutorials**, and click **Next**.



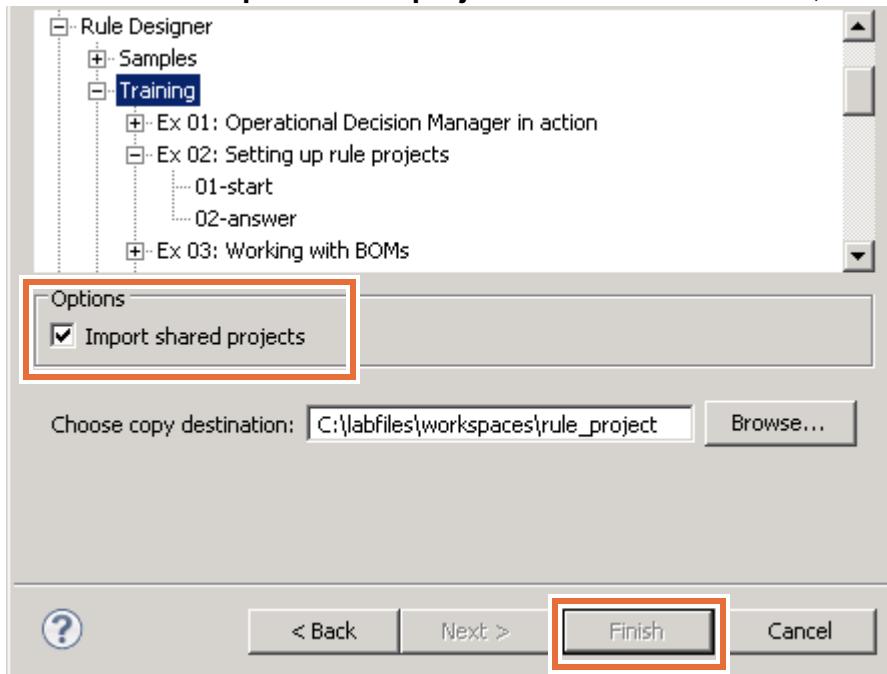
- 3. In the Sample or Tutorial page, expand the **Rule Designer > Training** node.

**Hint**

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



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



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

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

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

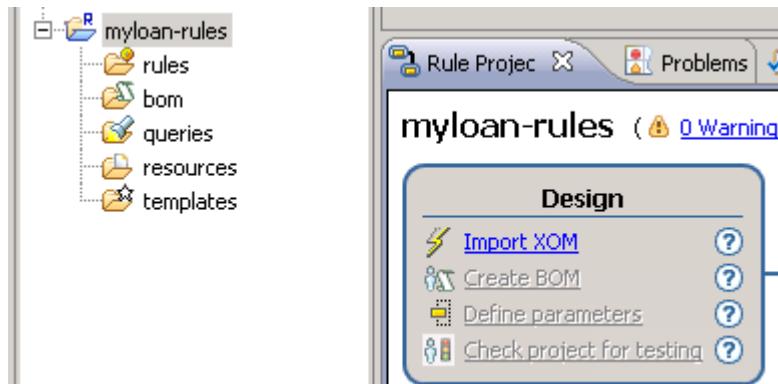
**Important**

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

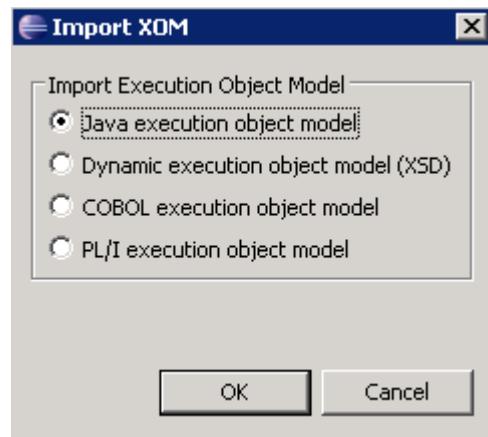
## 1.4. Creating a reference to the XOM in the rule project

Now that you have a XOM in your workspace, use the Rule Project Map to import it into your rule project:

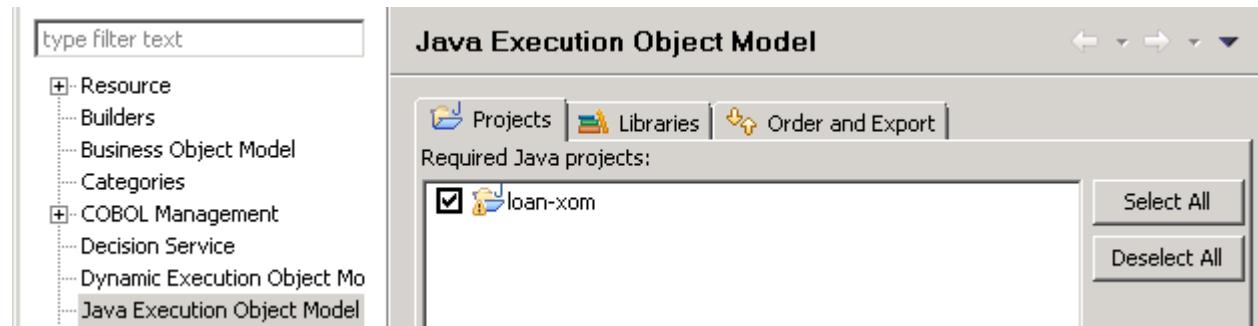
- 1. With the `myloan-rules` project selected in Rule Explorer, click the **Import XOM** link in the Rule Project Map to import a XOM for your rule project.



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



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



By selecting `loan-xom`, the `myloan-rules` project now references the `loan-xom` Java project.

You can expand the `loan-xom` project, which is now available in the Rule Explorer, to view the Java code implementation.

## Section 2. Setting up the BOM

In the **Design** part of the Rule Project Map, the **Create BOM** link is now enabled.



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

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

### 2.1. Creating the BOM

- 1. Click the **Create BOM** link in the Rule Project Map to create a BOM entry.

The **New BOM Entry** wizard opens.



#### Note

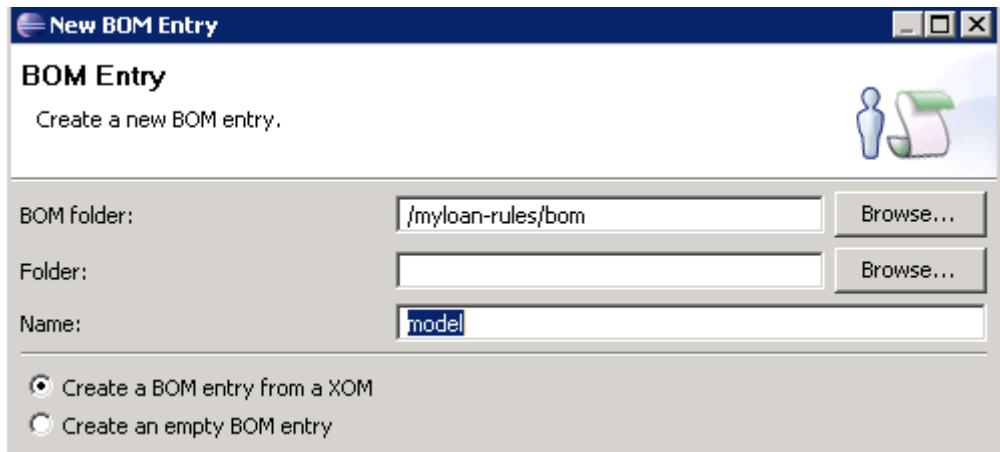
You can also right-click the `bom` folder in the `myloan-rules` project, and click **New > BOM Entry**.



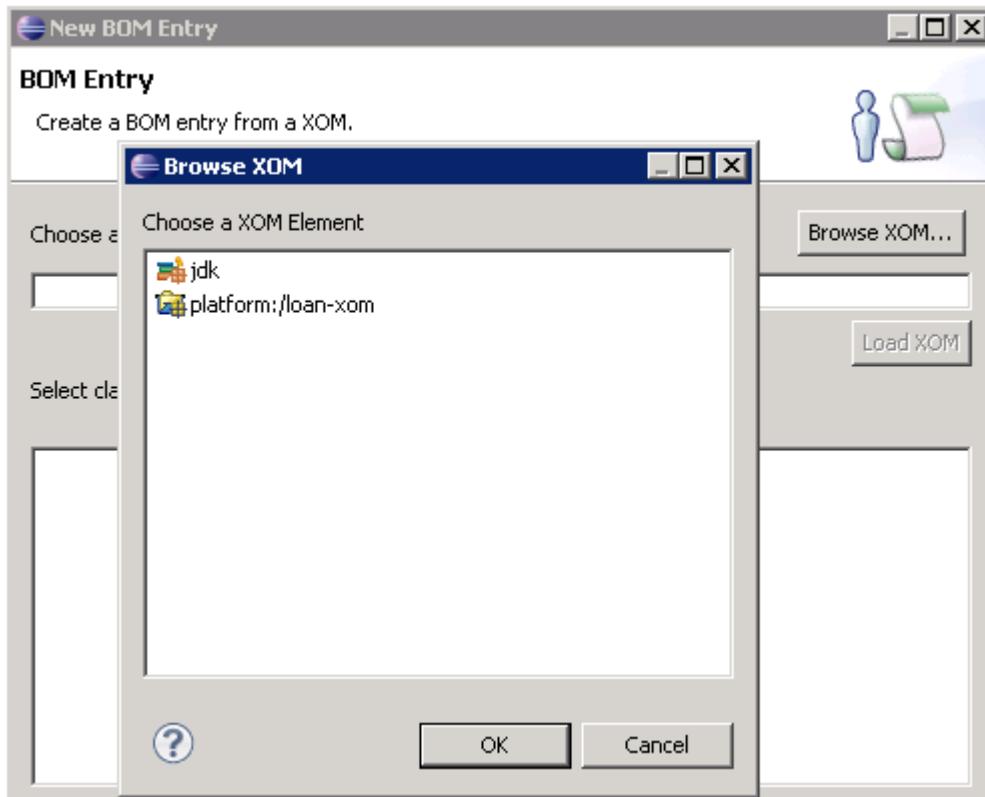
#### Information

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

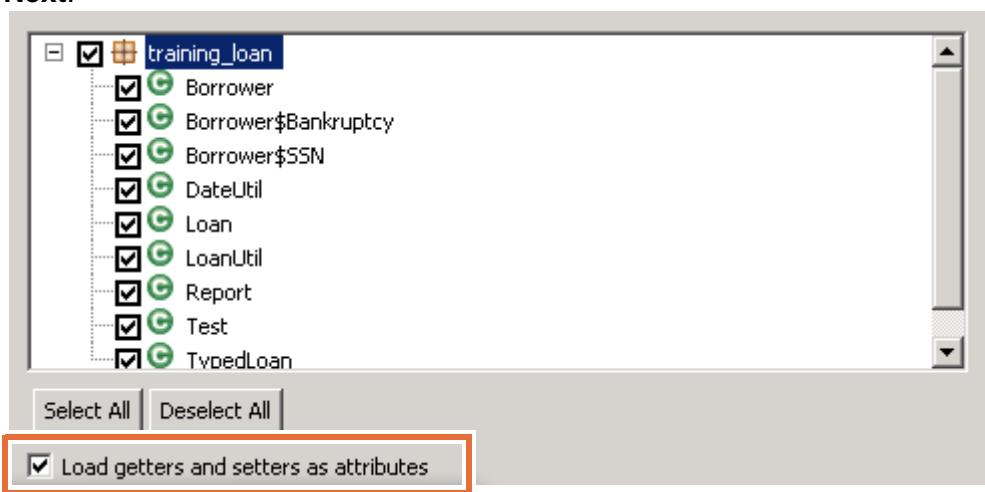
- 2. Make sure that the **Create a BOM entry from a XOM** option is selected, keep the default values for the other fields, and click **Next**.



- \_\_\_ 3. Click **Browse XOM**, select **platform:/loan-xom** in the Browse XOM window, and click **OK**.

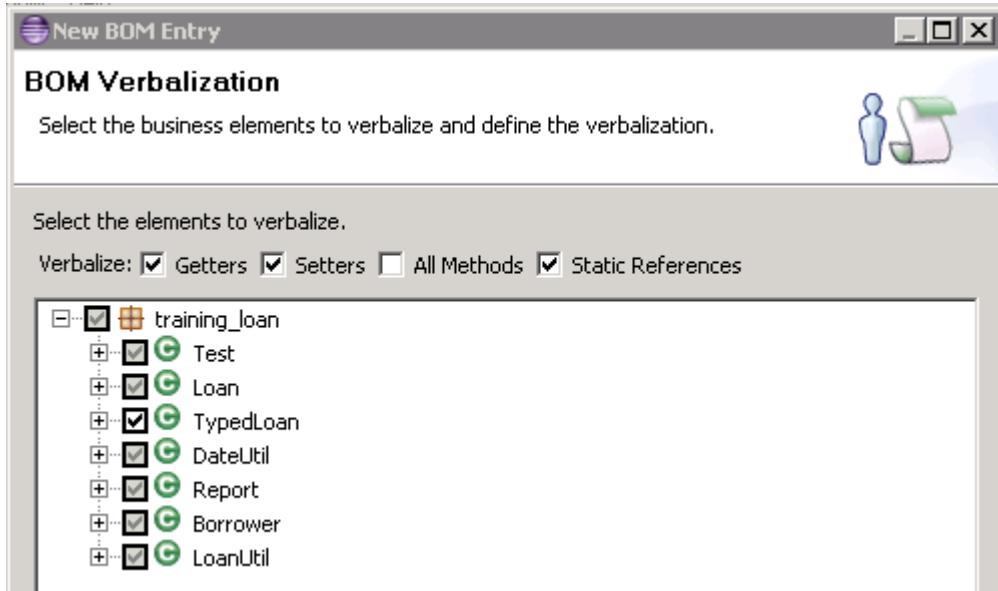


- \_\_\_ 4. Make sure that your BOM entry fully reflects the classes and methods in the XOM as follows:
- Expand **training\_loan** and select all classes by clicking **Select All**.
  - Make sure the **Load getters and setters as attributes** check box is selected and click **Next**.



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

The BOM Verbalization page opens.



- \_\_\_ 5. Click **Finish** to accept the default verbalization.

 **Information**

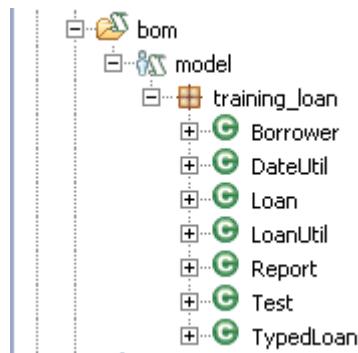
Verbalization attaches natural-language terms to the BOM elements to create the vocabulary for your business rules. You learn more about verbalization later in this course.

In your rule project, the BOM entry that is called `model` is now available with a default vocabulary.

## 2.2. Exploring the BOM

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

- \_\_\_ 1. In your `myloan-rules` project, expand **bom > model > training\_loan**.



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

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

**Hint**

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

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

**Class Verbalization**

[Remove](#) the verbalization.  [Edit](#) the documentation.  
 Generate automatic variable

Term:   [Edit term.](#)

[i](#) the borrower, a borrower, the borrowers....

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

**Members**  
Specify the members of this class.

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

[New...](#)  [Delete](#)  [Edit](#)

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

the first name of a borrower

**Member Verbalization**

[Remove](#) the verbalization.  
 [Create](#) a navigation phrase.  
 [Edit](#) the subject used in phrases.

**Navigation : "the first name of a borrower"**

Template:

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

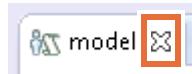
{first name} of {this}



### Note

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

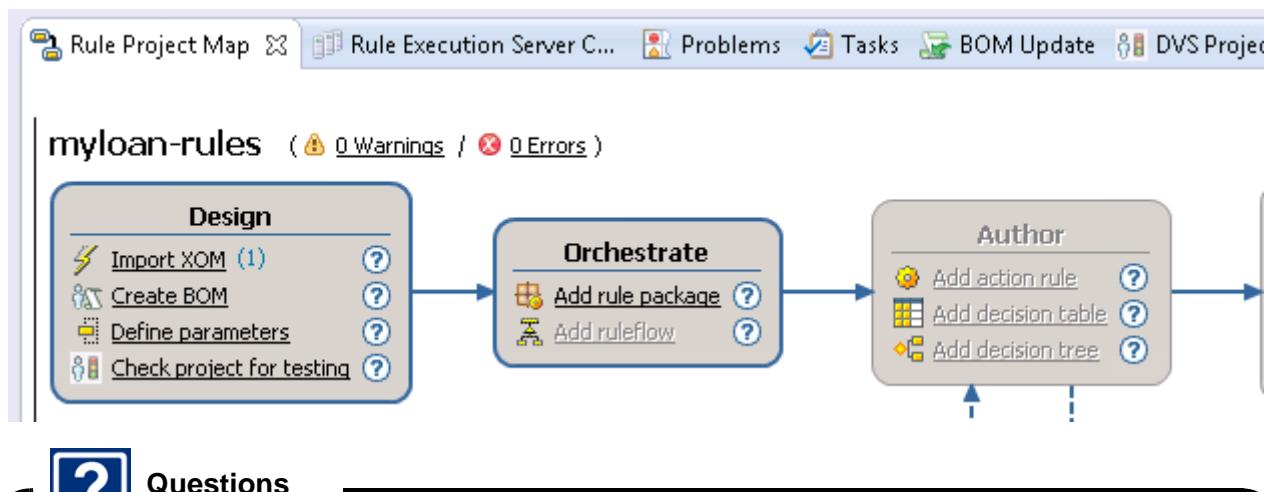
- \_\_\_ 7. Close the BOM editor.



## 2.3. Identifying the next tasks in rule project development

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

- \_\_\_ 1. Select the myloan-rules project.  
 \_\_\_ 2. In the Rule Project Map, identify the newly enabled links:
- In the **Design** part: **Define parameters** and **Check project for testing**
  - In the **Orchestrate** part: **Add rule package**



Which task should you do first?

**Answer**

The fact that these links are enabled means that your rule project already has enough information for you to continue with either the Design part or go to the Orchestrate part. For this exercise, you continue with Design.

Notice that the Author part is not enabled, so the project is not yet ready for rule authors.

## Section 3. Creating the ruleset parameters

In the **Design** part of the Rule Project Map, the **Define parameters** link is used to define the ruleset parameters, that is, the interface between your decision service and the calling application.

You can add parameters either by using the **Rule Project Map** link or by right-clicking your rule project and clicking **Properties** from the menu.



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

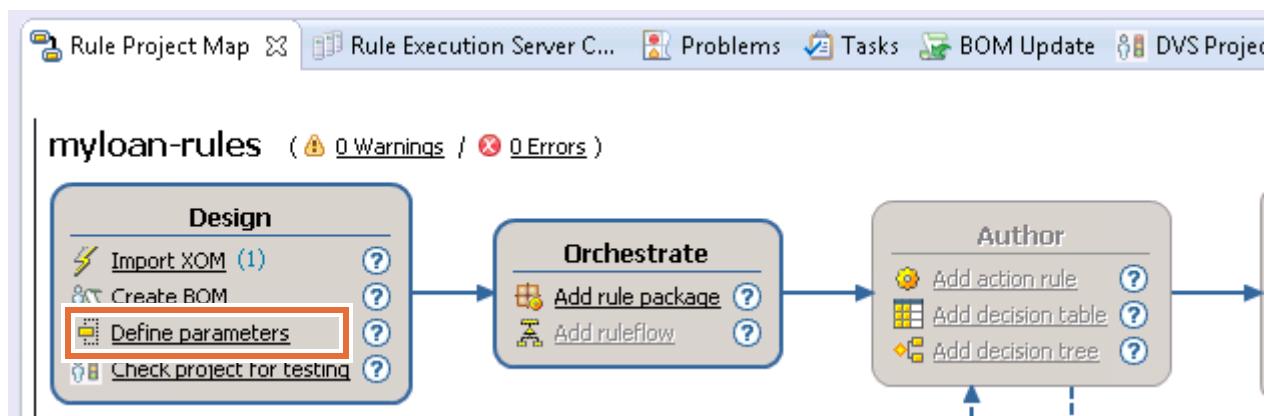
You are to implement the following ruleset parameters to pass the data between the calling application and the rule engine:

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

These parameters do not have default values.

To define the rule parameters as discussed with the business analysts.

- 1. In the Design part of the Rule Project Map, click the **Define parameters** link.



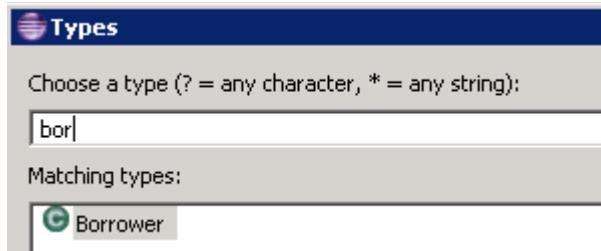
The Properties window for the rule project opens to the Ruleset Parameters page.

The Ruleset Parameters page presents all the defined ruleset parameters for the `loan-rules` project. It is now empty.

- 2. In the Ruleset Parameters page, add the `borrower` ruleset parameter details as described earlier in the requirements.
  - a. Click **Add**.
  - b. In the **Name** column, type over `myParam` to enter: `borrower`

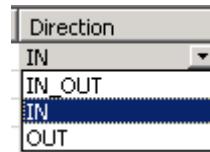
\_\_\_ c. Set the **Type** to: training\_loan.Borrower

- Click the **Type** field.
- Click the **Ellipsis** (...).
- In the Types window, select **Borrower**. (You can scroll through the list or start typing **Borrower** to filter the list. As you start typing, the field is automatically filtered to find types that match your spelled entry.)



- In the **Qualifiers** field, verify that the value is now equal to `training_loan` to ensure that the selected type is `training_loan.Borrower` as specified.
- Click **OK**.

\_\_\_ d. In the **Direction** column, select **IN**.



\_\_\_ e. Leave the **Default Value** column empty.

\_\_\_ f. In the **Verbalization** column, type over **myParam** to enter: the borrower

- 3. Click **Add** again and follow the instructions in Step 2 to add a `loan` ruleset parameter of direction `IN_OUT`, as described in the requirements.
- 4. Click **Add** again and follow the instructions in Step 2 to add a `report` ruleset parameter of direction `OUT`, as described in the requirements.

The completed parameters should now match the requirements.

| Ruleset Parameters         |                        |           |               |                 |  |
|----------------------------|------------------------|-----------|---------------|-----------------|--|
| Define ruleset parameters. |                        |           |               |                 |  |
| Name                       | Type                   | Direction | Default Va... | Verbalization   |  |
| borrower                   | training_loan.Borrower | IN        |               | the borrower    |  |
| loan                       | training_loan.Loan     | IN_OUT    |               | the loan        |  |
| report                     | training_loan.Report   | OUT       |               | the loan report |  |
|                            |                        |           |               |                 |  |
|                            |                        |           |               |                 |  |
|                            |                        |           |               |                 |  |
|                            |                        |           |               |                 |  |
|                            |                        |           |               |                 |  |

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

## Section 4. Creating a ruleset variable

Ruleset parameters carry the objects that are passed between the calling application and the rule engine. These objects are evaluated by the rules during rule execution. To avoid directly handling these objects, you can define ruleset variables to manipulate the objects during rule execution. When rule execution is complete, the final value of the ruleset variable can be assigned back to an output ruleset parameter and returned to the external application.



### Requirements

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

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

To meet these requirements, create a ruleset variable with these characteristics:

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

1. Before you can create a ruleset variable, create a *variable set* in your rule project.

a. Right-click the `myloan-rules` project, and click **New > Variable Set**.

The New Variable Set wizard opens.

Because you selected the rule project, the **Source folder** is already correctly set.

b. In the **Name** field, type: `local var`

c. Click **Finish**.

The Variable Set editor opens and is empty.



### Hint

If the Variable Set editor does not open automatically, double-click the `local var` variable set, or alternatively, right-click the `local var` variable set and click **Open With > Variable Set Editor**.

2. In the `local var` variable set, add the `loanApproved` ruleset variable.

a. In the Variable Set editor, click **Add**.

b. In the **Name** column, type over `myVar` to enter: `loanApproved`

- \_\_\_ c. Click the **Type** field to set the value:
- Click the Ellipsis (...).
  - In the Types window, select **boolean** (lowercase).
  - Click **OK**.
- \_\_\_ d. In the **Verbalization** column, type over **myVar** to enter: the loan is approved
- \_\_\_ e. In the **Initial Value** column, type: true
- \_\_\_ 3. Save your work by pressing Ctrl+S.
- \_\_\_ 4. Click the Problems view to make sure that there are no errors.
- \_\_\_ 5. Close the “local var” window.

| Name         | Type    | Verbalization        | Initial Value |  |
|--------------|---------|----------------------|---------------|--|
| loanApproved | boolean | the loan is approved | true          |  |
|              |         |                      |               |  |
|              |         |                      |               |  |
|              |         |                      |               |  |

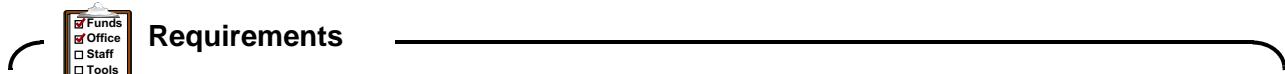
Add  
Remove  
Refactor

## Section 5. Creating rule packages

Recall the Rule Project Map with the **Add rule package** link that is enabled in the Orchestrate part.



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



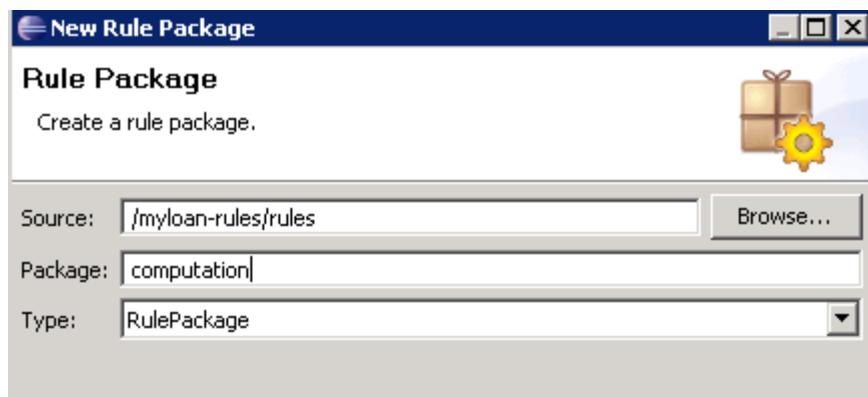
The business analysts provide you with an initial outline of how the rules can be organized and the naming conventions to use. Based on these requirements, the rule artifacts should be logically managed within the following packages and subpackages:

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

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

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

- \_\_\_ 1. Click the **myloan-rules** project to make sure that the Rule Project Map is open.
- \_\_\_ 2. Create the **computation** package.
  - \_\_\_ a. In the Orchestrate part of the Rule Project Map, click the **Add rule package** link.  
The New Rule Package window opens.
  - \_\_\_ b. In the **Package** field of the Rule Package window, enter `computation` and click **Finish** (or press Enter).



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

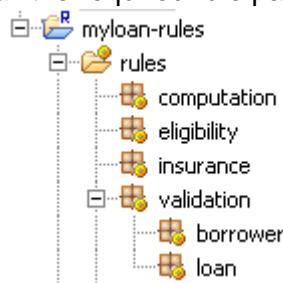
- 3. Add the remaining packages by using the instructions in Step 2:
  - `eligibility`
  - `insurance`
  - `validation`
- 4. Create the `borrower` and `loan` subpackages of the `validation` package.
  - a. In the Orchestrate part of the Rule Project Map, click the **Add rule package** link.
  - b. In the **Package** field, type `validation.borrower` and press Enter.
  - c. Add another package, type `validation.loan` in the **Package** field, and press Enter.



### Note

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

The rule project now contains all the required rule packages.



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



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

## Section 6. Publishing from Rule Designer to Decision Center

In this part of the exercise, you publish the rule project from Rule Designer to Decision Center.

Notice that the links in the Author part of the Rule Project Map are now enabled.



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

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

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

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

- 1. If the sample server is not already started, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server.**



#### Reminder

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

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

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

- 2. In Rule Designer, right-click the **myloan-rules** project (in the Rule Explorer view) and click **Decision Center > Connect**.

The Decision Center configuration wizard opens.

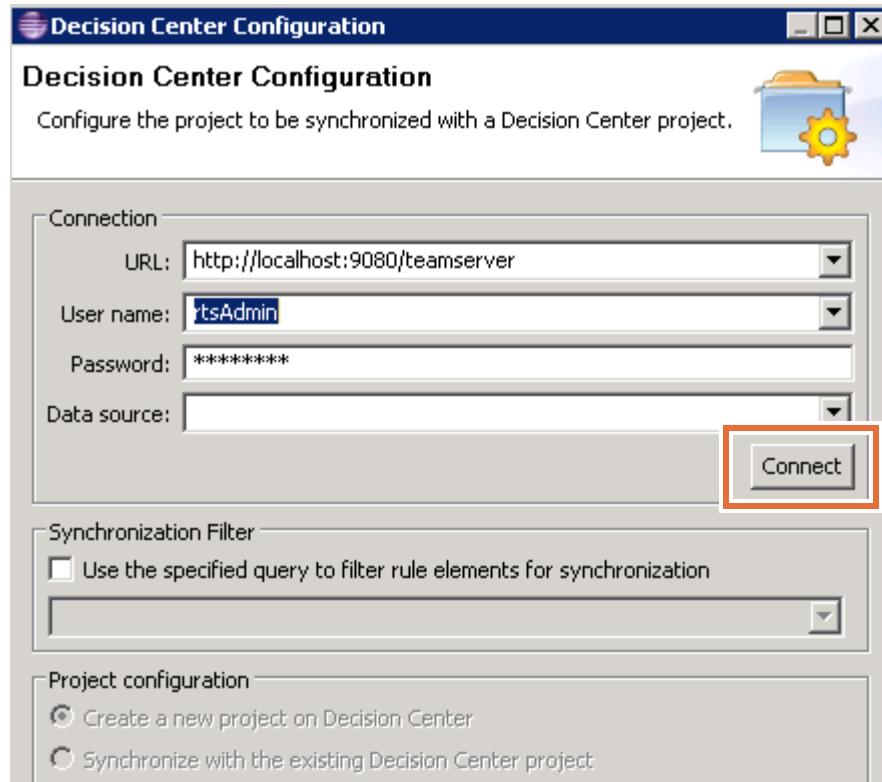
- 3. Enter the following values in the Decision Center Configuration window fields.

- **URL:** `http://localhost:9080/teamserver`

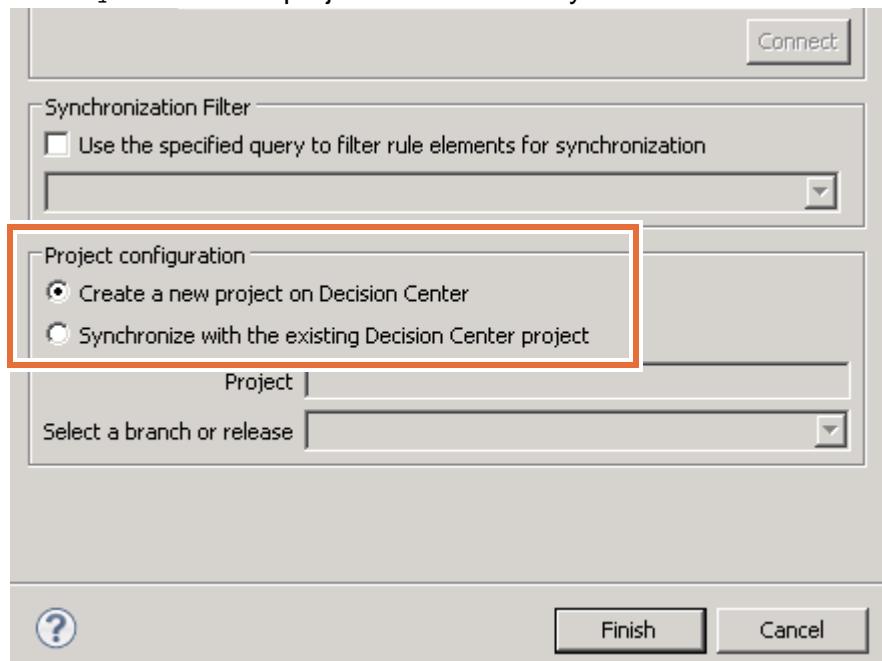
Make sure that you use the correct port for your environment.

- **User name:** `rtsAdmin`
- **Password:** `rtsAdmin`
- **Data source:** (Leave the field empty)

- 4. Click **Connect**.



After the connection is successfully established, the “Project configuration” section is enabled. The **Create a new project on Decision Center** option should be selected because the `myloan-rules` project does not exist yet on Decision Center.



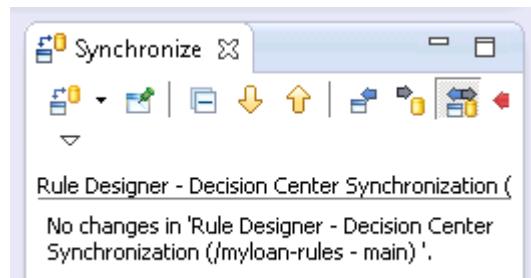
- 5. Click **Finish**.

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



- \_\_\_ 6. Click **OK** to close the window.
- \_\_\_ 7. When prompted to switch to the Team Synchronizing perspective, click **No**.

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



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



#### Important

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

## 6.2. Opening the project in Decision Center Business console

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

- \_\_\_ 1. Open Decision Center Business console.
- \_\_\_ a. Click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Decision Center Business console**.

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

`http://localhost:9080/decisioncenter`

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

- \_\_\_ b. Sign in with `rtsAdmin` as both the **User name** and the **Password**.

The console opens to the home page.

- 2. Click the **Library** tab and click **Classic Rule Projects** and notice the new `myloan-rules` project is now listed.

- 3. Click **myloan-rules** to open it.

The project does not contain any rules.

- 4. Log out and close the Business console.

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

### 6.3. Opening the project in the Decision Center Enterprise console

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

- 1. Open the Decision Center Enterprise console.
- a. Click the Enterprise console shortcut on your desktop or click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Decision Center Enterprise console**.

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

`http://localhost:9080/teamserver`

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

- \_\_\_ b. Sign in with `rtsAdmin` as both the **User name** and the **Password**.
- \_\_\_ 2. On the **Home** tab of the Decision Center Enterprise console, you can see the different projects and decision services that are accessible to business users.
- \_\_\_ 3. In the **Project in use** field, select **myloan-rules** (and leave the other fields unchanged).

Work on a project

Project in use:

Branch in use:

Current action:

Work on a Decision Service

Decision Service in use:

Branch in use:

Current action:

- \_\_\_ 4. Click the **Explore** tab.
- \_\_\_ 5. In the Smart Folders list, expand the **validation** folder.

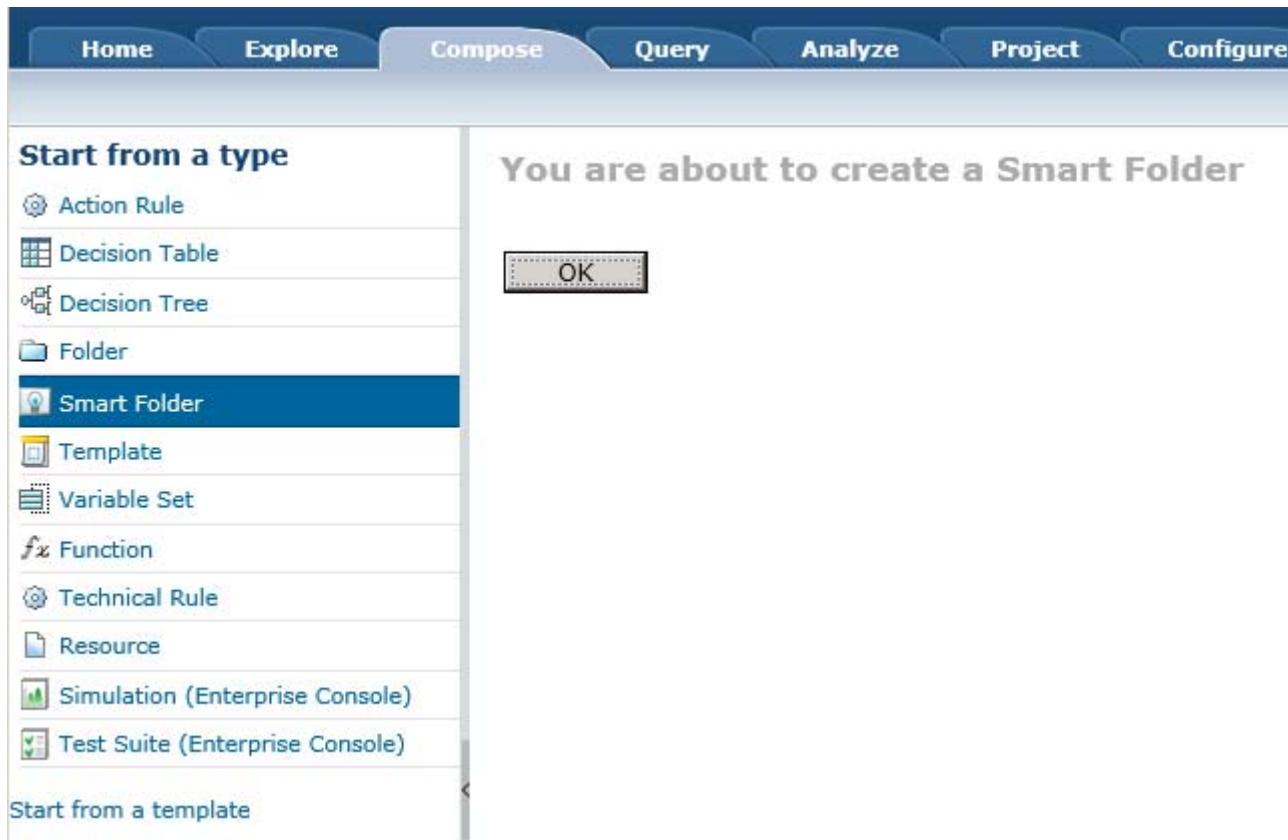
The **Explore** tab lists all the artifacts of the `loan-rules` project. Artifacts are organized within smart folders. The project does not contain any rules yet.

| Actions                                   | Name | Status |
|-------------------------------------------|------|--------|
| There are no business rules at this level |      |        |

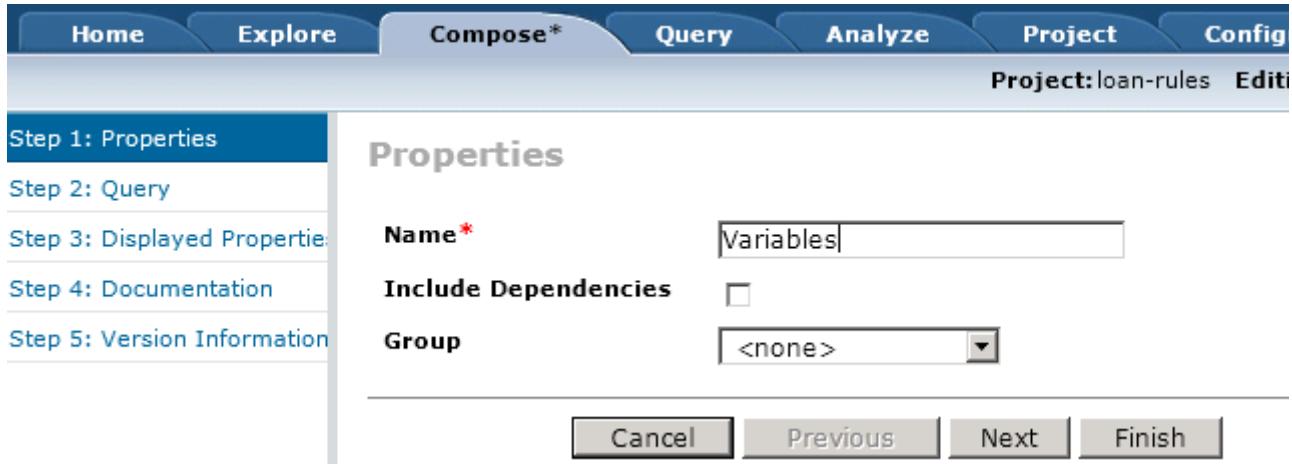
By default, all business rules are under the **Business Rules** smart folder, and organized in subfolders that correspond to rule packages in Rule Designer.

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

- \_\_\_ 6. Click the **Compose** tab.
- \_\_\_ 7. In the **Start from a type** pane, click **Smart Folder**, and click **OK**.

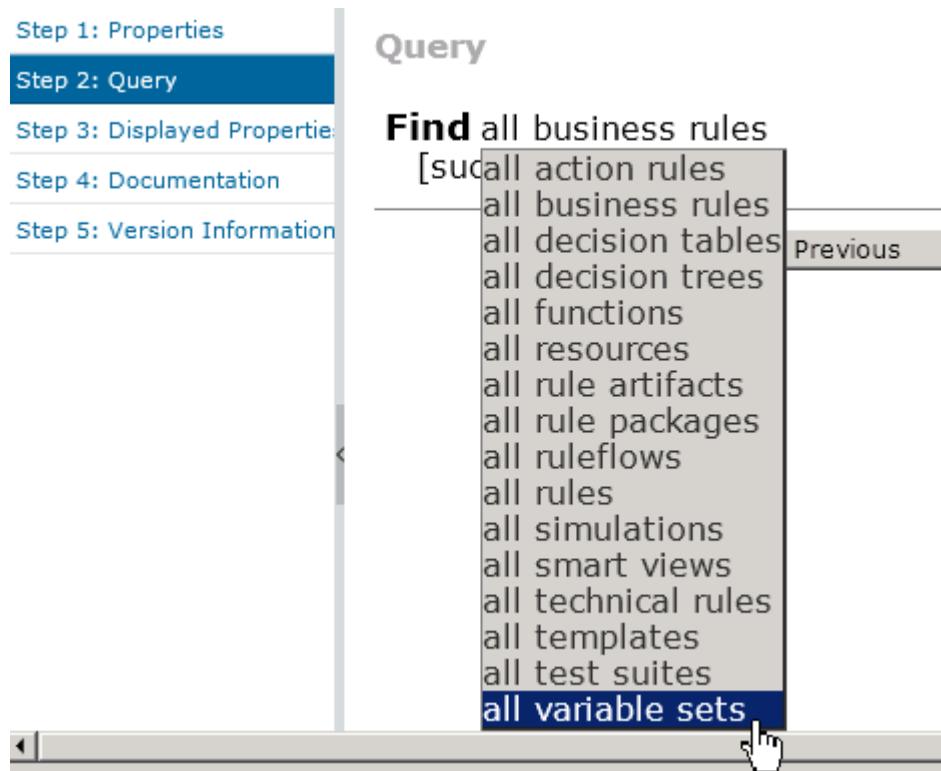


- \_\_\_ 8. Name the smart folder **Variables** and click **Next**.



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

- \_\_\_ 9. In the **Find** statement, click **all business rules** and select **all variable sets**.



- \_\_\_ 10. Click **Finish**.

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

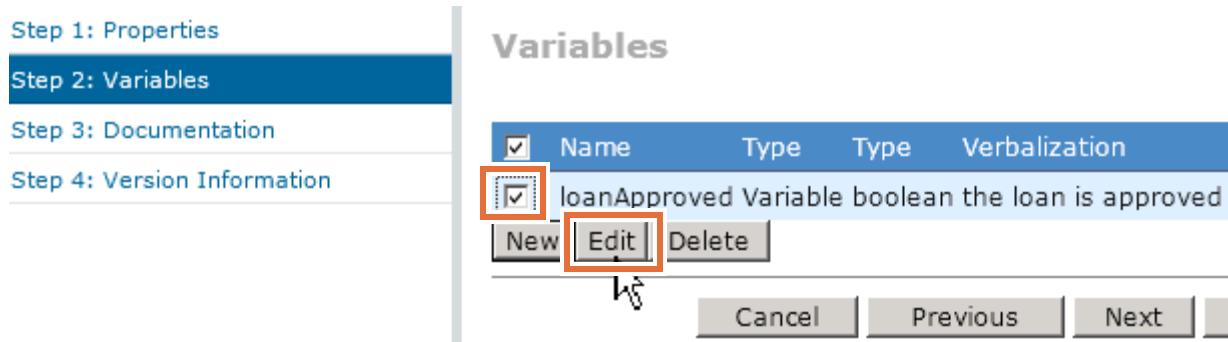
#### 6.4. Modifying the ruleset variable in Decision Center

- \_\_\_ 1. On the **Explore** tab, select the **local var** variable set in the table and click **Edit**.

| Actions                             | Name      |
|-------------------------------------|-----------|
| <input checked="" type="checkbox"/> | local var |

- \_\_\_ 2. The **Compose** tab opens. Click **Next** to open the Variables page.

- \_\_\_ 3. Select the **loanApproved** variable and click **Edit**.



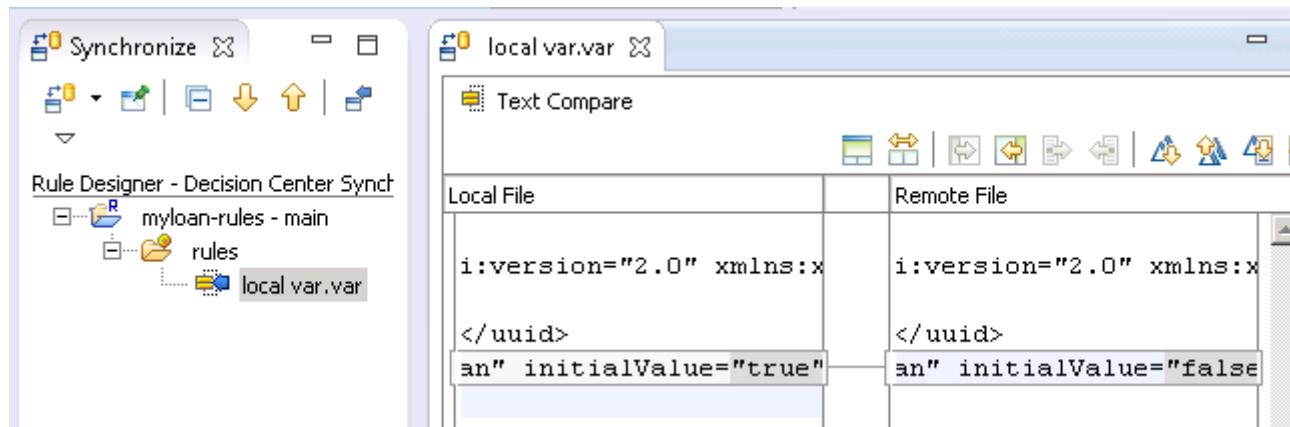
- \_\_\_ 4. In the **Initial Value** field, change the value from `true` to `false` and click **Apply**.

- \_\_\_ 5. Click **Finish**.

The variable is updated and opens in the Details page.

## 6.5. Synchronizing Rule Designer with Decision Center

- \_\_\_ 1. Go back to Rule Designer in the Rule perspective.
- \_\_\_ 2. Synchronize the `myloan-rules` project with Decision Center.
  - \_\_\_ a. Right-click `myloan-rules` and click **Decision Center > Synchronize with Decision Center**.
  - \_\_\_ b. In the Synchronization Settings window, click **Finish**.
  - \_\_\_ c. When prompted to switch to the Team Synchronizing perspective, click **Yes**.
- \_\_\_ 3. In the Synchronize view, expand **myloan-rules - main > rules** to see that your ruleset variable is changed.
- \_\_\_ 4. Double-click the variable to open it in the Text Compare view and check your changes.



- \_\_\_ 5. Close the Text Compare view.

**Information**

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

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

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

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

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

## Section 7. Deleting a project from Decision Center

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

- \_\_\_ 1. If you closed the Decision Center Enterprise console, sign in again with `rtsAdmin` as the user name and password.
- \_\_\_ 2. On the **Home** tab, select **Work on a project**, then select the `myloan-rules` project as the project in use.
- \_\_\_ 3. In the **Configure** tab, under Administration, click **Erase Current Project**.
- \_\_\_ 4. Click **Yes** in the window to confirm the project deletion.
- \_\_\_ 5. Close the Enterprise console.

## Section 8. Creating a rule project from Decision Center

In this part of the exercise, you create a rule project in Rule Designer from an existing rule project in Decision Center.

 Requirements

Business analysts indicate that they worked with another team and developed a complementary project, called `loanvalidation-rules`. The latest version of this `loanvalidation-rules` project is in Decision Center. Business analysts ask you to import this project in Rule Designer to look at it.

### 8.1. Creating a project in Rule Designer from Decision Center

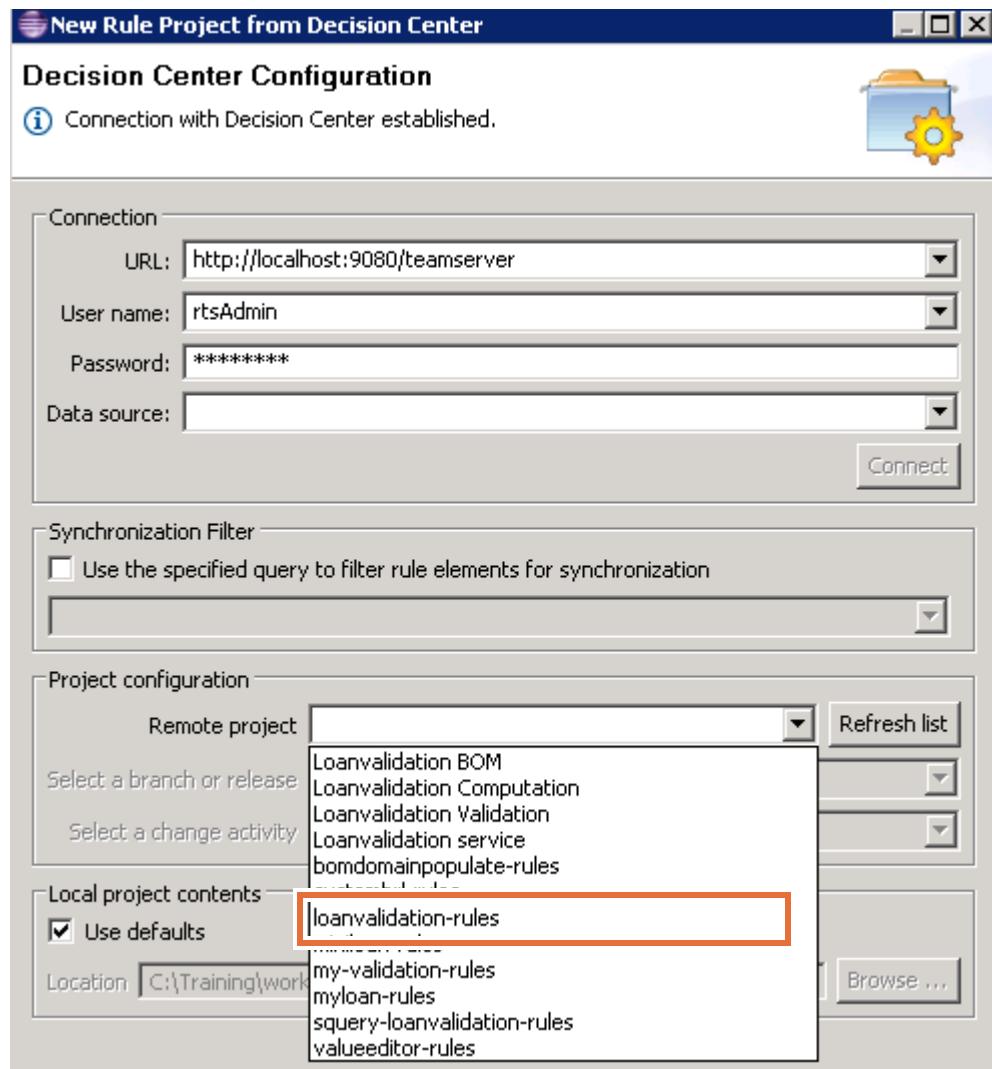
- \_\_\_ 1. Go back to Rule Designer and make sure that you are in the Rule perspective.
- \_\_\_ 2. Click **File > New > Project**, select **Rule Project from Decision Center**, and click **Next**.



The New Rule Project from Decision Center wizard opens.

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

4. In the Project configuration section, select the `loanvalidation-rules` project from the **Remote project** list to import it in Rule Designer.



#### Note

Click **Refresh List** if you cannot see the project in the list.

The **Remote project** list shows all the rule projects in Decision Center for which you have the **View** permission. This permission depends on the **user name** that you used to connect. Here, because you are connected as `rtsAdmin`, the Decision Center administrator, you can see all the rule projects in Decision Center.

5. In the **Select a branch or release** menu, select the `main` branch.  
 6. Click **Connect** to establish the server connection.  
 7. Click **Finish**.

Rule Designer imports all the items in the rule project into your workspace and creates the `loanvalidation-rules` project in Rule Designer based on its content in Decision Center.



## Troubleshooting

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

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



## Reminder

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

- \_\_\_ 12. Stop the sample server by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Stop server**.

## 8.2. Finalizing the rule project in Rule Designer

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

- \_\_\_ 1. In the Rule perspective, wait for the `loanvalidation-rules` project to finish building.

2. Look at the messages in the Problems view and notice that they are prefaced with "B2X".

| 10 errors, 0 warnings, 0 others                                                                        |                |
|--------------------------------------------------------------------------------------------------------|----------------|
| Description                                                                                            | Resource       |
| Errors (10 items)                                                                                      |                |
| [B2X] Cannot find execution class "loan.Borrower.Bankruptcy" for translating business object model.bom | model.bom      |
| [B2X] Cannot find execution class "loan.Borrower.SSN" for translating business object model.bom        | model.bom      |
| [B2X] Cannot find execution class "loan.Borrower" for translating business object model.bom            | model.bom      |
| [B2X] Cannot find execution class "loan.DateUtil" for translating business object model.bom            | model.bom      |
| [B2X] Cannot find execution class "loan.Loan" for translating business object model.bom                | model.bom      |
| [B2X] Cannot find execution class "loan.LoanUtil" for translating business object model.bom            | model.bom      |
| [B2X] Cannot find execution class "loan.Report" for translating business object model.bom              | model.bom      |
| [B2X] Cannot find execution class "loan.Test" for translating business object model.bom                | model.bom      |
| [B2X] Cannot find execution class "loan.TypedLoan" for translating business object model.bom           | model.bom      |
| The project path platform:/loanvalidation-xom cannot be resolved.                                      | loanvalidati.. |

3. Open the loanvalidation-rules project properties.

- a. Right-click the **loanvalidation-rules** project and click **Properties** from the menu.
- b. In the Properties window, select **Business Object Model**.
- c. In the Business Object Model page, expand the list in the Required BOM entries to see the origin or source for the **loanvalidation-rules** BOM.



### Questions

Why does your new project have errors?

### Answer

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



### Questions

Which executable elements are required?

## Answer

The loanvalidation-rules project cannot build because it requires the loanvalidation-xom project, which is not present in your Rule Designer workspace.



**Stop**

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

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

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

## End of exercise

## Exercise review and wrap-up

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

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

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



### Note

The solution project does not include the imported `loanvalidation-rules` project.



# Exercise 3. Working with the BOM

## What this exercise is about

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

## What you should be able to do

After completing this exercise, you should be able to:

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

## Introduction

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

The exercise includes these tasks:

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

## Requirements

You should complete this exercise before proceeding:

- Exercise 2, "Setting up rule projects"

## Section 1. Finalizing the XOM

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

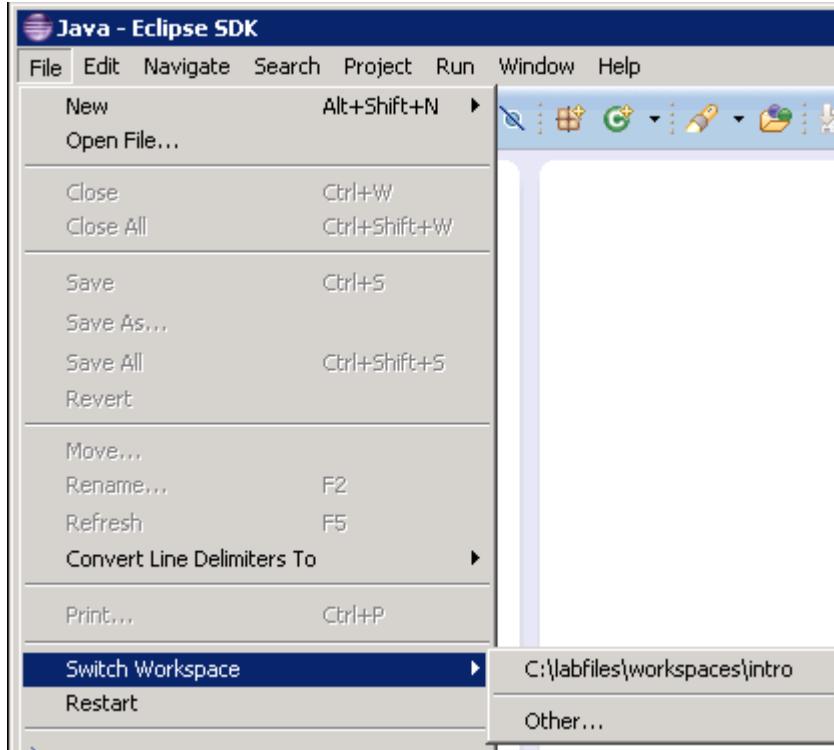


### Note

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

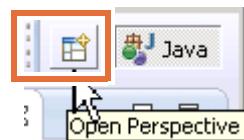
### 1.1. Setting up your environment for this exercise

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

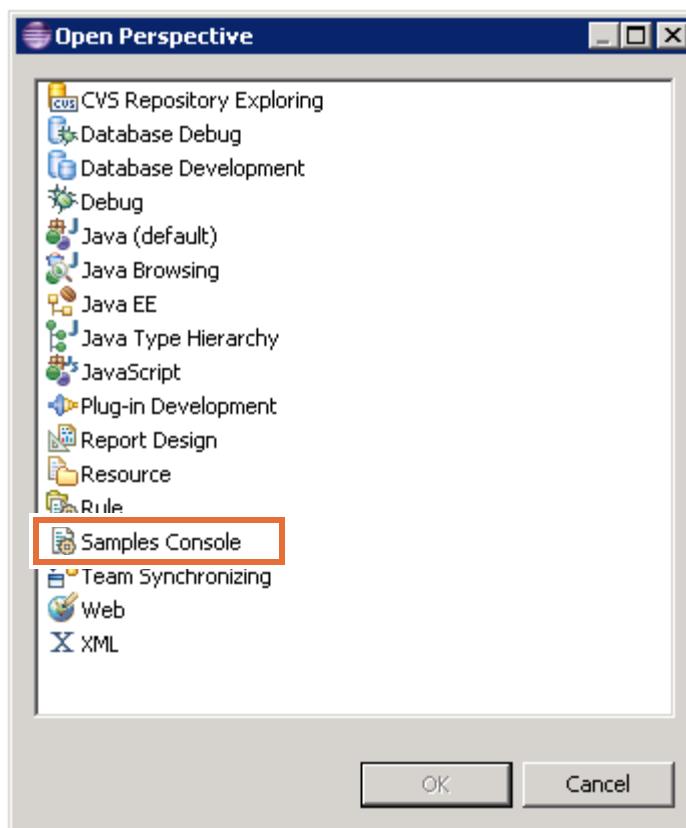


- b. In the **Workspace Launcher** window, enter the path:  
`<LabfilesDir>\workspaces\bom`
- c. Ignore **Copy Settings**, and click **OK**.

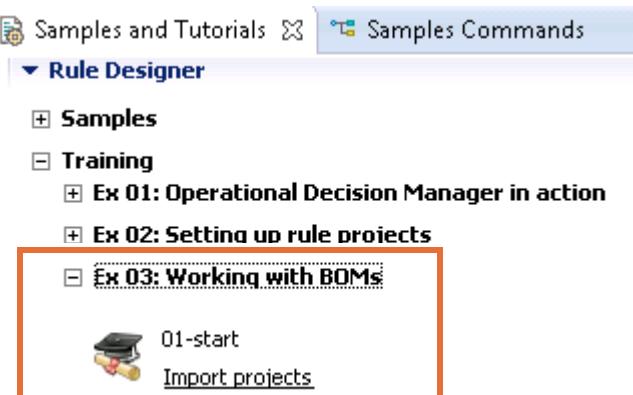
- \_\_\_ 2. Close the Welcome view and switch to the Samples Console perspective.
- \_\_\_ a. Click the **Open Perspective** icon in the upper-right part of the Eclipse window.



- \_\_\_ b. Select **Samples Console**, and click **OK**.



- \_\_\_ 3. Go to **Rule Designer > Training > Ex 03: Working with BOMs > 01-start** and click **Import projects**.

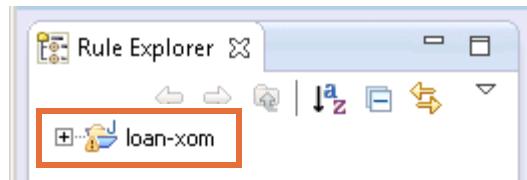


The Rule perspective opens.

- \_\_\_ 4. Close the Help view.

5. In Rule Explorer, notice that you now have the `loan-xom` Java project available in your workspace.

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



## 1.2. Understanding the problem

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

 A screenshot of the Problems view in the Rule perspective. The tab bar at the top has "Problems" highlighted with a red box. Below the tab bar is a message: "0 errors, 1 warning, 0 others". The main area is a table with columns: Description, Resource, Path, Location, and Type. There is one entry under "Warnings": "The value of the field Borrower.spouse is not used" in file "Borrower.java" at line 27, categorized as a "Java Problem".
 

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

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

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

 A screenshot of the Tasks view in the Rule perspective. The tab bar at the top has "Tasks" highlighted with a red box. Below the tab bar is a table with columns: Description and Resource. There are five entries:
 

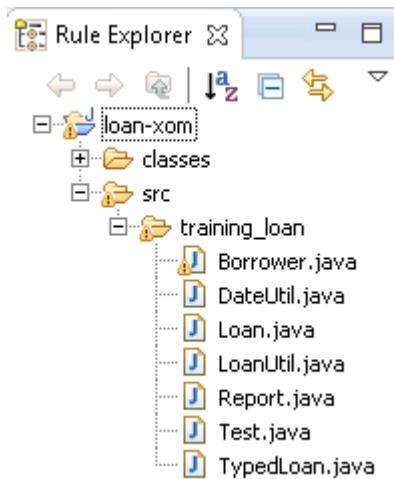
| Description                                                                                                            | Resource      |
|------------------------------------------------------------------------------------------------------------------------|---------------|
| TODO : Uncomment when the getters and setters of the Borrower's attributes are created                                 | Test.java     |
| TODO : Uncomment when the getters and setters of the Borrower's attributes are created                                 | Test.java     |
| TODO: Create a getter called getBirthDate for the birth attribute.                                                     | Borrower.java |
| TODO: Create the getters (not the setters) for the following attributes: firstName, lastName, and SSN.                 | Borrower.java |
| TODO: Create the getters and the setters for the following attributes: zipCode, yearlyIncome, creditScore, and spouse. | Borrower.java |

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

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

### 1.3. Finish implementing the Borrower class

- 1. In Rule Explorer, expand **loan-xom > src > training\_loan** to see the **Borrower.java** class file.



- 2. In the **Borrower** class, create the getters (not the setters) for the following attributes:

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



#### Example

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

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

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

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

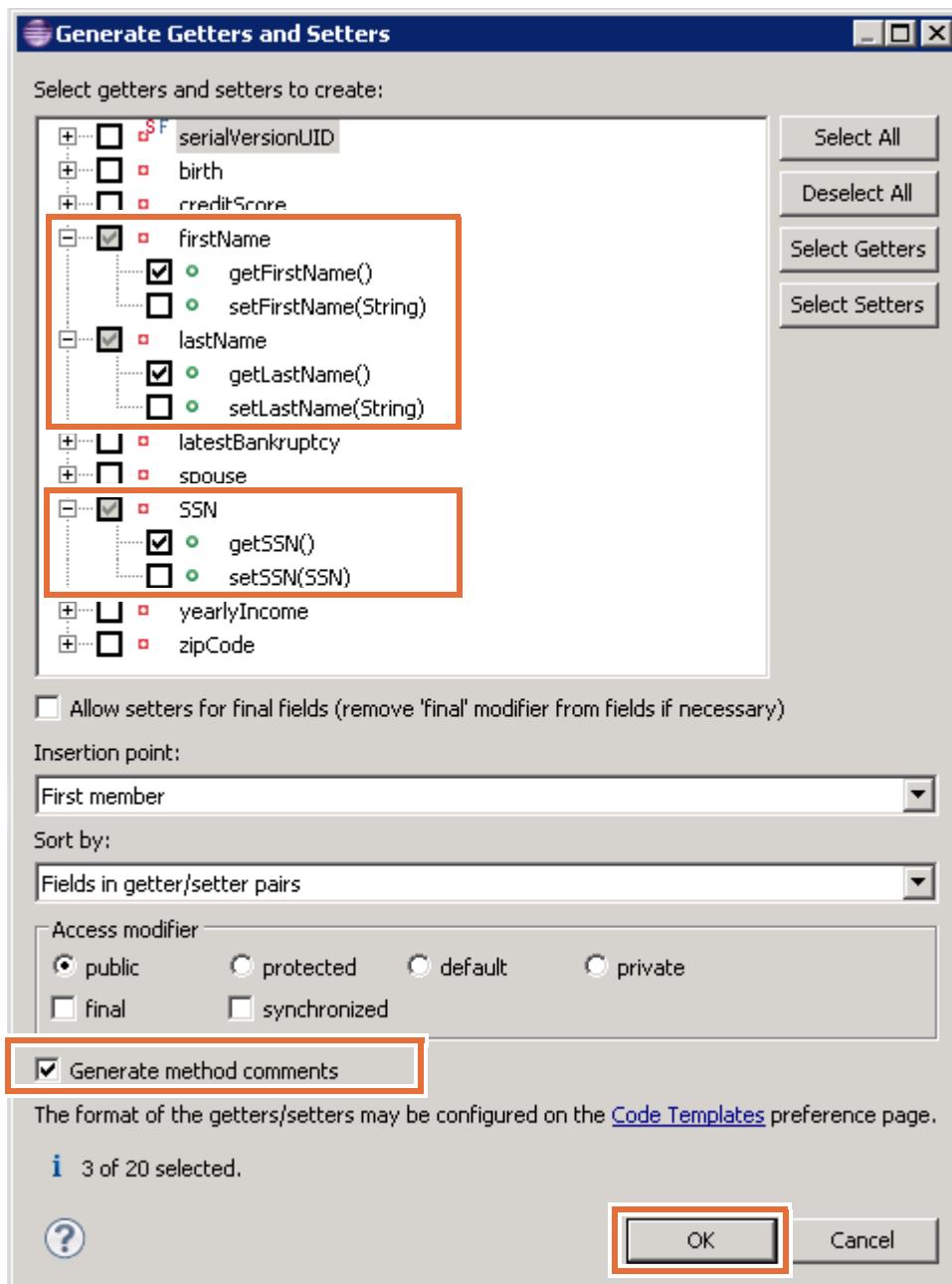
- \_\_ b. In the Borrower class code, select the Borrower class name, and right-click to select **Source > Generate Getters and Setters**.

The screenshot shows a Java code editor window with the file 'Borrower.java' open. The code defines a class 'Borrower' that implements the 'Serializable' interface. The class has several private fields: 'String name', 'String address', 'String SSN', 'int age', 'String phone', 'int id', 'Borrower parent', 'Calendar birthDate', and 'Bankruptcy bankruptStatus'. There are also annotations '@Override' and '@SuppressWarnings("unchecked")' at the bottom.

A context menu is open over the 'Borrower' class name. The menu is organized into sections:

- Source**:
  - Undo (Ctrl+Z)
  - Revert File
  - Save (Ctrl+S)
- Generate**:
  - Open Declaration (F3)
  - Open Type Hierarchy (F4)
  - Open Call Hierarchy (Ctrl+Alt+H)
  - Show in Breadcrumb (Alt+Shift+B)
  - Quick Outline (Ctrl+O)
  - Quick Type Hierarchy (Ctrl+T)
  - Open With
  - Show In (Alt+Shift+W)
- Refactor**:
  - Cut (Ctrl+X)
  - Copy (Ctrl+C)
  - Copy Qualified Name
  - Paste (Ctrl+V)
- Code Generation**:
  - Quick Fix (Ctrl+1)
  - Source (Alt+Shift+S)
- Comment**:
  - Toggle Comment
  - Add Block Comment
  - Remove Block Comment
  - Generate Element Comment
- Format**:
  - Correct Indentation
  - Format
  - Format Element
- Imports**:
  - Add Import
  - Organize Imports
  - Sort Members...
  - Clean Up...
- Overrides**:
  - Override/Implement Methods...
  - Generate Getters and Setters... (highlighted with a cursor)
- Utilities**:
  - Generate Delegate Methods...
  - Generate hashCode() and equals()...
  - Generate toString()...
  - Generate Constructor using Fields...
  - Generate Constructors from Superclass...
- Localization**:
  - Externalize Strings...

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



- \_\_\_ d. Select **Generate method comments** and click **OK**.

- \_\_ e. Go back to the `Borrower.java` file to see the newly added getter methods.

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

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

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



### Note

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

- \_\_ 3. In the `Borrower` class, create both the getters and the setters for the following attributes:

- creditScore
- spouse
- yearlyIncome
- zipCode



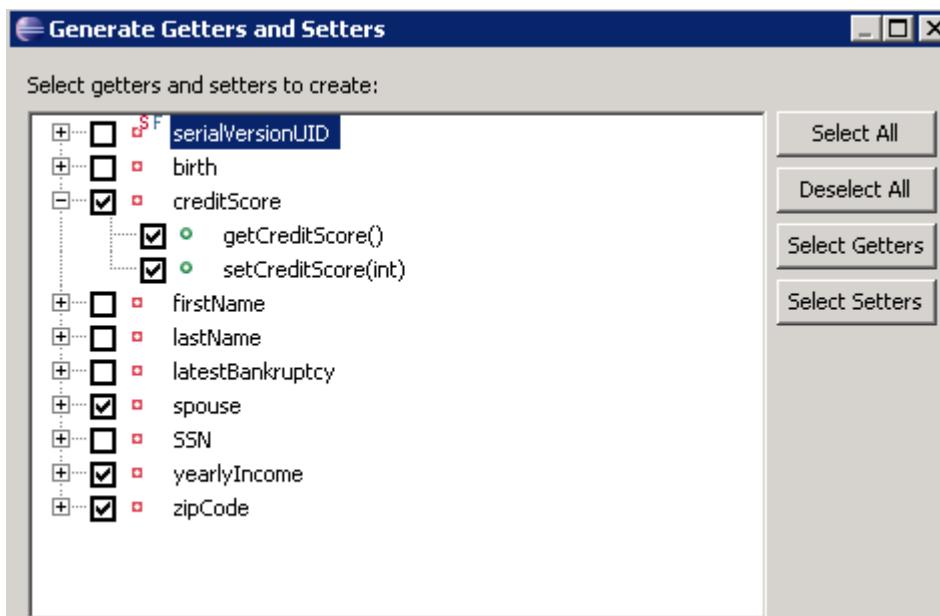
## Example

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

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

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

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



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

**Note**

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

- \_\_\_ 4. In the `Borrower` class, create a getter that is called `getBirthDate` for the `birth` attribute.

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

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

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

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

- \_\_\_ 5. In the `Borrower` class, look for the following comments and delete them:

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

- \_\_\_ 6. Save the `Borrower.java` file by pressing **Ctrl+S**.

## 1.4. Finish implementing the Test class

- \_\_\_ 1. In Rule Explorer, double-click **Test.java** to open it in the code editor and follow the instructions of the `TODO` comments.

- \_\_\_ a. Look through the `Borrower` class to find the following method calls and delete the forward slashes `//` to obtain these lines of code:

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

These lines are in various places in the file.

```
public class Test {

 public static void main(String[] args) {
 Borrower b1 = new Borrower("John", "Doe", DateUtil.makeDate(1968, Calend
 b1.setCreditScore(600);
 b1.setYearlyIncome(100000);
 b1.setLatestBankruptcy(DateUtil.makeDate(1990, Calendar.JANUARY, 01), 7,
 System.out.println(b1);

 Borrower b2 = new Borrower("John", "Doe", DateUtil.makeDate(1970, Calend
 System.out.println(b2);

 Borrower b3 = new Borrower("John", "Doe", DateUtil.makeDate(1970, Calend
 System.out.println(b3);

 Loan l1 = new Loan(DateUtil.makeDate(2005, Calendar.JUNE, 1), 60, 100000
 System.out.println(l1);

 Report r1 = new Report(b1, l1);
 r1.setCorporateScore(b1.getCreditScore());
 r1.addCorporateScore(12);

 System.out.println(r1);
```

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

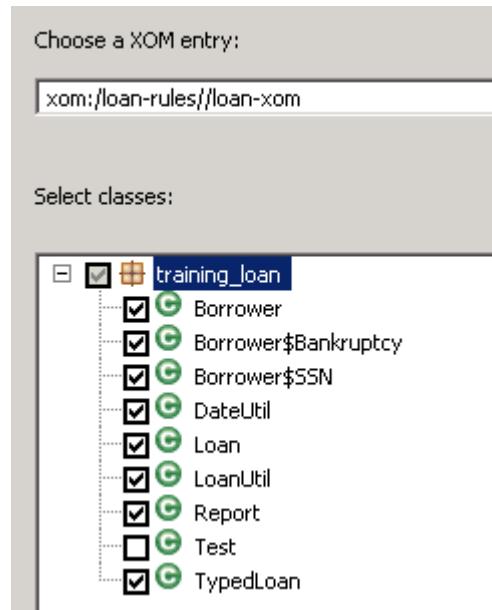
## Section 2. Creating a rule project and its BOM

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

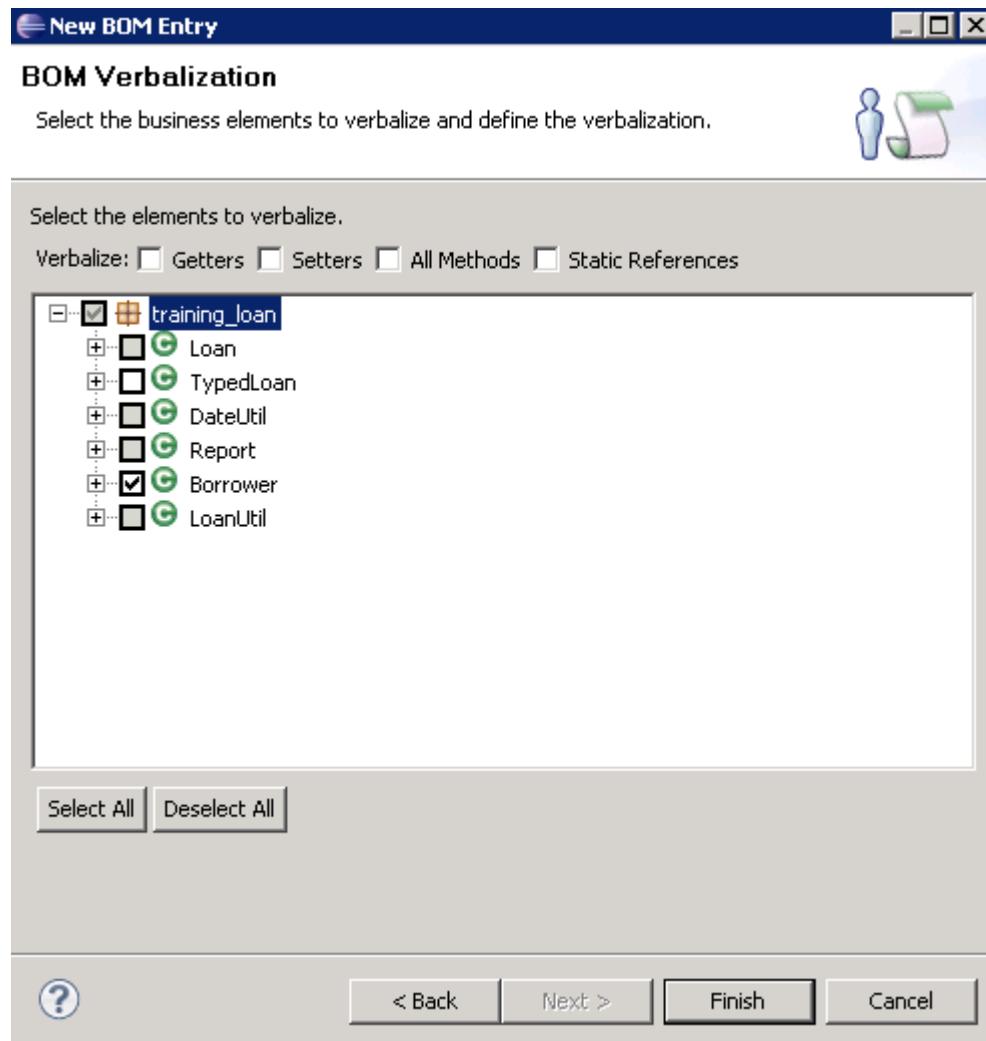
- \_\_ 1. Create a standard rule project.
  - \_\_ a. Right-click anywhere in the Rule Explorer and select **New > Rule Project**.
  - \_\_ b. In the “Select a template” page, go to the Classic Rule Projects section.
  - \_\_ c. Select **Standard Rule Project** and click **Next**.
  - \_\_ d. In the **Project name** field, type `loan-rules` and click **Next**.
  - \_\_ e. Click **Next** to skip the **Rule Project References** page.
  - \_\_ f. In the Rule Project XOM Settings page, select the `loan-xom` Java project and click **Finish**.

The new rule project opens in the rule editor.

- \_\_ 2. Create a BOM entry in the `loan-rules` project.
  - \_\_ a. In Rule Explorer, right-click **loan-rules > bom** and click **New > BOM Entry**.
  - \_\_ b. In the New BOM Entry window, keep `model` in the **Name** field, select the **Create a BOM entry from a XOM** option, and click **Next**.
  - \_\_ c. For the **Choose a XOM entry** field, click **Browse XOM**.
  - \_\_ d. In the Browse XOM window, select `platform:/loan-xom`, and click **OK** to close the window.
  - \_\_ e. In the “Select classes” section, expand **training\_loan**, select the check boxes for all classes except `Test`, and click **Next**.



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



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

Only the `Borrower` class is verbalized.

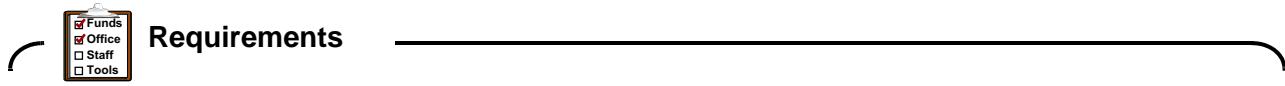
- \_\_\_ 3. Make sure that everything is saved by pressing **Ctrl+Shift+S**.

You can find an intermediate solution to this point of the exercise by importing the project **Ex 03: Working with BOMs > 02-intermediate answer**.

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

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

### 3.1. Creating the default vocabulary

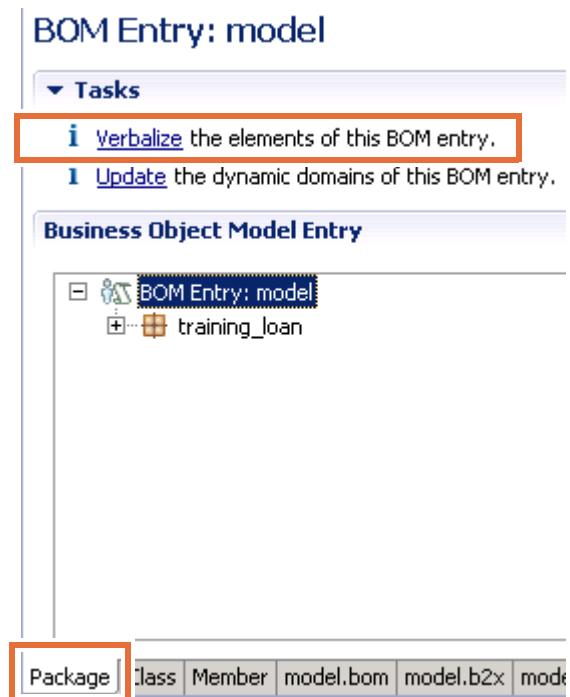


Business analysts examined the BOM that you created for the project and found that some elements must be reworked. They now ask for a complete verbalization of the `Report` class and of the `Loan` class. They also ask you to revise the vocabulary that is associated with the `Borrower` class.

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

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

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

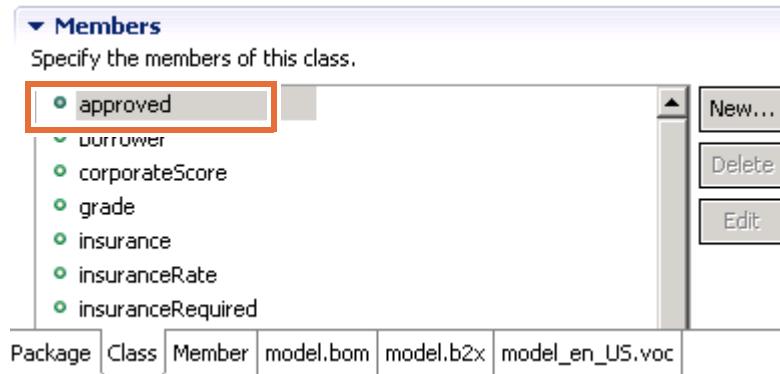


The Verbalize BOM wizard opens.

- \_\_\_ b. In the Configure Verbalization page of the Verbalize BOM wizard, click **Deselect All**, and then select **Report**.

You want to verbalize the `Report` BOM class and all its members.

- \_\_\_ c. Click **Finish**.
- \_\_\_ d. Save your work (Ctrl+S).
- \_\_\_ e. In the **Business Object Model Entry** section of the BOM editor, expand `training_loan` and double-click **Report** to open it in the Class page of the BOM editor.
- \_\_\_ f. In the **Members** section of the Class page, double-click **approved** to open it in the Member page.



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

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

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

**Class Verbalization**

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

Generate automatic variable



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

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

- \_\_\_ 6. Create the default verbalization for the `duration` member of the `Loan` class.
  - \_\_\_ a. In the Class page for the `Loan` class, double-click the `duration` member to edit it in the Member page of the BOM editor.
- \_\_\_ b. In the Member Verbalization section of the Member page, click **Create a default verbalization**.

**Member Verbalization**

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

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

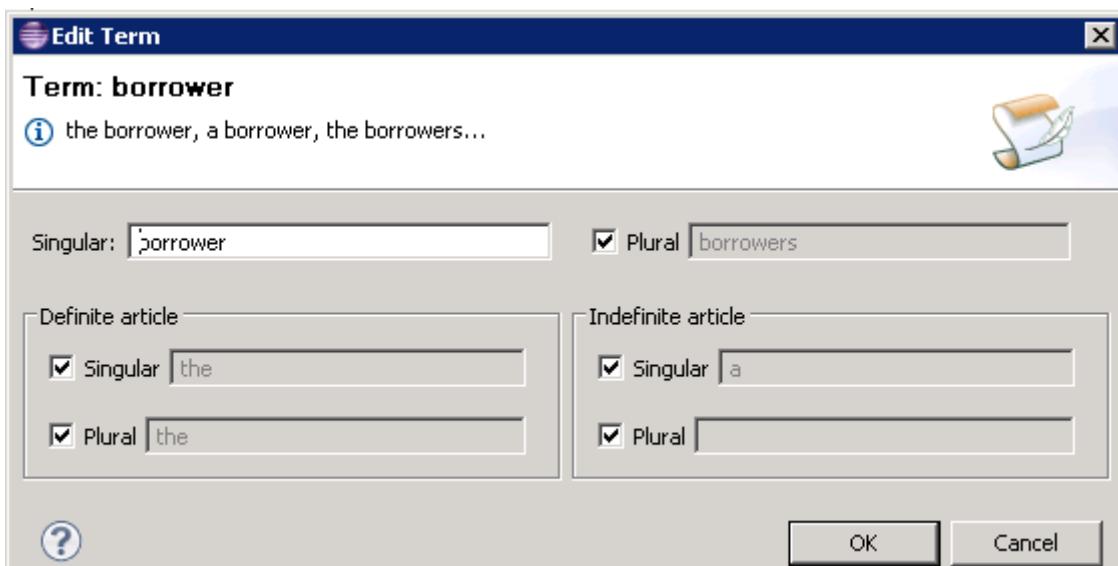
### 3.2. Examining the verbalization for members of the `Borrower` class

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

- \_\_ 2. Click **Edit term** in the Class Verbalization section.

The screenshot shows a user interface for class verbalization. At the top, there's a link labeled 'Class Verbalization'. Below it, there are two buttons: 'Remove' (with a red X icon) and 'Edit' (with a pencil icon). A checkbox labeled 'Generate automatic variable' is also present. A text input field contains the word 'borrower'. To the right of the input field is a button labeled 'Edit term.' with a hand cursor icon pointing at it. Below the input field, a tooltip-like message says 'the borrower, a borrower, the borrowers....'

The Edit Term window opens.



- \_\_ a. Notice that you can edit the `borrower` term to set the vocabulary for singular and plural forms, and for the definite and indefinite articles.
- \_\_ b. Click **Cancel** to close the Edit Term window.
- \_\_ 3. Open the `birthDate` member of the `Borrower` class in the BOM editor to explore how you can edit its verbalization in the **Template** field in the Member Verbalization section.
- \_\_ 4. Open the `hasLatestBankruptcy` method of the `Borrower` class and examine its default verbalization:

`{this} is has latest bankruptcy`



### Questions

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

**Answer**

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

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

**Important**

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

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

5. Open the `setLatestBankruptcy` method of the `Borrower` class and examine its default verbalization in the **Template** field.

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

▼ Member Verbalization

✗ Remove the verbalization.

+ Create an action phrase.

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

Template:

**Questions**

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

**Answer**

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

**Questions**

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

**Answer**

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

**Arguments**  
Edit the arguments of this member.

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

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

### 3.3. Editing the default verbalization

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

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

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

```
{this} has latest bankruptcy
```



Note

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

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

- 
- 2. Enter the following new verbalization for the `setLatestBankruptcy` method in the **Template** field in the Member Verbalization section:

`set the latest bankruptcy of {this} to date {0}, chapter {1} and reason {2}`

Notice that the BOM editor notifies you of missing placeholders as you edit the **Template** field.

- 3. Save your work.
- 4. Close the BOM editor.

## End of exercise

## Exercise review and wrap-up

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

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

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



# Exercise 4. Refactoring

## What this exercise is about

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

## What you should be able to do

After completing this exercise, you should be able to:

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

## Introduction

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

The exercise includes these tasks:

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

## Requirements

There are no specific requirements for this exercise.

## Section 1. Refactoring rule artifacts after vocabulary changes

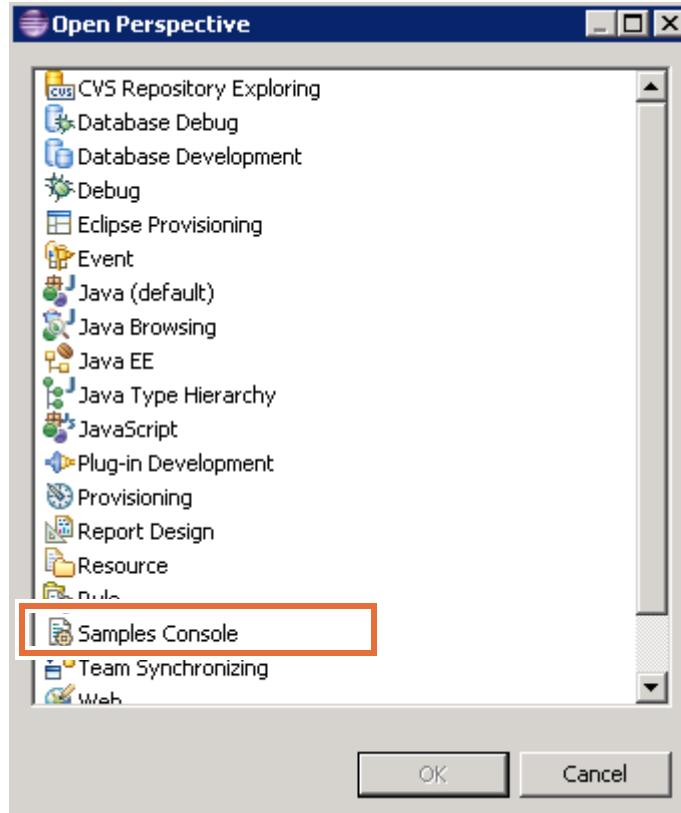
In this part of the exercise, you see how a change in the vocabulary affects rule artifacts, and learn more about the relationship between the rule artifacts and the vocabulary in the BOM.

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

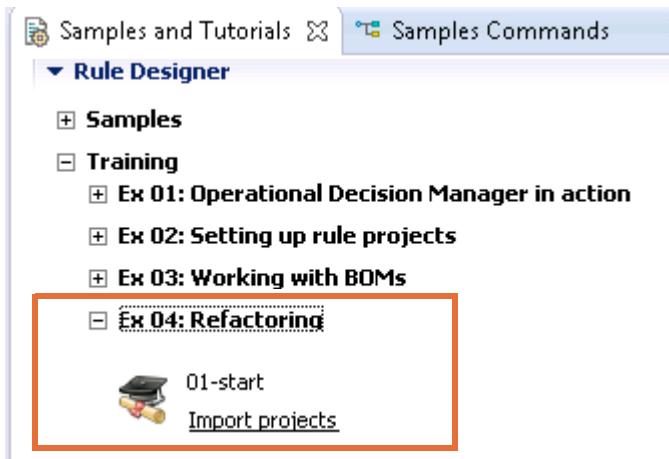
- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the **Workspace Launcher** window, enter the path:  
*<LabfilesDir>\workspaces\refactoring*
- 2. Close the Welcome view and switch to the Samples Console perspective.
  - a. Click the **Open Perspective** icon in the upper-right part of the Eclipse window.



- b. Select **Samples Console**, and click **OK**.



3. Select Rule Designer > Training > Ex 04: Refactoring > 01-start > Import projects.



The Rule perspective opens.

4. Close the Help view.

## 1.2. Exploring the default verbalization

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

The condition statement uses this vocabulary:

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



### Questions

Is this rule condition easy to understand and unambiguous?

### Answer

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



### Questions

How can you solve this ambiguity issue?

### Answer

Change the vocabulary to use clear language.



### Important

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



### Requirements

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

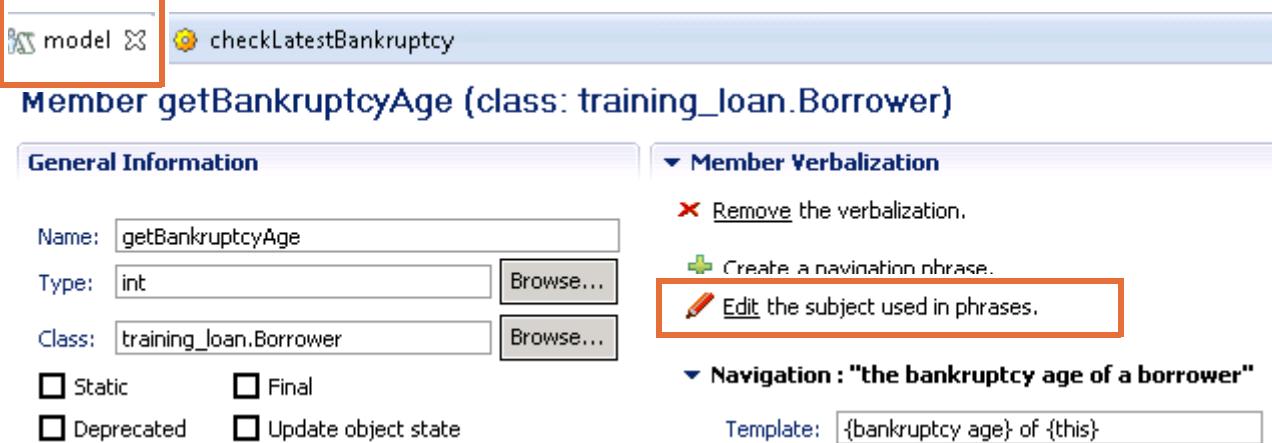
- The current “bankruptcy age” term should indicate the *latest* bankruptcy.
- The navigation template for the duration member of the Loan class should indicate that the duration is in *years*.

### 1.3. Changing the default verbalization of the getBankruptcyAge method

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

- \_\_\_ 1. Edit the `getBankruptcyAge` BOM member verbalization to become: age of the latest bankruptcy

- \_\_\_ a. Switch back to the **model** tab.



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

**General Information**

Name: `getBankruptcyAge`

Type: `int`

Class: `training_loan.Borrower`

Static     Final  
 Deprecated     Update object state

**Member Verbalization**

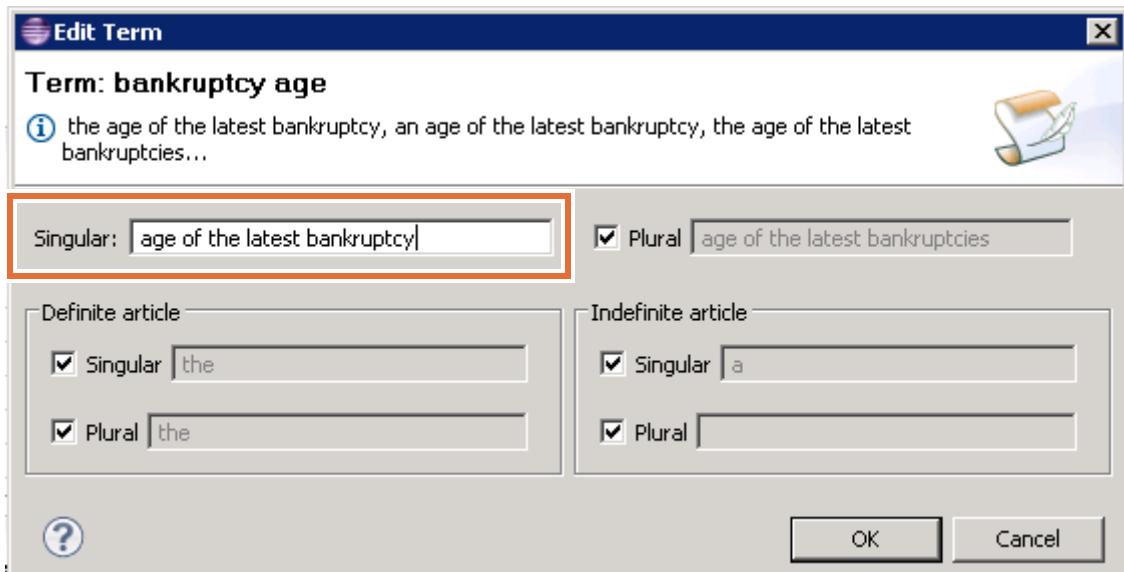
Remove the verbalization.  
 Create a navigation phrase.  
 Edit the subject used in phrases.

**Navigation : "the bankruptcy age of a borrower"**

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

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

age of the latest bankruptcy



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

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

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

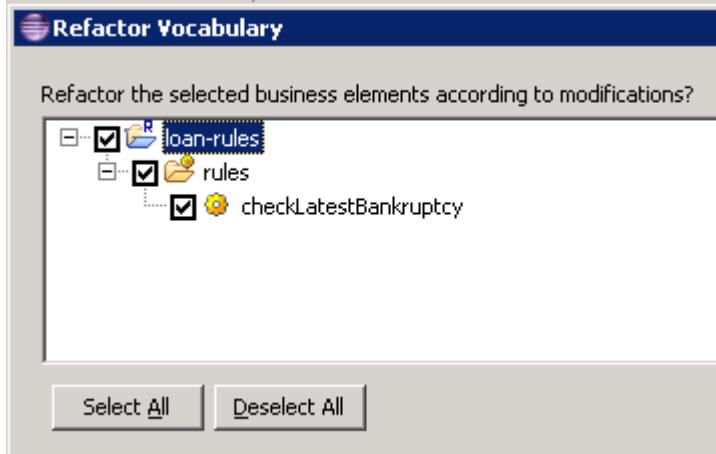
Template: `{age of the latest bankruptcy} of {this}` 

 Changes of this verbalization may impact business rules.

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

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

The Refactor Vocabulary window opens.



Because the previous vocabulary term is no longer valid, you must update any rule artifact that uses that BOM member. The Refactor Vocabulary window automatically lists the rules that are affected so you can update them all at the same time.

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

## 1.4. Changing the default verbalization of the duration member

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



### Questions

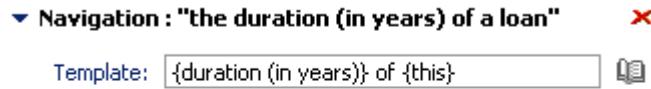
Before you change anything, which rule artifacts would be affected if you change the verbalization of the `duration` member to "duration in years"?

**Answer**

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

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

`{duration (in years)} of {this}`



Changes of this verbalization may impact business rules.

- \_\_\_ 3. Save your BOM.

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

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

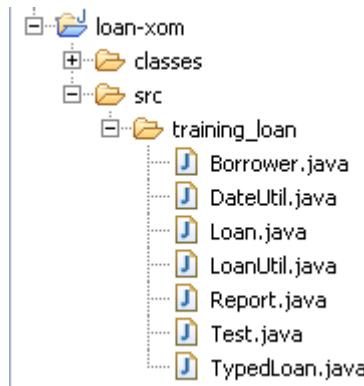
## Section 2. Maintaining consistency between the BOM and the XOM

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

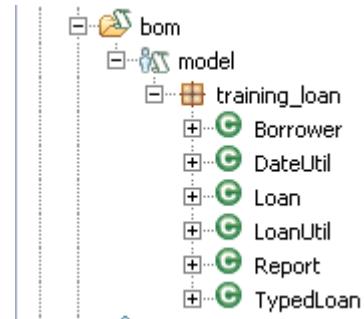
### 2.1. Adding a XOM class that is missing in the BOM

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

- \_\_\_ 1. In Rule Explorer, expand **loan-xom > src > training\_loan** and notice the list of classes.



- \_\_\_ 2. Expand the BOM in the **loan-rules** project (**loan-rules > bom > model > training\_loan**).



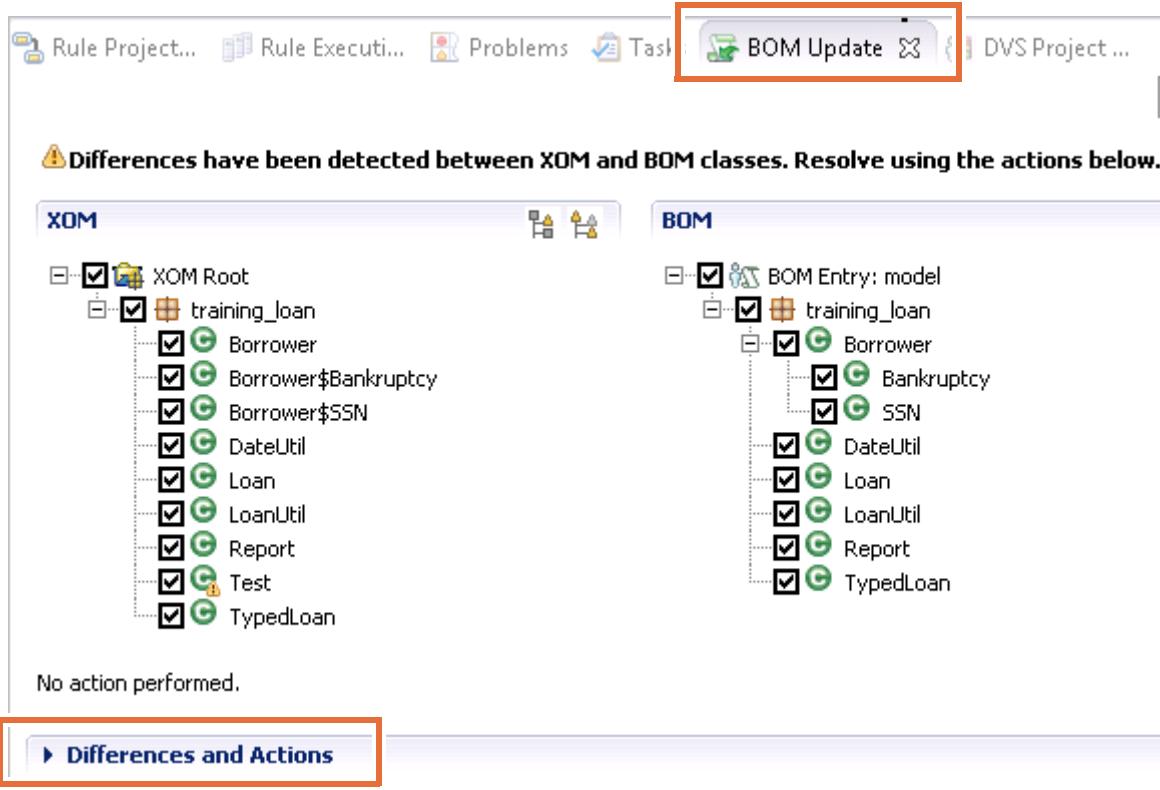
#### Questions

Do the lists match?

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

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

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

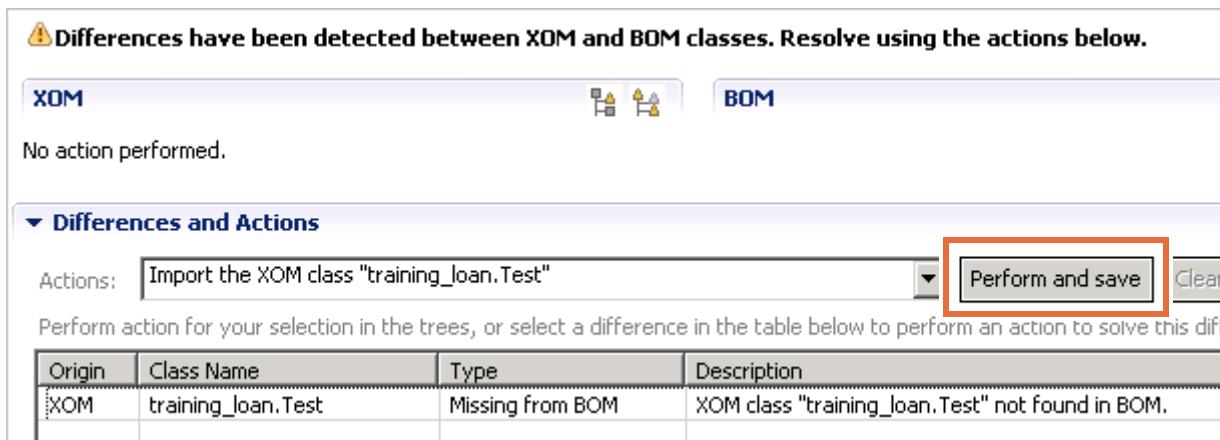


- 4. In the BOM Update view, expand **Differences and Actions**.

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

There is only one suggested action: Import the `training_loan.Test` XOM class.

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

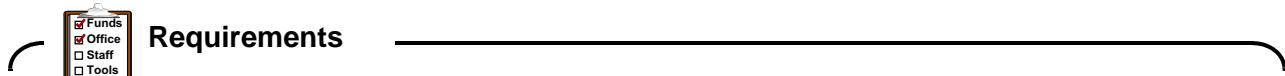


The BOM is automatically updated, and the `training_loan.Test` BOM class is created.

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

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

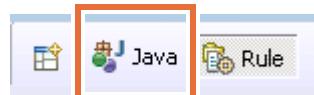
## 2.2. Adding a XOM method that is missing in the BOM



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

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

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



- \_\_\_ 2. In the Package Explorer of the Java perspective, expand `loan-xom > src > training_loan` and double-click `Borrower.java` to edit it.
- \_\_\_ 3. Add a method called `getFullName` to the `Borrower` class of the `loan-xom` project.

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

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

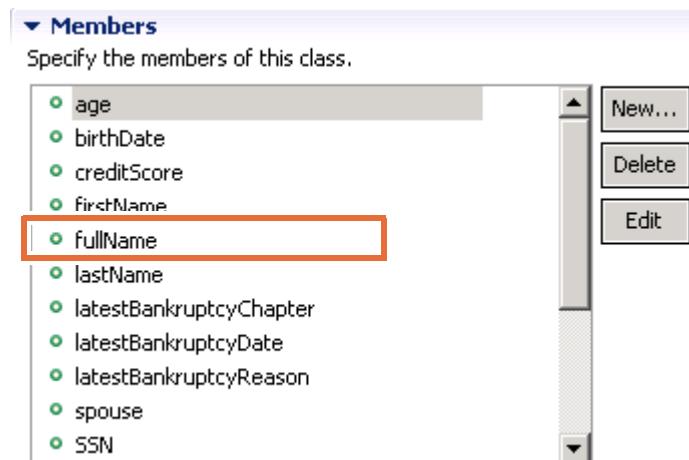
You can append this method at the end of the `Borrower` class before the final closing brace `}`.

- \_\_\_ 4. Save your work.
- \_\_\_ 5. Switch back to the Rule perspective.
- \_\_\_ 6. Go to the BOM Update view, and manage the new difference between the BOM and the XOM.
  - \_\_\_ a. Refresh the BOM Update view by clicking **Refresh** ( ) (upper-right corner).



- \_\_\_ b. Expand **Differences and Actions** and notice the new line that states that the XOM attribute `training_loan.Borrower.fullName` was not found in the `training_loan.Borrower` BOM class.
- \_\_\_ c. In the **Actions** field, select the following line and click **Perform and save**.  
Update the BOM class "training\_loan.Borrower"
- \_\_\_ d. When the Verbalize BOM window opens, click **Select All** and click **Finish** to create a default verbalization for the new member.

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



- \_\_\_ e. Refresh the BOM Update view, and notice that there are no differences between the XOM and the BOM.

### 2.3. Creating a rule to use this new attribute

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

The Intellirule editor opens.

- \_\_\_ 2. In the **Content** section of the rule editor, type the following rule:  
  
then  
print the full name of the borrower of 'the loan report';

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

The screenshot shows the IBM ODM Rule Perspective. A rule is being authored:

```

then
print the full name of the borrower of 'the loan report'

```

A dropdown menu is open under the "print" action, listing various predicates. The "full name of <a borrower>" option is highlighted with a red box. To the right of the dropdown, a tooltip-like box displays the phrase "the age".

### Information

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

- \_\_\_ 4. Save your work.

## 2.4. Deprecating BOM members

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



### Requirements

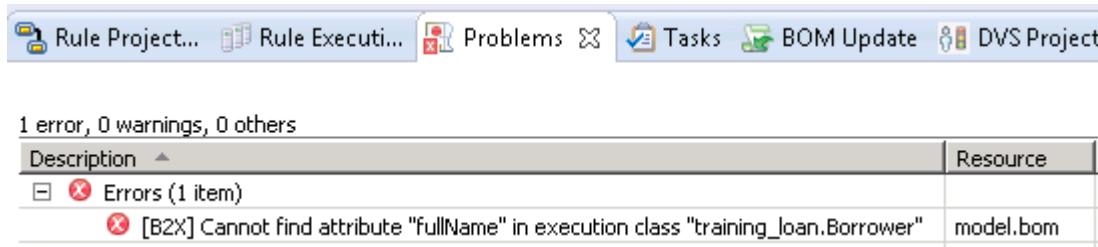
Following a subsequent meeting, business analysts now indicate that the `getFullName` method in the `Borrower` class of the `loan-xom` project is superfluous and must be removed.

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

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

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

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



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

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

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

▼ Differences and Actions

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

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

5. Select: **Deprecate the BOM attribute “`training_loan.Borrower.fullName`”** and click **Perform and save**.
6. View the deprecation result in the `Borrower.fullName` BOM member in the BOM editor.
- Expand `loan-rules > bom > model > training_loan > Borrower`.
  - Double-click the `fullName` attribute.

Member `fullName` (class: `training_loan.Borrower`)

General Information

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

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

\_\_\_ 7. View the deprecation result in the `displayFullName` rule.

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

### Action Rule: `displayFullName`

**General Information**

Name : `displayFullName`

**Documentation**

**Content**

**then**

**Warning** : 'training\_loan.Borrower: fullName' is deprecated.

\_\_\_ 8. View the result in the Problems view.

- \_\_\_ a. Notice the error:

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

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

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



### Questions

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

### Answer

Several options exist to resolve this problem:

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

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



## Requirements

The decision is made to implement the first listed solution.

- \_\_\_ 9. Delete the `fullName` attribute of the `Borrower` BOM class.
  - \_\_\_ a. Open the `Borrower` BOM class in the BOM editor.
  - \_\_\_ b. In the Members list, select **fullName** and click **Delete**.
  - \_\_\_ c. When prompted to confirm removal of this BOM member, click **Yes**.
  - \_\_\_ d. Save your work.
- \_\_\_ 10. Delete the `displayFullName` action rule.
  - \_\_\_ a. Right-click the `displayFullName` action rule.
  - \_\_\_ b. Click **Delete**.
  - \_\_\_ c. When prompted to confirm deletion, click **Yes**.
- \_\_\_ 11. Save your work.

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

## End of exercise

## Exercise review and wrap-up

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

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

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

# Exercise 5. Working with ruleflows

## What this exercise is about

This exercise teaches you how to create a ruleflow.

## What you should be able to do

After completing this exercise, you should be able to:

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

## Introduction

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

The exercise involves these tasks:

- Section 1, "Exploring a ruleflow diagram"
- Section 2, "Creating a ruleflow"

## Requirements

There are no specific requirements for this exercise.

## Section 1. Exploring a ruleflow diagram

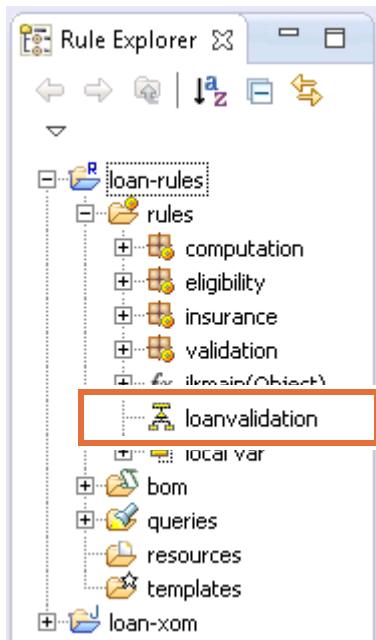
In this section, you explore the parts of an existing ruleflow in the Ruleflow editor.

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

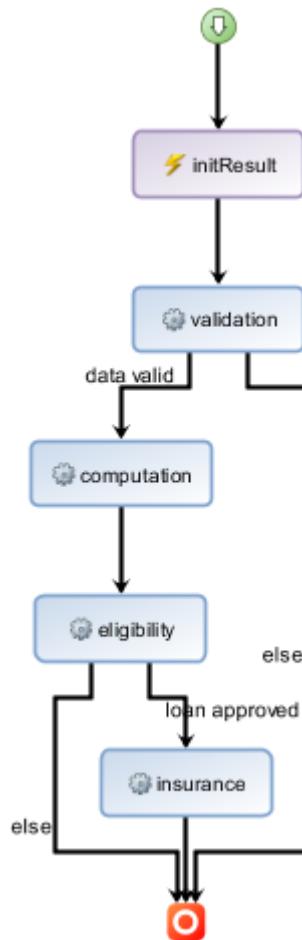
- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the Workspace Launcher window, enter the path:  
*<LabfilesDir>\workspaces\ruleflow*
- 2. Close the Welcome view and switch to the Samples Console perspective.
- 3. Select **Rule Designer > Training > Ex 05: Ruleflows > 01-start > Import projects**.
- 4. Close the Help view.

### 1.2. Exploring the ruleflow

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



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



3. Click the **Properties** tab in the lower-right corner of the perspective, and notice that the **main flow task** property is set to **true**.

| Properties     |                |
|----------------|----------------|
| Property       | Value          |
| documentation  | en-US          |
| main flow task | true           |
| name           | loanvalidation |
| tags           |                |



### Questions

Do you recognize the ruleflow tasks?

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

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

## **Answer**

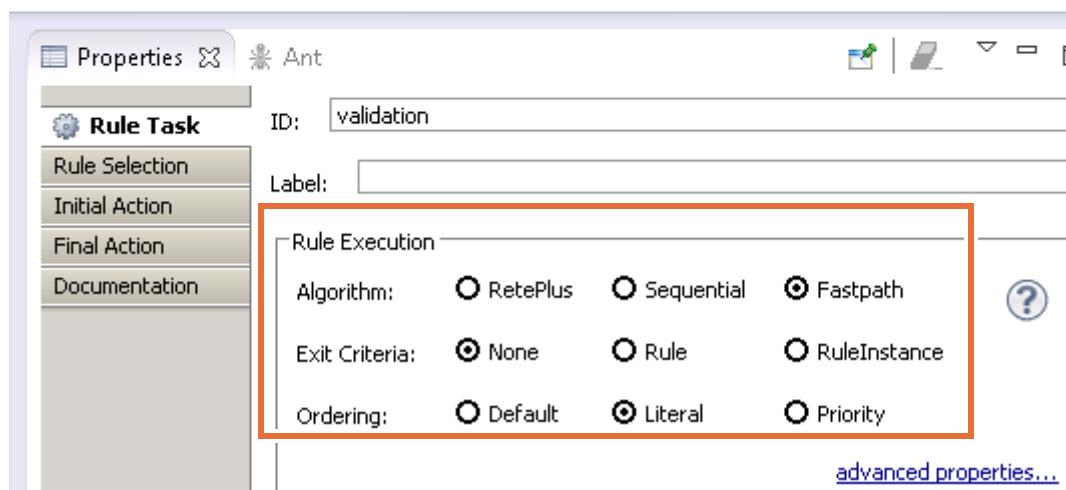
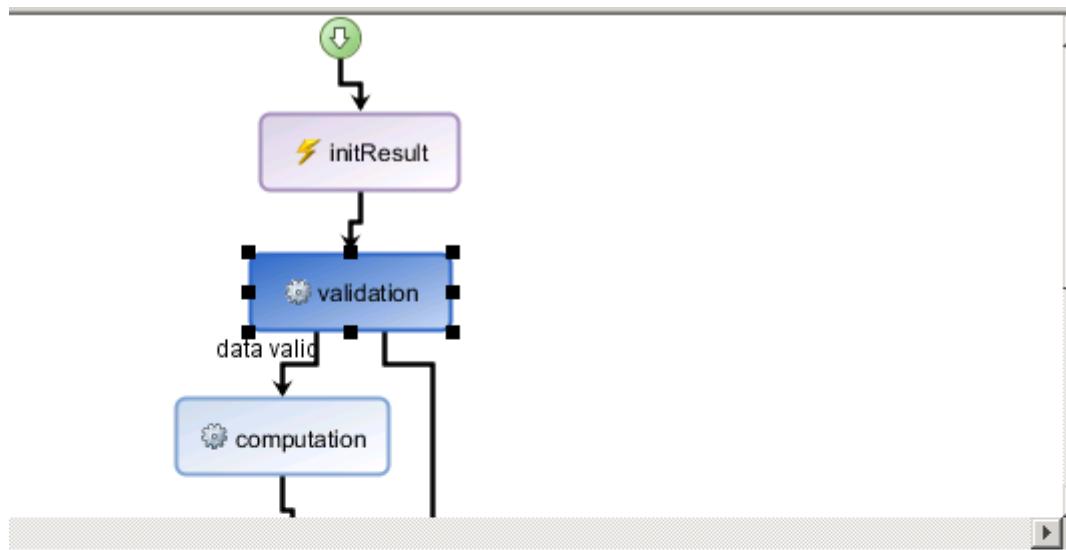
The ruleflow organizes the flow of rule execution in a ruleset to produce a decision.

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

Evaluation of each rule task produces a result or decision. For example, after executing the rules in the validation task, the result would be either *valid* or *not valid*. If the result is *valid*, the ruleflow follows the path to the computation task. Otherwise, the ruleflow goes directly to the end and the rule execution terminates.

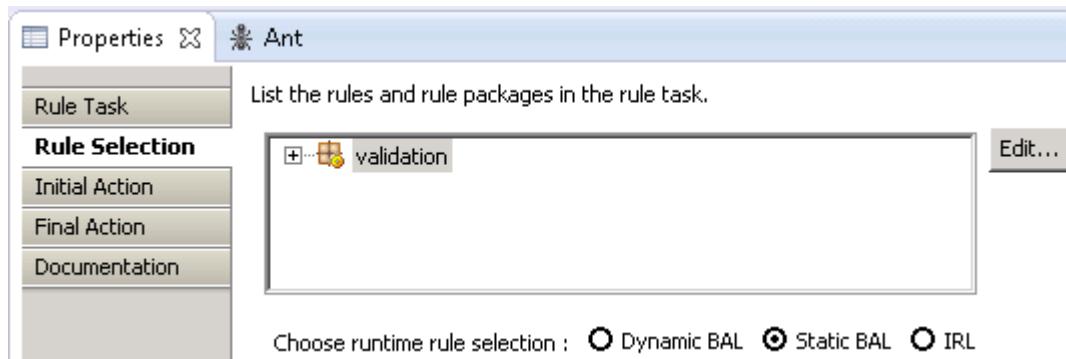
Transitions between tasks define the path through the ruleflow. You set conditions on transitions to define which path to take according to the results of evaluating a rule task.

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

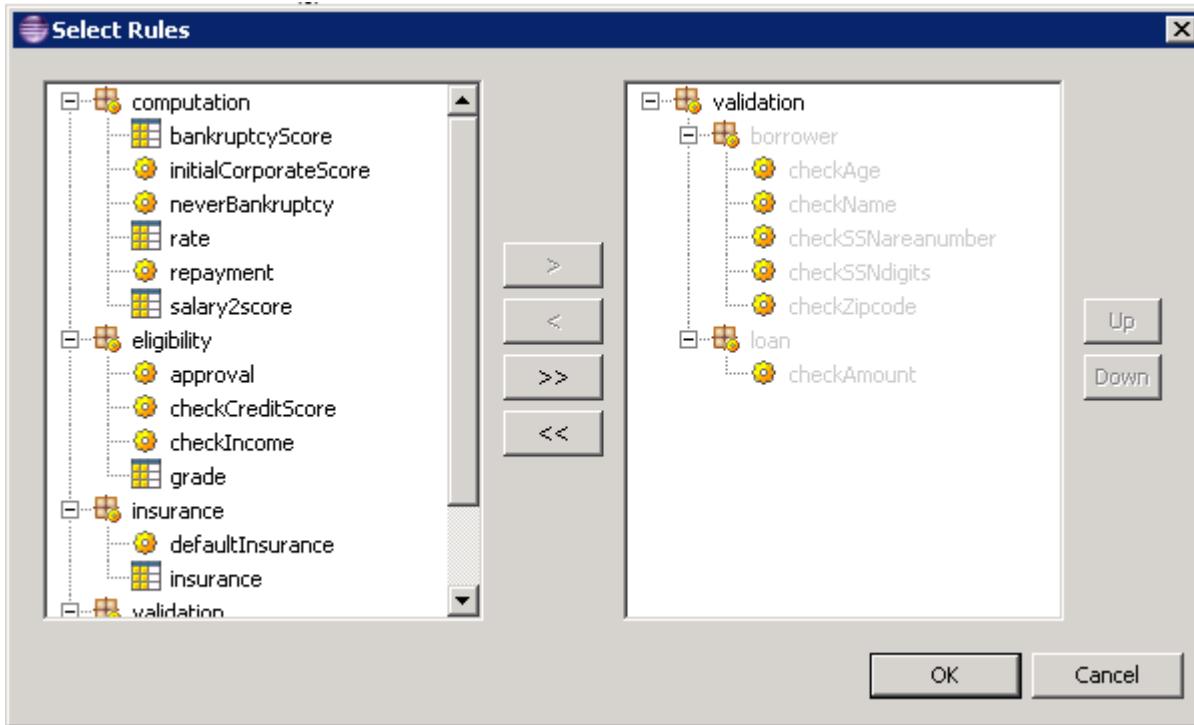


These execution properties dictate how instances of the rules in this rule task are executed.

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

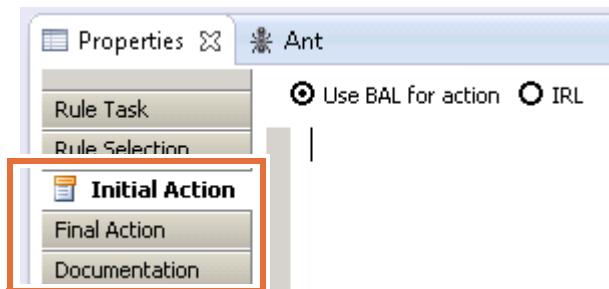


- \_\_\_ c. In the **Rule Selection** tab of the Properties view, click **Edit** to open the Select Rules window.
- \_\_\_ d. Take some time to experiment with selecting which rules to include in this task.



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

- \_\_\_ e. Click **Cancel** to close the Select Rule window.
- \_\_\_ f. Take some time to explore the **Initial Action** tab, the **Final Action** tab, and the **Documentation** tab of the validation task properties.



- In the **Initial Action** tab, you define the actions to take before this task executes. You can write these actions either in BAL (Business Action Language) or in IRL (ILOG Rule Language).
- In the **Final Action** tab, you define the actions to take after this task executes. You can write these actions either in BAL or in IRL.
- In the **Documentation** tab, you can write comments to describe this task.

- \_\_\_ 5. Create a rule task.
- Drag any rule package from the Rule Explorer into the Ruleflow editor.
  - Delete the task from the diagram when you are finished.

### 1.3. Exploring action tasks and transitions

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



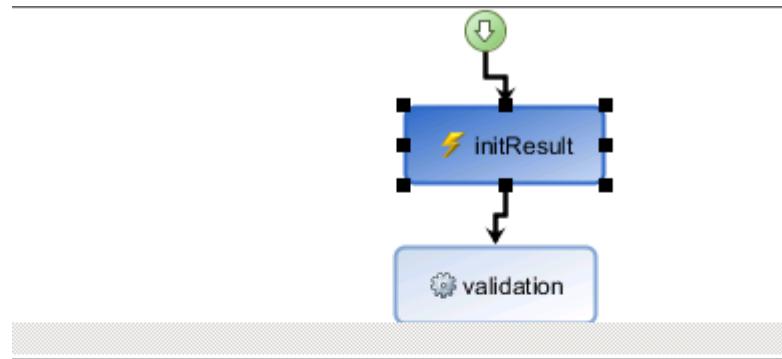
#### Questions

Can you identify an action task in the ruleflow?

#### Answer

The `initResult` task is an action task.

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



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

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

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



### Questions

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

### Answer

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

3. In the `loanvalidation` ruleflow, click the `data valid` transition to see its properties in the Properties view.



### Questions

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

### Answer

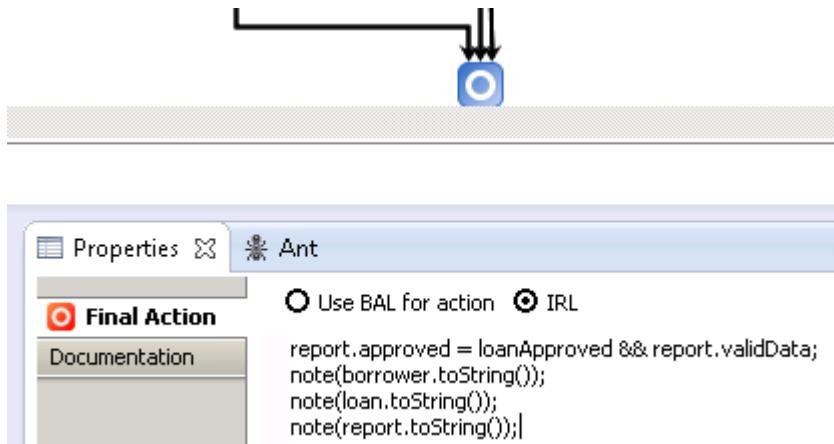
You set the label and the condition of a transition in its **Condition** tab. The condition might be written in either Business Action Language (BAL) or ILOG Rule Language (IRL). You learn what BAL and IRL are later in this course.

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

This transition condition is written in BAL.

4. Take some time to explore the relationship between the `validData` attribute of the `training_loan.Report` class, its verbalization that you can see in the BOM of the `loan-rules` project, and the way this transition condition is expressed.

5. In the loanvalidation ruleflow, click the end node to view the **Final Action** in the Properties view.



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

This Final Action is written in IRL.

## 1.4. Using ruleset parameters and variables in rules

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

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



### Questions

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

### Answer

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

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

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

3. Close the rule editor windows for these rules.

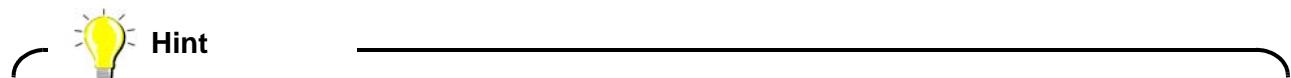
## Section 2. Creating a ruleflow

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

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

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

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

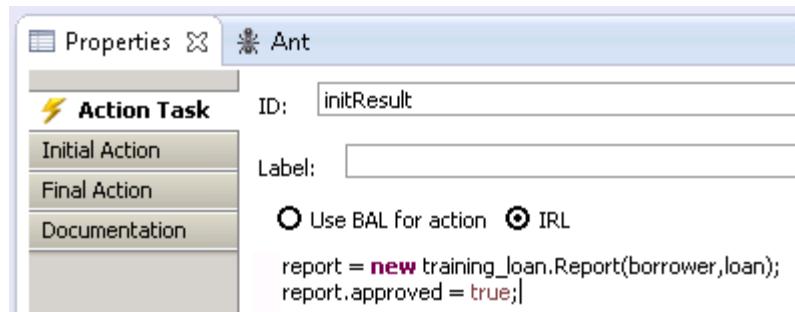


Keep the `loanvalidation` ruleflow open while you work on the `loanvalidation_2` ruleflow in a second Ruleflow editor. By switching between windows and tabs, you can see what you must update when you work through the next steps.

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

\_\_ d. In the **Action Task** tab:

- In the **ID** field, enter: `initResult`
- Select the **IRL** option.
- Enter the following IRL function code:  
`report = new training_loan.Report(borrower,loan);  
report.approved = true;`



\_\_ e. Save your work.

## Troubleshooting

You can ignore the errors that you see.

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

- Ordering: **Priority**
  - **insurance**
    - Algorithm: **RetePlus**
    - Exit Criteria: **None**
    - Ordering: **Default**
- \_\_\_ 7. Create the transitions between the tasks.
- \_\_\_ a. In the Ruleflow editor palette, click  **Create a transition**. You can now create multiple transitions in the diagram.
  - \_\_\_ b. Click the start node and then click the `initResult` action task.
  - \_\_\_ c. Click the `initResult` action task, and then click the `validation` rule task.
  - \_\_\_ d. Click the `validation` rule task, and then click the `computation` rule task.
  - \_\_\_ e. Click the `computation` rule task, and then click the `eligibility` rule task.
  - \_\_\_ f. Click the `eligibility` rule task, and then click the `insurance` rule task.
  - \_\_\_ g. Click the `insurance` rule task, and then click the end node.
  - \_\_\_ h. Create another transition between the `validation` rule task and the end node.



**Hint**

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

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



## Questions

Are you missing any tasks or transitions?

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

## Answer

To complete the `loanvalidation_2` diagram so that it matches the `loanvalidation` diagram, you must complete the following tasks.

- \_\_\_ 1. Add a transition between the `eligibility` rule task and the end node.
- \_\_\_ 2. Add a label and a condition to the transition between the `eligibility` and `insurance` tasks.
  - \_\_\_ a. Select the transition between the `eligibility` and `insurance` rule tasks.
  - \_\_\_ b. Click the **Condition** tab.
  - \_\_\_ c. In the **Label** field, enter: `loan approved`
  - \_\_\_ d. Select **Use BAL for transition condition**.
  - \_\_\_ e. In the condition editor, enter: 'the loan is approved'
- \_\_\_ 3. Save your work.

## End of exercise

## Exercise review and wrap-up

This exercise looked at how to design a ruleflow.

To see a possible solution to this exercise, switch to a new workspace and import the project **Ex 05: Ruleflows > 02-answer**.



# Exercise 6. Exploring action rules

## What this exercise is about

This exercise teaches you how action rules are written.

## What you should be able to do

After completing this exercise, you should be able to:

- Identify the parts of an action rule
- Explain the difference between using automatic variables or rule variables

## Introduction

To learn how to author action rules, you review the Trucks and Drivers rule project, which is a complete project with rules that use various BAL constructs. You explore and run these rules to understand rule behavior during rule execution.

The exercise involves these tasks:

- Section 1, "Exploring rule structure"
- Section 2, "Exploring rule behavior"
- Section 3, "Working with automatic variables"

## Requirements

There are no specific requirements for this exercise.

## Section 1. Exploring rule structure

In this section, you explore the behavior of rules in the Trucks and Drivers rule project. This rule project is designed to demonstrate specific BAL constructs. You can open each rule, review the constructs that are used, and then view the results that are produced after rule execution.

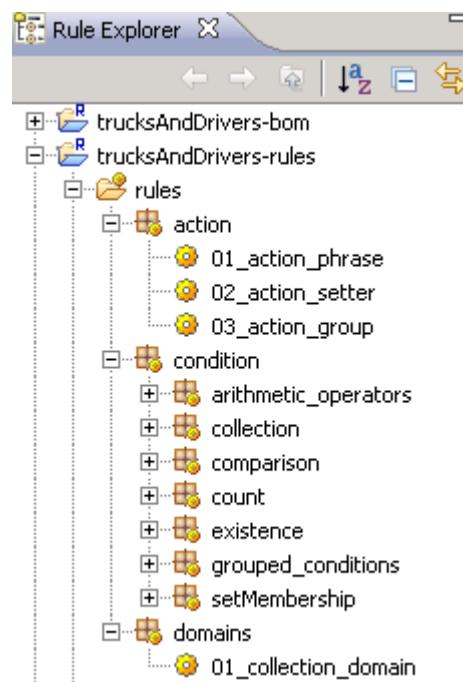
### 1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the **Workspace Launcher** window, enter the path:  
`<LabfilesDir>\workspaces\explore_rules`
- 2. Close the Welcome view and switch to the Sample Console perspective.
- 3. Select **Rule Designer > Training > Ex 06: Exploring rules > 01-start > Import projects**.
- 4. Close the Help view.

### 1.2. Exploring the rules

The action rules to explore are provided in different rule packages.

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



- 2. Notice the **condition** package, which includes the following subpackages to illustrate these BAL constructs in condition statements:
  - **Arithmetic operators:** Action rules in this rule package show how to use mathematical operators to carry out calculations in the condition of an action rule.

- **Collections:** Action rules in this rule package show how to define a variable to retrieve a list of objects (`java.util.Collection`).
- **Comparisons:** Action rules in this rule package show how to use BAL operators to compare the values of strings or the number of objects in working memory.
- **Count tests:** Action rules in this rule package show how to use operators that count the number of occurrences of an object.
- **Existence tests:** Action rules in this rule package show how to use BAL operators to test the existence of objects with certain members in working memory.  
Conditions are qualified with the `where` keyword.
- **Grouped conditions:** Action rules in this rule package show how to use the `all`, `any`, and `none` BAL constructs to combine conditions, as an alternative to logical operators (and, it is not true that, or) between condition statements.
- **Tests for membership in a set:** Action rules in this rule package show how to write conditions that test whether a member of an object in working memory is part of a set (`java.util.Collection`).

- 3. In Rule Explorer, open the BOM by expanding **trucksAndDrivers-bom > bom > model**.
- 4. Expand **drivers** and review the objects and vocabulary.



### Questions

Consider these questions as you review the **rules** and **vocabulary** in this rule project:

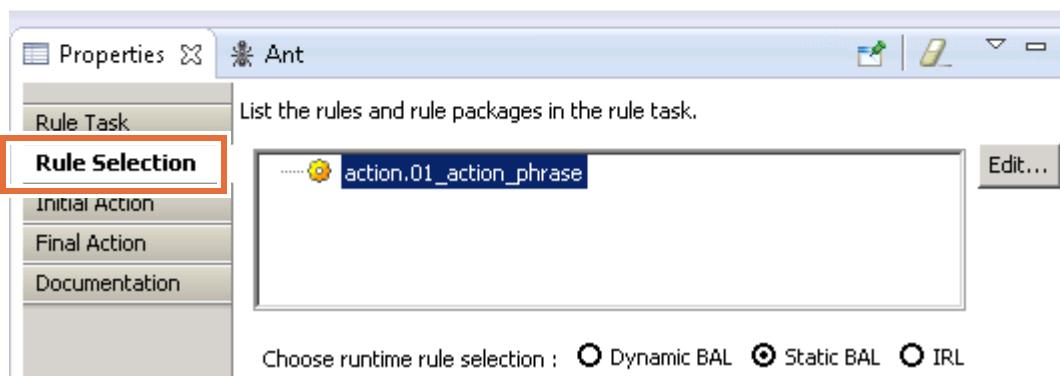
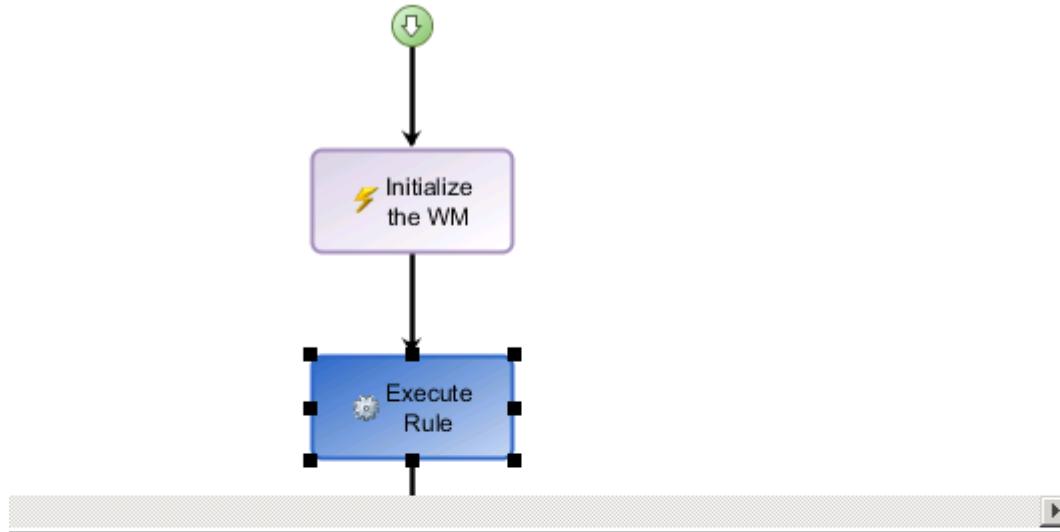
- Can you identify the objects on which the action rules work?
- Are there any objects for which you cannot identify the origin?
- Based on your experience with Java programming, can you determine what the terms *definitions*, *conditions*, *actions*, and *variables* mean in relation with the design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

## Section 2. Exploring rule behavior

In this part of the exercise, you execute at least one action rule from each rule package in the trucksAndDrivers-rules project.

### To execute a rule:

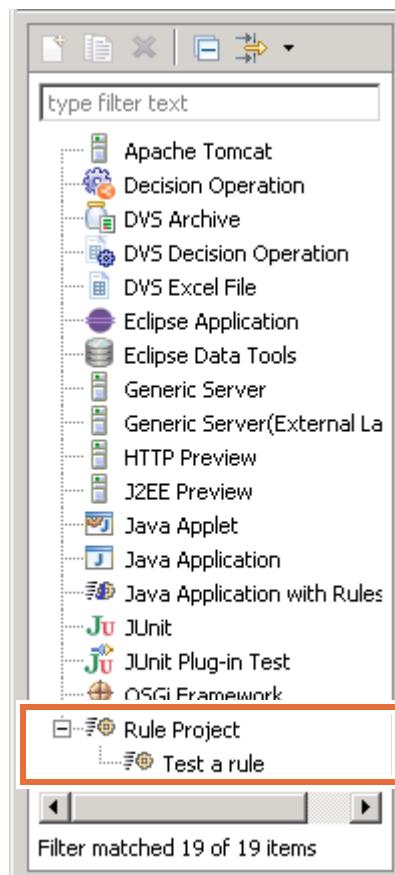
- \_\_\_ 1. In Rule Explorer, expand **trucksAndDrivers-tests > rules**.
- \_\_\_ 2. Double-click the **testFlow** ruleflow to open it in the Ruleflow editor.
- \_\_\_ 3. In the ruleflow diagram, select the **Execute Rule** rule task and open the Properties view.
- \_\_\_ 4. In the Properties view, click the **Rule Selection** tab, and click **Edit** to choose a rule to run.



You can choose any rule to start with, and repeat these steps to try other rules.

- \_\_\_ 5. From the **Run** menu, click **Run Configurations** to set up rule execution.

6. In the Run Configurations window, expand **Rule Project** and select **Test a rule**.



7. In the Test a rule launch configuration, click **Run** to start the execution of the trucksAndDrivers-tests ruleflow.  
 8. At the “Save and Launch” window, click **OK**.

The console opens and you can see that the rule executed correctly by looking at the message, which should match the action statement in the rule.

```
<terminated> Test a rule [Rule Project] C:\Program Files\IBM\ODM87\ODM\jdk\bin\javaw.exe (Dec 23, 2014 9:09:44 AM)
The number of deliveries in the working memory is 10
the truck overloaded TRUCK-T2000 has its smallest delivery removed
```

The screenshot shows the Eclipse IDE interface with the 'Console' tab active. The console output window displays the results of running the 'Test a rule' configuration. It shows the termination message, the current number of deliveries in memory (10), and a specific message indicating a delivery was removed from an overloaded truck named 'TRUCK-T2000'.

## Section 3. Working with automatic variables

In this part of the exercise, you create automatic variables in your BOM and use them in your rule artifacts. By doing so, you learn the differences in using an automatic variable instead of a rule variable in action rules.

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

**Class Verbalization**

Generate automatic variable

Term: `truck` [Edit term.](#)

**i** the truck, a truck, the trucks....

A variable that is called `the truck` is automatically made available in the rule editors. You can use the automatic variable to author rules based on objects in the working memory that are instances of the `Truck` BOM class.

- 3. Save the BOM (Ctrl+S) and wait for the workspace to rebuild.
- An error message appears on the rule project.
- 4. In the Problems view, double-click the error to open the problem rule.

**Content**

```

definitions
 x set !truck! to a truck ;
if
 the model of truck is one of { the F150 , the MACK TRUCK }
then
 display the message "The truck model of " + the serial number of

```

Intellirule IRL | 01\_setMembership\_isOneOf.brl

Rule Proj... Rule Exec... Problems Tasks BOM Up... DVS Proj... □ □

1 error, 0 warnings, 0 others

| Description                                                                            | Resource     | Path               | Location |
|----------------------------------------------------------------------------------------|--------------|--------------------|----------|
| <input checked="" type="checkbox"/> Errors (1 item)                                    | 01_SetMem... | /trucksAndDrive... | line 2   |
| <input checked="" type="checkbox"/> An automatic variable 'truck' is already declared. | 01_SetMem... | /trucksAndDrive... | line 2   |

You can also open the rule directly in the `trucksAndDrivers-rules` project by expanding **rules > condition > setMembership** and double-clicking `01_SetMembership_isOneOf`.

**Questions**

Why does this rule no longer compile?

**Answer**

The `01_setMembership_isOneOf` action rule cannot compile because it declares a variable that is called `truck`, which conflicts with the name of the automatic variable that you created.

**Information**

In the following steps, you gain hands-on experience with the rule editor by trying to intuitively use its authoring functions.

**Questions**

How can you rewrite the rule so that it compiles?

**Answer**

- To avoid conflicting variables, you can delete the definitions part of the `01_setMembership_isOneOf` rule.
- As indicated in the BOM, the automatic variable for the `Truck` class is verbalized as `the truck`, meaning that everywhere you previously had `truck` only, you must now use: `the truck`

— 5. Delete the **definitions** statement.

- \_\_\_ 6. Modify the `01_setMembership_isOneOf` action rule to use the automatic variable as shown here:

```
if
 the model of the truck is one of { the F150 , the MACK TRUCK }
then
 display the message "The truck model of " + the serial number of the truck
 + " is F150 or MACK_TRUCK";
```

 **Hint**

You can click the error icons on the sides of the rule editor to see how to fix the error. Or you can right-click `truck`, click **Quick Fix** for a solution, and double-click the solution that is proposed.

You can also double-click `truck` and select the correct variable from the list, but if you use this option, be careful not to overwrite any other part of the rule statement.

- \_\_\_ 7. Save the rule.
- \_\_\_ 8. In the `trucksAndDrivers-rules` project, expand **rules > action** and double-click the `01_action_phrase` rule to open it.



**Questions**

Can you figure out how you can simplify this rule with the new automatic variable?

Do not change it yet.

**Answer**

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

9. To use the automatic variable, delete the definition statement and modify the `01_action_phrase` rule in the `action` rule package to match this content:
- ```
if
  the current load of the truck is more than the capacity of the model of the
  truck
then
  remove the smallest delivery of the truck;
  display the message "the truck overloaded " + the serial number of the
  truck + " has its smallest delivery removed";
```
10. Save the `01_action_phrase` action rule and execute it again to verify that the changes did not change the execution results.
11. You can rerun the `Test a rule` run configuration by clicking the **Run** icon on the toolbar, and clicking **Test a rule** from the list.



The traces in the Console view are:

the truck overloaded TRUCK-T2000 has its smallest delivery removed

These traces should match the traces you had in Section 2, "Exploring rule behavior," on page 6-4.



Optional

Optionally, you can make similar changes in the other action rules in the `trucksAndDrivers-rules` project.

- Delete all definitions of the rule variable called '`'the truck'`' (with single quotation marks).
- Where you had the rule variable '`'the truck'`' (with single quotation marks), you can use the automatic variable `the truck` instead (without quotation marks).

End of exercise

Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 06: Exploring rules > 02-answer**.

The first part of the exercise looked at how the action rules are structured and demonstrated their behavior during rule execution. You also saw the difference between using automatic variables or rule variables.

Exercise 7. Authoring action rules

What this exercise is about

This exercise teaches you how to author action rules.

What you should be able to do

After completing this exercise, you should be able to:

- Use the Intellirule editor and Guided editor to author action rules
- Use rule variables, automatic variables, and ruleset parameters in rule statements

Introduction

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

The exercise includes these tasks:

- Section 1, "Authoring action rules in the Intellirule editor"
- Section 2, "Authoring actions rules in the Guided editor"
- Section 3, "Authoring rules with ruleset parameters"
- Section 4, "Creating an action rule template"
- Section 5, "Authoring action rules from your action rule template"

Requirements

You should complete these exercises before proceeding:

- Exercise 2, "Setting up rule projects"
- Exercise 6, "Exploring action rules"

Section 1. Authoring action rules in the Intellirule editor

In this exercise, you author rules in the main Rule Designer rule editors:

- Intellirule editor
- Guided editor



Note

You can also edit the action rules as pure text, in the Eclipse Text editor.

1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace:
 - a. From the **File** menu, click **Switch Workspace > Other**.
 - b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\author_rules`
- 2. Close the Welcome view and switch to the Samples Console perspective.
- 3. Select **Rule Designer > Training > Ex 07: Authoring rules > 01-start > Import projects**.
- 4. Close the Help view.

The workspace includes a predefined `loan-xom` XOM project and a separate `loan-bom` project.

As you saw during Exercise 6, "Exploring action rules", the BOM can be maintained in a separate project for modularity.

- 5. In Rule Designer, create a standard rule project named: `loan-rules`
 - a. Right-click the Rule Explorer and click **New > Rule Project**.
 - b. In the **New Rule Project > Classic Rule Projects** section, select **Standard Rule Project** and click **Next**.
 - c. In the **Project name** field, enter `loan-rules` and click **Next**.
 - d. In the Rule Project References page, select `loan-bom`, and click **Next**.

This reference page indicates that the `loan-rules` project references the `loan-bom` project. The `loan-rules` project uses the vocabulary from this BOM project.
 - e. In the Rule Project XOM Settings page, make sure that `loan-xom` is not selected, and click **Finish**.
- 6. In the `loan-rules` project, use the **Add rule package** link in the Rule Project Map to create a rule package named: `loan`

1.2. Authoring rules with the Intellirule editor



Requirements

Scenario:

A loan company must implement its loan policies, which evolve regularly, and must provide a loan application to its agents.

First, the application must verify input data from a form. Then, it must check customer eligibility, which is based on personal profile, score, and the type of loan requested. The application can also compute some insurance rates, as a function of the computed score.

Use the Intellirule editor to author some of the action rules that implement the Loan scenario.

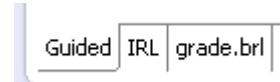
- 1. In the `loan` rule package, add a rule that is called: `grade`
 - a. Right-click the `loan` rule package, and click **New > Action Rule**.
The New Action Rule window opens.
 - b. In the **Name** field, enter: `grade`
 - c. Click **Finish** to close the New Action Rule window.
The empty rule opens in the Intellirule editor.



Note

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

- **Guided** for the Guided editor



- **Intellirule** for the Intellirule editor

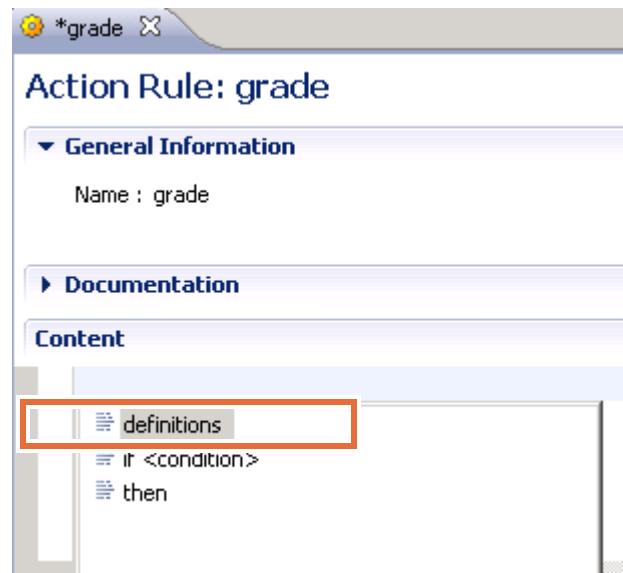


If the Intellirule editor is not the default editor that opens when you create an action rule, right-click this action rule in the Rule Explorer and click **Open With > Intellirule Editor**.

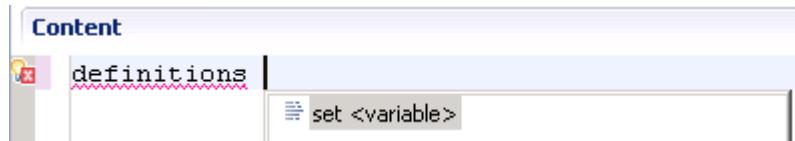
- 2. Use the Intellirule editor to start authoring the `grade` rule.

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

- ___ a. Press Ctrl+Spacebar to open the Content Assist menu and double-click **definitions**.



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

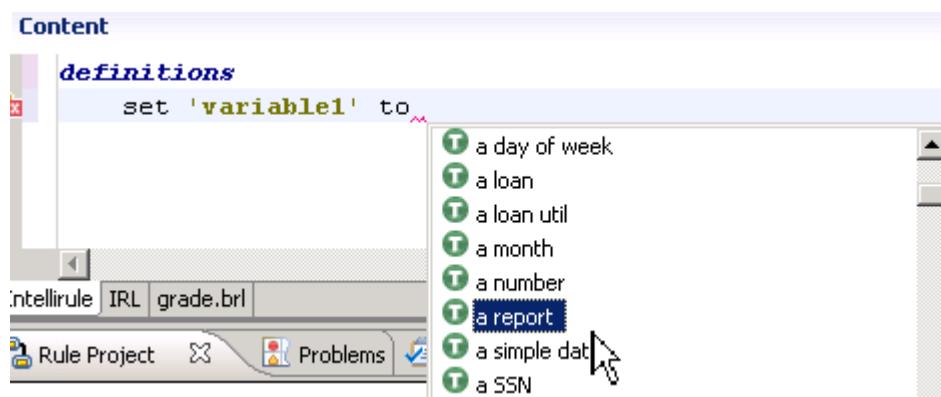


- ___ c. Put the cursor in the space after `set 'variable1'` to and press Ctrl+Spacebar to open Content Assist.

- ___ d. Double-click **to <binding type>**.

The Content Assist menu now prompts you with a vocabulary list.

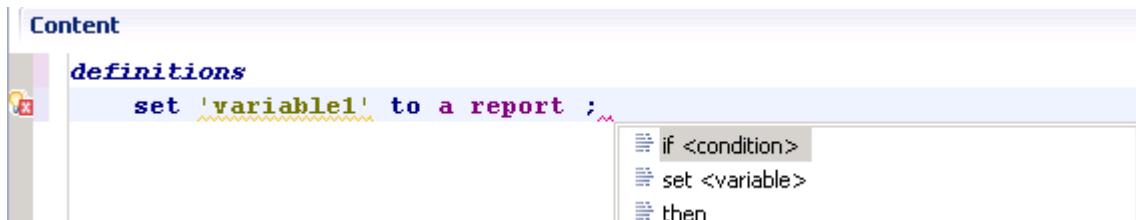
- ___ e. You want the variable to be of type Report, so scroll through the vocabulary list to find and double-click **a report**.



- ___ f. Finally, you are prompted to complete the definition statement for your variable with a semicolon (;), which is required in the Intellirule editor, or to add another clause to your statement. For this rule, the variable definition is complete, so you can double-click the semicolon.



- ___ g. When prompted, select **if <condition>**.



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

- ___ h. Before you continue with the condition statement, go back to the variable definition and type over 'variable1' to change it to: 'the report'



Important

Make sure that you keep your variable name enclosed with single quotation marks (').

- ___ 3. Now that you see how Content Assist can guide you to build the rule, complete the rule statement to match this content:

```
definitions
    set 'the report' to a report;
if
    the amount of the loan of 'the report' is more than 1000000
then
    set the grade of 'the report' to "GOLD";
```

- ___ 4. Save your work and close the editing window.

- ___ 5. In the loan rule package, create a second action rule called: invalid corporate score
This rule must refuse a loan application when the corporate score is less than 1000.

- ___ 6. For this rule, instead of using the Content Assist menu, type the rule content directly into the editor to match this content:

```
definitions
    set 'the report' to a report;
if
    the corporate score in 'the report' is less than 1000
then
    in 'the report', refuse the loan with the message "Corporate score less
    than 1000";
```



Hint

You can copy the rule content from the `author.txt` file and paste directly into the Intellirule editor. The `author.txt` file is in the `<LabfilesDir>\code` directory.



Note

Notice that you must pay attention to punctuation to avoid errors in the rule statement.

- ___ 7. Save the rule (Ctrl+S).

- ___ 8. Create another action rule in the `loan` rule package called: `too big loan`

This rule should refuse a loan application when the loan is more than 10,000,000.



Questions

How do you write this action rule in the Intellirule editor?

Answer

The body of the action rule that is called `too big loan` must be:

```
definitions
    set 'the report' to a report;
if
    the amount of the loan of 'the report' is more than 10000000
then
    in 'the report', refuse the loan with the message "The loan amount is too
    big";
```

**Hint**

You can find the text of the `too big loan` rule in the `author.txt` file in the `<LabfilesDir>\code` directory.

- ___ 9. Save your work and close the editing windows.

Section 2. Authoring actions rules in the Guided editor

Use the Guided editor to author the remaining action rules to fully implement the Loan scenario. As in the previous section, you author these action rules by using rule variables and automatic variables.

- ___ 1. Create an action rule in the loan rule package, named: too long loan duration
The rule must refuse a loan application when the loan duration is strictly greater than 200.
By default, the rule opens in the Intellirule editor.
- ___ 2. Close the editing window for the rule, and then in Rule Explorer, right-click the rule and click **Open With > Guided Editor**.



- ___ 3. Define the rule content to match this text:

```
definitions
    set the report to a report
if
    the duration (in years) of the loan of the report is more than 200
then
    in the report, refuse the loan with the message The loan duration is too
    long
```

- ___ a. Start with the variable definition statement:
 - Click **definitions**.
 - Click **variable1** and set the variable name by typing: the report
 - Click **select a choice** and click **<an object>** from the list.
 - Click **select an object** and select **a report** from the list.
- ___ b. Continue editing the condition and action statements to complete the rule.



Hint

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



Questions

What differences did you identify between the Intellirule editor and the Guided editor?

Answer

With the Intellirule editor, you must use punctuation, such as semicolons (;) at the end of definitions and actions, single quotation marks (') around variables, and double quotation marks ("") around String constants. The Guided editor does not require these extra characters.

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

Section 3. Authoring rules with ruleset parameters

For this part of the exercise, you use a ruleset parameter instead of the local rule variable.



Questions

What direction should the report parameter be: IN, OUT, IN_OUT? What default values should it have, if any?



Questions

Which members of the Report class might be useful for recording output results from the rule engine?

Answer

Here are some examples:

- messages
- insuranceRequired
- insuranceRate
- validData

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

- ___ 3. In the `loan-bom` project, create an OUT ruleset parameter that is named `report` of type `training_loan.Report`, and verbalize it as: the loan report
 - ___ a. In the Rule Explorer, right-click the `loan-rules` project and click **Properties**.
 - ___ b. In the Properties window, select **Ruleset Parameters**.

- ___ c. In the Ruleset Properties pane, click **Add** and define the name, type, direction, and verbalization of your new parameter.

| Name | Type | Direction | Default Value | Verbalization |
|--------|----------------------|-----------|---------------|-----------------|
| report | training_loan.Report | OUT | | the loan report |



Reminder

For detailed steps about defining ruleset parameters, see Section 3, "Creating the ruleset parameters" in Exercise 2, "Setting up rule projects".

- ___ d. Click **OK** to save your work and close the Properties window.
- ___ 4. Rewrite the `grade` rule to use the `report` ruleset parameter.

You can use your existing local rule variable to manipulate the `report` ruleset parameter (which shows up in the vocabulary list as "the loan report"). You can also delete the local rule variable and use the ruleset parameter directly.

```

Content

definitions
  set 'the report' to a report ;
  if
    the corporate score in 'the report' is less than 1000
  then
    in 'the report', refuse the loan with the message "Corporate score less than 1000"

```

- ___ a. In Rule Explorer, expand `loan-rules > rules > loan` and double-click the `grade` rule.
- ___ b. Delete the **definitions** part of the rule, which includes these lines:

```

definitions
  set 'the report' to a report

```

You should now see errors in the rule.

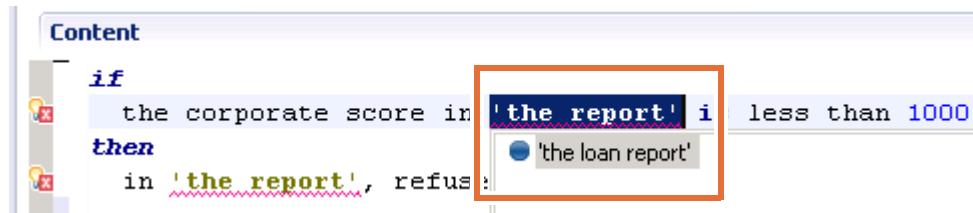
```

Content

if
  the corporate score in 'the report' is less than 1000
then
  in 'the report', refuse the loan with the message "Corporate score less than 1000"

```

- ___ c. In the condition and action statements, click ‘**the report**’ and select the new ruleset parameter.



The errors should now be gone.

- ___ 5. Save your work (Ctrl+Shift+S).
- ___ 6. When you are finished editing rules, close the rule editors.
- ___ 7. In the Problems view, verify that there are no messages.

Section 4. Creating an action rule template

The `loan-rules` project contains a series of action rules that share a similar structure. These action rules all check some data, accept or reject these data based on certain conditions, and generate a message that explains the rejection reason.

An efficient way to author all the similar rules of this project is to create an action rule template, and apply it to the creation of the rules.

In this section, you create the required business template.

To create the template:

- 1. In the `loan-rules` project, create an action rule template that is called: `checkData`
 - a. Right-click the `loan-rules` project, and click **New > Action Rule Template**.
The New Action Rule Template wizard opens.
 - b. Keep the default value in the **Template folder** field:
`/loan-rules/templates`
 - c. Leave the **Folder** field empty.
 - d. Enter `checkData` in the **Name** field of your action rule template.
 - e. In the **Type** field, leave the default value as **ActionRule**.
 - f. Click **Finish**.

The Guided editor opens for you to edit the `checkData` action rule template, which is visible now in the `loan-rules/templates` folder.

- 2. With the Guided editor, edit the `checkData` action rule template so that the action statement matches this content:

```
if
<select a condition>
then
    in the loan report, reject the data with the message <enter a string>
```

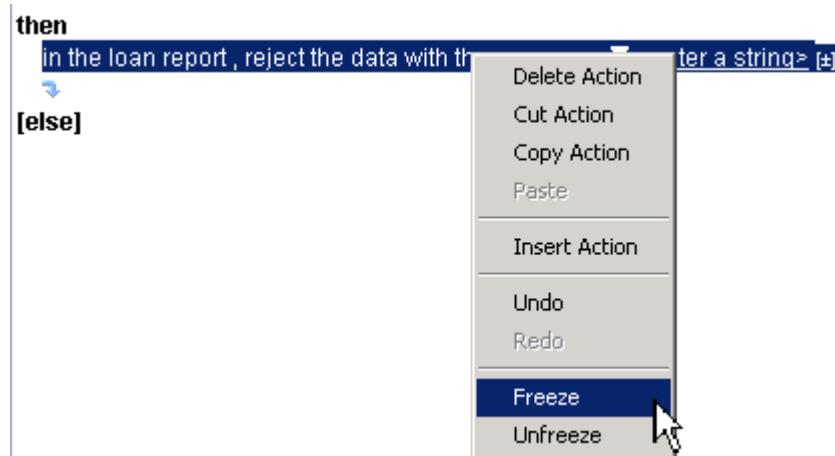


Important

Do not change the condition statement. This template is for the action statement only.

- 3. Follow these steps to freeze the actions **except** the `[±]` sign after `<enter a string>`, which you must leave unfrozen.
 - a. Select the rule action statement with your mouse.

- __ b. Right-click the selected statement and click **Freeze** to freeze all actions.



- __ c. Right-click the plus-minus sign (\pm) at the end of the actions and click **Unfreeze**.

If this plus-minus sign (\pm) is frozen, the message can contain only one `String` element. However, in many cases you must define a message by using multiple `String` elements. You see an example with the `check SSN` action rule (in Section 5, "Authoring action rules from your action rule template," on page 7-15). You must unfreeze that sign before you proceed.

- __ 4. Save the `checkData` action rule template.

The frozen parts of the action rule template are gray. Notice that the placeholders `<select a condition>` and `<enter a string>` are still shown in blue, which indicates that you can still edit them.



Note

You can also freeze everything except the placeholders by right-clicking anywhere in the Guided editor and clicking **Freeze All**. Placeholders can still be editable (they are shown in blue).

To unfreeze everything, right-click anywhere in the Guided editor and click **Unfreeze All**.

- __ 5. Close the template.

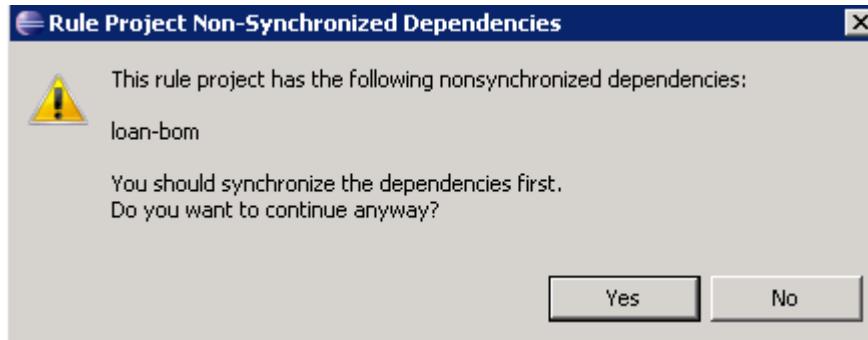
Section 5. Authoring action rules from your action rule template

In this section, you see how the rule authors work with the templates you provide. This section illustrates how you can support rule authors during project development.

5.1. Publishing the BOM and the rules to Decision Center

- ___ 1. Make sure that the sample server is started.
- ___ 2. Publish the loan-rules project:
 - ___ a. In the Rule Explorer, right-click the `loan-rules` project and click **Decision Center > Connect**.
 - The Decision Center configuration wizard opens.
 - ___ b. Enter the connection entries:
 - **URL:** `http://localhost:9080/teamserver`
 - **User name:** `rtsAdmin`
 - **Password:** `rtsAdmin`
 - ___ c. Click **Connect**.
 - ___ d. Select **Create a new project on Decision Center** and click **Finish**.

A window warns you that you should synchronize the BOM first and asks if you want to continue.



- ___ e. Click **No**.



Important

When you have multiple projects that depend on each other, make sure that you publish all of them to Decision Center. Because the BOM is stored in a project that is separate from the rules, you must publish the BOM first.

- ___ 3. Repeat Step 2 on page 7-15 to publish the `loan-bom` project to Decision Center.
 - ___ a. In the “Project configuration” section, make sure that **Create a new project on Decision Center** is selected because the `loan-bom` project does not exist yet on Decision Center.
 - ___ b. Click **Finish**.
 - ___ c. When synchronization completes, no changes are found, so you can click **OK** to close the window.
 - ___ d. Click **No** when you are prompted to switch to the Team Synchronizing perspective. You do not have to open the Team Synchronizing perspective because it is empty.
- ___ 4. Repeat Step 2 on page 7-15 to again publish the `loan-rules` project to Decision Center. This time, you do not see any warnings about dependencies because the BOM is already available in Decision Center.

5.2. Authoring the rule in Decision Center

- ___ 1. Open the Decision Center Enterprise console at this URL:
`http://localhost:9080/teamserver`
If the console was already open, it might be necessary to refresh your browser and sign in again to see the newly published `loan-rules` and `loan-bom` projects.
- ___ 2. Sign in with `rtsAdmin` for the user name and password:
- ___ 3. On the **Home** tab, select **Work on a project** and choose **loan-rules** as the project to use,
- ___ 4. Click the **Compose** tab.

5. On the **Compose** tab, in the **Start from a template** section, you see your newly created template.

Start from a type

- Action Rule
- Decision Table
- Decision Tree
- Folder
- Smart Folder
- Template
- Variable Set
- Function
- Technical Rule
- Resource
- Simulation (Enterprise Console)
- Test Suite (Enterprise Console)

Start from a template

checkData

Rule Type

Content

if
<condition>
then
in 'the loan report', reject the data with the message **<a>**

Initial Values

| Name | Value |
|--------|-----------|
| Name | checkData |
| Active | True |
| Locale | en_US |
| Status | New |

Documentation

OK

6. Click the `checkData` template and click **OK**.
7. Click **OK**.
8. On the Properties page, define the rule properties.
- Change the name to: `check name`
 - Set the **Folder** to: `/loan`
 - Click **Next**.

The template opens in the rule editor on the Content page. Notice that this editor is like the Guided editor in Rule Designer and does not use special punctuation to distinguish parts of the rule.

9. On the Content page, edit the `check name` rule to state:

```
if
  the last name of the borrower of the loan report is empty
then
  in the loan report, reject the data with the message The borrower's name
  is missing
```

**Hint**

To change the condition phrase from `is <a string>` to `is empty`, click **is**.

[definitions]**if**

the last name of the borrower of the report [± **is** ▼ `<a string>` [±] **X** ↴

___ 10. Notice that you cannot change the beginning of the action parts:

in the loan report, reject the data with the message

___ 11. Click **Finish**.

The completed rule opens in the Details view.

___ 12. Click the **Explore** tab and note that the new `check name` rule is now visible in the loan rule package.

5.3. Deleting projects

- ___ 1. While you are still in the `loan-rules` project, click the **Configure** tab.
- ___ 2. Go to the Administration section.
- ___ 3. Click **Erase Current Project** and click **Yes** when prompted to confirm deletion.
- ___ 4. On the **Home** tab, in the **Work on project** section, select the `loan-bom` project as the project in use, and click the **Configure** tab.
- ___ 5. Click **Erase Current Project** and click **Yes** when prompted to confirm deletion.
- ___ 6. Close the Enterprise console.

End of exercise

Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 07: Authoring rules > 02-answer**.

This exercise looked at how to author action rules in the Intellirule editor and in the Guided editor, by using ruleset parameters, rule variables, or automatic variables. You also saw how to create and work with rule templates to support the rule authors in Decision Center.

Exercise 8. Authoring decision tables and decision trees

What this exercise is about

This exercise teaches you how to author decision tables and decision trees.

What you should be able to do

After completing this exercise, you should be able to:

- Use the decision table editor to create a decision table
- Use the decision tree editor to create a decision tree

Introduction

In this exercise, you work with patterns of rules to implement them as decision tables or decision trees. This exercise includes these tasks:

- Section 1, "Authoring a decision table"
- Section 2, "Authoring a decision tree"

Requirements

You should complete these exercises before proceeding:

- Exercise 7, "Authoring action rules"

Section 1. Authoring a decision table

In this part of the exercise, you author the following decision table, which was given as an example during the lectures.

| | Loan duration (years) | | Yearly rate (%) |
|---|-----------------------|-----|-----------------|
| | min | max | |
| 1 | < 5 | | 5 |
| 2 | 5 | 8 | 5.8 |
| 3 | [9 | 12[| 6.7 |
| 4 | 12 | 16 | 7.4 |
| 5 | ≥ 17 | | 7.9 |

This decision table defines the following five rules:

- If the duration of the loan is less than five years
then set the yearly interest rate of the loan to 5.0
- If the duration of the loan is between five years and eight years
then set the yearly interest rate of the loan to 5.8
- If the duration of the loan is at least nine years and less than 12 years
then set the yearly interest rate of the loan to 6.7
- If the duration of the loan is between 12 years and 16 years
then set the yearly interest rate of the loan to 7.4
- If the duration of the loan is at least 17 years
then set the yearly interest rate of the loan to 7.9

By creating this table, you also learn how the operators that you can use in the decision table editor (<, [...], and others) correspond to BAL operators (is less than, is between, and others).

1.1. Setting up your environment for this exercise

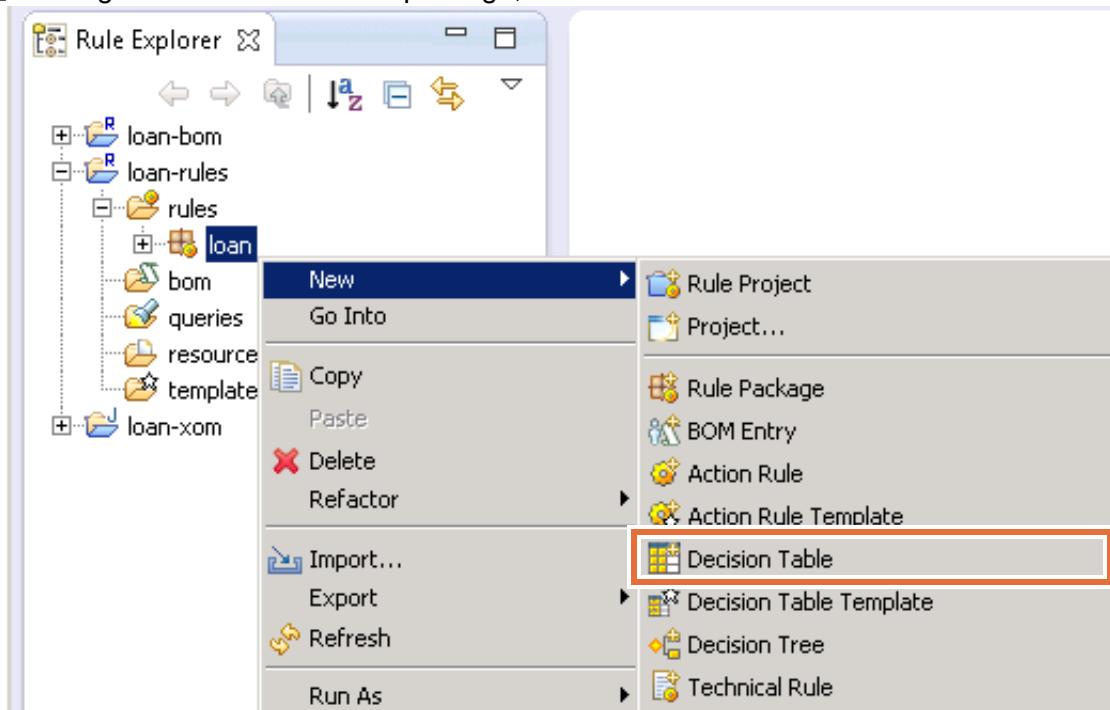
- ___ 1. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\dtabc_dtrees`
- ___ 2. Close the Welcome view and switch to the Samples Console perspective.
 - ___ a. Click the **Open Perspective** icon in the upper-right part of the Eclipse window.

- ___ b. Select **Samples Console**, and click **OK**.
 - ___ 3. Select **Rule Designer > Training > Ex 08: Authoring decision tables and trees > 01-start > Import projects**.
- The Rule perspective opens.
- ___ 4. Close the Help pane.

1.2. Creating the table

- ___ 1. In the `loan` rule package, create a decision table named: `rate`

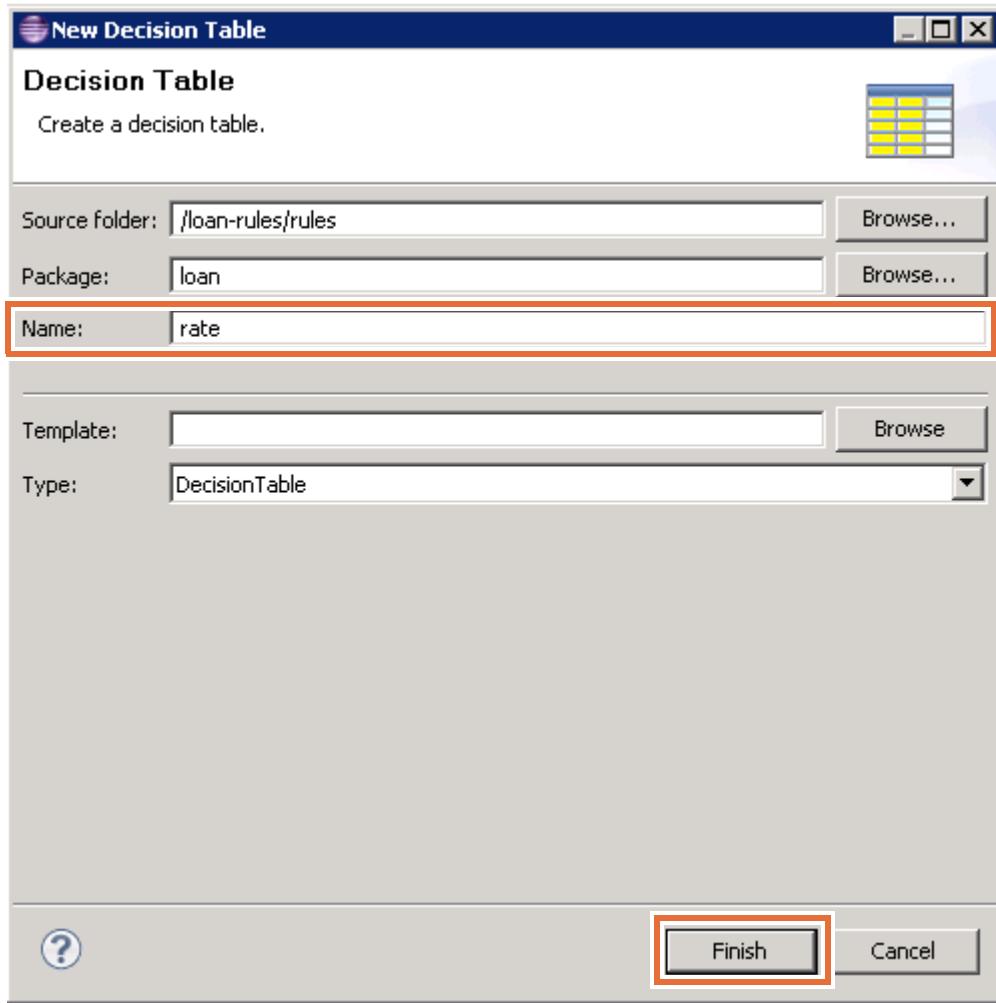
 - ___ a. Expand the `loan-rules > rules` project.
 - ___ b. Right-click the `loan` rule package, and click **New > Decision Table**.



The New Decision Table wizard opens.

- ___ c. In the **Name** field, type: `rate`

- __ d. Click **Finish**.



The new decision table opens in the decision table editor, with three condition columns and one action column.

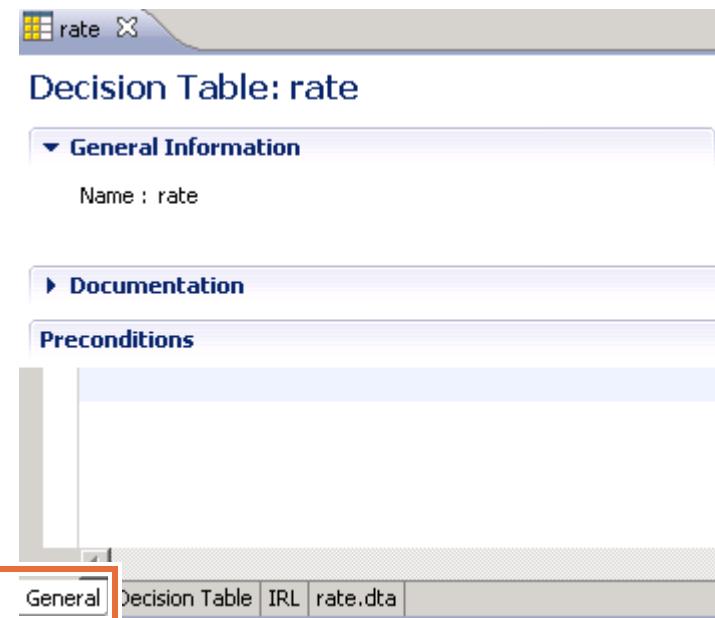
The screenshot shows the 'rate' decision table editor. The interface includes a toolbar with various icons, a header row with columns labeled A, B, C, and D, and a main grid area with 8 rows labeled 1 through 8. The 'General' tab is selected at the bottom left, and the status bar at the bottom right shows 'Decision Table rate.dta'.

The editor tab is called **Decision Table**.

- 2. Open the **General** tab of the decision table editor, and enter the following definitions in the Preconditions section:

definitions

```
set 'the loan' to the loan of 'the loan report';
```



Later, when you define the condition tests and action statements, you use `the loan` variable instead of the longer term: `the loan of 'the loan report'`

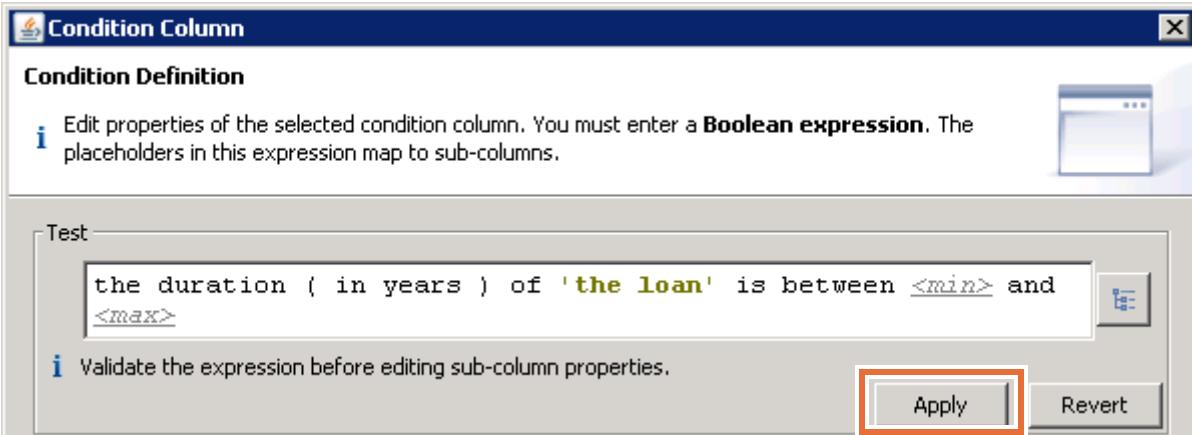


Hint

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

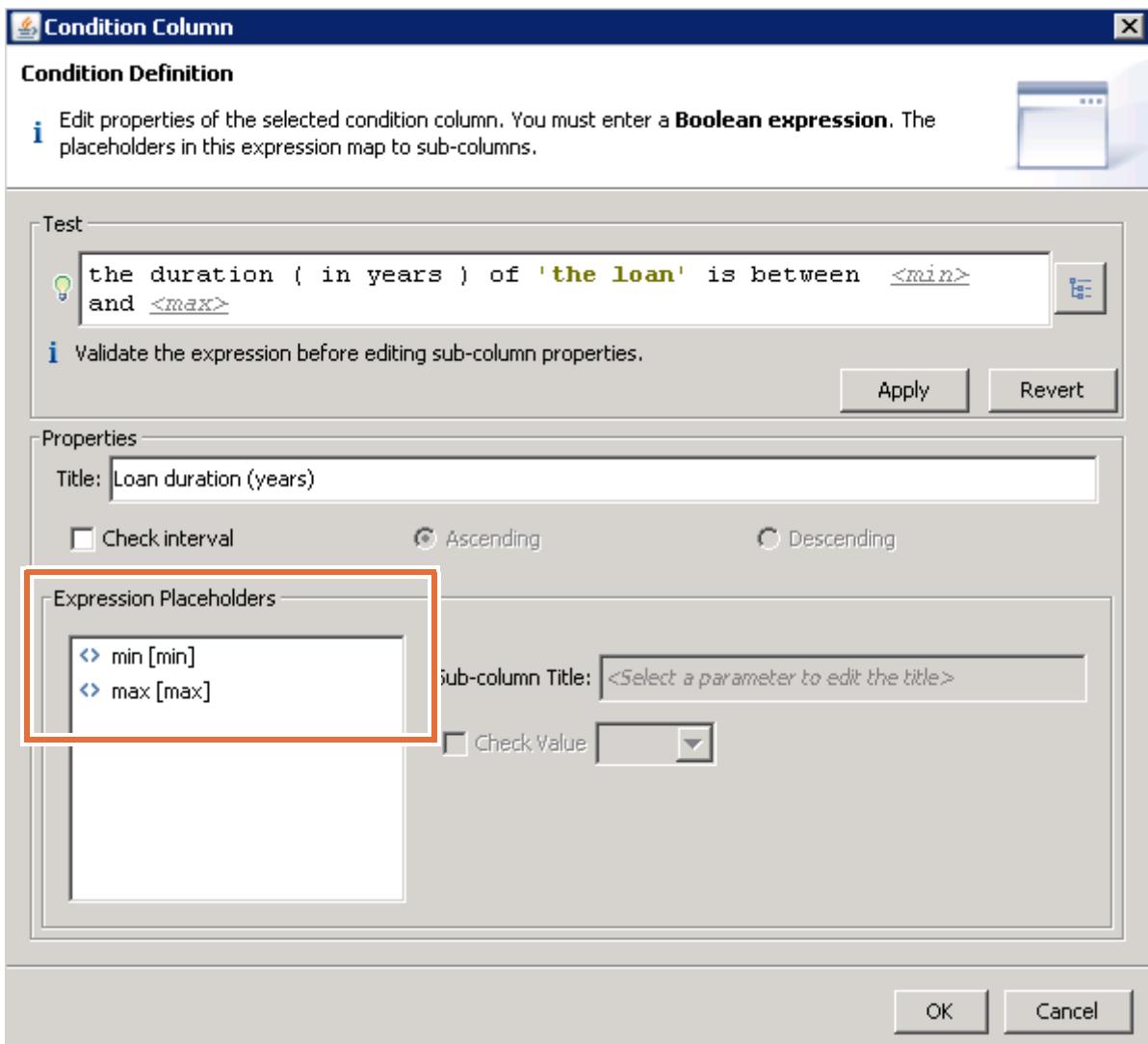
- 3. Save your work.
- 4. Click the **Decision Table** tab to return to the table editor and define the first condition column.
 - a. Double-click the header of the first condition column, which is labeled `A`, to open the Condition Column window.
 - b. In the **Test** section, click `<condition>` and select the appropriate vocabulary from the vocabulary list to build this condition:
`the duration (in years) of 'the loan' is between <min> and <max>`

- ___ c. When you are finished editing the test, click **Apply**.



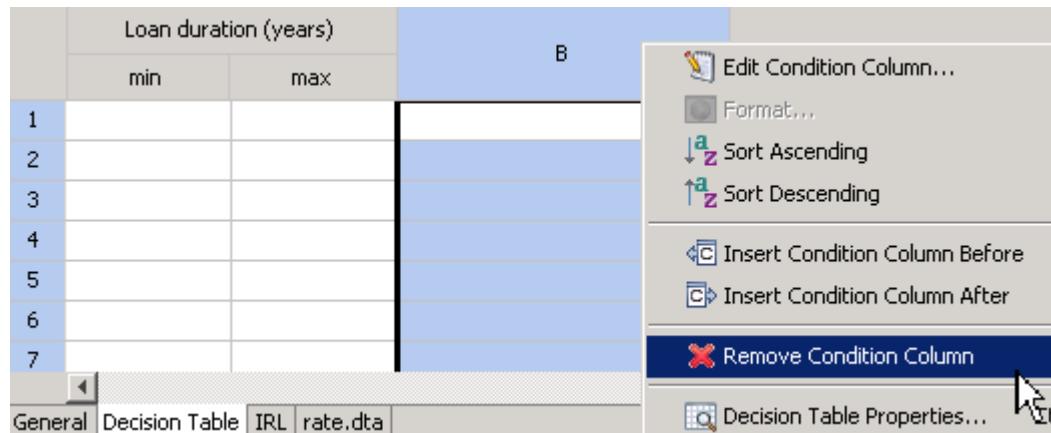
- ___ d. In the **Title** field, replace A with: Loan duration (years)

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



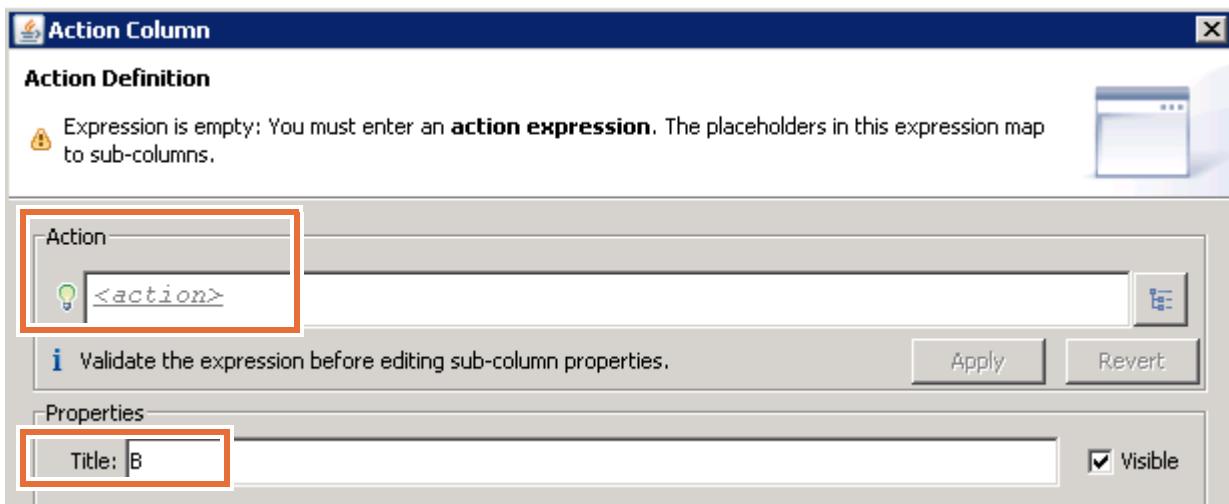
- ___ e. Click **OK**.

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



- ___ 6. Define the action column.

- ___ a. Double-click the action column header, which is now labeled **B**, to open the Action Column window.

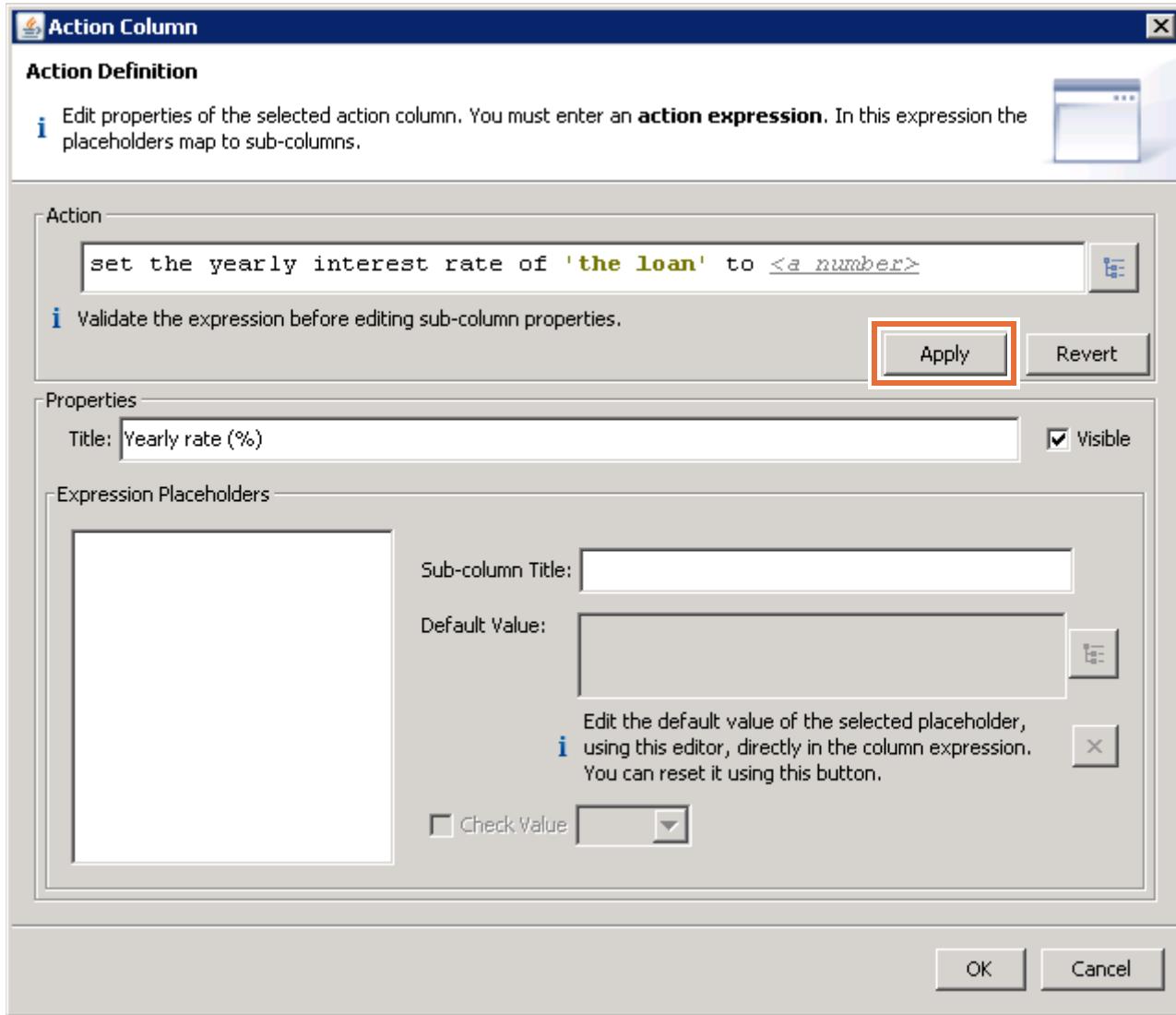


- ___ b. In the **Action** field, click **<action>**, and select the appropriate vocabulary from the vocabulary list to write this action statement:

set the yearly interest rate of 'the loan' to <a number>

- ___ c. In the **Title** field, replace **B** with: Yearly rate (%)

- ___ d. When you are finished editing the action, click **Apply**.



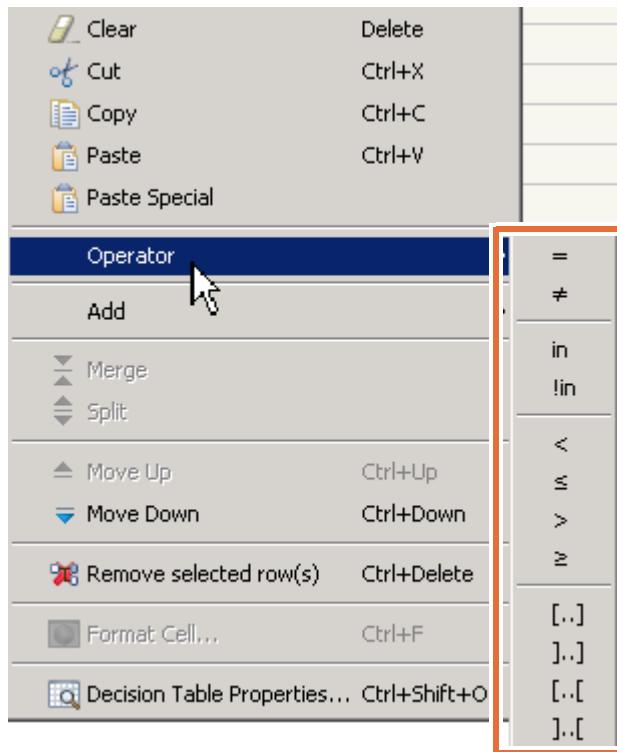
- ___ e. Click **OK** to close the window.

1.3. Completing the table cells with values

The cells in the `rate` decision table are currently empty. To complete the `rate` decision table, you work with operators in the next step.

**Hint**

To show the list of available operators in the decision table editor, right-click a decision table cell and select **Operator**. To enter an operator in a cell of the decision table, you can select the required operator from this list.

**Questions**

Can you relate the operators in the decision table editor (such as $>$, $<$) to BAL operators?

Answer

The operators in the decision table editor and the corresponding BAL constructs are as follows:

| Decision table editor operators | Corresponding BAL number operators |
|---------------------------------|------------------------------------|
| = | equals |
| != | does not equal |
| in | is one of { ... } |
| !in | is not one of { ... } |
| < | is less than |
| <= | is at most |
| > | is more than |
| >= | is at least |
| [...] | is between ... and ... |
| [...[| is at least ... and less than ... |
|]...] | is more than ... and at most ... |
|]...[| is strictly between ... and ... |

You are now ready to complete the `rate` decision table.

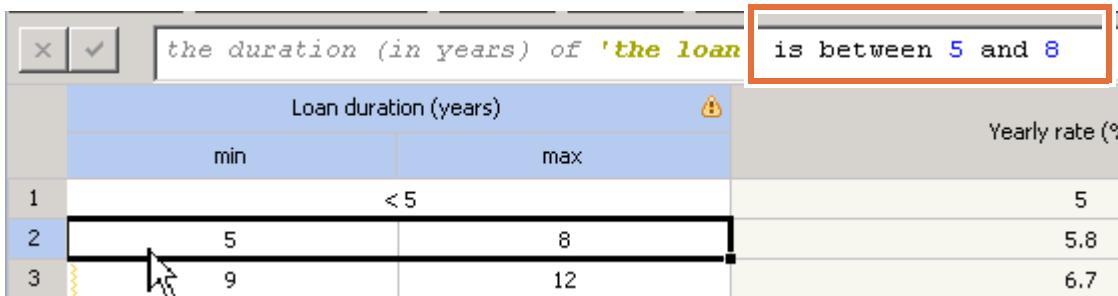
- 1. Review the five rules that you must write and the corresponding lines in the decision table, recalled at the beginning of this exercise. This table lists the values and operators that you use to complete the table.
- 2. In the decision table editor, first enter the following values in cells of the `rate` decision table:

| Loan duration (years) | Yearly rate (%) |
|-----------------------|-----------------|
| < 5 | 5 |
| >= 5 AND <= 8 | 5.8 |
| >= 9 AND < 12 | 6.7 |
| >= 12 AND <= 16 | 7.4 |
| >=17 | 7.9 |

For rows that have only a single value, enter that value in the first column only.

- 3. Next, edit the operators that are used for each row in the condition.
 - a. Click the first row of the condition column and select the “less than” (<) operator.
- The subcolumns merge automatically to match the operator. Notice that the condition test changes from **is between** to **is less than**.

- ___ b. Click the second row of the condition column, and consider the condition test (**is between**), which is equivalent to the BAL operator: [...]

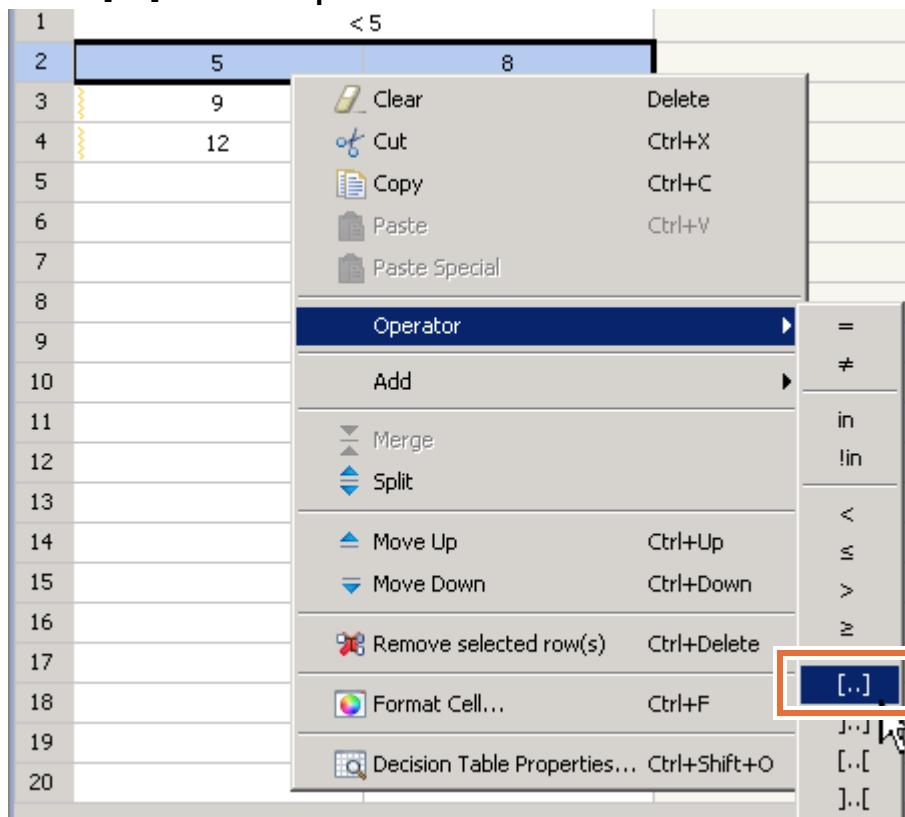


The screenshot shows a decision table with three columns: 'Loan duration (years)', 'min', and 'max'. Row 1 has 'min' as < 5 and 'max' as 5. Row 2 has 'min' as 5 and 'max' as 8. Row 3 has 'min' as 9 and 'max' as 12. A red box highlights the condition 'is between 5 and 8' in the header row.

| | | the duration (in years) of 'the loan' | is between 5 and 8 |
|---|-----|---------------------------------------|--------------------|
| | | Loan duration (years) | Yearly rate (%) |
| | min | max | |
| 1 | | < 5 | 5 |
| 2 | 5 | 8 | 5.8 |
| 3 | 9 | 12 | 6.7 |

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

In this case, these operators are equivalent to **is between** or [...]. You can right-click that row and click [...] from the **Operator** menu.



Notice that the cells do not change.

- ___ c. Right-click the condition column for row 3, and set the condition to **is at least and less than**, which is represented in BAL as: [...]
- ___ d. Complete the remaining rows of the condition column.
- ___ 4. Delete the empty rows of the decision tables.
- ___ a. Select all of the empty rows.
- ___ b. Click  **Remove selected row(s)** on the toolbar.

Your table should have five rows, and have the following values.

| Loan duration (years) | | Yearly rate (%) |
|-----------------------|-----|-----------------|
| Min | Max | |
| < 5 | | 5 |
| 5 | 8 | 5.8 |
| [9 | 12[| 6.7 |
| 12 | 16 | 7.4 |
| >=17 | | 7.9 |

| Loan duration (years) | | Yearly rate (%) |
|-----------------------|-----|-----------------|
| min | max | |
| < 5 | | 5 |
| 5 | 8 | 5.8 |
| [9 | 12[| 6.7 |
| 12 | 16 | 7.4 |
| ≥ 17 | | 7.9 |



Information

In the decision table editor, when you select a row of a decision table, you can see the corresponding rule at the top of the decision table. You can use this useful function to verify what you are writing.

5. Verify the rule statement for each row in the decision table.

- __ a. Click 1 in row 1, and hover your mouse to see the full rule statement, including the precondition.

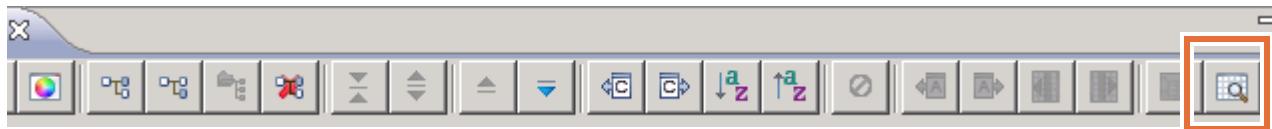
1 < 5

definitions
set 'the loan' to the loan of 'the loan report';
if
all of the following conditions are true :
- (the duration (in years) of 'the loan' is less than 5),
then
set the yearly interest rate of 'the loan' to 5 ;

- __ b. Click the remaining rows to verify that the rule conditions are correct for each row (or rule).
- __ c. Save your work.

6. Take some time to examine the Decision Table Properties window.

- a. Edit the decision table properties by clicking  **Open Decision Table Properties Editor** on the toolbar.



The Decision Table Properties window opens.

Decision Table Properties

General

i Set the general properties of the decision table.

Table view

Show comments Show rules BAL comments in IRL View

Table

| | | |
|---|---|---|
| <input checked="" type="checkbox"/> Auto-resize columns | <input checked="" type="checkbox"/> Expand / collapse handles | <input type="checkbox"/> Show hidden columns |
| <input checked="" type="checkbox"/> Show column tooltip | <input checked="" type="checkbox"/> Show row tooltip | <input checked="" type="checkbox"/> Show cell tooltip |
| <input type="checkbox"/> Render Boolean values | | |

Table checks

| | Cell values | Symmetry | Overlap | Gap |
|---------------------|-------------------------------------|--------------------------|-------------------------------------|-------------------------------------|
| -Table | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| ... ● Loan duration | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | Interval |
| ... ● Yearly rate (| <input checked="" type="checkbox"/> | N/A | N/A | N/A |

Report checking as:

| | | | |
|--------------|---------|-----------|---------|
| Cell values: | Warning | Gap: | Warning |
| Overlap: | Warning | Symmetry: | Info |

General **Lock** **Format**

With this window, you can specify some general or lock properties of the decision table.

- b. Open each of the **General**, **Lock**, and **Format** tabs to see the available options.
- c. On the **General** tab, select **Show rules** and click **OK**.

- __ d. Click 1 in row 1 again to see the rule in the rule pane that is now open In the decision table editor.

```

definitions
  set 'the loan' to the loan of 'the loan report';
  if
    all of the following conditions are true :
      - ( the duration (in years) of 'the loan' is less than 5 ) ,
  then
    set the yearly interest rate of 'the loan' to 5 ;

```

General Decision Table IRL rate.dta



Questions

Can you determine how you can lock the preconditions of the decision table to prevent changes?

Answer

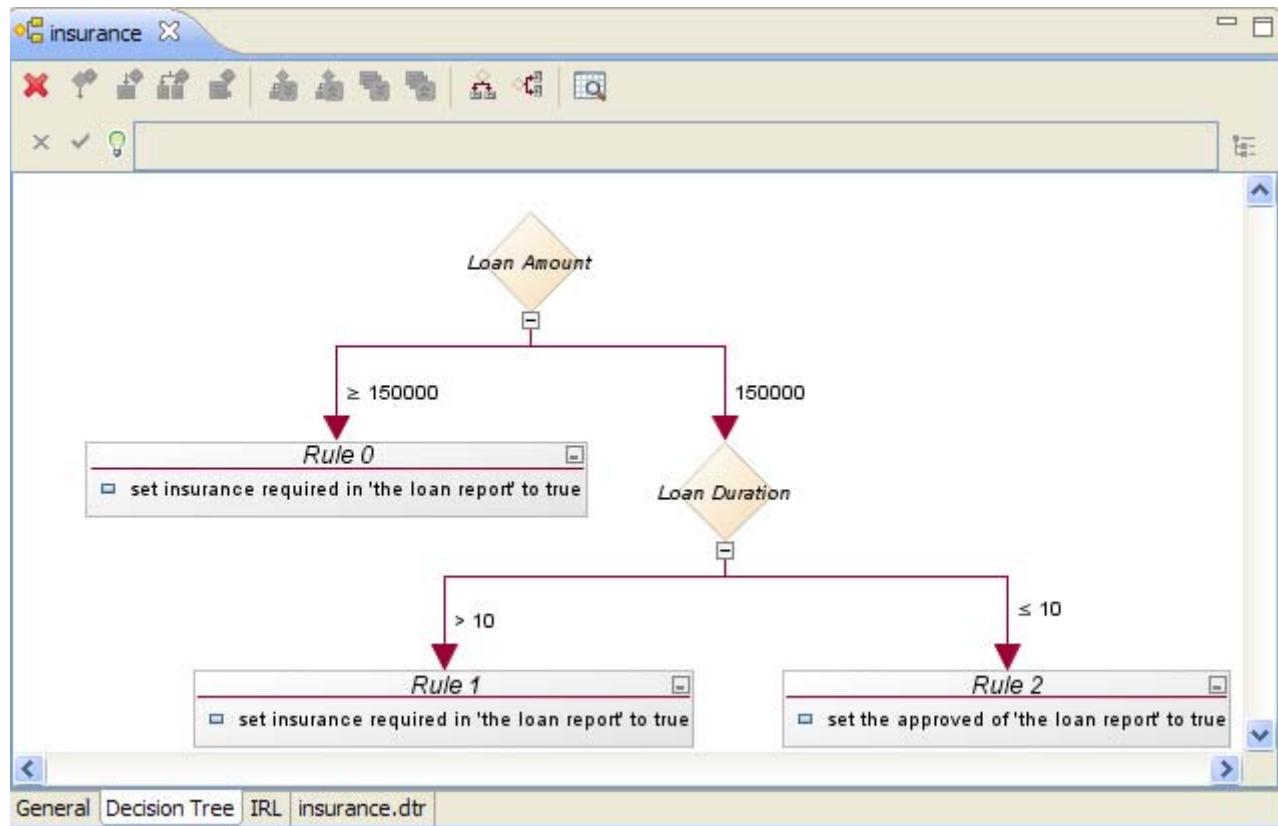
To lock the preconditions of the decision table, you must:

- Select the **Lock Preconditions** check box in the **Lock** tab of the Decision Table Properties window
- Then, select the **Enforce locking rules on this table** check box in the **Lock** tab for locking to take effect

- __ 7. Save your work.
__ 8. Close the decision table editor.

Section 2. Authoring a decision tree

In this part of the exercise, you author a decision tree that is called `insurance` in the `loan` rule package of the `loan-rules` project.



Requirements

The decision tree implements rules to determine whether insurance is required to approve the loan. These rules are based on the amount and the duration of the loan, as follows:

- If the amount of the loan ≥ 150000 , then an insurance is required
- If the amount of the loan < 150000 then:
 - If the duration of the loan > 10 , then an insurance is required
 - If the duration of the loan ≤ 10 , then the loan is approved

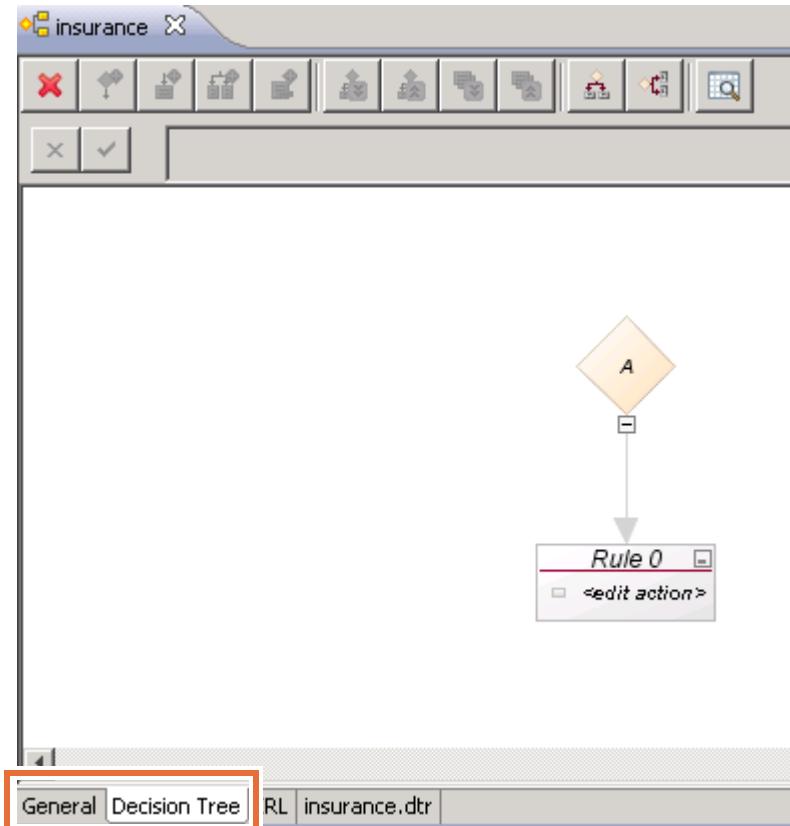
2.1. Creating the tree

- __ 1. In the `loan` rule package of the `loan-rules` project, create a decision tree called: `insurance`
 - __ a. Expand the `loan-rules` project.
 - __ b. Right-click the `loan` rule package, and click **New > Decision Tree**.

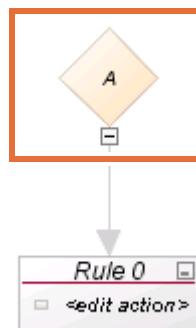
The New Decision Tree wizard opens.

- ___ c. In the **Name** field, enter: insurance
- ___ d. Click **Finish**.

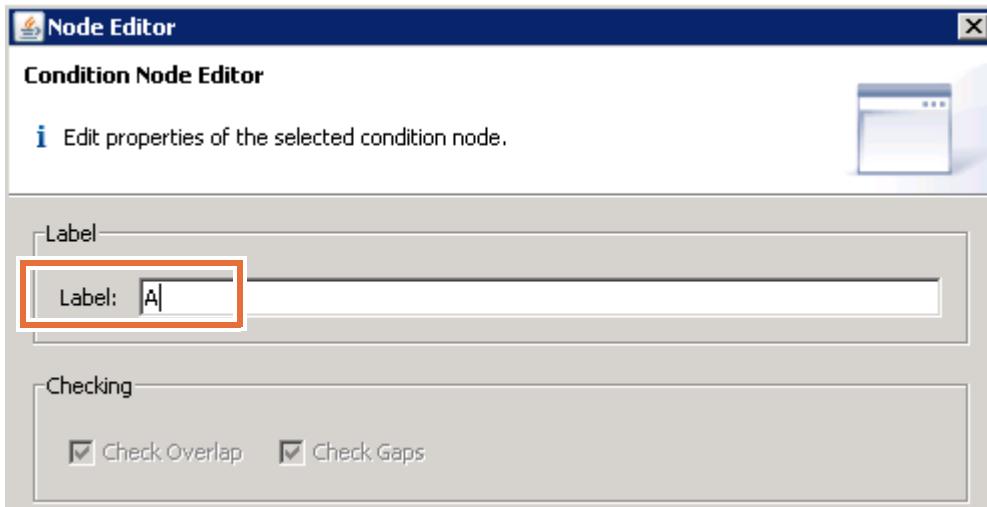
The new decision tree opens with one condition node, one branch, and one action. The tree diagram is on the **Decision Tree** tab.



- ___ 2. In the **General** tab of the decision tree editor, enter the following definitions in the Preconditions section:
- ```
definitions
set 'the loan' to the loan of 'the loan report';
```
- \_\_\_ 3. Click the **Decision Tree** tab to return to the decision tree editor and define the condition.
  - \_\_\_ a. Double-click the condition node.

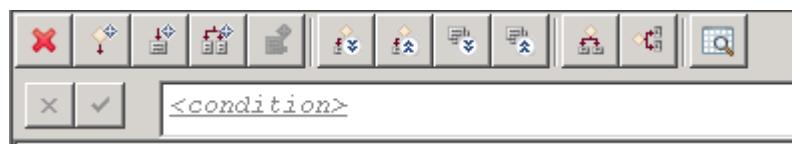


The Node editor opens.



- \_\_\_ b. In the **Label** field, replace A with Loan Amount and click **OK**.
- \_\_\_ c. In the edit bar, click <condition>, and select the appropriate vocabulary from the vocabulary list to write this condition:

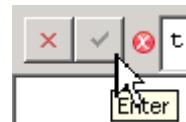
the amount of 'the loan' is less than <a number>



### Hint

You can also type directly or press Ctrl+Spacebar to open the Content Assist box when constructing your condition statement.

- \_\_\_ d. When you are finished editing the condition, click the check mark to save your edits.



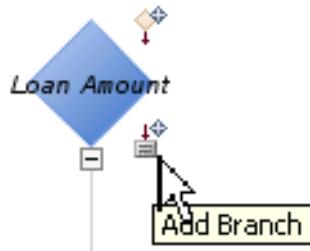
- \_\_\_ 4. Click the **Loan Amount** condition node (if it is not already selected) and notice the **Insert a Condition Node** and **Add Branch** icons beside it, which you can use to insert new conditions or rules.



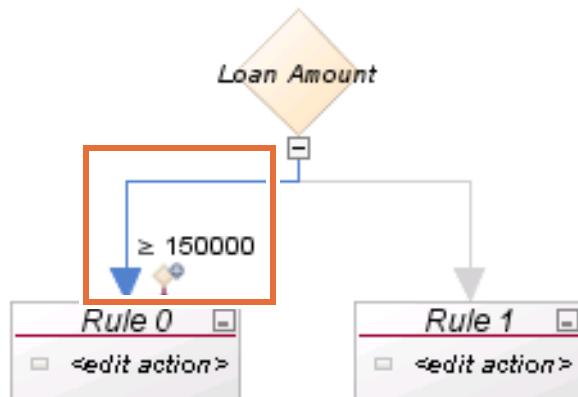
These icons are also available on the toolbar.



- \_\_\_ 5. Click **Add Branch** to create a branch.



- \_\_\_ 6. Click the branch (on the line) to **Rule 0**, and in the edit bar, modify the condition to state:  
the amount of 'the loan' is at least 150000

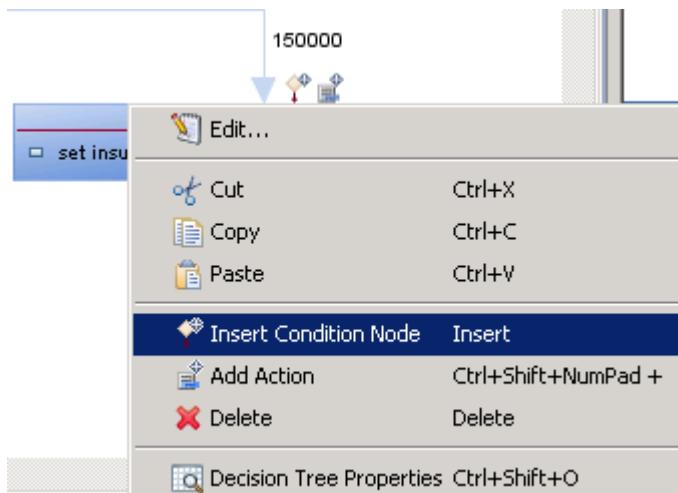


### Important

Remember to click the **check mark** to save your edit.

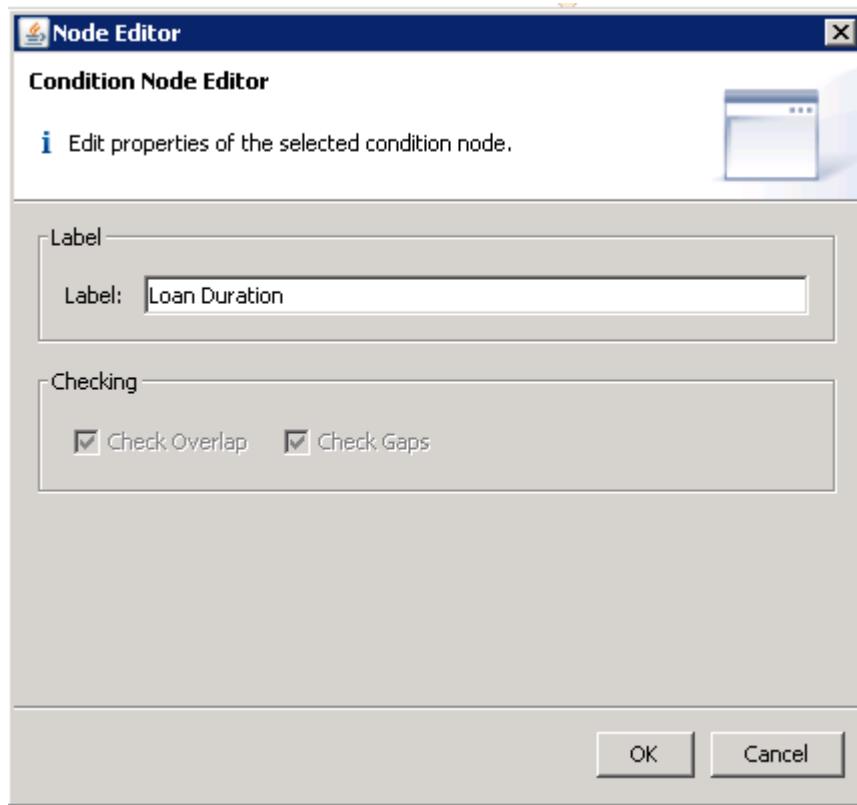
- \_\_\_ 7. Click the branch to **Rule 1**, and in the edit bar, modify the condition to state:  
the amount of 'the loan' is less than 150000  
The branches are now defined and visible in dark red.
- \_\_\_ 8. In Rule 0, click **<edit action>**, and in the edit bar, modify the action to state:  
set insurance required in 'the loan report' to true

- \_\_\_ 9. Right-click **Rule 1**, and click **Insert Condition Node**.



- \_\_\_ 10. Rename the new condition node as: **Loan Duration**

- \_\_\_ a. Right-click the new condition node and click **Edit** to open the Node editor.
- \_\_\_ b. In the Node editor, in the **Label** field, enter: **Loan Duration**
- \_\_\_ c. Click **OK** to close the Node editor.



- \_\_\_ 11. Edit the condition to state:

the duration ( in years ) of 'the loan' is at least <a number>

- \_\_ 12. Click the **Loan Duration** node, and add a branch.
- \_\_ a. Modify the branch to Rule 1 to state:  
the duration (in years) of 'the loan' is more than 10
  - \_\_ b. Modify the branch to Rule 2 to state:  
the duration (in years) of 'the loan' is at most 10
  - \_\_ c. Set the action for Rule 1 to:  
set insurance required in 'the loan report' to true
  - \_\_ d. Set the action for Rule 2 to:  
set the approved of 'the loan report' to true
- \_\_ 13. Save your work.

- \_\_ a. Click  **Decision Tree Properties**.
- \_\_ b. Click **Show rules** and then click **OK**.



- \_\_ c. In the tree diagram, click one of the rules to see the full rule statement in the rule pane that is now visible.

 **Note**

Try clicking the rule header to see the rule. Clicking outside of the header might not show the rule.



## End of exercise

## Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 08: Authoring decision tables and trees > 02-answer**.

The first part of the exercise looked at how to author a decision table.

The second part of the exercise looked at how to author a decision tree.



# Exercise 9. Working with static domains

## What this exercise is about

This exercise teaches you how to define static domains in the BOM to simplify rule authoring.

## What you should be able to do

After completing this exercise, you should be able to:

- Create various types of static domains
- Use domains in rules

## Introduction

In this exercise, you learn how to create and use static domains during these tasks:

- Section 1, "Exploring a collection domain"
- Section 2, "Working with an enumeration of literals"
- Section 3, "Defining an enumeration of static references"

## Requirements

There are no specific requirements for this exercise.

## Section 1. Exploring a collection domain

In this section, you explore a collection domain to learn how you can use it in a business rule.



### Hint

In this exercise, you must write some code and some rule statements. You can find the code snippets and the rules in the `<LabfilesDir>\code\create_domains.txt` file, and you can copy and paste them in Rule Designer.

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

- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the **Workspace Launcher** window, enter the path:  
`<LabfilesDir>\workspaces\static_domains`
- 2. Close the Welcome view and switch to the Samples Console perspective.
- 3. Select **Rule Designer > Training > Ex 09: Static domains > 01-start > Import projects**.
- 4. Close the Help pane.

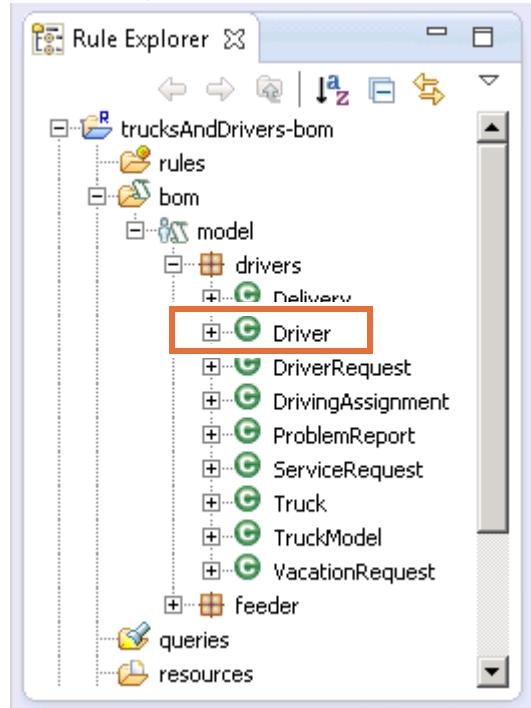
In your workspace, you now have the following projects:

- `trucksAndDrivers-bom`
- `trucksAndDrivers-rules`
- `trucksAndDrivers-tests`
- `trucksAndDrivers-xom`

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

## 1.2. Exploring the domain

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



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

**Class Driver (package: drivers)**

**General Information**

|               |                  |           |
|---------------|------------------|-----------|
| Name:         | Driver           |           |
| Namespace:    | drivers          | Change... |
| Superclasses: | java.lang.Object | Change... |
| Interfaces:   |                  | Change... |

Deprecated

**Members**

Specify the members of this class.

- age
- drivingAssignments
- gender
- knownDrivers
- licenseClass
- name
- nbAccidents
- totalVacationTime
- vacationRequests

New...  
Delete  
Edit

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

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

| General Information |                                 |
|---------------------|---------------------------------|
| Name:               | <code>drivingAssignments</code> |
| Type:               | <code>java.util.Vector</code>   |
| Class:              | <code>drivers.Driver</code>     |

- \_\_\_ 4. The vector is defined with a collection domain, and each member of the collection is of type `drivers.DrivingAssignment`.

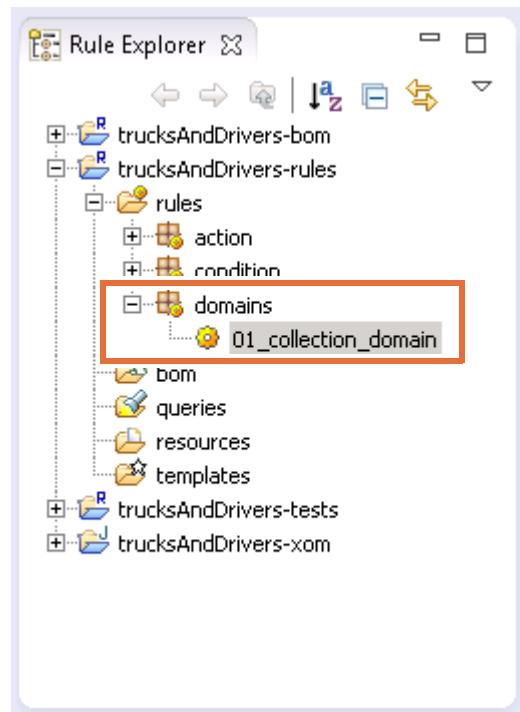
**▼ Domain**  
Create and edit a domain for this member.

[Edit](#) the domain.  
[Remove](#) the domain.

**Domain type: Collection**  
Element type: `drivers.DrivingAssignment`  
Cardinality: 0, \*

This collection domain was automatically defined when the BOM was generated from the XOM because this XOM attribute is defined as: `Vector<DrivingAssignment>`

- \_\_\_ 5. In Rule Explorer, expand **trucksAndDrivers-rules > rules > domains**.  
 \_\_\_ 6. Double-click the `01_collection_domain` rule to open it and see how this rule uses the collection domain.

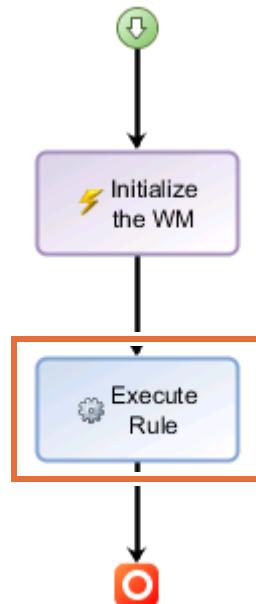


- \_\_\_ 7. When you are done, close the rule.

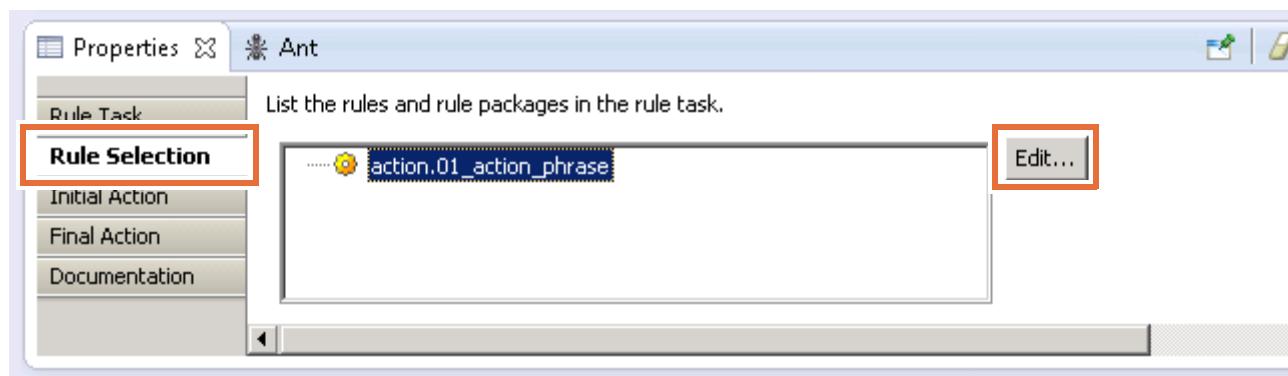
### 1.3. Testing the rule

With the next steps, you test the `domains\01_collection_domain` action rule. First, you verify that objects in the working memory exist that match the definitions of this rule. Then, you use the `trucksAndDrivers-tests` project to test this rule in the rule project.

- \_\_\_ 1. Expand the `trucksAndDrivers-xom > src > feeder` folder and double-click the `WMFeeder.java` file.
- \_\_\_ 2. Explore the `WMFeeder.java` file.
  - \_\_\_ a. Verify that a `DrivingAssignment` object is created for the `Driver` with the name: John Jongle
  - \_\_\_ b. Read through the code to see how the `DrivingAssignment` object is inserted into the working memory through the `IlrContext insert` method.
- \_\_\_ 3. Use the `trucksAndDrivers-tests` project to test the `01_collection_domain` rule.
  - \_\_\_ a. In the Rule Explorer, expand the `trucksAndDrivers-tests > rules` folder and double-click the `testFlow` ruleflow to open it.
  - \_\_\_ b. Double-click the **Execute Rule** rule task to edit its properties.

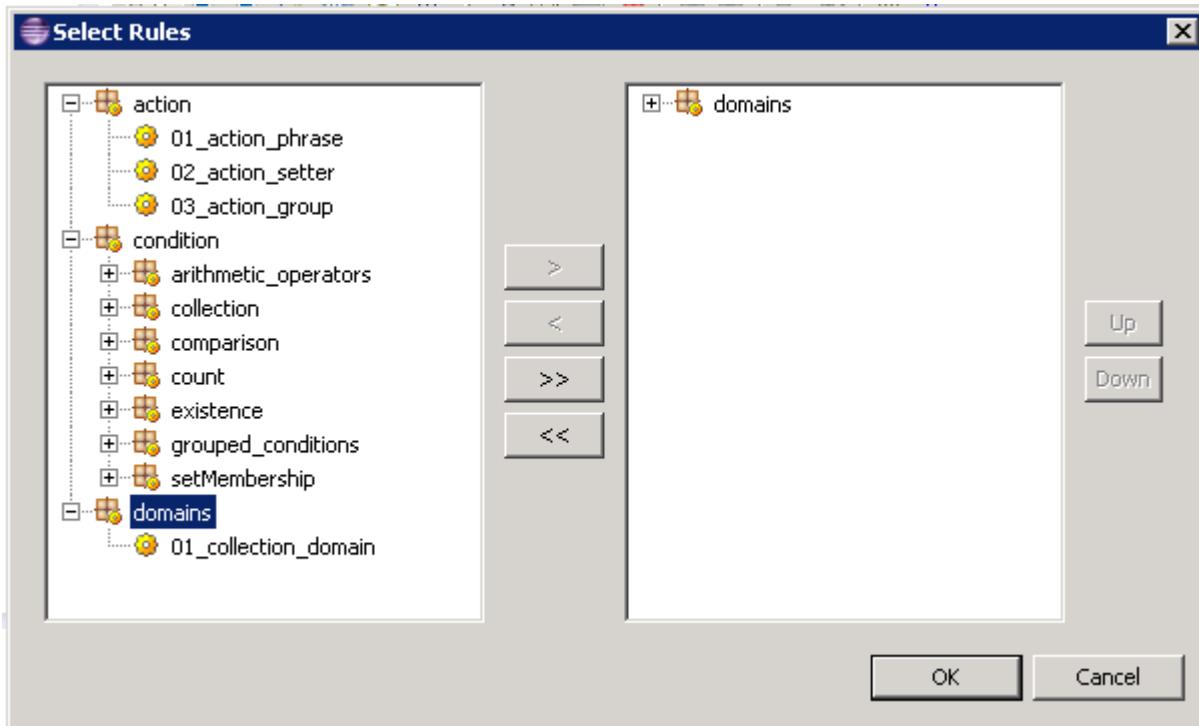


- \_\_\_ c. In the Rule Task properties, click **Rule Selection** and click **Edit...**.



The Select Rules window opens.

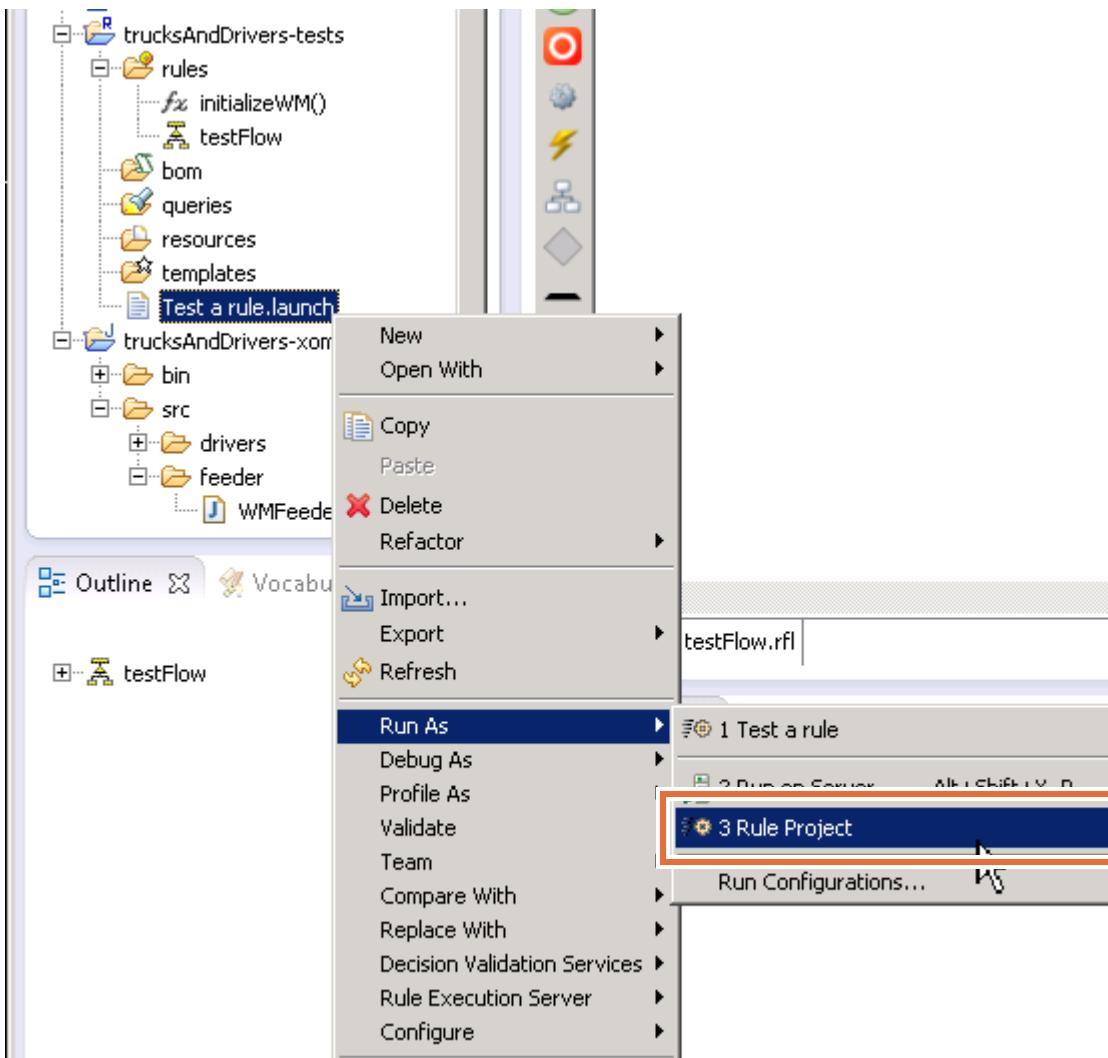
- \_\_ d. In the Select Rules window, replace the content on the right side with the `domains` package and click **OK**.



Now, only the rules of the `domains` package are executed in the `testFlow` ruleflow.

- \_\_ e. Save the ruleflow (Ctrl+S).

- \_\_\_ f. In the trucksAndDrivers-tests project, right-click the `Test a rule.launch` configuration, and click **Run As > Rule Project** to execute the `testFlow` ruleflow.



- \_\_\_ g. Verify that you have the following result in the Console view:

Added the pending driving assignment to the list of driving assignments of John Jongle



### Note

Here is another example of how you can use this collection domain.

definitions

```
set 'the driver' to a driver ;
if
 there is at least one driving assignment in the driving assignments of 'the
 driver'
then
 ...

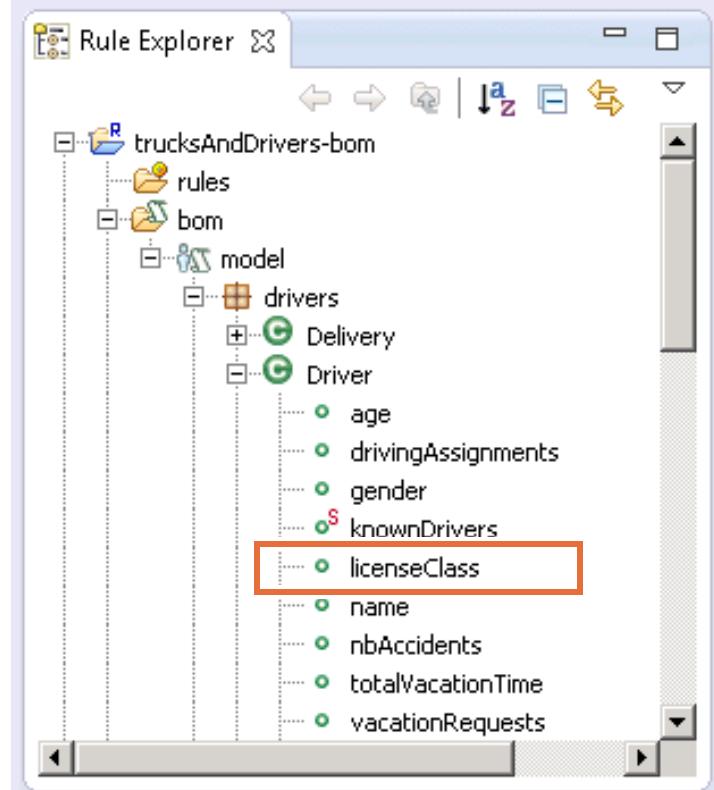
```

## Section 2. Working with an enumeration of literals

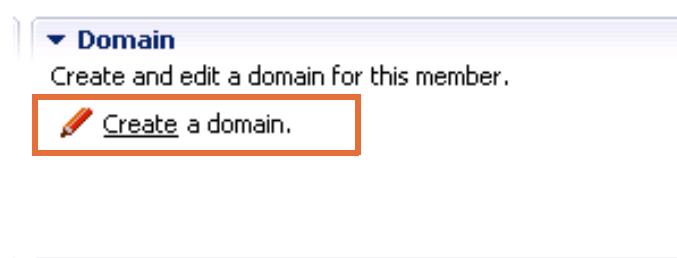
In this section, you define a domain as a list of literal values {A, B, C, D} for the `licenseClass` member of the `Driver` and `TruckModel` classes, and then use it to author an action rule.

### 2.1. Creating the domain

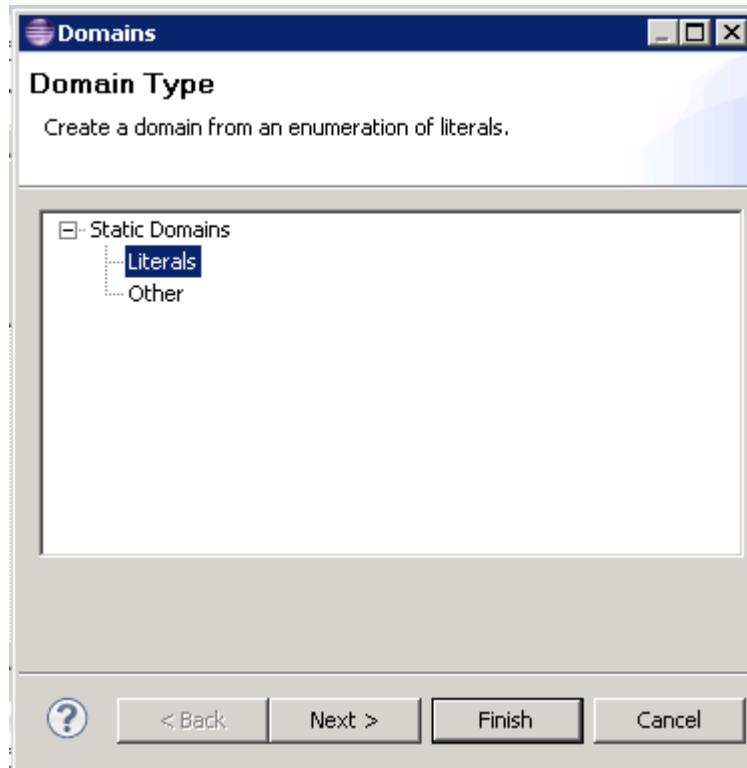
- 1. In the `trucksAndDrivers-bom` project, expand the `drivers.Driver` class and double-click the `licenseClass` BOM member to open it in the BOM editor.



- 2. In the BOM editor, in the **Domain** section, create a domain of literal values:
  - a. Click **Create a domain** to open the Domains window.



- \_\_\_ b. In the Domain Type page of the Domains windows, select **Literals** in the list and click **Next**.

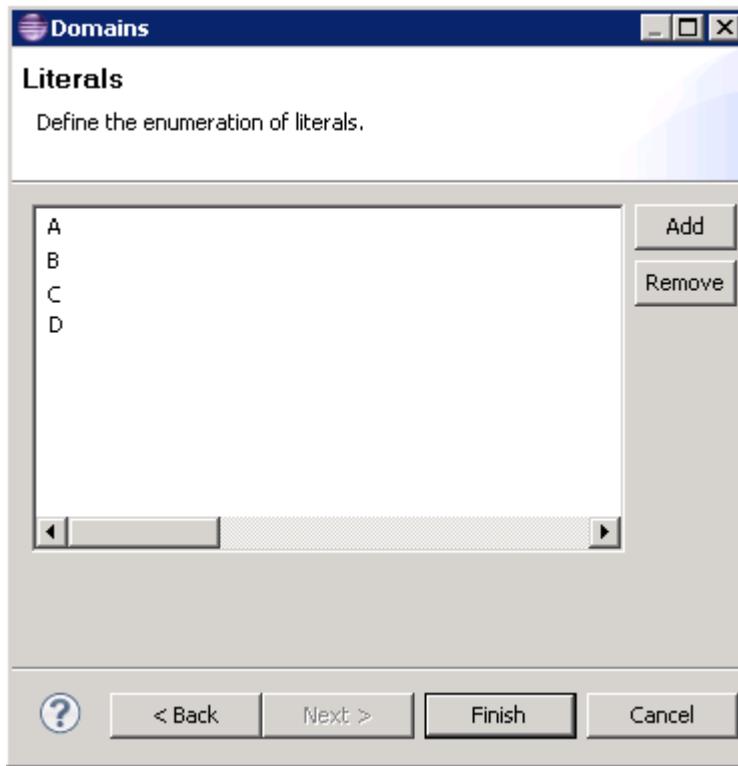


- \_\_\_ c. In the Literals page of the Domains window, click **Add** and type over the new domain value to rename it to: A



If you want, you can click **Add** several times to add the placeholders for your domain values, and then type over the placeholder names later.

- \_\_ d. Add these values to the domain: B, C, and D



- \_\_ e. Click **Finish** to close the Domains window.

You created the domain of type Literals {A, B, C, D}.

**Domain type: Literals**

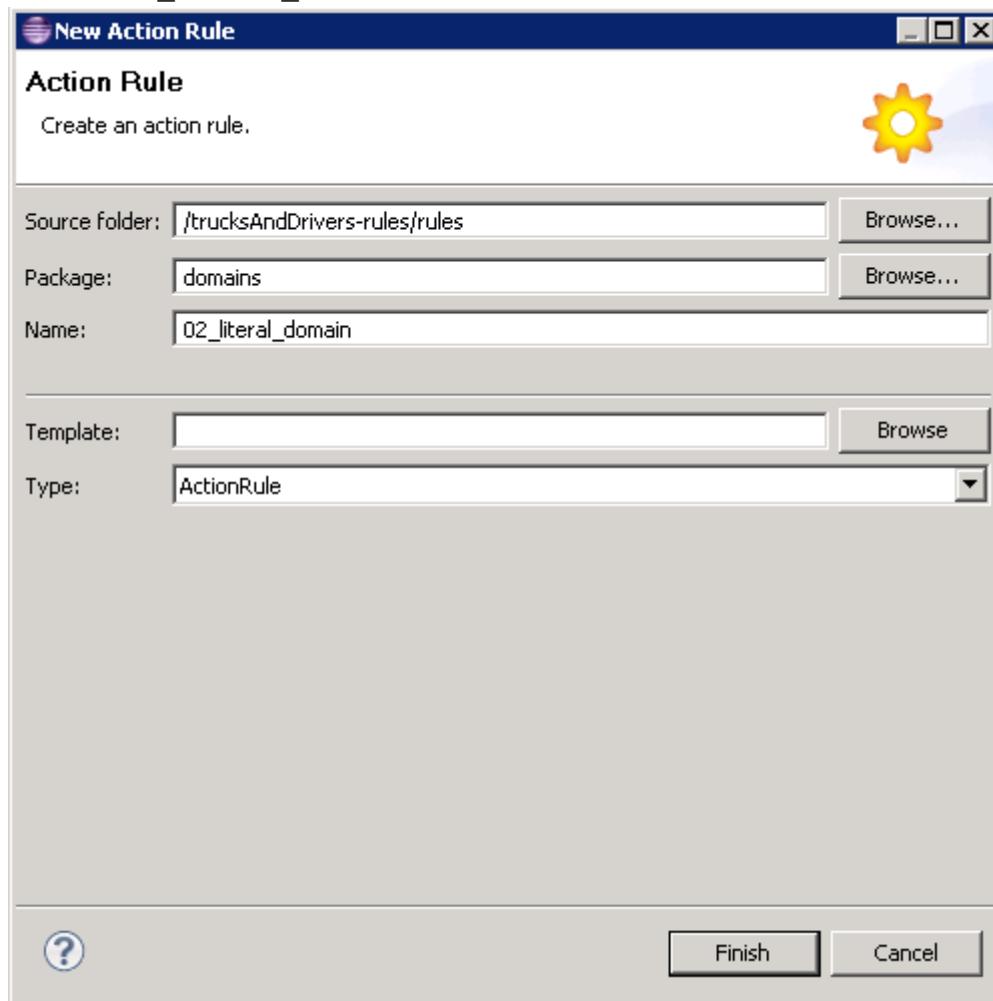
- A
- B
- C
- D

- \_\_ 3. Create the domain of type Literals {A, B, C, D} for the drivers.TruckModel.licenseClass BOM member as well.  
\_\_ 4. Save the BOM (Ctrl+S).

## 2.2. Testing the domain in a rule

In this section, you use your new domains to author a rule.

- 1. In the `domains` rule package of the `trucksAndDrivers-rules` project, create the action rule named: `02_literal_domain`



- 2. Enter this rule statement in the rule editor:

```

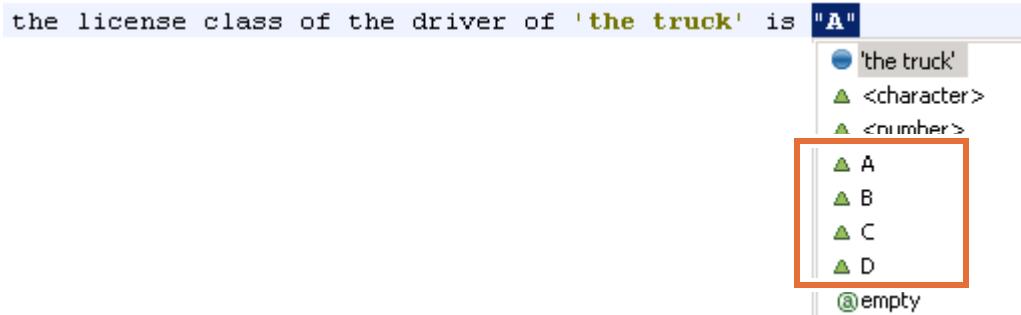
definitions
 set 'the truck' to a truck where the driver of this truck is not null;
if
 the license class of the driver of 'the truck' is "A" and the license
 class of the model of 'the truck' is one of {"B", "C", "D"}
then
 display the message "The driver (" + the name of the driver of 'the truck'
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

```

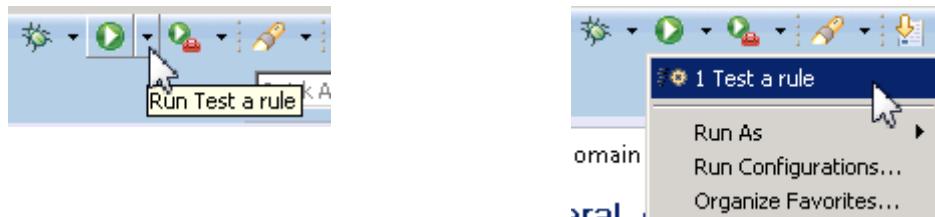
**Hint**

Recall that you can type directly in the Intellirule editor or you can paste the code from the code snippet in the <LabfilesDir>\code\create\_domains.txt file.

Because the `licenseClass` BOM member type is a domain of Literals, an enumeration of values is available to you when you author an action rule that uses this `licenseClass` member.



- \_\_\_ 3. After you finish editing, save the rule.
- \_\_\_ 4. Rerun the test project from the toolbar by clicking **Run > Test a rule** on the toolbar.

**Information**

The `testFlow` ruleflow is already correctly configured to execute all rules of the `domains` package. You do not have to explicitly add your new rule to the Execute task.

- \_\_\_ 5. Verify that, in addition to the result from the previous section, you also have the following result:

The driver (John Jongle) is not eligible to drive the truck TRUCK-F150

- \_\_\_ 6. Close the rule.

## Section 3. Defining an enumeration of static references

A domain that is set as an enumeration of static references specifies a list of references to constants, for example:

```
{static GroupA, static GroupB, static GroupC}
```

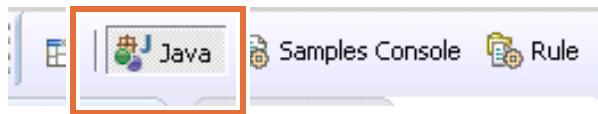
You can define attribute types, method return types, and arguments as follows:

- If you have an attribute of type A, you can define a domain of static references on it by using the static attributes of the class A (classic Java enumeration pattern)
- If you have an attribute of a primitive type, you can define a literal domain on it

### 3.1. Creating the static references Java class

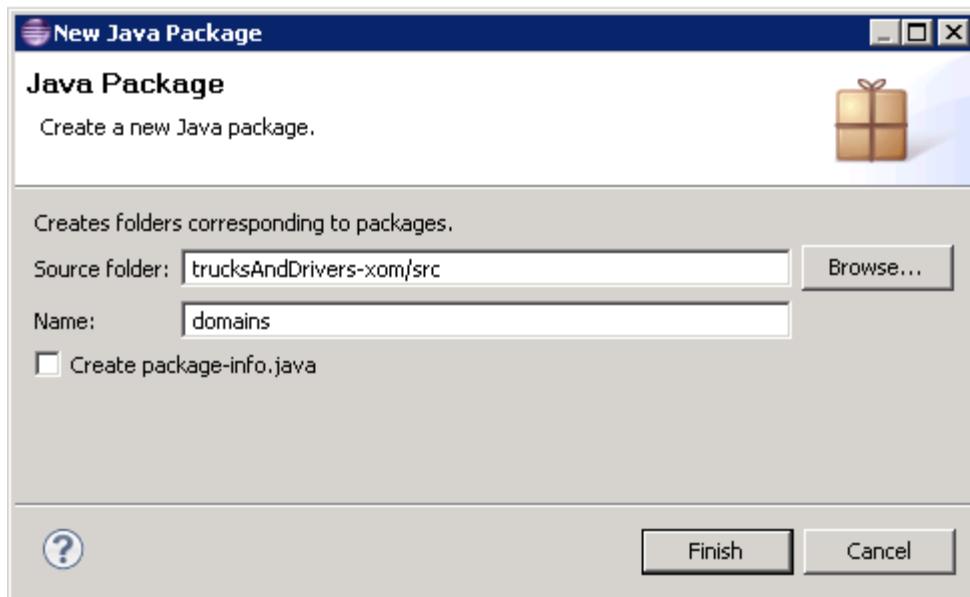
In this section, you change the attribute `gender` in the `Driver` class to use the static attributes of a new class called `GenderType`.

- 1. Switch to the Java perspective.



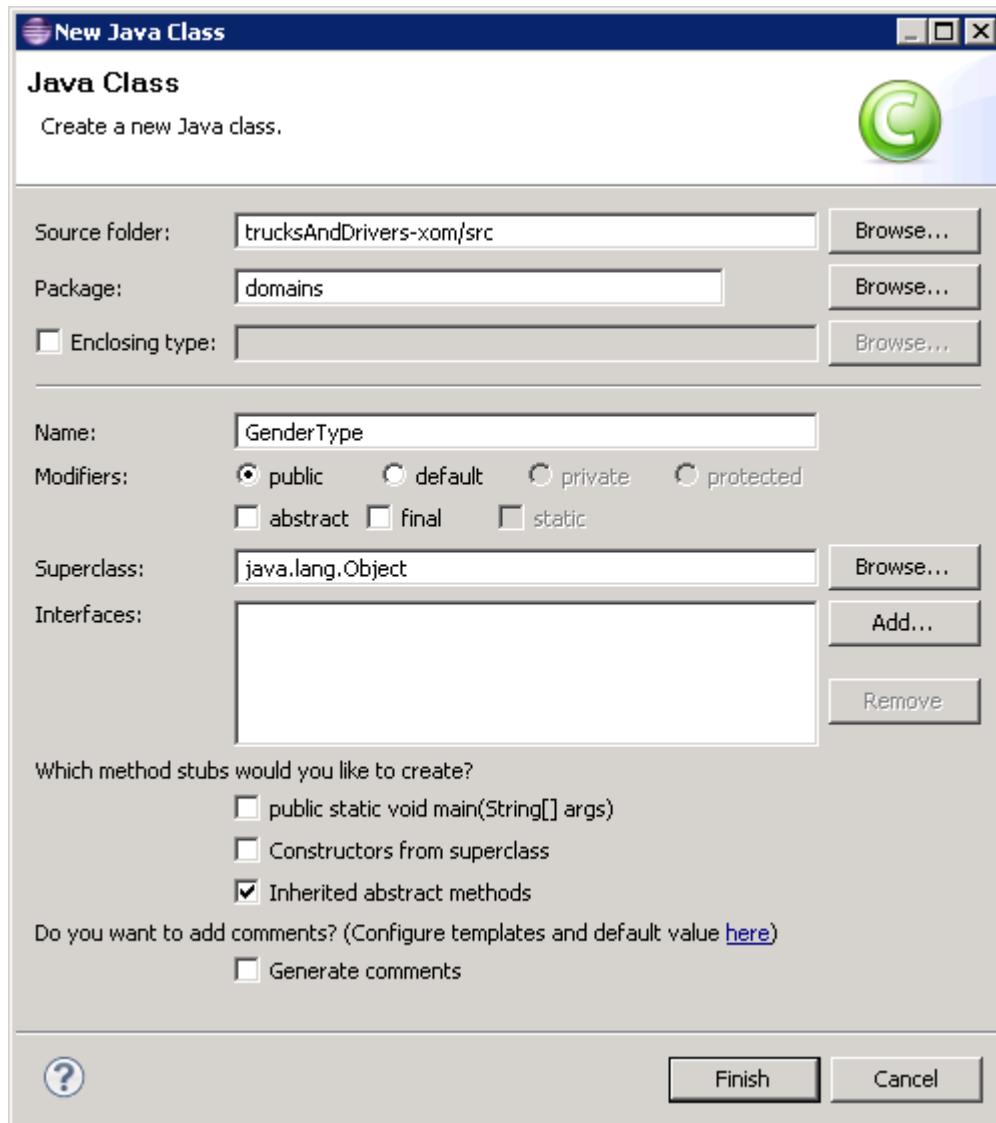
- 2. Create a package and class in the `trucksAndDrivers-xom > src` project.

- a. Expand `trucksAndDrivers-xom > src`, right-click `src`, and click **New > Package**.
- b. Name the package: `domains`
- c. Click **Finish**.



- d. Right-click the `domains` package, and click **New > Class**.
- e. Name the class: `GenderType`

- \_\_\_ f. Leave the other default values and click **Finish**.



- \_\_\_ 3. When the `GenderType.java` file opens in the editor, define the static attributes in the Java file with this content:

```
package domains;
public class GenderType {
 private final String name;
 public static final GenderType MALE = new GenderType("male");
 public static final GenderType FEMALE = new GenderType("female");
 public static final GenderType UNKNOWN = new GenderType("unknown");
 private GenderType(String _name) {
 this.name = _name;
 }
 public String toString() {
 return this.name;
 }
}
```

**Hint**

You can find this code snippet in the `<LabfilesDir>\code\create_domains.txt` file.

- 4. In the Package Explorer, find the `drivers.Driver` class in the **trucksAndDrivers-xom > src** folder, and change the type from `String` to `GenderType` for the following class members:

- gender attribute
- `getGender` method
- `setGender` method

- a. Expand **trucksAndDrivers-xom > src > drivers.Driver**.
- b. Double-click **drivers.Driver.gender**.
- c. Find the following line:

```
private String gender;
```

```
public class Driver {
 private int age;
 private int nbAccidents;
 private String name; // name
 private String licenseClass; // driver's license class
 private Vector<DrivingAssignment> drivingAssignments; // current driving assignments
 private Vector<VacationRequest> vacationRequests; // vacation requests
 private String gender;
```

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

It should now read:

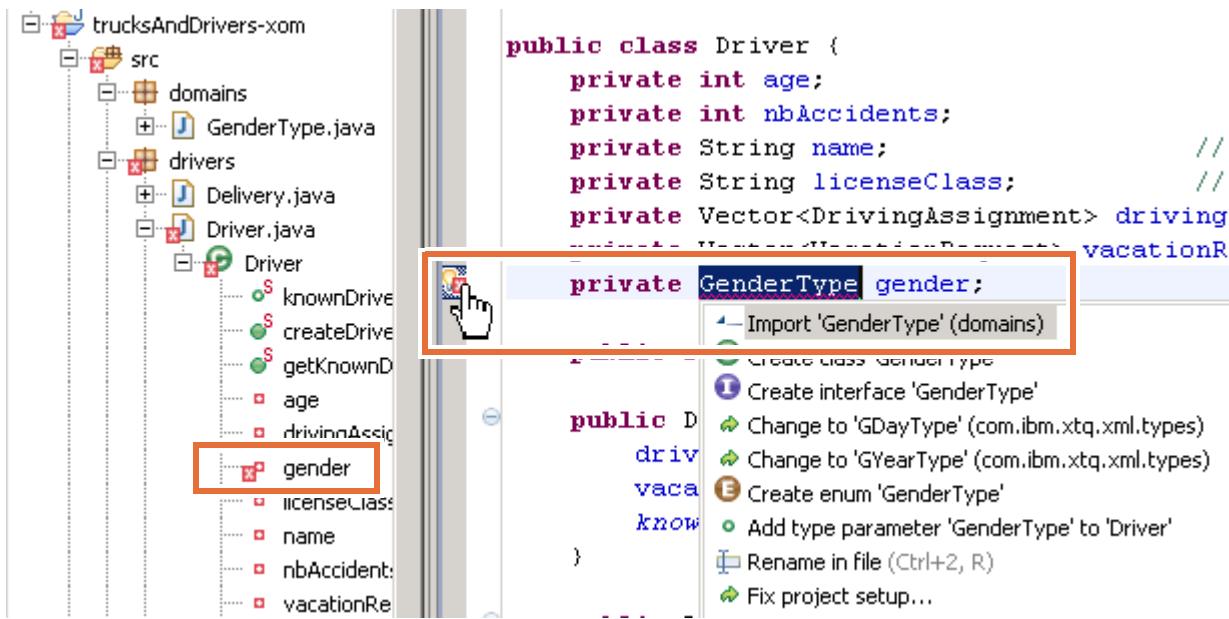
```
private GenderType gender;
```

```
public class Driver {
 private int age;
 private int nbAccidents;
 private String name; // name
 private String licenseClass; // driver's license class
 private Vector<DrivingAssignment> drivingAssignments; // current driving assignments
 private Vector<VacationRequest> vacationRequests; // vacation requests
 private GenderType gender;
```

- e. Change the `String` attribute for `drivers.Driver.getGender` and `drivers.Driver.setGender` to `GenderType`.
- f. Save your work (Ctrl+S).

When you save the file, notice that you get an error on `GenderType`.

5. To resolve the problem, hover your mouse over the error icon and select the **Import Gender Type (domains)** quick fix.



6. Save your work (Ctrl+Shift+S).
7. Update the `feeder.WMFeeder` class that uses the setter method.
- In the `WMFeeder` Java file, replace all occurrences of "MAN" with: `GenderType.MALE`
  - In the `WMFeeder` class, replace all occurrences of "UNKNOWN" with: `GenderType.UNKNOWN`

```

private void createModel() {
 // Definitions of the drivers
 Driver da20 = new Driver("George Smith", TruckModel.MACK_TRUCK.getLicenseC
 da20.setAge(20);
 da20.setNbAccidents(3);
 da20.setGender(GenderType.MALE);

 Driver da18 = new Driver("John Jongle", TruckModel.MACK_TRUCK.getLicenseCl
 da18.setAge(18);
 da18.setNbAccidents(0);
 da18.setGender(GenderType.MALE);

 Driver dB23 = new Driver("Marc Lansen", TruckModel.F150.getLicenseClass());
 dB23.setAge(23);
 dB23.setNbAccidents(0);
 dB23.setGender(GenderType.MALE);

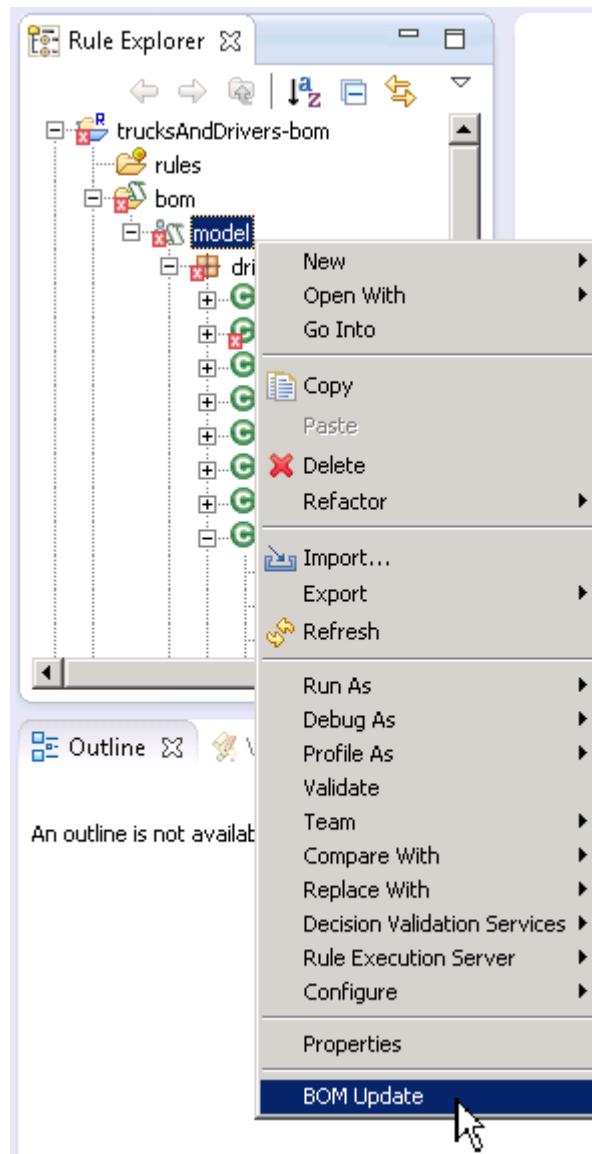
 Driver dB37 = new Driver("Jill Northwood", TruckModel.T2000.getLicenseClas
 dB37.setAge(37);
 dB37.setNbAccidents(1);
 dB37.setGender(GenderType.UNKNOWN);
}

```

- \_\_\_ c. If you have errors on these values, use the **Import Gender Type (domains)** quick fix as you did in Step 4.

After you save the XOM, the Problems view lists an error on the BOM. You correct this error now by using the BOM Update view.

- \_\_\_ 8. Save your work and close the editing windows for the Java files.
- \_\_\_ 9. Switch back to the Rule perspective.
- \_\_\_ 10. In the `trucksAndDrivers-bom` project, right-click `bom.model` and click **BOM Update**.



You find some differences in the **Differences and Actions** section of the BOM Update view.

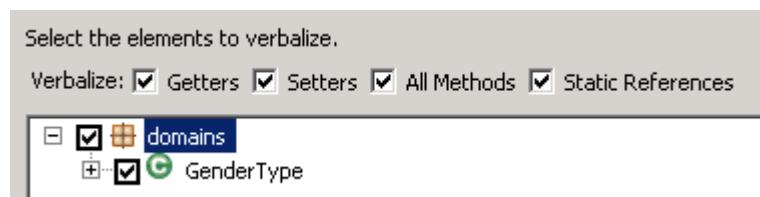
- The XOM class `domains.GenderType` is not found in the BOM.
- The definition of the attribute `drivers.Driver.gender` differs between the BOM and the XOM.

- \_\_ a. Select the **Import the XOM class** action and click **Perform and save**.

**Differences and Actions**

| Actions:                                                                                                                                                            |                    |                  |                                                         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|------------------|---------------------------------------------------------|
| <input drivers.driver""="" type="button" value="Update the BOM class "/> <input type="button" value="Perform and save"/> <input type="button" value="Clear table"/> |                    |                  |                                                         |
| Perform: <input domains.gendertype""="" type="button" value="Import the XOM class "/> <small>Select an action to solve this difference</small>                      |                    |                  |                                                         |
| Origin                                                                                                                                                              |                    |                  |                                                         |
| XOM                                                                                                                                                                 | domains.GenderType | Missing from BOM | XOM class "domains.GenderType" not found in BOM.        |
| XOM                                                                                                                                                                 | drivers.Driver     | Modified         | Difference between attributes. XOM attribute "drivers.D |

- \_\_ b. In the Verbalize BOM window, make sure that you click **Select All** and that you select the **All Methods** check box to verbalize all members and methods of the BOM class `GenderType`.



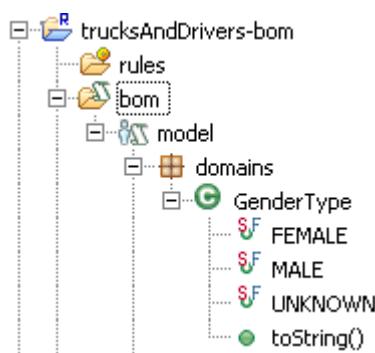
- \_\_ c. Click **Finish**.

- \_\_ 11. Go back to the BOM Update view, select the **Update the BOM class** action, and click **Perform and save**.

- \_\_ 12. In the Configure Verbalization window, click **Finish**.

- \_\_ 13. After the BOM is updated, verify the following points:

- \_\_ a. A new `GenderType` BOM class exists under `domains` in the BOM and is defined as a domain of type `Static References` with these values: `MALE`, `FEMALE`, and `UNKNOWN`



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

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

#### General Information

|        |                    |           |
|--------|--------------------|-----------|
| Name:  | gender             |           |
| Type:  | domains.GenderType | Browse... |
| Class: | drivers.Driver     | Browse... |

- \_\_\_ 14. Close the BOM editor.

## 3.2. Authoring an action rule that uses a domain of static references

In this section, you use the `GenderType` static domain of type Static References to author an action rule.

- \_\_\_ 1. In the `domains` rule package of the `trucksAndDrivers-rules` project, create the action rule `03_static_references_domain`:

```
definitions
 set 'the driver' to a driver;
if
 the gender of 'the driver' is UNKNOWN
then
 display the message "Update the personal data of " + the name of 'the
 driver';
```



#### Reminder

You can type directly in the Intellirule editor or you can paste the code from the code snippet in the `<LabfilesDir>\code\create_domains.txt` file.

- \_\_\_ 2. Save and close the rule.  
 \_\_\_ 3. Rerun the `trucksAndDrivers-tests` project from the toolbar by clicking **Run > Test a rule** on the toolbar.  
 \_\_\_ 4. Verify that the following line is now included in the result:

Update the personal data of Jill Northwood

## End of exercise

## Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 09: Static domains > 02-answer**.

This exercise looked at how to create static domains in the BOM, and how to work with them in rules. You also saw the relationship between domains and the XOM. You explored an existing domain of type Collection and learned how you can use it in an action rule.

# Exercise 10. Working with dynamic domains

## What this exercise is about

This exercise teaches you how to define and use dynamic domains with Microsoft Excel spreadsheets.

## What you should be able to do

After completing this exercise, you should be able to:

- Create dynamic domains in Microsoft Excel spreadsheets
- Update and use dynamic domains in rules
- Access and update dynamic domains in Decision Center
- Synchronize dynamic domains between Rule Designer and Decision Center

## Introduction

In this exercise, you work in Rule Designer to create a dynamic domain with Microsoft Excel, update the values of a domain, and discover the effects of such updates.

You learn how to resolve inconsistencies across the BOM, XOM, and rules after modifying domain classes. You also learn how to synchronize updated domain values between Rule Designer and Decision Center.

The exercise involves these tasks:

- Section 1, "Creating a dynamic domain in Rule Designer"
- Section 2, "Using the dynamic domain in a rule"
- Section 3, "Updating the dynamic domain"
- Section 4, "Updating the XOM"
- Section 5, "Publishing the BOM and rule projects to Decision Center"
- Section 6, "Examining rules in Decision Center"
- Section 7, "Modifying the dynamic domain in Decision Center"
- Section 8, "Updating Rule Designer from Decision Center"

## Requirements

You should complete these exercises before proceeding:

- Exercise 3, "Working with the BOM"
- Exercise 6, "Exploring action rules"
- Exercise 9, "Working with static domains"



### **Important**

For this exercise, you work with a series of Microsoft Excel spreadsheets to define the values of the dynamic domain, and their updates. The Excel spreadsheets that are stored in the `<LabfilesDir>\code` directory.

During the exercise, you open and read these Excel spreadsheets, but you do not have to modify them.

You can read the Microsoft Excel spreadsheets by using either:

- Microsoft Excel Viewer, which is installed on the computer lab environment for this course
- Microsoft Excel, which is **not** installed on the computer lab environment for this course

If you have access to Microsoft Excel, you can *optionally* edit the Microsoft Excel spreadsheets.

## Section 1. Creating a dynamic domain in Rule Designer

In this section, you create a dynamic domain in the BOM to represent the values that are listed in an Excel spreadsheet.



### Hint

In this exercise, you must write some pieces of code. You can find the pieces of code to create in the `<LabfilesDir>\code\create_domains.txt` file that is installed on your computer, and you can copy and paste them in Rule Designer.

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

- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the **Workspace Launcher** window, enter the path:  
`<LabfilesDir>\workspaces\dynamic_domain`
- 2. Close the Welcome view and switch to the Samples Console perspective.
- 3. Select **Rule Designer > Training > Ex 10: Dynamic domains > 01-start > Import projects**.
- 4. Close the Help pane.

In your workspace, you now have the following projects:

- trucksAndDrivers-bom
- trucksAndDrivers-rules
- trucksAndDrivers-tests
- trucksAndDrivers-xom

These projects define the business rule solution to which you must add the dynamic domain.

#### Scenario:

In the Trucks and Drivers rule application, drivers might submit requests for vacations. In their current state, requests for vacations are not associated with any specific type.

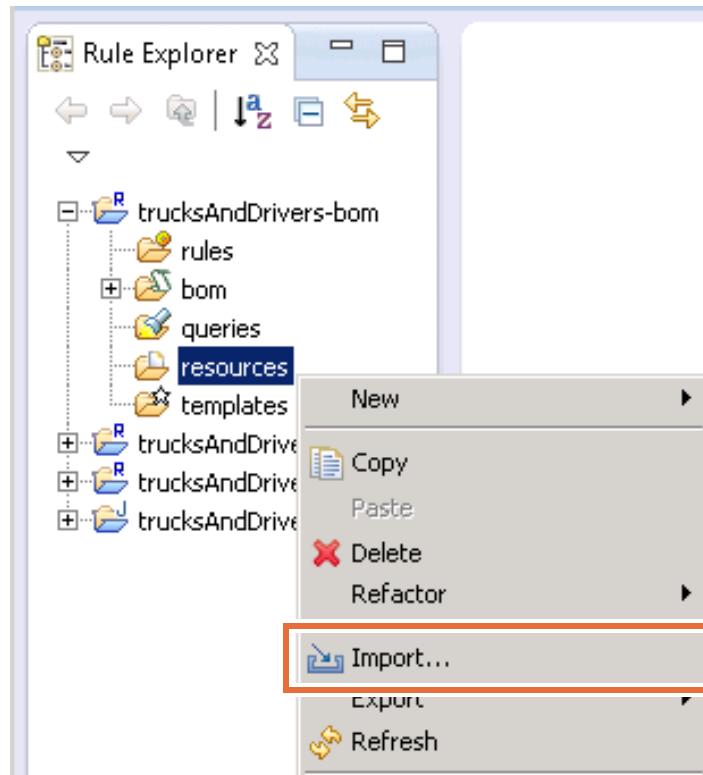
After a discussion with the human resources department, business analysts tell you that the type of vacation request must be differentiated for statistical purposes.

To start the work, the human resources department provides you with an Excel spreadsheet that lists their current list of possible types of vacation requests. They also indicate that this list might change regularly, and ask that it remain easy to update in case of changes.

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

## 1.2. Creating the domain

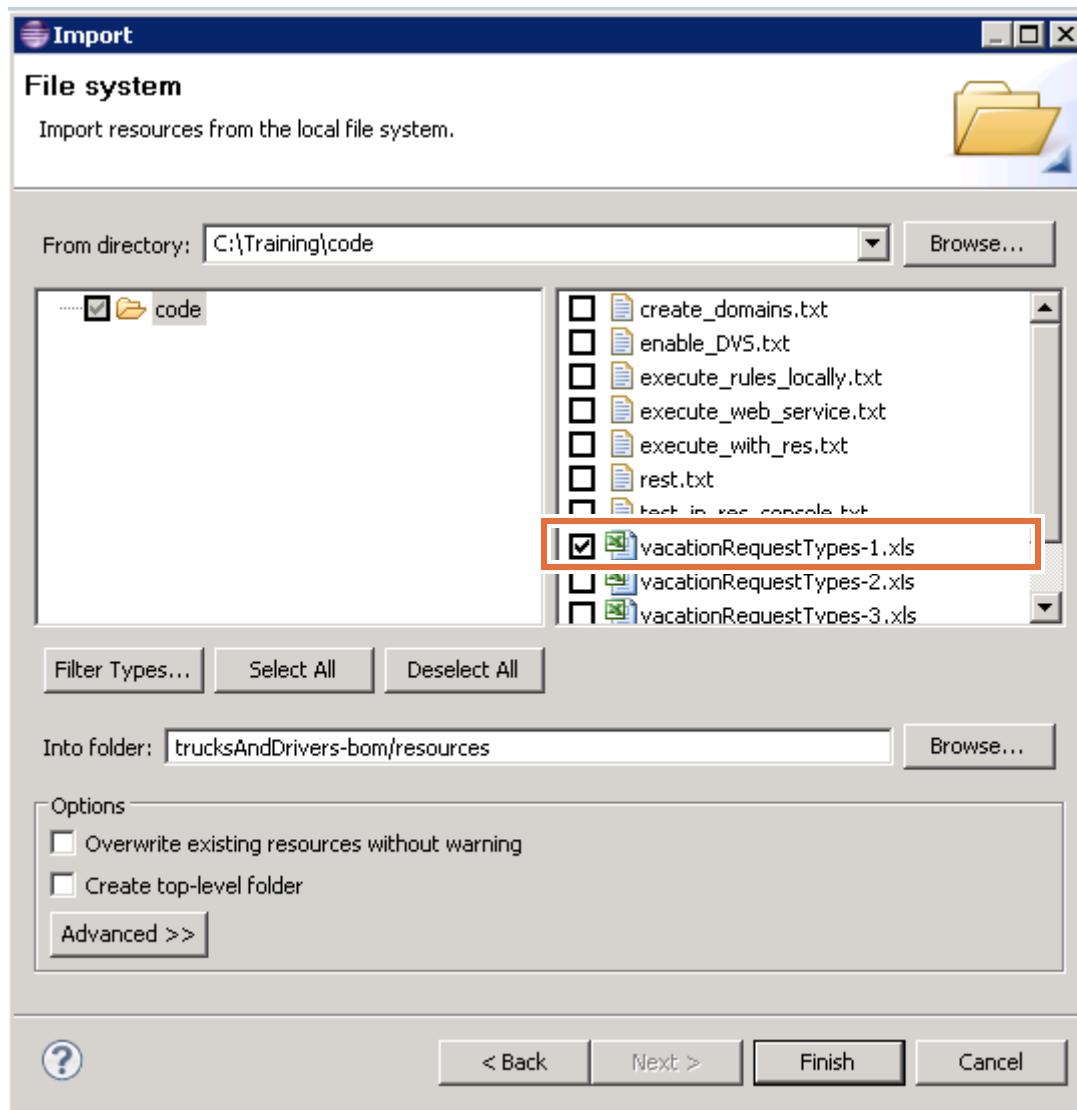
- 1. Open the `<LabfilesDir>\code\vacationRequestTypes-1.xls` file and verify that each row of this file contains three columns, where:
  - The first column represents the name to the domain value.  
Example: JuryDuty
  - The second column represents the code in the BOM-to-XOM mapping.  
Example: `return "V78";`
  - The third column represents the verbalization of the domain value.  
Example: Jury Duty
- 2. In Rule Designer, import the Excel domain file into the `resources` folder of your BOM project.
  - a. Right-click the `resources` folder of the `trucksAndDrivers-bom` project, and click **Import**.



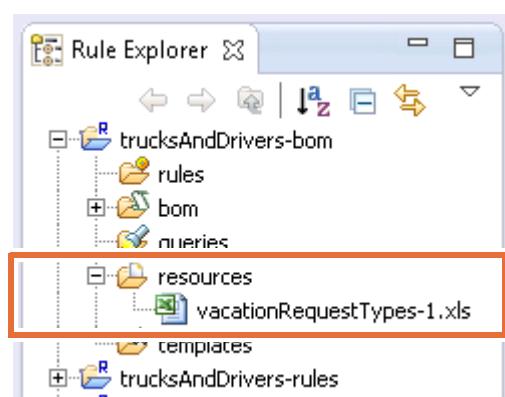
The Import window opens.

- b. In the Select pane of the Import window, select **General > File System**, and click **Next**.

- \_\_ c. In the File System pane, click **Browse**, which is next to the **From directory** field, and select the <LabfilesDir>\code folder.
- \_\_ d. Select **vacationRequestTypes-1.xls** and click **Finish**.



The `vacationRequestTypes-1.xls` file is now visible under the `resources` folder of the `trucksAndDrivers-bom` project.



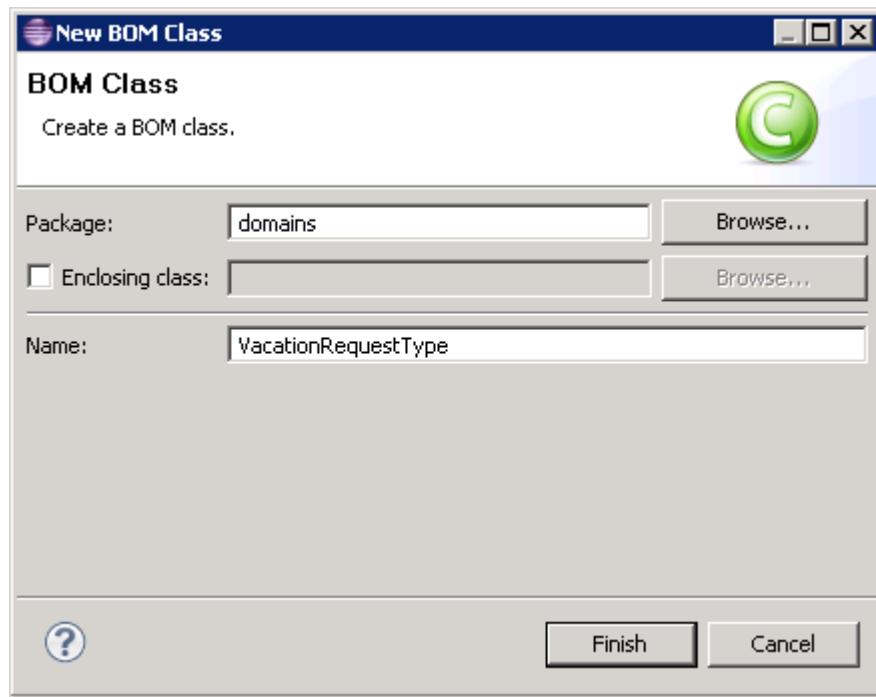
- \_\_ e. In your workspace, rename the `vacationRequestTypes-1.xls` file as: `vacationRequestTypes.xls`

**Hint**

In Rule Designer, to rename a file, click it and press F2 to open the Rename Resource window. Or, you can right-click the file, and click **Refactor > Rename**.

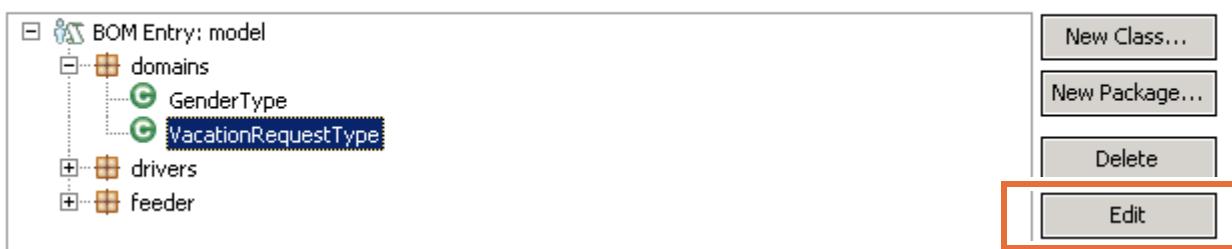
In the Rename Resource window, you type a value in the **New name** field and click **OK** to close the window and rename the file.

- \_\_ 3. Expand `trucksAndDrivers-bom > bom > model` and double-click **domains** to open the **domains** package in the BOM editor.
- \_\_ 4. In the **Business Object Model Entry: model** section, make sure that **domains** is selected and click **New Class** to create a BOM class called: `VacationRequestType`

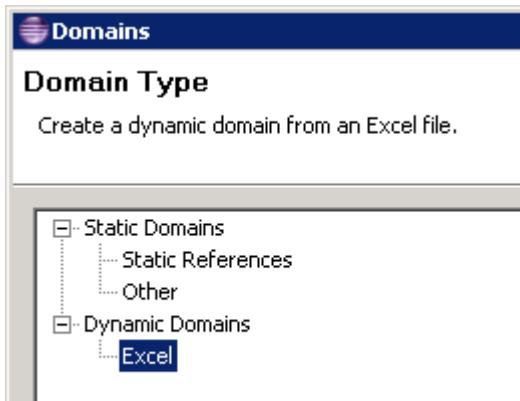


5. Edit the new `VacationRequestType` BOM class in the BOM editor to associate a dynamic domain with that class.

### Business Object Model Entry

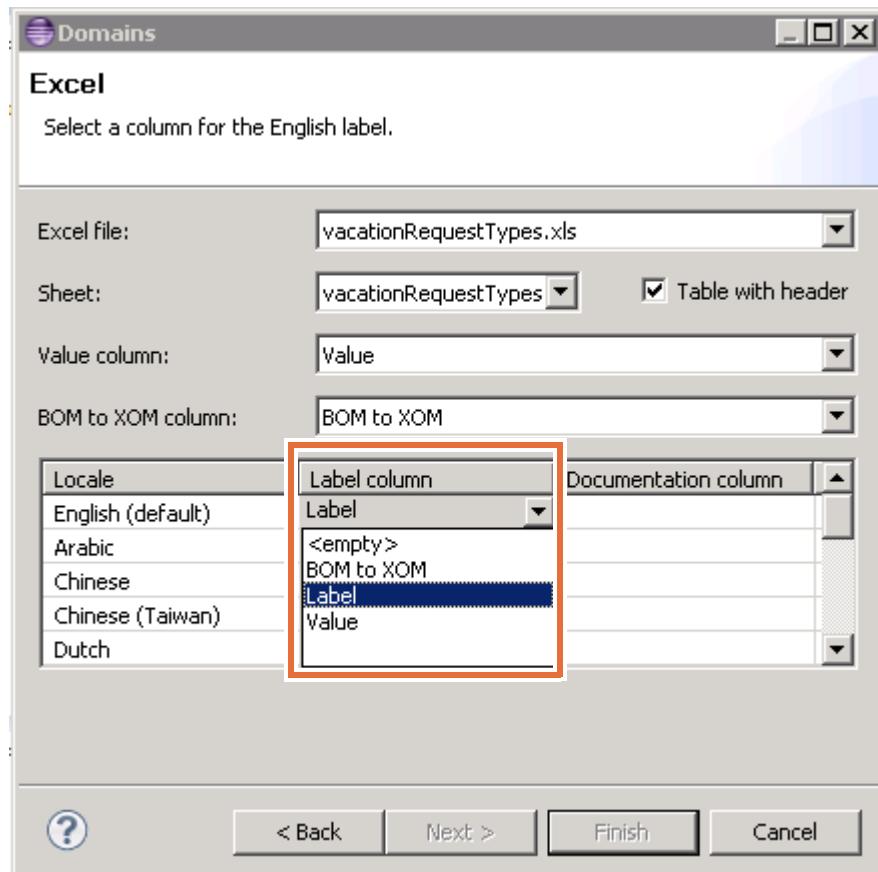


- \_\_\_ a. In the **Domain** section, click **Create a domain**.  
The Domains window opens.
- \_\_\_ b. In the Domains window, select **Excel**, and click **Next**.



- The Excel pane of the Domains window opens.
- \_\_\_ c. In the Excel pane, set the **Excel file** field to `vacationRequestTypes.xls`, which is the only available choice in the list.  
This choice is available in this window because the `vacationRequestTypes.xls` file is present in the `resources` folder of the BOM project.  
The `vacationRequestTypes.xls` file contains only the `vacationRequestTypes` sheet. Rule Designer automatically sets the value of the **Sheet** field to the name of that unique sheet.
  - \_\_\_ d. Select **Table with header** to indicate that the first row in the `vacationRequestTypes` sheet is the header for the columns in that sheet.  
It is a good practice to set a header row in the Excel spreadsheet for dynamic domains. This header helps distinguish the purpose of each column. Also, Rule Designer uses the header values as indications for you to select the appropriate columns when you configure the dynamic domain, as you see next.
  - \_\_\_ e. In the **Value column** field, select **Value**, which is the header name for the column that gives the values of the dynamic domain in the spreadsheet.
  - \_\_\_ f. In the **BOM to XOM column** field, select **BOM to XOM**, which is the header name for the column that provides the BOM-to-XOM mapping in the spreadsheet.

- \_\_\_ g. In the table, click the cell at the intersection of the row **English (default)** and the column **Label column**, and select **Label** in the list.

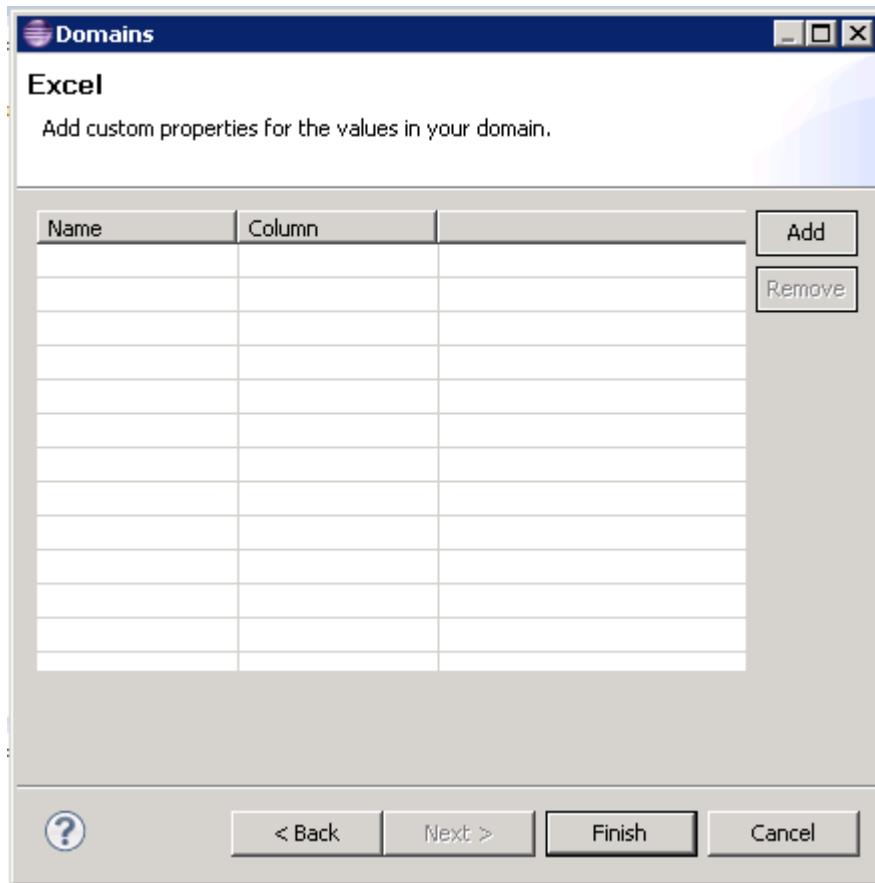


**Label** is the header name for the column that gives the labels of the dynamic domain in the spreadsheet.

The `vacationRequestTypes.xls` file does not define a documentation column, so you can leave the cells of the **Documentation column** empty.

- \_\_\_ h. Click **Next**.

The next pane opens, where you can add custom properties. For this domain, you have no properties.



- \_\_\_ i. Click **Finish** to close the Domains window.
- \_\_\_ 6. Save the BOM.

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

Class page. However, the Problems view indicates an error because the XOM does not

▼ Domain

Create and edit a domain for this class.

>Edit the domain.

Remove the domain.

**Domain type: Excel**

Synchronize with dynamic values.

- Administrative
- Compensatory
- Educational
- FamilyPersonal
- JuryDuty
- Military
- Overtime
- Pregnancy
- Recognition
- Sick
- Strike
- Vacation
- VolunteerFireAndRescue

include a domains.VacationRequestType class to translate your new BOM class.



**Questions**

How can you resolve this problem?

**Answer**

The domains.VacationRequestType class in the XOM does not exist. To resolve this problem, set the **Execution name** in the BOM-to-XOM mapping of the domains.VacationRequestType BOM class to an existing class.



**Questions**

Can you figure out which class you must use to set the **Execution name** in the BOM-to-XOM mapping?

**Hint**

Look at the content of the cells in the **BOM to XOM** column in the `vacationRequestTypes.xls` file.

**Answer**

The class that you require depends on the type for the values that the BOM-to-XOM mapping returns. In the `vacationRequestTypes.xls` file, the **BOM to XOM** column defines a BOM-to-XOM mapping. For example, the mapping for Jury Duty is:

```
return "V78";
```

The execution values associated with the values in the BOM dynamic domain are `String` objects, such as `"V78"`. The **Execution name** in the BOM-to-XOM mapping must be: `java.lang.String`

- \_\_\_ 7. To correct the problem in the `VacationRequestType` BOM class, define the BOM-to-XOM mapping.
  - \_\_\_ a. Scroll down to the **BOM to XOM Mapping** section and expand this section.
  - \_\_\_ b. In the **Execution name** field, click **Browse**.
  - \_\_\_ c. In the **Choose a type** field, type `String`, select `String` from the list, and click **OK**. The type field is now set to: `java.lang.String`

**▼ BOM to XOM Mapping**

Edit the mapping between this BOM class and the XOM.

Execution name:

Extender name:

- \_\_\_ 8. Save the BOM.
- \_\_\_ 9. Verify that the Problems view no longer indicates any errors on this new class. (You can ignore any warnings for now.)

### 1.3. Examining the dynamic domain in Rule Designer

In this section, you continue working in the BOM editor to examine your new dynamic domain.

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

**▼ Custom Properties**

Define custom properties for this class.

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

- The `domainProviderResource` property gives the name of the Excel spreadsheet that is used as the source for the values of the dynamic domain.

You must modify this value if you want to change the source Excel spreadsheet for your domain.

- The `domainValueProviderName` property gives the name of the classes that are used to manipulate this spreadsheet and transform its values into the dynamic domain.

- \_\_\_ 2. In the **Members** section, verify that you can see all the values that are defined in the `vacationRequestTypes.xls` file.

- \_\_\_ a. Double-click the **VolunteerFireAndRescue** BOM member to open it.
- \_\_\_ b. Verify that this BOM member is both static and final.

**General Information**

|                                                                                                              |                             |           |
|--------------------------------------------------------------------------------------------------------------|-----------------------------|-----------|
| Name:                                                                                                        | VolunteerFireAndRescue      |           |
| Type:                                                                                                        | domains.VacationRequestType | Browse... |
| Class:                                                                                                       | domains.VacationRequestType | Browse... |
| <input type="radio"/> Read/Write <input checked="" type="radio"/> Read Only <input type="radio"/> Write Only |                             |           |
| <input checked="" type="checkbox"/> Static <input checked="" type="checkbox"/> Final                         |                             |           |
| <input type="checkbox"/> Deprecated <input type="checkbox"/> Update object state                             |                             |           |
| <input type="checkbox"/> Ignore for DVS                                                                      |                             |           |

- \_\_\_ c. Verify that the **Type** of this BOM member is `domains.VacationRequestType`.
- \_\_\_ d. Verify that the verbalization of this BOM member is Volunteer Fire and Rescue as shown in the Microsoft Excel file.

**▼ Member Verbalization**

~~✖ Remove~~ the verbalization.

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

- \_\_\_ e. Expand the **BOM to XOM Mapping** section and verify that the getter method of this BOM member is as shown in the Microsoft Excel file:

```
return "K29";
```

▼ **BOM to XOM Mapping**  
Edit the mapping between this BOM member and the XOM.

✎ [Edit the imports.](#)

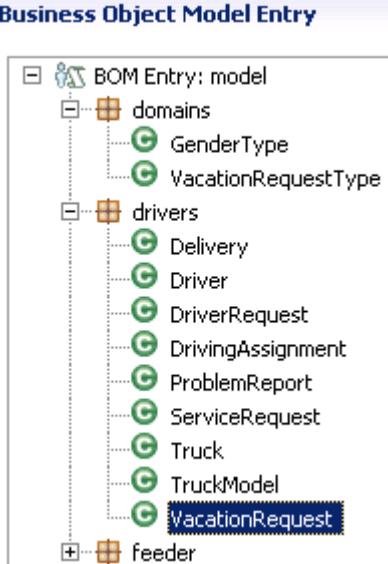
▼ Getter

```
return "K29";
```

## Section 2. Using the dynamic domain in a rule

Now that the dynamic domain exists in the BOM of the rule project, use it to complement the `VacationRequest` BOM class per the request from the human resources department. You then author rules that are based on this type.

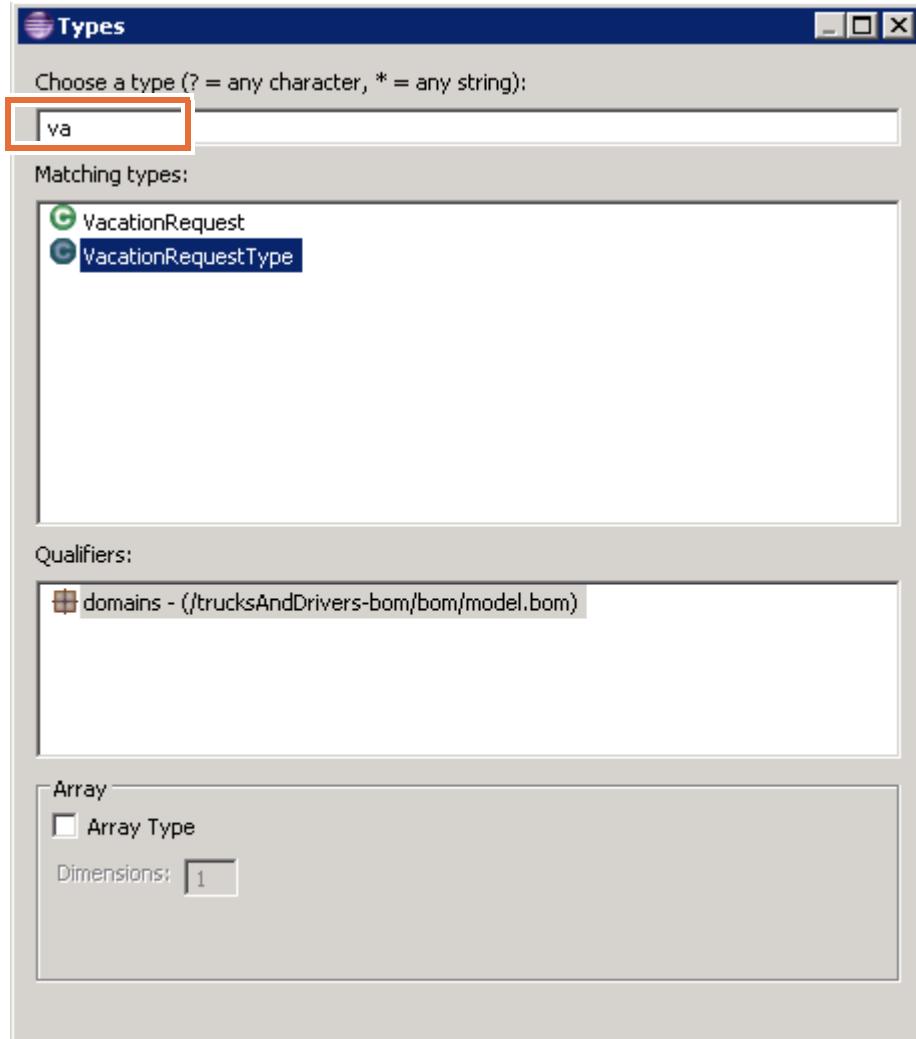
- 1. First, you update the `VacationRequest` BOM class by adding a `type` BOM attribute that business users can read and that is based on the dynamic domain.
  - a. Return to the Package page of the BOM editor and double-click the `drivers.VacationRequest` BOM class to open it in the Class page.



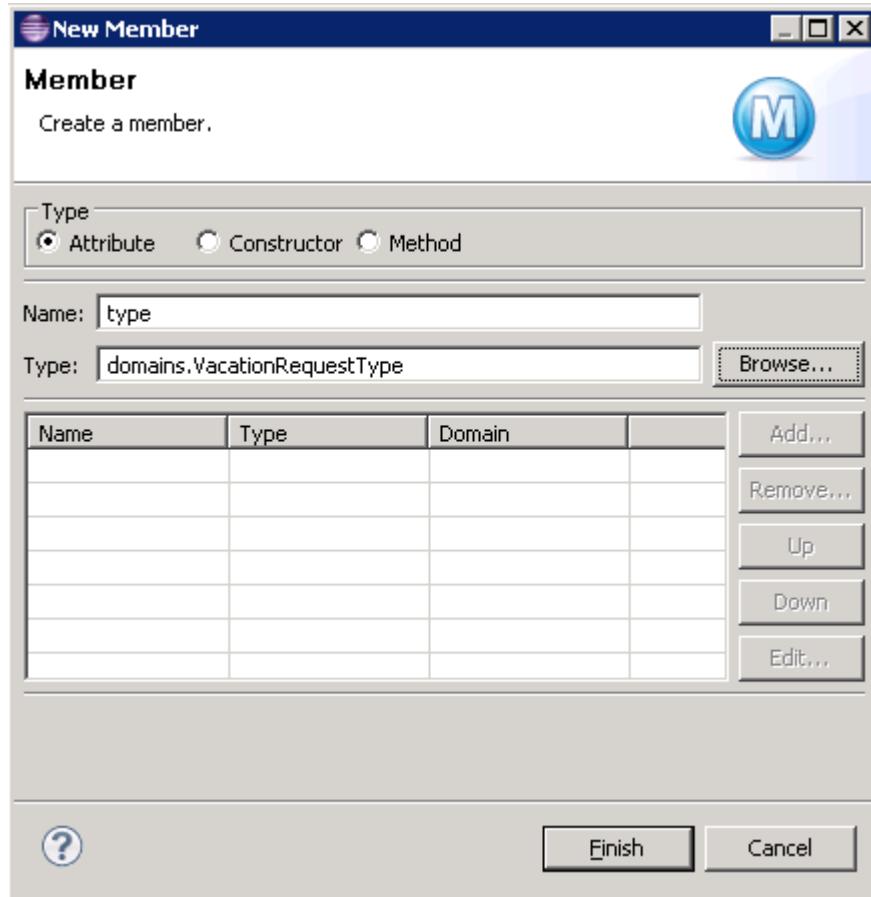
- b. In the **Members** section, add a BOM attribute named: `type`
- c. Set the **Type** field to: `domains.VacationRequestType`

**Hint**

If you click **Browse** to find the type, you can filter the list of options by starting to type the class name that you are looking for.



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



- \_\_ e. After the `type` attribute is created, set it to **Read Only**.

- \_\_ f. Create a default verbalization for it.

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

| General Information                                                                                                 |                                                                                    | Member Verbalization                                                         |  |
|---------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------|--|
| Name:                                                                                                               | <input type="text" value="type"/>                                                  | ⚠ This member is not verbalized <a href="#">Create</a> default verbalization |  |
| Type:                                                                                                               | <input type="text" value="domains.VacationRequestType"/> <a href="#">Browse...</a> |                                                                              |  |
| Class:                                                                                                              | <input type="text" value="drivers.VacationRequest"/> <a href="#">Browse...</a>     |                                                                              |  |
| <input type="radio"/> Read/Write <input checked="" type="radio"/> <b>Read Only</b> <input type="radio"/> Write Only |                                                                                    |                                                                              |  |

- \_\_ 2. Save the BOM.

 **Questions** \_\_\_\_\_

Are you done preparing this attribute?

**Answer**

Your new `type` attribute for the `VacationRequest` BOM class is not associated with any attribute in the XOM. You cannot execute this business rule solution without an error at run time.

The Problems view shows this error:

B2X cannot find attribute "type" in execution class "drivers.VacationRequest"

You solve this problem later in this exercise, in "Updating the XOM" on page 10-23. For now, ignore the error and try to use the new dynamic domain in a rule.

- 3. Now, write an action rule that uses this new `type` BOM attribute in Rule Designer.
- a. Create an action rule that is named `05_dynamic_domain` in the `domains` rule package of the `trucksAndDrivers-rules` project, with the following code:
 

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

**Hint**

You can find the code for this rule in the `create_domains.txt` file in the `<LabfilesDir>\code` directory.

- \_\_\_ b. Notice how the rule editor shows the verbalization of the values in the dynamic domain, while you edit one of { }.

**if**

the type of 'a vacation request' is one of { **Administrative**, Compensatory, Edu

**then** Domain value:

Administrative : domains.VacationRequestType

**print**

Administrative

Compensatory

Educational

Family/Personal

Jury Duty

Military

Overtime

Pregnancy

Recognition

Sick

...

- \_\_\_ 4. Save the rule.

## Section 3. Updating the dynamic domain



### Requirements

The human resources department indicates that the `Compensatory` value is no longer valid and must be replaced with the `Other Vacation` value. You must update the dynamic domain by modifying the Excel file and dynamically updating the dynamic domain of the BOM in Rule Designer. You also verify the consequences of the update.

#### To modify the source Excel spreadsheet and update the dynamic domain:

- 1. In Rule Designer, import the `<LabfilesDir>\code\vacationRequestTypes-2.xls` file into the `resources` folder of the `trucksAndDrivers-bom` project.
- 2. Open the `vacationRequestTypes-2.xls` file, and verify that its content is the same as the content of the `vacationRequestTypes.xls` file, except for the line:

**Value:** Compensatory

**BOM to XOM:** return "B34" ;

**Label:** Compensatory

This line is no longer present, and is replaced with a line that defines the `Other` value, with the following three cells:

**Value:** Other

**BOM to XOM:** return "O22" ;

**Label:** Other Vacation

- 3. Close the `vacationRequestTypes.xls` file, and in Rule Designer, rename this file back to: `vacationRequestTypes-1.xls`
- 4. Close the `vacationRequestTypes-2.xls` file, and in Rule Designer, rename this file as `vacationRequestTypes.xls` to use this new Excel spreadsheet as the source for the values in the dynamic domain.

- \_\_\_ 5. In the BOM editor, open the `VacationRequestType` BOM class, and click **Synchronize with dynamic values** in the Domain section.

**Domain type: Excel**

 [Synchronize](#) with dynamic values.

- Administrative
- Compensatory
- Educational
- FamilyPersonal
- JuryDuty
- Military
- Overtime
- Pregnancy
- Recognition
- Sick
- Strike
- Vacation
- VolunteerFireAndRescue

The removed `Compensatory` value is still visible, both under **Synchronize with dynamic values** and in the Members section. However, this value is marked as *deprecated* in the BOM editor (as you verify next).

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



**Important**

The Members section lists all the static references that are, or were, part of this dynamic domain. The references that correspond to values that are no longer present in the dynamic domain are marked as *deprecated*. Deprecated values are not removed and might still be used to author rule artifacts. However, because they are marked as deprecated, these values are underlined in the edited rule artifacts that use them, and associated warnings are added in the Problems view of Rule Designer.

- \_\_\_ 6. Save the BOM.
- \_\_\_ 7. Verify that the `Compensatory` value is deprecated.
- \_\_\_ a. In the Problems view, verify that the following warning messages exist:
- 'Compensatory' is deprecated
  - [BOM] GBRMO0011W: Member domains.VacationRequestType.Compensatory is deprecated
- \_\_\_ b. In the **Members** section of the `VacationRequestType` BOM class, double-click the `Compensatory` member.
- \_\_\_ c. In the Member page for the `Compensatory` BOM member, verify that the **Deprecated** check box is selected.

After modifying the values of the dynamic domain, you must verify that the rule artifacts based on this dynamic domain are still correct. In the present case, you must verify only the 05\_dynamic\_domain rule.

- 8. Reopen the 05\_dynamic\_domain rule in the domains rule package of the trucksAndDrivers-rules project.
  - a. Verify that the term Compensatory is underlined, and that the associated warning is the same as in the Problems view:  
'Compensatory' is deprecated
  - b. In the **if** and **then** parts of the rule, replace Compensatory with Other Vacation.

```
if
 the type of 'a vacation request' is one of { Administrative , Compensatory ,
then
 print "The driver " + the name of 'the driver' + " requested v
 
```

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

You obtain the following rule:

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

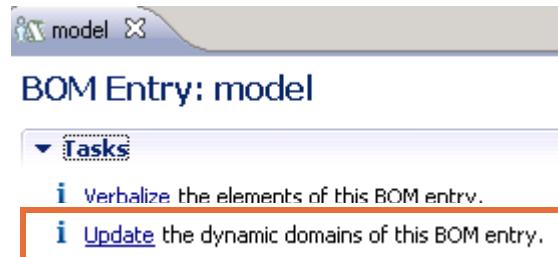
- 9. Save the 05\_dynamic\_domain rule.
- 10. Verify that the 'Compensatory' is deprecated warning is no longer present in the Problems view.
- 11. Close the rule.



## Information

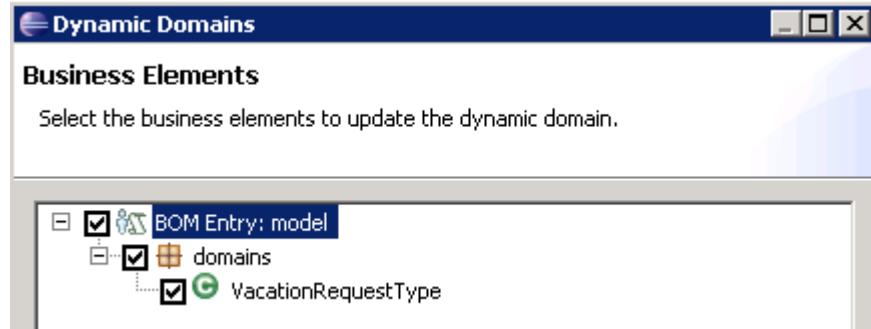
If you define multiple dynamic domains in a BOM entry, you can update or repopulate the values of all of them from the Package page of the BOM editor.

- 1. In the Package page of the BOM editor, click **Update the dynamic domains of this BOM entry** in the Tasks section.



The Dynamic Domains window opens.

- 2. In the Dynamic Domains window, select the check boxes that are associated with the dynamic domains you want to update.



- 3. Click **Finish** and save your work.

The Dynamic Domains window closes, and Rule Designer updates the selected dynamic domains.

## Section 4. Updating the XOM

You must now make sure that the complete business rule solution is executable by updating its XOM to reflect the additions in the BOM.

First, you identify the inconsistency in the rule project.

- \_\_\_ 1. Reopen the Problems view (if it is closed) by clicking **Window > Show View > Other > Problems**.
- \_\_\_ 2. Verify that the Problems view contains the following message:

[B2X] Cannot find attribute "type" in execution class  
"drivers.VacationRequest"

This message indicates that the `type` attribute cannot be found in the XOM class that is associated with the `drivers.VacationRequest` BOM class.

- \_\_\_ 3. If you cannot see this error message in the Problems view, make sure that the BOM is selected against the XOM while you build the project, and rebuild the project.
  - \_\_\_ a. Go to **Window > Preferences**, and then go to **Rule Designer > Build**.
  - \_\_\_ b. In the Build page, make sure that the **Perform BOM to XOM checks during build** check box is selected, and then click **OK**.
  - \_\_\_ c. Click **Project > Clean** to clean and rebuild all projects in your workspace.

**To correct the inconsistency problem:**

- \_\_\_ 1. Expand **trucksAndDrivers-xom > src > drivers** and open the `VacationRequest` class to add a private attribute:

- **Name:** type
- **Type:** String

- \_\_\_ 2. Create the corresponding setter and getter methods.

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

```
public String getType() {
 return type;
}
```

- \_\_\_ 3. Save the XOM.

The error that is related to this inconsistency in the BOM-to-XOM mapping is no longer visible in the Problems view.

- \_\_\_ 4. Close the `VacationRequest.java` file.

## Section 5. Publishing the BOM and rule projects to Decision Center

In this section, you publish the `trucksAndDrivers-bom` project and the `trucksAndDrivers-rules` project to Decision Center to create the corresponding projects in Decision Center.

### 5.1. Publishing the BOM and the rules

- \_\_\_ 1. If the sample server is not started, start it now by using the **Start server** desktop shortcut or by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.

Starting the server might take several minutes.

- \_\_\_ 2. Publish the `trucksAndDrivers-bom` project:

- \_\_\_ a. In the Rule Explorer, right-click the `trucksAndDrivers-bom` project and click **Decision Center > Connect**.

The Decision Center configuration wizard opens.

- \_\_\_ b. Enter the connection entries:

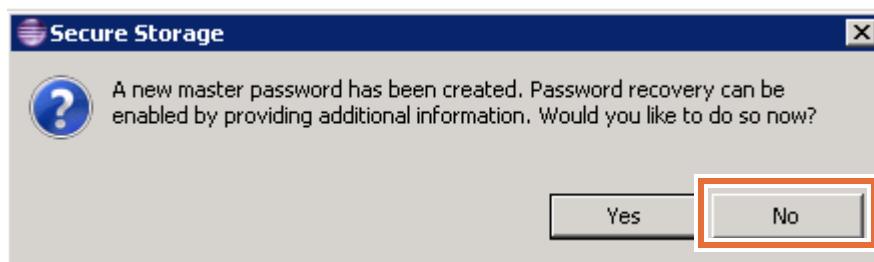
- **URL:** `http://localhost:9080/teamserver`
    - **User name:** `rtsAdmin`
    - **Password:** `rtsAdmin`

- \_\_\_ c. Click **Connect**.

- \_\_\_ d. In the Project configuration section, make sure that **Create a new project on Decision Center** is selected because the `trucksAndDrivers-bom` project does not exist yet on Decision Center.

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

- \_\_\_ f. If you see a Secure Storage window, click **No**.



- \_\_\_ g. When synchronization completes, no changes are found, so you can click **OK** to close the Synchronize Complete window.

- \_\_\_ h. Click **No** when prompted to switch to the Team Synchronizing perspective.

You do not have to open the Team Synchronizing perspective because it is empty.

- \_\_\_ 3. Repeat Step 2 to publish the `trucksAndDrivers-rules` project to Decision Center.

**Important**

After you publish the `trucksAndDrivers-rules` project to Decision Center, do not disconnect. By keeping Rule Designer and Decision Center connected, you do not have to establish this connection again in the next steps.

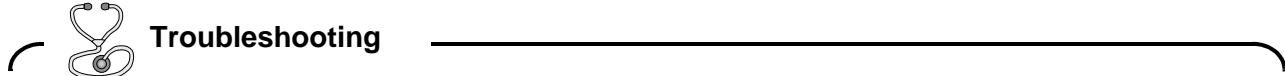
## Section 6. Examining rules in Decision Center

In this section, acting as a business user in Decision Center, you work with the dynamic domain and the rule artifacts you published from Rule Designer.

### 6.1. Opening the published projects in Decision Center

- \_\_ 1. Sign in to Decision Center as an administrator.
  - \_\_ a. Start Mozilla Firefox and type the following URL into a browser:

http://localhost:9080/teamserver



Make sure that you use Mozilla Firefox as the browser.

- \_\_ b. Sign in with rtsAdmin for both the **Username** field and the **Password** field.
- \_\_ 2. In the **Home** tab, switch to the trucksAndDrivers-rules project.
  - \_\_ a. Make sure that **Work on a project** is selected.
  - \_\_ b. In the **Project in use** field, select **trucksAndDrivers-rules**.

A screenshot of the IBM Decision Center interface. The top navigation bar has tabs for Home, Explore, Compose, Query, and Analyze. The Home tab is active. Below the navigation bar, the title "Welcome to the Decision Center Home Page" is displayed. Underneath the title, there is a configuration section:

- Work on a project
  - Project in use:
  - Branch in use:
  - Current action:

- \_\_ 3. Click the **Explore** tab, and click the **domains** folder.

4. Select the `05_dynamic_domain` action rule and click **Quick Edit** icon to open the rule in the editor.

|                                     | Actions | Name                                        |
|-------------------------------------|---------|---------------------------------------------|
| <input type="checkbox"/>            |         | <a href="#">01_collection_domain</a>        |
| <input type="checkbox"/>            |         | <a href="#">02_literal_domain</a>           |
| <input type="checkbox"/>            |         | <a href="#">03_static_references_domain</a> |
| <input type="checkbox"/>            |         | <a href="#">04_bounded_domain</a>           |
| <input checked="" type="checkbox"/> |         | <a href="#">05_dynamic_domain</a>           |

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

```

Name* 05_dynamic_domain
Status* New
definitions
set the driver to ▼ a driver [from/in] ✕
 [where] ✕
set a vacation request to ▼ a vacation r
 [where] ✕
if
 the type of a vacation request is one of
then
 print ▼ "The driver " + the name of the c
 " requested vacation for Administrative, C
if
then
else
end if
end if
end rule

```

6. Click **Cancel** to exit the edit mode.  
 7. On the **Home** tab, switch to the `trucksAndDrivers-bom` project in the **Project in use** field.  
 8. Click the **Explore** tab.



### Questions

Do you see the `vacationRequestType.xls` file that you added to the `trucksAndDrivers-bom` project in Rule Designer? If not, how can you access it?

**Answer**

The `vacationRequestType.xls` file is stored in the `resources` folder of the `trucksAndDrivers-bom` project. However, Decision Center does not have a default smart folder that lists the content of the `resources` folder. To see the `vacationRequestType.xls` file, you must create a smart folder that lists the content of the `resources` folder.

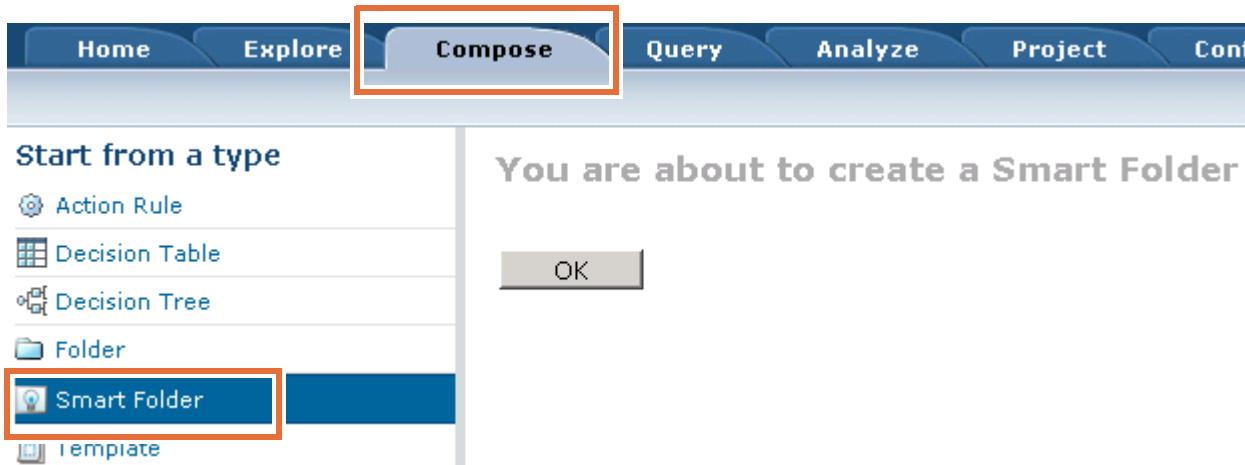
**Information**

The `resources` folder of a rule project is not automatically accessible in Decision Center when you publish the project from Rule Designer. This folder is used mainly for technical data that must be shared between Decision Center and Rule Designer. Hiding this folder in Decision Center can help prevent non-technical business users from modifying the data by mistake.

## 6.2. Creating a Resources smart folder

In this section, you learn how to create a smart folder that lists the content of the `resources` folder.

- 1. If you are not already in the `trucksAndDrivers-bom` project, go to the **Home** tab and change the **Project in use** to: `trucksAndDrivers-bom`
- 2. On the **Compose** tab, select **Smart Folder** in the **Start from a type** list and click **OK**.

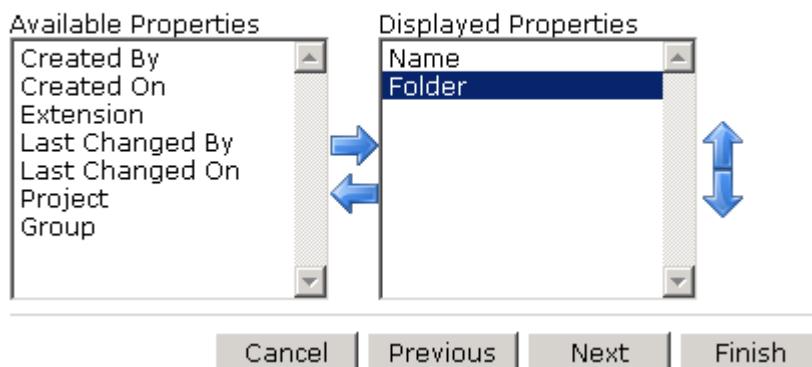


- 3. In the Properties page, enter `Resources` in the **Name** field and click **Next**.
- 4. In the Query page, write the query to define this smart folder.
  - a. In the **Find** statement, click **all business rules** and select **all resources**.
  - b. Click **Next**.

5. In the Displayed Properties page, set up the list of the elements that you want the smart folder to display.
- In the Available Properties section, select **Folder** and click the right-arrow that goes from the **Available Properties** list on the left to the **Displayed Properties** list on the right.
  - Repeat to add the **Name** property to the Displayed Properties list.
  - Make sure that **Folder** is listed before **Name** in the **Displayed Properties** section.

Use the up-arrow to move the property, if required.

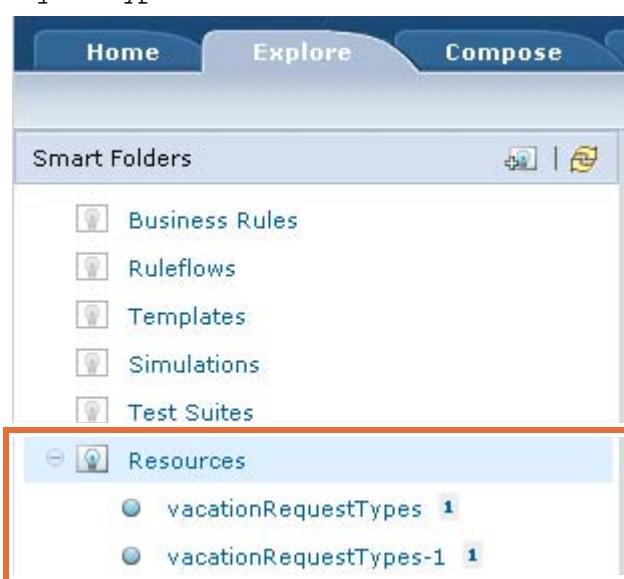
### Displayed Properties



- Click **Next**.
- In the Documentation pane, enter a description for this folder, such as:  
Smart folder to list the content of the resources folder
- Click **Finish**.

The **Resources** smart folder is now listed on the **Explore** tab and should contain the two Excel spreadsheets that you imported in Rule Designer:

- vacationRequestTypes
- vacationRequestTypes-1



## Section 7. Modifying the dynamic domain in Decision Center

You learned how to use Excel source files for dynamic domains in rules that are authored in Rule Designer. Next, you see how the domain can be modified and used in rules that are authored in Decision Center.

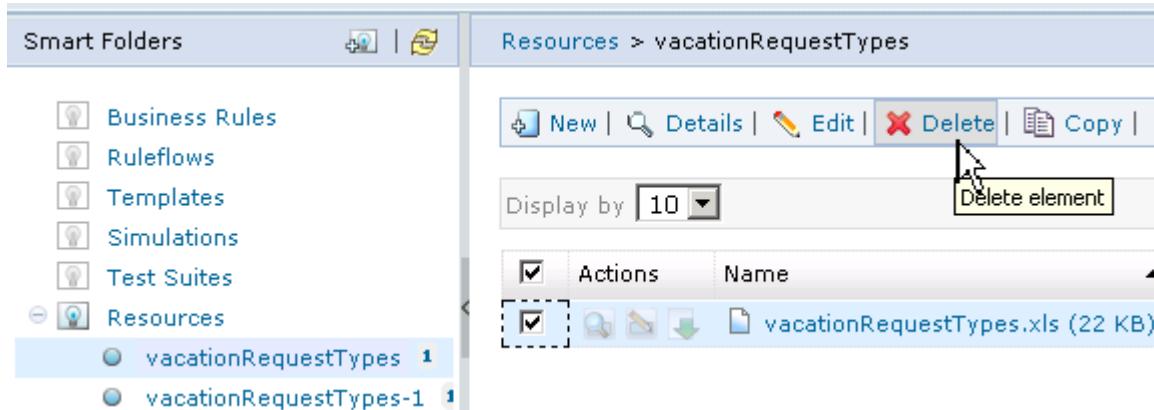


### Note

If you are using the VMware image that was developed for this course, you cannot modify Excel files. Instead, the modified domain values are provided in the `vacationRequestTypes-3.xls` file in the `<LabfilesDir>\code` directory. You delete the Excel file that is in use and replace it with: `vacationRequestTypes-3.xls`

### 7.1. Modifying the values of the dynamic domain

- 1. In the `<LabfilesDir>\code` directory, open the `vacationRequestTypes-3.xls` Excel file to view the modified domain values.  
The first line, which defined the `Administrative` value, is now replaced with the `Training` value.
- 2. Make a backup copy of this file by renaming it as: `vacationRequestTypes-3-copy.xls`
- 3. Rename the `vacationRequestTypes-3-copy.xls` file as: `vacationRequestTypes.xls`
- 4. In Decision Center, delete the existing `vacationRequestTypes.xls` file in the `trucksAndDrivers-bom` project.
  - a. On the **Explore** tab, expand the **Resources** smart folder and click the `vacationRequestTypes.xls` file.
  - b. In the table of artifacts, select **vacationRequestTypes.xls**, and click **Delete** on the toolbar.



- c. When prompted to confirm the deletion, click **Yes**.
- 5. Import the new file to Decision Center.
  - a. On the **Explore** tab, click the **Resources** smart folder and click **New** on the toolbar.

- \_\_\_ b. On the Properties page, click **Browse** next to the **File** field, select the <LabfilesDir>\code\vacationRequestTypes.xls Excel spreadsheet, and click **Open**.
  - \_\_\_ c. Click **Finish**.
  - \_\_\_ d. Click the **Explore** tab and verify that the vacationRequestTypes.xls file is now visible in the **Resources** smart folder.
- \_\_\_ 6. Click the **Project** tab.
- \_\_\_ 7. On the **Project** tab, under **Business Object Model**, click **Update dynamic domains**.  
The Dynamic Domains page opens.
- \_\_\_ 8. Select **domains.VacationRequestType**.

You can expand `domains.VacationRequestType` to see the current list of domain values.

- \_\_\_ 9. Click **Update** on the toolbar.
- \_\_\_ 10. Click **OK** after the domains are successfully reloaded.
- \_\_\_ 11. Expand **domains.VacationRequestType** again to see the updated list.

## 7.2. Modifying the domain value in the rule

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

- \_\_\_ 1. On the **Home** tab, switch to the `trucksAndDrivers-rules` project as the **Project in use**.

2. On the **Explore** tab, click the Preview icon for the `05_dynamic_domain` rule in the **domains** folder to confirm that the rule has a warning.

**Rule Preview**

**Edit**

**Name** 05\_dynamic\_domain  
**Status** New

**'Administrative' is deprecated.**

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

**Questions**

Why is there a warning?

**Answer**

The `Administrative` value no longer exists in the dynamic domain, so it is marked as deprecated (similar to what occurs in Rule Designer).

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

**Rule Preview**

**Edit**

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

**if**

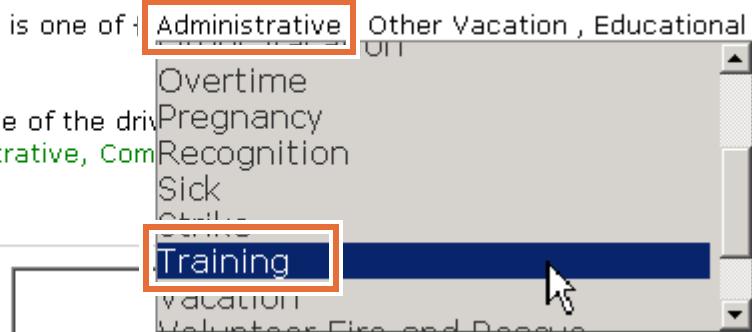
the type of a vacation request is one of { **Administrative** Other Vacation , Educational }

**then**

print "The driver " + the name of the driver  
" requested vacation for **Administrative**, Com

**[else]**

Add a comment to this version



- \_\_\_ 5. The rule should state:

```
definitions
 set 'the driver' to a driver ;
 set 'a vacation request' to a vacation request in the vacation requests of
 'the driver' ;

if
 the type of 'a vacation request' is one of { Training, Other Vacation,
 Educational }

then
 print "The driver " + the name of 'the driver' + " requested vacation
 for Training, Other Vacation or Educational reason";
```

- \_\_\_ 6. Save the rule 05\_dynamic\_domain and verify that it no longer has a warning.

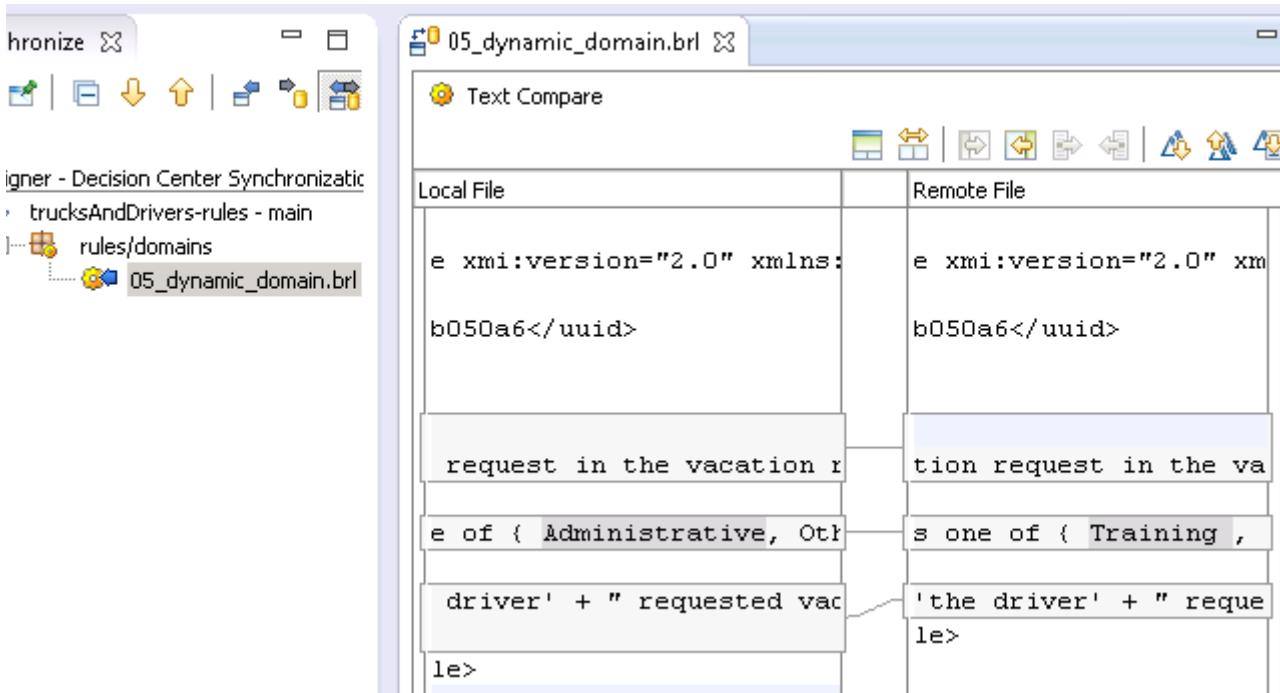
- \_\_\_ 7. Sign out and close the Enterprise console.

## Section 8. Updating Rule Designer from Decision Center

In Decision Center, you changed both the rule project and the BOM. You must now update the Rule Designer copy of the BOM and rule project by synchronizing with Decision Center.

### 8.1. Synchronizing the rule project

- 1. In Rule Explorer, synchronize the trucksAndDrivers-rules project with Decision Center.
    - a. Right-click the trucksAndDrivers-rules project, and click **Decision Center > Synchronize with Decision Center**.
    - b. In the Synchronization Settings window, click **Finish**.
  - 2. When prompted to open the Team Synchronize perspective, click **Yes**.
  - 3. In the Team Synchronize perspective, expand **trucksAndDrivers-rules** to view which artifacts must be synchronized.
- The `05_dynamic_domain` rule is listed. The blue arrow that points left indicates that Decision Center has a more recent version than Rule Designer. The suggested action is to *update* Rule Designer.
- 4. Double-click `05_dynamic_domain.brl` to open it in the Text Compare view.



- 5. In the Team Synchronize perspective, right-click the rule `05_dynamic_domain` and click **Update** to update this rule with the latest changes from Decision Center.
- 6. Return to the Rule perspective, and open the rule `05_dynamic_domain` in the rule editor.

7. Verify that the value `Training` is underlined and that, when moving the cursor over this value, the following error is visible:

Variable 'Training' is not declared

**Content**

```

definitions
 set 'the driver' to a driver ;
 set 'a vacation request' to a vacation request in the vacation request;
 then
 print "The driver " + the name of 'the driver'

```

**Error : Variable 'Training' is not declared**. 'request' is one of { **Training** , Other Vacation }

- ✖ Add a new variable 'Training'
- ✖ Did you mean 'a vacation request' ?
- ✖ Did you mean 'the driver' ?
- ✖ Did you mean 'the truck' ?
- ✖ Remove this 'item'
- ✚ Insert a 'item' before
- ✚ Insert a 'item' after



### Questions

Why does this error exist?

### Answer

The error exists because you added `Training` in the dynamic domain and used this value in the rule in Decision Center, but you did not yet update the BOM project in Rule Designer.

The next step would be to synchronize the BOM project that contains the dynamic domain.

## End of exercise

## Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 10: Dynamic domains > 02-answer**. The solution project does not include the Training vacation request type.

This exercise looked at how to define and use a dynamic domain, how to update dynamic domains in Decision Center, and how to synchronize them between Rule Designer and Decision Center. You also saw the effects of changing dynamic domain values in both Decision Center and Rule Designer.



### Information

As you learned, the values of the dynamic domains are shared between business users and developers, and can be updated independently. To ensure consistency across projects, you must establish governance guidelines:

- Clarify who can make these types of updates, and whether to enforce restrictions on when and how updates are made.
- Make sure that all roles that are involved in the business rule solution are aware of these changes, and update their working environment to reflect the changes.



### Important

If you want to redo this exercise, you must delete the projects that you published to Decision Center.

- 1. If the Decision Center console is closed, sign in again with `rtsAdmin` as the user name and password.
- 2. Delete the `trucksAndDrivers-rules` project:
  - a. On the **Home** tab, select the `trucksAndDrivers-rules` project as the project in use.
  - b. In the **Configure** tab, under Administration, click **Erase Current Project** and click **Yes** when prompted to confirm the project deletion.
- 3. Repeat step 2 to delete the `trucksAndDrivers-BOM` project.

# Exercise 11. Queries and ruleset extraction

## What this exercise is about

This exercise teaches you how to define queries and rule extractors on rule projects.

## What you should be able to do

After completing this exercise, you should be able to:

- Search for rule artifacts and find rules according to their dependencies
- Define and run queries and apply actions on query results
- Create query-based ruleset extractors to extract ruleset archives

## Introduction

In many cases, you must analyze your ruleset to find dependencies between your rules. In this exercise, you search your rule project for rule artifacts that have specific properties, or rules that can enable or disable some other rule artifacts. You also learn how to create and run queries on rule projects.

In some cases, you must extract a subset of your rule artifacts to create the ruleset archive. For this purpose, you create ruleset extractors that are based on queries, and use these ruleset extractors to create the ruleset archives that match your requirements.

The exercise involves these tasks:

- Section 1, "Searching for rule artifacts"
- Section 2, "Querying rule projects"
- Section 3, "Extracting a ruleset archive"

## Requirements

There are no specific requirements for this exercise.

## Section 1. Searching for rule artifacts

In this part of the exercise, you search for rule artifacts within a rule project. You also search for which rules are enabled or disabled by others, or which ruleflows might select them.

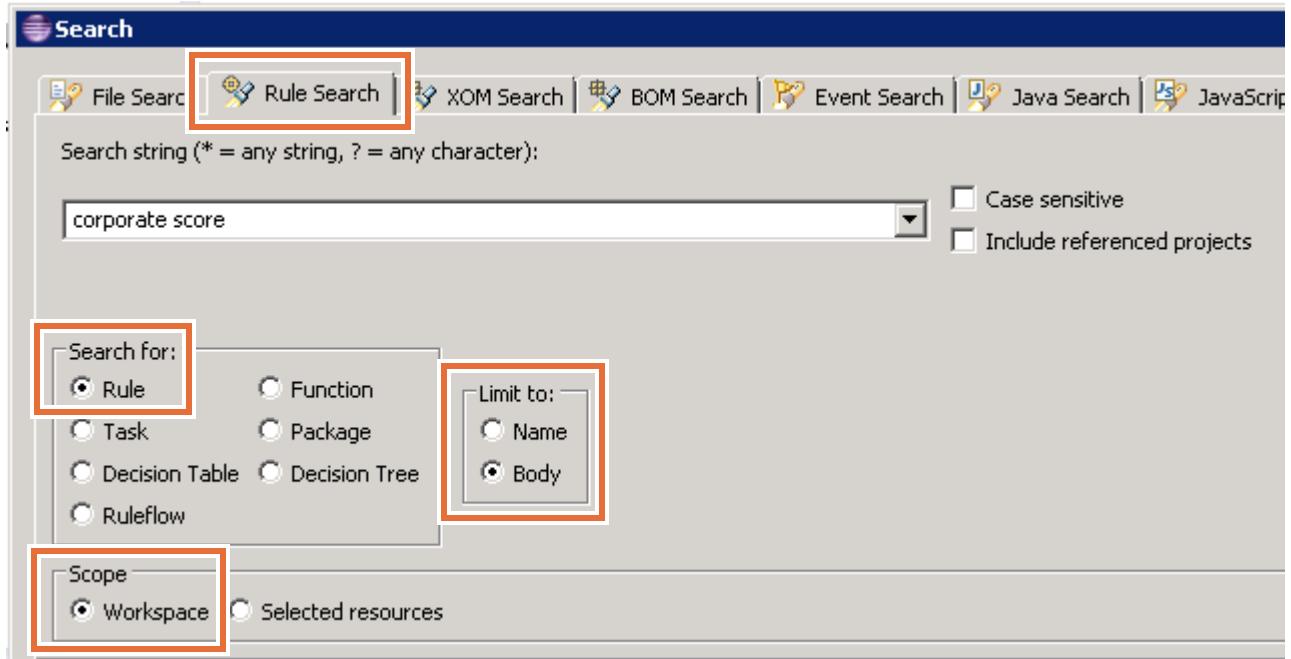
### 1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace:
  - a. From the **File** menu, click **Switch Workspace > Other**.
  - b. In the **Workspace Launcher** window, enter the path:  
`<LabfilesDir>\workspaces\queries`
- 2. Close the Welcome view and switch to the Samples Console perspective.
- 3. Select **Rule Designer > Training > Ex 11: Queries and ruleset extraction > 01-start > Import projects**.
- 4. Close the Help pane.

### 1.2. Searching for specific criteria across the rules

- 1. From the **Search** menu, click **Search** to open the Search window.
- 2. In the Search window, click the **Rule Search** tab and take some time to figure out what the different elements on this tab are used for.
- 3. Search all rules in the workspace that contain the `corporate score` string in their names or bodies.
  - a. In the search field, type: `corporate score`
  - b. Make sure that **Rule** is selected in the **Search for** section.
  - c. Make sure that **Body** is selected in the **Limit to** section.

- \_\_\_ d. Make sure that **Workspace** is selected in the **Scope** section.



- \_\_\_ e. Click **Search**.

All the results are listed in the Search view.

The screenshot shows the 'Search' view with a toolbar at the top featuring Properties, Ant, and Search buttons. The search term 'corporate score' is entered, and the results count is displayed as '8 elements found.' Below the toolbar is a list of eight search results, each with a small icon and a link to its location:

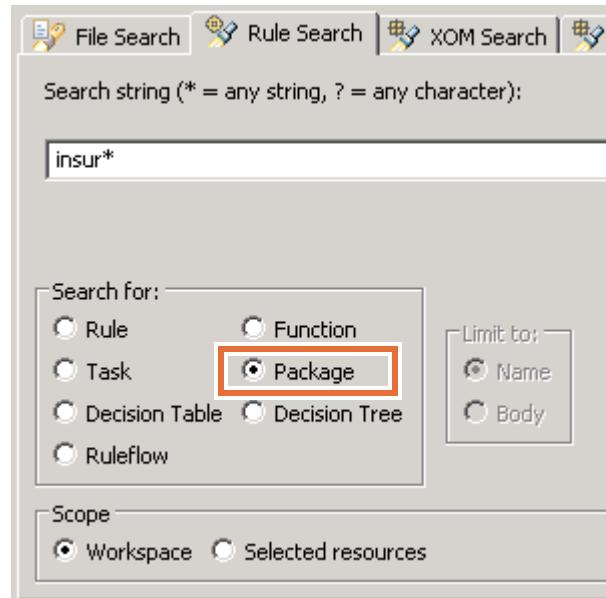
- bankruptcyScore - [loan-rules] - /loan-rules/rules/computation (2 matches)
- grade - [loan-rules] - /loan-rules/rules/eligibility (2 matches)
- initialCorporateScore - [loan-rules] - /loan-rules/rules/computation
- neverBankruptcy - [loan-rules] - /loan-rules/rules/computation
- salary2score - [loan-rules] - /loan-rules/rules/computation (2 matches)

- \_\_\_ f. In the Search view results, double-click **initialCorporateScore**.

The `initialCorporateScore` action rule opens. The `corporate score` string that you searched for is highlighted in the rule artifact.

- \_\_\_ 4. Close the `initialCorporateScore` rule.

- \_\_\_ 5. From the **Search > Search** menu, do a new search for the `insur*` string in all packages in the workspace.

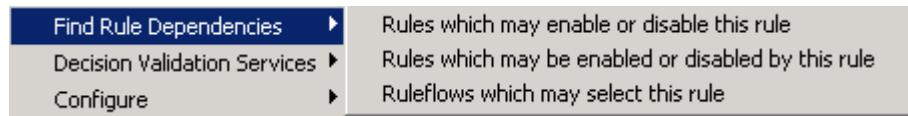


The `insurance` package is returned as a match.

- \_\_\_ 6. Open the Search window again and take some time to discover the purpose of the **XOM Search** tab and the **BOM Search** tab.

### 1.3. Searching for dependencies between rules

In this section, you search for dependencies between rules by using the **Find Rule Dependencies** menu.



- \_\_\_ 1. In the `loan-rules` project, expand `rules > insurance`.
- \_\_\_ 2. Right-click the `insurance` decision table, and click **Find Rule Dependencies > Rules which may enable or disable this rule**.

The Search view lists the `grade` decision table.

 **Questions**

Why can the `grade` decision table enable or disable the `insurance` decision table that you selected?

**Answer**

The actions that are defined in the `grade` decision table set the grade of 'the loan report', which in turn determines which of the rows of the `insurance` decision table can be executed.

- 3. In the `eligibility` rule package, right-click the `grade` decision table, and click **Find Rule Dependencies > Rules which may be enabled or disabled by this rule**.

The search results show the `insurance` decision table, which you expected, along with the `approval` action rule.

**Questions**

Can you figure out why the `grade` decision table that you selected might enable or disable the `insurance` decision table and the `approval` action rule?

**Answer**

The `grade` decision table might enable or disable the `insurance` decision table because it sets the 'the loan report' grade, which is later used in the conditions of the `insurance` decision table.

The same mechanism applies to the `approval` action rule, where the condition is also based on this grade:

the loan grade in 'the loan report' is one of { "A" , "B" , "C" }

- 4. In the `computation` rule package, right-click the `initialCorporateScore` action rule in the rule project, and click **Find Rule Dependencies > Ruleflows which may select this rule**.

The Search view lists the `loanvalidation` ruleflow and the `computation` rule task.

**Questions**

Can you figure out why the `loanvalidation` ruleflow and its `computation` rule task are the answer?

## **Answer**

The Search view shows the `loanvalidation` ruleflow because this ruleflow selects all the rule artifacts in the rule project.

The ruleflow includes rule tasks that correspond directly to all the rule packages that are defined in the project, and can thus select all the rule artifacts of this rule project. If the path through the `loanvalidation` ruleflow includes the `computation` rule task, then the `initialCorporateScore` rule is executed.

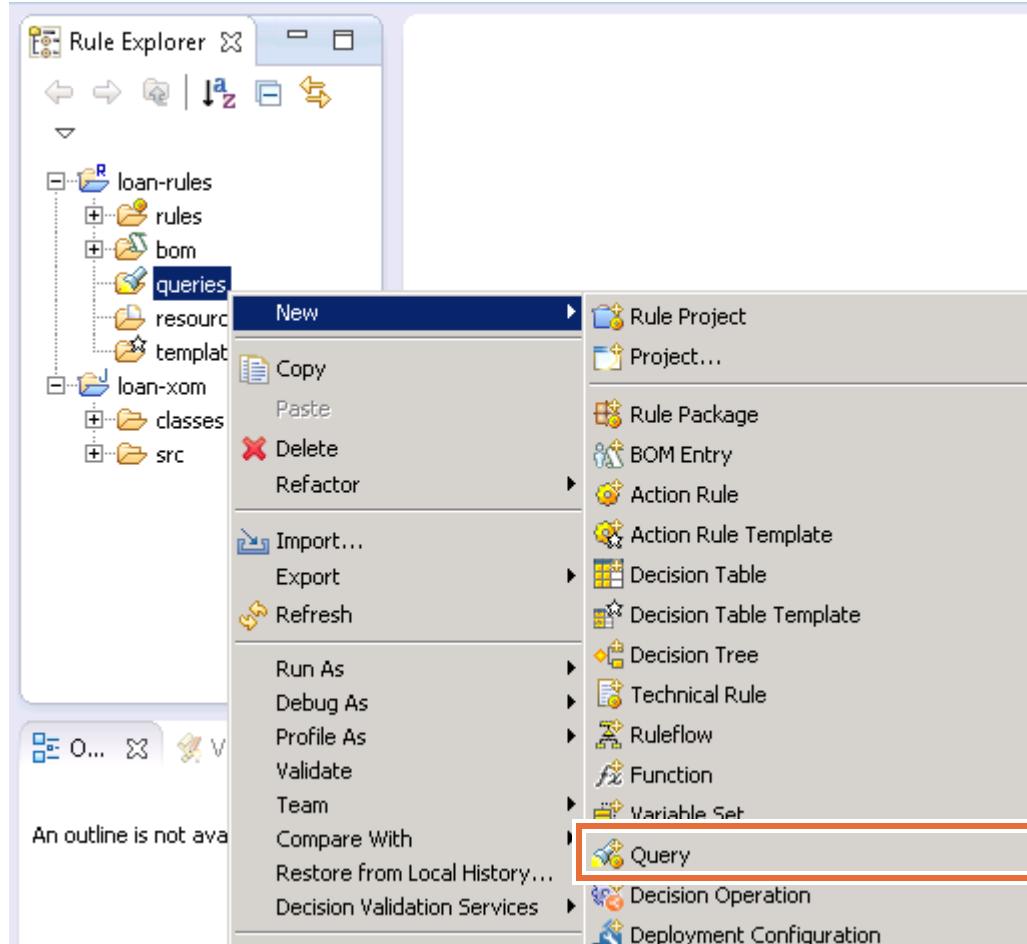
## Section 2. Querying rule projects

In this part of the exercise, you create queries on rule projects to search for rules. You create different queries by using different criteria. You also run an action on the rules that the query returns.

### 2.1. Searching for rules that may become applicable

In this section, you search for rules that might become applicable if another rule is executed.

- \_\_\_ 1. Create a query in the `loan-rules` project.
  - \_\_\_ a. In the Rule Explorer, right-click the `queries` folder and click **New > Query**.



- \_\_\_ b. In the **Name** field, type: `may become applicable`
- \_\_\_ c. Click **Finish**.
- \_\_\_ 2. In the new query, define the **such that** statement.
  - \_\_\_ a. Click **enter a condition** and double-click **each business rule** to select it from the vocabulary list.

- \_\_ b. Complete the statement to match this content:

Find all business rules  
such that each business rule may become applicable  
when [ 'a borrower' has filed a bankruptcy ]

- \_\_ c. Save the query.



#### Hint

To see the list of choices, press Ctrl+Spacebar at the location in the rule where you want to add text, or type directly in the Query editor.

- \_\_ 3. In the **Run** section, click **Run query** option of the query to run the query on the rules.



The Search view indicates all rows of the `bankruptcyScore` decision table.



#### Questions

Do you know why you get this result?

#### Answer

The Search view indicates all rows of the `bankruptcyScore` decision table. All rows in this decision table have conditions that are based on the age of the latest bankruptcy of the borrower. The conditions are met only if the borrower filed a bankruptcy (which is the query condition).

The `neverBankruptcy` rule also relies on the bankruptcy status of the borrower for its conditions. However, the `neverBankruptcy` action rule is not applicable if the borrower filed a bankruptcy because its conditions start with: it is not true that

No other rule artifact in the rule project relies on the bankruptcy status of the borrower.

- \_\_ 4. Close the query.

## 2.2. Searching for rules that may lead to a state

In this section, you search for rules that can lead to a specific state of the application objects.

- 1. Create a query that is called `may lead to a state` with the following code:

```
Find all business rules
such that each business rule may lead to a state
where ['a report' is insurance required]
```

- 2. Save the query and click **Run query**.

The Search view lists the `defaultInsurance` action rule and some of the rows in the insurance decision table.



### Questions

Can you figure out why you obtain this result?

### Answer

The Search view includes the `defaultInsurance` action rule because this action rule sets the `insuranceRequired` member of 'the loan report' to `true`.

The insurance decision table defines rules that set the `insuranceRequired` member of 'the loan report' to either `true` or `false`. All rows that are indicated in the Search view set this member to `true`.

The other rows (1 and 5) set this member to `false` and are not indicated in the Search view.

- 3. Verify that the results in the Search view are accurate and that all rows of the insurance decision table, except 1 and 5, set the `insuranceRequired` member of 'the loan report' to `true`.
  - a. In the Search view, right-click the insurance decision table and click **Show In > Rule Explorer**. Notice that the insurance decision table is highlighted in the Rule Explorer.
  - b. In the Rule Explorer, double-click the insurance decision table, which is highlighted, to open it in the decision table editor.
  - c. Read the values that are given in the `Insurance required` column and verify that all rows contain `true` except for rows 1 and 5, which contain `false`.

- \_\_ d. Select a row (for example, row 3), and read the complete action rule that corresponds to this row of the insurance decision table.

| Grade | Amount of loan                                                                                                                                                                                                                                                                                                                                                          |         | Insurance required | Insurance rate |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|--------------------|----------------|
|       | Min                                                                                                                                                                                                                                                                                                                                                                     | Max     |                    |                |
| 1     | < 100,000                                                                                                                                                                                                                                                                                                                                                               |         | false              | 0              |
| 2     | 100,000                                                                                                                                                                                                                                                                                                                                                                 | 300,000 | true               | 0.001          |
| 3     | 300,000                                                                                                                                                                                                                                                                                                                                                                 | 600,000 | true               | 0.003          |
| 4     | > 600,000                                                                                                                                                                                                                                                                                                                                                               |         | true               | 0.005          |
| 5     | ( the loan grade in 'the loan report' is not empty )<br>and all of the following conditions are true :<br>- ( the loan grade in 'the loan report' is "A" )<br>- ( the amount of 'the loan' is at least 300000 and less than 600000 ),<br><b>then</b><br>set insurance required in 'the loan report' to true ;<br>set the insurance rate in 'the loan report' to 0.003 ; |         |                    |                |
| 6     | 100,000                                                                                                                                                                                                                                                                                                                                                                 | 300,000 | true               | 0.006          |
| 7     | 300,000                                                                                                                                                                                                                                                                                                                                                                 | 600,000 | true               | 0.0085         |
| 8     | > 600,000                                                                                                                                                                                                                                                                                                                                                               |         | true               | 0.0115         |

General Decision Table IRL insurance.dta

- \_\_ 4. Close the decision table and the query.

### 2.3. Searching for rules that use a BOM member

In this section, you search for rules that use a specific BOM member.

- \_\_ 1. Create a query that is called: using BOM member

The query should state:

Find all business rules

such that each business rule is using the BOM Member  
"training\_loan.Loan.amount"

- \_\_ 2. Save the query and run it.

The Search view indicates the checkAmount action rule, the insurance decision table, and the repayment action rule.



#### Questions

Can you figure out why you obtain this result?

**Answer**

The Search view indicates the `checkAmount` action rule because the conditions of this action rule are based on the value of the amount of 'the loan', that is, on the `training_loan.Loan.amount` BOM member.

The Search view also indicates the `insurance` decision table because the conditions for all the rows in this decision table are based on the same value.

Finally, the Search view indicates the `repayment` rule because the actions of this rule use the same value to compute the monthly repayment.

No other rule artifact in the rule project relies on the value of the amount of 'the loan'.

## 2.4. Applying actions with queries

In this section, you run a query and apply actions to the results. First, you search for rules that have a high priority, and modify their priority as an action that is associated with the query.

- \_\_\_ 1. Create a query that is called: the `priority of the rules`

The query should state:

Find all business rules

such that the priority of each business rule is "1000000"

Do set the priority of each business rule to "-1000000"



### Important

The query action is introduced with the `Do` keyword and modifies the priority of the business rules that match the query criteria.

Before you run a query that potentially modifies your rule project, you first want to determine whether your query is correct, and verify which rule artifacts might be affected. For this reason, the **Run query actions** task is separate from the **Run query** task, which provides an opportunity to review the query results before applying actions.

- \_\_\_ 2. Save your work.

- \_\_\_ 3. Select the **Run query** option of the query "the priority of the rules".

You have two results:

- The `grade` decision table
- The `initialCorporateScore` action rule



### Questions

Are these results correct?

### Answer

In the loan-rules project, only the grade decision table and the initialCorporateScore action rule have their **priority** property that is defined with the value 1000000.

4. Now, apply the actions to the found rule artifacts by clicking **Run query actions** in the Run section.



5. Validate your results.
- In Rule Explorer, open the initialCorporateScore rule in the computation package.
  - In the Properties view, verify that the **priority** property of the initialCorporateScore action rule is now equal to -1000000.
  - Do the same verification for the grade decision table.
  - Close the rule and decision table.
6. Run the priority of the rules query again.

No business rules match.



### Questions

Why are there no matches?

**Answer**

The action of this query replaces the priority of 1000000 with a priority of -1000000. After this action is applied, the two rule artifacts found initially now have a priority of -1000000, and are thus not found if you run the query again.

- \_\_\_ 7. Close the query and the rules.

## Section 3. Extracting a ruleset archive

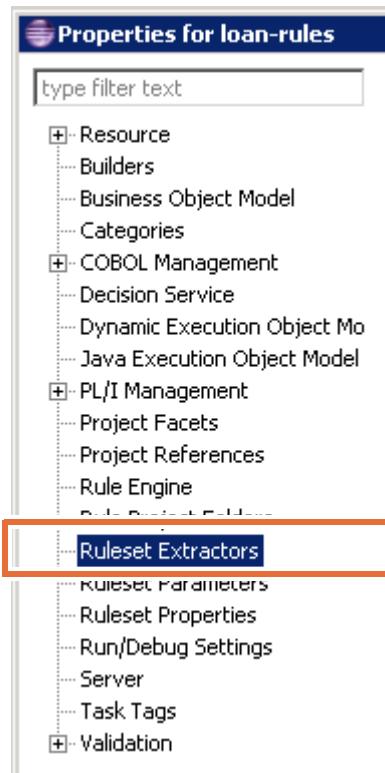
In this section, you learn how to extract a ruleset archive from a rule project, both with and without a ruleset extractor.

### 3.1. Defining a query-based ruleset extractor

In this part of the exercise, you create a ruleset extractor that is based on the **may become applicable** query that you created earlier. You later apply this ruleset extractor to extract a ruleset archive that contains only the results of the query.

To create a ruleset extractor that uses the **may become applicable** query:

- \_\_\_ 1. In Rule Designer, right-click the **loan-rules** project and click **Properties**.
- \_\_\_ 2. In the Properties window, click **Ruleset Extractors**.



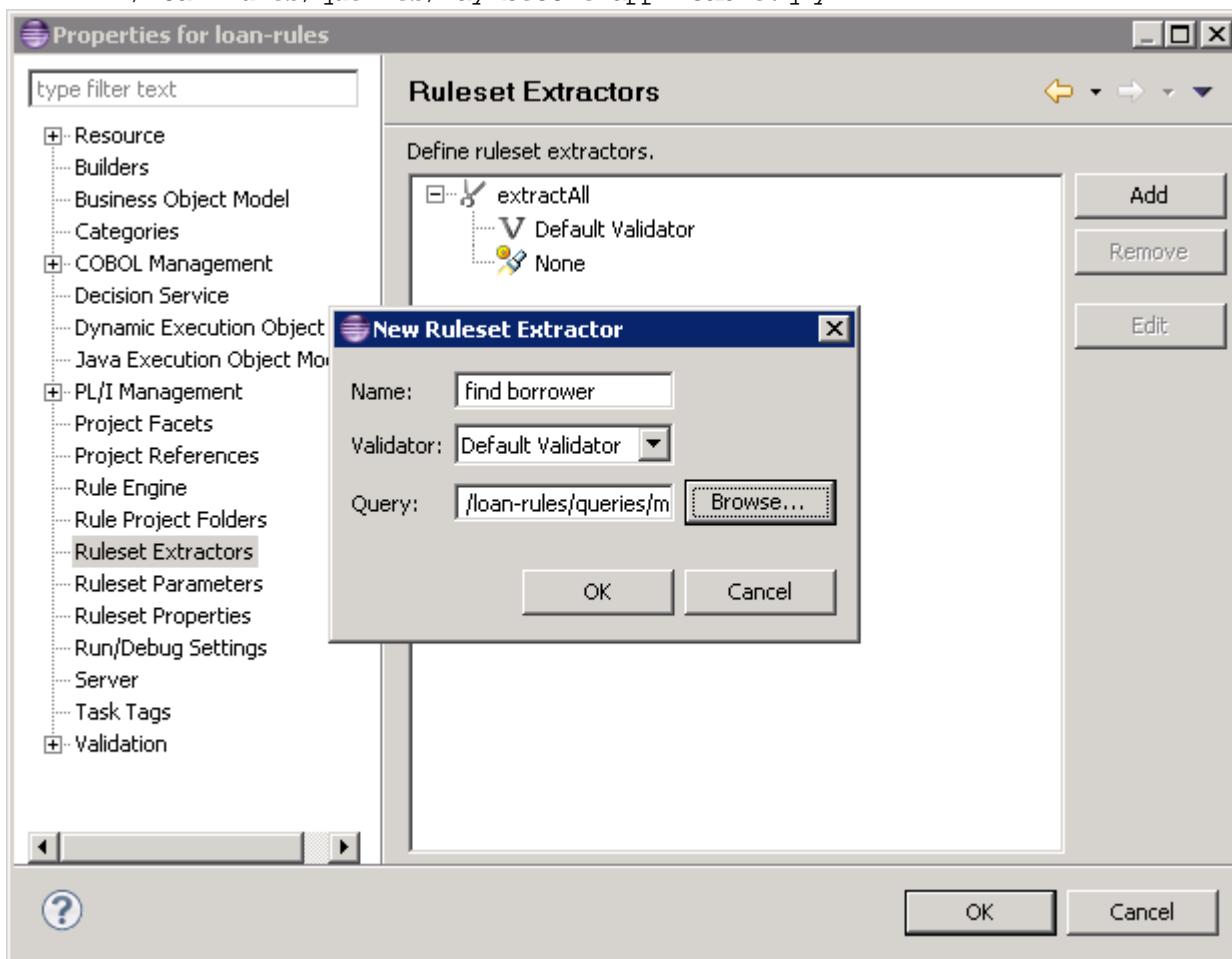
- \_\_\_ 3. In the Ruleset Extractors pane, click **Add**.

The New Ruleset Extractor window opens.

- \_\_\_ 4. Define the extractor in the New Ruleset Extractor window.
  - \_\_\_ a. Enter **find borrower** in the **Name** field of the new ruleset extractor.
  - \_\_\_ b. Keep the default value of the **Validator** field.
  - \_\_\_ c. For the **Query** field, click **Browse** to open the "Select Query as Extractor" window.
  - \_\_\_ d. Choose **may become applicable** and click **OK**.

- \_\_\_ e. The **Query** field is now set to:

/loan-rules/queries/may become applicable.qry



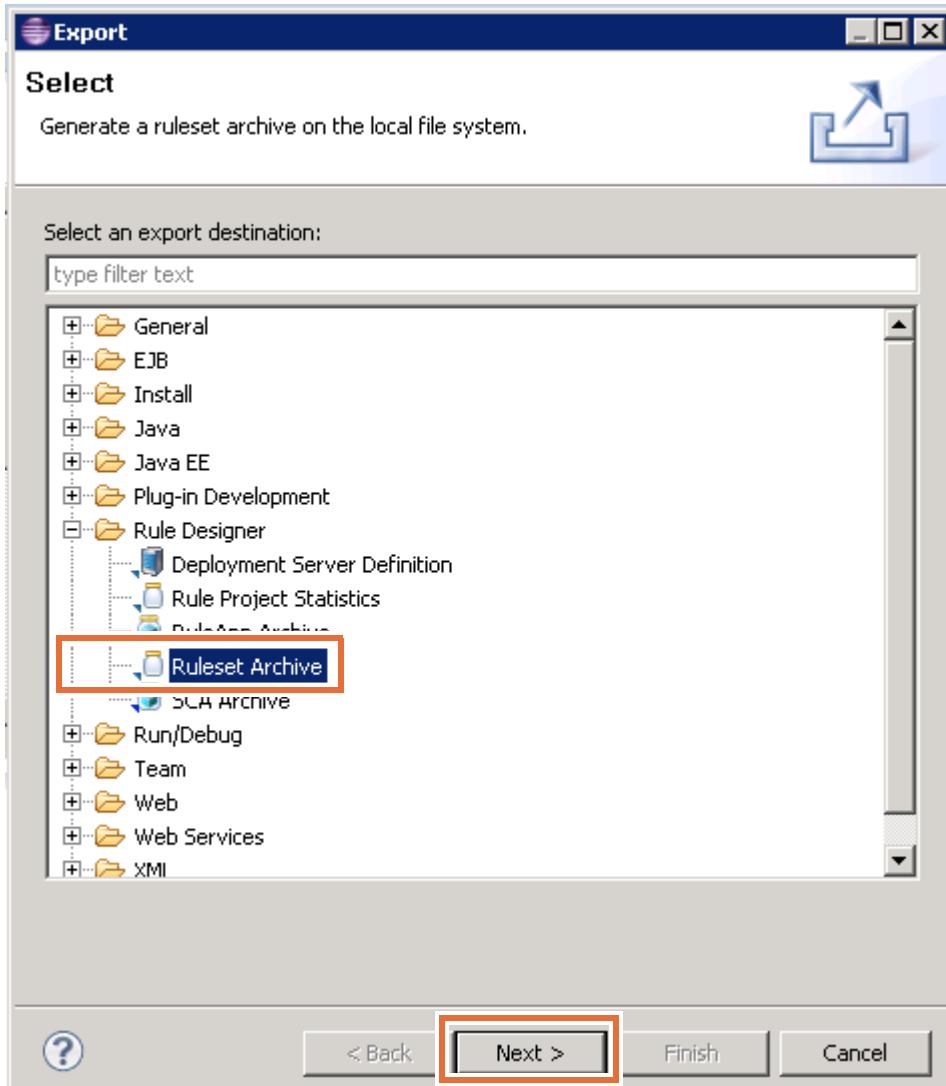
- \_\_\_ 5. Click **OK** to close the New Ruleset Extractor window.  
 \_\_\_ 6. Click **OK** to close the Properties window.

### 3.2. Using a ruleset extractor to extract a ruleset archive

- \_\_\_ 1. In Rule Designer, right-click the **loan-rules** project, and click **Export > Export**.



2. In the Export window, expand Rule Designer, select Ruleset Archive, and click Next.



The “Export a Ruleset Archive” window opens.

3. In the “Export a Ruleset Archive” window, keep loan-rules in the **Rule project** field.  
4. In the **Archive file** field, type the following name: `rulesetArchive.jar`



### Important

Make sure that the ruleset archive file has `.jar` as its extension.

5. Click **Next**.  
6. In the “Export a ruleset archive” window, select the **Use a ruleset extractor** check box.

The **Ruleset extractor** list is now enabled.

7. In the **Ruleset extractor** list, select the `find borrower` extractor from the list.  
8. Click **Finish**.

Rule Designer creates the specified ruleset archive as a JAR file in your workspace.

You can see the `rulesetArchive.jar` file in the Rule Explorer in the `loan-rules` project.

### 3.3. Exploring the content of the ruleset archive

- 1. In Rule Explorer, rename `rulesetArchive.jar` to **`rulesetArchive.zip`**.
  - a. In the `loan-rules` project, select the `rulesetArchive.jar` file and press F2.
  - b. Rename the `rulesetArchive.jar` file extension to: `.zip`
  - c. Click **OK**.
- 2. Open `rulesetArchive.zip` with the system editor and explore its contents.
  - a. Right-click the `rulesetArchive.zip` file and click **Open With > System Editor**. A Windows Explorer window opens. You can now explore the content of the ruleset archive.
  - b. Take some time to relate this content with the result that you had in "Searching for rules that may become applicable" on page 11-7.



#### Information

The `bankruptcyScore` decision table that is listed in the `IRL\computation` folder (`bankruptcyScore-dta.irl`) matches the results from running this same query earlier.

- 3. When you are finished, close Windows Explorer and rename the ruleset archive file extension back to: `.jar`

### 3.4. Exporting a ruleset archive without a ruleset extractor

In this section, you create a ruleset archive that uses all the rules.

- 1. Export the `loan-rules` project again.
  - a. In the Rule Explorer, right-click the `loan-rules` project and click **Export > Export**.
  - b. In the Export page, expand **Rule Designer**, select **Ruleset Archive**, and click **Next**.
- 2. In the **Archive file** field of the Export a Ruleset Archive window, enter: `rulesetArchive2.jar`
- 3. Click **Finish**.
 

Rule Designer creates the `rulesetArchive2.jar` ruleset archive, which you can see in Rule Explorer.
- 4. Explore `rulesetArchive2.jar` by following the steps from "Extracting a ruleset archive" on page 11-14, and compare these files with the files extracted in your first ruleset archive. For example, compare the contents of the `IRL/eligibility` folder.



### Questions

Can you anticipate any advantages of using a ruleset extractor versus not using one?

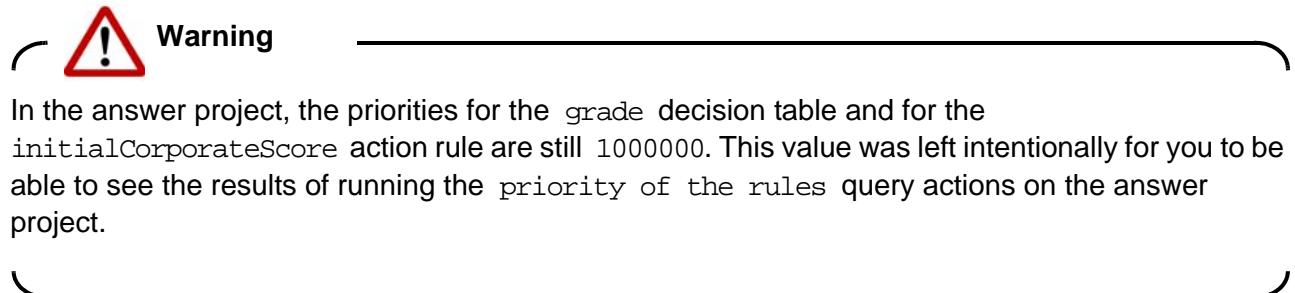
### Answer

An extractor can reduce the number of rules that are included in your ruleset, which can improve performance when you have thousands of rules to evaluate during rule execution.

### End of exercise

## Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 11: Queries and ruleset extraction > 02-answer**.



The first part of the exercise looked at how to search for rule artifacts within a rule project and how to find which rules are enabled or disabled by others, or which ruleflows might select them.

The second part of the exercise looked at how to create and run queries on rule projects. It also looked at how to create ruleset extractors that are based on queries, and how to use these ruleset extractors to create ruleset archives that match the queries.

The third part of the exercise looked at how to define ruleset extractors and how to extract ruleset archives with or without a ruleset extractor.



# Exercise 12. Executing rules locally

## What this exercise is about

This exercise teaches you how to run rule projects locally to ensure the correctness of rulesets.

## What you should be able to do

After completing this exercise, you should be able to:

- Create launch configurations to execute rulesets locally
- Create Java applications to execute rulesets locally with an embedded rule engine
- Explore the rule engine API

## Introduction

In this exercise, you create a launch configuration to execute the rule artifacts in your rule project locally in Rule Designer, without creating any code. Then, you create a Java application to execute your ruleset with an embedded rule engine. You use this application to execute the rule artifacts in your rule project locally in Rule Designer.

The exercise includes these tasks:

- Section 1, "Executing rules locally by using a launch configuration"
- Section 2, "Executing rules locally by using a runner application"

## Requirements

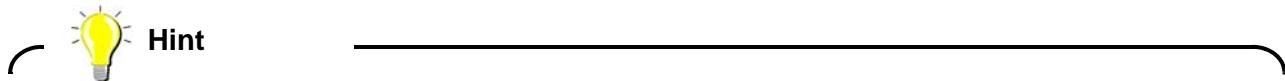
There are no specific requirements for this exercise.

## Section 1. Executing rules locally by using a launch configuration

In this part of the exercise, you create a launch configuration. You then use it to execute the rule artifacts in your rule project locally in Rule Designer.

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

- \_\_\_ 1. In Rule Designer, switch to a new workspace:
  - \_\_\_ a. From the **File** menu, click **Switch Workspace > Other**.
  - \_\_\_ b. In the **Workspace Launcher** window, enter the path:  
*<LabfilesDir>\workspaces\execute\_rules\_locally*
- \_\_\_ 2. Close the Welcome view and switch to the Samples Console perspective.
- \_\_\_ 3. To import the start projects, select **Rule Designer > Training > Ex 12: Executing rules locally > 01-start**.
- \_\_\_ 4. Close the Help pane.



In this exercise, you write some code. To help you, you can copy and paste the code snippets from the *<LabfilesDir>\code\execute\_rules\_locally.txt* file into Rule Designer.

### 1.2. Creating a launch configuration

In this section, you create a launch configuration for the `loan-rules` project.

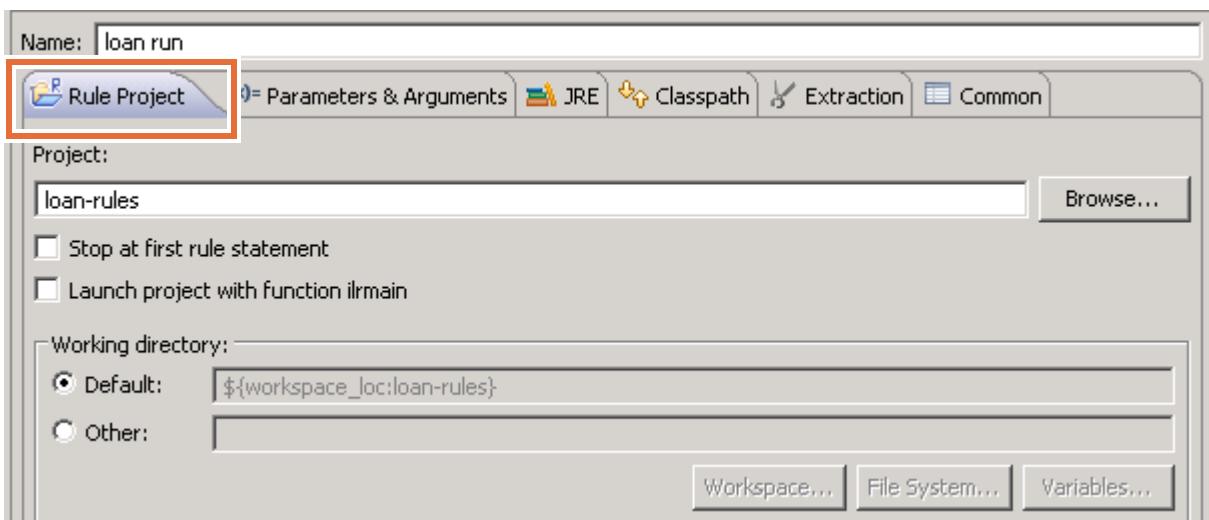
- \_\_\_ 1. From the menu, click **Run > Run Configurations**.



The Run Configurations window opens.

- \_\_\_ 2. In the Run Configurations window, right-click **Rule Project** in the left pane, and click **New**.  
The new configuration is created, with the default name `New_configuration`.
- \_\_\_ 3. In the **Name** field, enter: `loan run`

4. On the **Rule Project** tab, configure the following properties.
- \_\_ a. To set the **Project** field, click **Browse**, select **loan-rules** from the list of available rule projects, and click **OK**.
  - \_\_ b. Make sure that the **Stop at first rule statement** check box is not selected.  
This check box is used only when you run the Rule Debugger, which you work with later.
  - \_\_ c. Clear the **Launch project with function ilrmain** check box, if it is selected.  
When this check box is selected, you must have an `ilrmain` function within the rule project to initialize the ruleset parameter values within the rule project.  
In this exercise, you do not require such a function. Instead, you set the ruleset parameter values by using the **Parameters & Arguments** tab directly in a later step.
  - \_\_ d. In the **Working directory** section, select **Default** if it is not already selected.



5. Click the **Parameters & Arguments** tab and notice the warning that states that some parameters are not set.  
You can ignore this warning message because you are about to set the ruleset parameter values.
6. Configure the ruleset parameters that are required to execute your rule project.
- \_\_ a. Select the **borrower** ruleset parameter in the **Name** column, and click **Edit Value** to open the Edit Parameter Value window.
  - \_\_ b. In the Edit Parameter Value window, select the **Function body** option and enter the following code to overwrite and define the initial value of the **borrower** ruleset parameter:

```
training_loan.Borrower borrower = new
training_loan.Borrower("John", "Doe", training_loan.DateUtil.makeDate(1968,
java.util.Calendar.MAY, 12), "123456789");
borrower.zipCode = "91320";
borrower.creditScore = 600;
borrower.yearlyIncome = 100000;
return borrower;
```



## Hint

You can find all code snippets for this exercise in the  
 <LabfilesDir>\code\execute\_rules\_locally.txt file.

| Parameters: |           |          |       |
|-------------|-----------|----------|-------|
| Name        | Direction | Type     | Value |
| borrower    | IN        | Borrower |       |
| loan        | IN_OUT    | Loan     |       |

**Edit Parameter Value**

Name: borrower

Expression value:

Function body:

```
training_loan.Borrower borrower = new
training_loan.Borrower("John", "Doe", training_loan.DateUtil.make
borrower.zipCode = "91320";
borrower.creditScore = 600;
borrower.yearlyIncome = 100000;
return borrower;
```

OK Cancel

- \_\_\_ c. Click **OK** to close the Edit Parameter Value window.
  - \_\_\_ d. Repeat the previous steps to set **Function body** for the **loan** ruleset parameter with the following code:
- ```
training_loan.Lean loan = new
training_loan.Lean(training_loan.DateUtil.makeDate(2015,
java.util.Calendar.JULY,15),72,100000,0.7d);
return loan;
```
- ___ e. Click **Apply**.
 - ___ 7. Click the **Extraction** tab and make sure that **Type of extraction** is set to **Use all rules contained in project**.

**Information**

The other option on this tab is to use an extractor to filter the rules so that only the extracted rules are run.

- 8. Click the **Common** tab to save your launch configuration as a shared file that you can reuse later.
 - a. Select **Shared file** and click **Browse**.
 - b. Select `loan-rules` and click **OK**.

The launch configuration file is now created within the rule project directory itself.
- c. Look at the other elements on this tab, and take some time to figure out what they are for.
- 9. Click **Apply** to save the changes.
- 10. Click **Close** to close the Run Configurations window.
- 11. In the Rule Explorer, expand the `loan-rules` project to see the `loan run.launch` file that is now visible.

1.3. Running the launch configuration

Now that you created the launch configuration, you can run it to execute your ruleset.

- 1. In the Rule Explorer, right-click the `loan run.launch` file in the `loan-rules` project and click **Run As > loan run**.

Rule Designer executes the rules in your rule project locally by using the ruleset parameters that are defined in the `loan run` launch configuration.

The Console view shows the following execution traces:

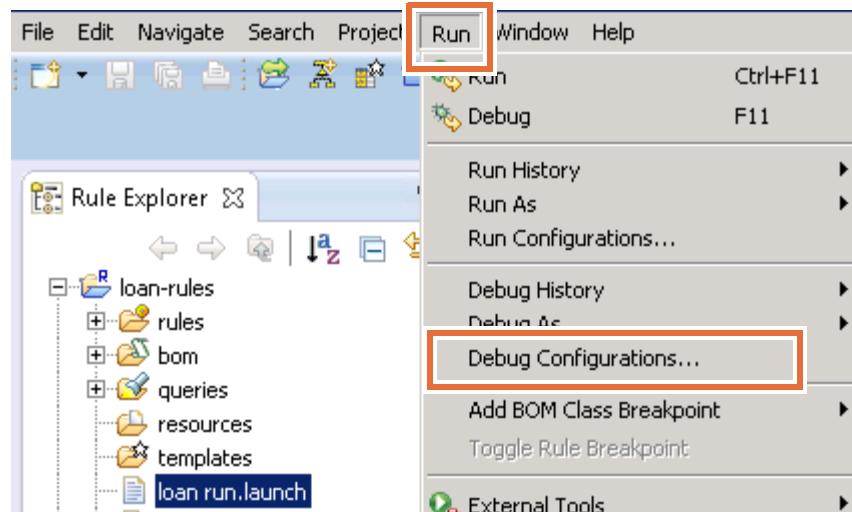
```
Borrower( firstName: John lastName: Doe born: May 12, 1968 SSN: 123-45-6789
zip code: 91320 income: 100000 credit score: 600)
Loan( amount: 100000 starting: Jun 1, 2015 duration: 72 month LTV: 70% rate:
5.7% monthly repayment: 1,643.16)
Report( validData: true approved: true score: 820 grade: B insuranceRequired:
true insuranceRate: 2% message: Low risk loan
Congratulations! Your loan has been approved
)
```

- 2. Take some time to review the execution traces in the Console view and compare them to the defined ruleset parameters.

1.4. Debugging with a launch configuration

In the previous section, you created the `loan run.launch` configuration to execute your ruleset locally without the Rule Debugger. In this section, you modify this launch configuration to use the Rule Debugger.

- With the `loan run.launch` configuration selected in Rule Explorer, click the **Run > Debug Configurations** menu item.



Hint

Alternatively, you can right-click the `loan run.launch` configuration and click **Debug As > Debug Configurations**.

The Debug Configurations window opens.



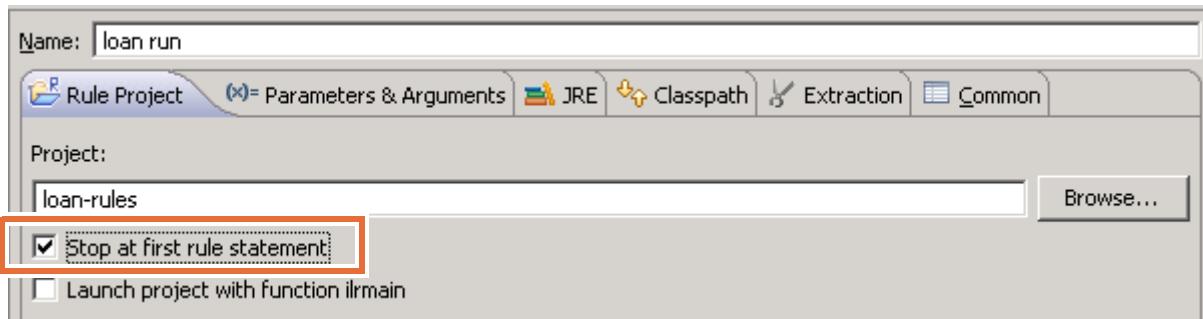
Questions

Do you see any differences between the Debug Configurations window and the Run Configurations window that you worked in earlier?

Answer

The Debug Configurations window and Run Configurations window share content except that **Debug** replaces **Run**.

2. In the Rule Project pane, select **Stop at first rule statement**.



The **Stop at first rule statement** option is used when you run this configuration with **Run > Debug Configurations**. If this check box is selected, the Rule Debugger stops before any business rule is executed. You can then complete any steps that are required for your debugging session (such as setting breakpoints or modifying ruleset parameters).

3. Click **Apply**.



Do not click **Debug**. You use the Rule Debugger in a later exercise.

4. Click **Close** to exit.
 5. In the Rule Explorer, double-click the `loan run.launch` configuration in the `loan-rules` project to view the contents as plain text.
 6. Look for this text in the `loan run.launch` configuration:

```
<booleanAttribute key="ilog.rules.studio.launching.STOP_AT_FIRST_STATEMENT"
value="true"/>
```



You can press **Ctrl+F** to search for this text.

This line was added because you selected the **Stop at first rule statement** check box.

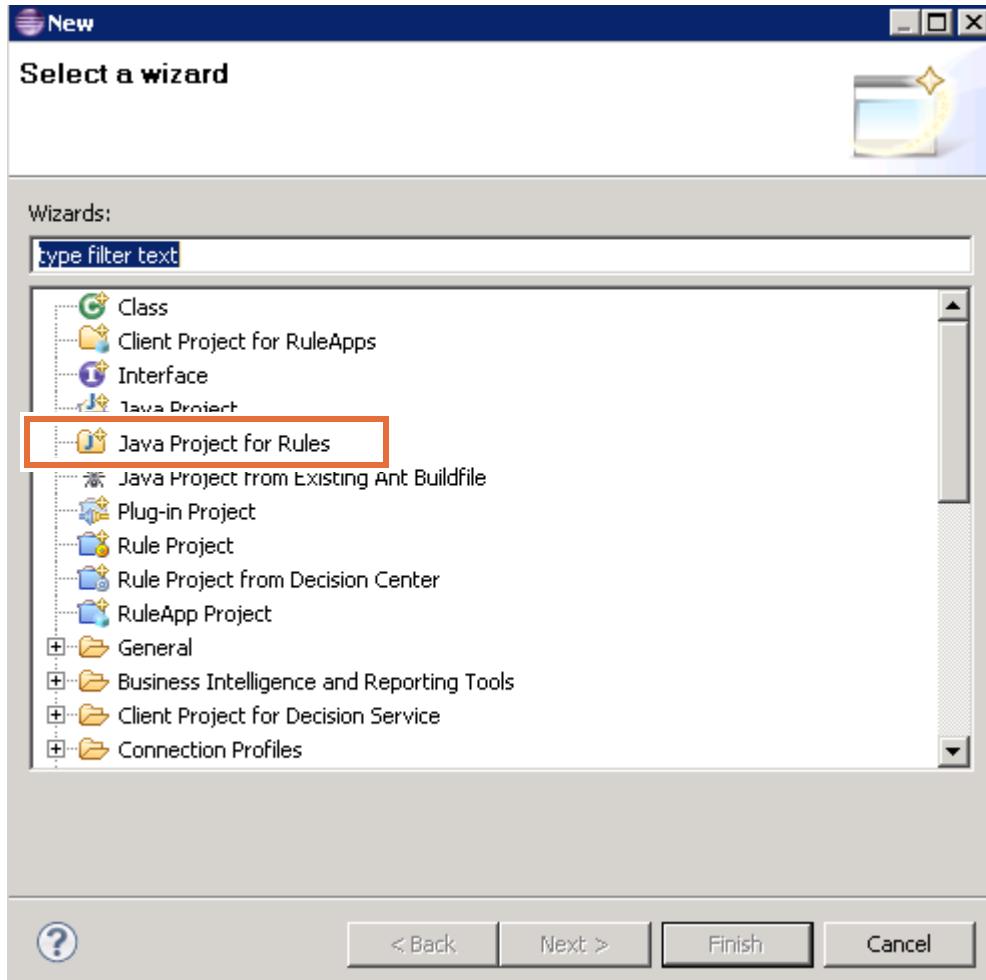
7. Close the window with the `loan run.launch` file.

Section 2. Executing rules locally by using a runner application

In this part of the exercise, you create a Java application to execute your ruleset with an embedded rule engine. By creating this “runner” application, you simultaneously package the ruleset to execute it as the appropriate ruleset archive, ready for its execution in your runner application.

2.1. Creating a Java project for rules

- 1. Right-click anywhere in the Rule Explorer, and click **New > Other**.
The “Select a wizard” window opens.
- 2. In the “Select a wizard” window, select **Java Project for Rules**, and click **Next**.



The “Java Project for Rules” wizard opens. The “Java Project for Rules” is also available in the **Rule Designer** folder of this window.

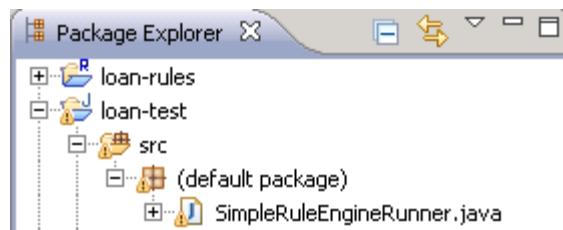
- 3. In the “Java Project for Rules” page:
 - a. Enter `loan-java` in the **Project name** field as the name of the runner application project.
 - b. Ensure that **Use default location** is still selected, and click **Next**.

- ___ 4. In the “Java Project for Rules Settings” page, select the `loan-rules` project and click **Next**.
By doing so, you indicate that you want to create a runner application that executes the rule artifacts of the `loan-rules` project.
- ___ 5. In the “Java Project for Rules Templates” page, select the **Simple Rule Engine Runner** template and click **Finish**.
The Open Associated Perspective window opens and asks whether you want to switch to the Java perspective.
- ___ 6. When prompted to switch to the Java perspective, click **Yes**.
Rule Designer opens the Java perspective. The `loan-java` Java project is now visible in the Package Explorer view.

2.2. Exploring the runner application

In this section, you explore the created runner application and learn more about the API required to embed a rule engine and request the ruleset execution.

- ___ 1. In the Package Explorer, expand `loan-java > src > default package` to see the contents of the `SimpleRuleEngineRunner.java` file.



The main class is named `SimpleRuleEngineRunner` and can execute a ruleset.

The code for this class is incomplete but contains the required steps to:

- Load the ruleset, knowing its deployment location
 - Instantiate a rule engine
 - Execute the rule engine against objects that are passed as parameters
- ___ 2. In the Package Explorer, double-click `SimpleRuleEngineRunner.java` to open the file in the editor.
- ___ 3. Find the following lines in the `SimpleRuleEngineRunner` class:

```
IlrContext context = new IlrContext(ruleset);
IlrParameterMap inputs = new IlrParameterMap();
context.setParameters(inputs);
//context.insert(object);
IlrParameterMap outputs = context.execute();
```



Questions

What do these lines of code do?

Answer

- `IlrContext` is the class that represents a rule engine, which you create by passing the ruleset as the argument:

```
IlrContext context = new IlrContext(ruleset);
```

- `IlrParameterMap` is the class that represents the ruleset parameters, which you pass to the rule engine by calling:

```
context.setParameters();
```

- You insert objects in the working memory of the rule engine by calling:

```
context.insert();
```

- You request the execution of the rulesets by the rule engine by calling:

```
context.execute();
```

The rule engine executes the rule artifacts in the ruleset against the objects that are provided in the working memory, the ruleset parameters, or both.

- The result of `context.execute()` is an `IlrParameterMap` object that contains the ruleset output parameters, if any.



Hint

Use the product documentation to find more information about the rule engine API. From the documentation home page, you find the API documentation under **Operational Decision Manager V8.7 > Decision Server Rules > Reference > Rule Designer reference > Rule Designer API**.

2.3. Defining the objects that are passed to the rules



Note

The current code can execute, but is not useful because there are no objects that your rules can work on. Do not run it now.

**Questions**

Where and how can you define the objects on which your rules work?

Answer

The code to initialize the objects must be written after the following comment:

```
// Initialize the inputs
```

Depending on how the rule project is designed, the rule engine identifies these objects through different means:

- When you use `context.setParameters()` to pass objects to the rule engine, the rule engine considers the elements of the `IlrParameterMap` instance as ruleset parameters.

This line defines each "ruleset parameter" as an element in "inputs".

```
IlrParameterMap inputs = new IlrParameterMap();
```

Then, this line passes the map.

```
context.setParameters(inputs);
```

- When you use `context.insert()` to pass objects to the rule engine, the rule engine recognizes that the objects are in the working memory.

First, you initialize the working memory by defining each object to insert in the working memory. Then, the `context.insert(object)` command inserts each object.

```
// Initialize the working memory
context.insert(object);
```

**Questions**

Based on the way that the `loan-rules` project is defined, can you determine which option is applicable here?

Answer

The loan-rules project defines action rules that work on objects that are provided as ruleset parameters. To execute these action rules, you must define the values of these ruleset parameters. You do not have to define objects to insert in the working memory.

To define the ruleset parameters:

1. Open the `SimpleRuleEngineRunner` class of the `loan-java` Java project if it is not open yet.
2. Find the `Initialize the inputs` section of the code and insert the following object definitions ***between*** the definition of `IlrParameterMap inputs` and the call to `context.setParameters(inputs)` that inserts this map into the context.

```

    // Create the engine
    IlrContext context = new IlrContext(ruleset);

    // Initialize the inputs
    IlrParameterMap inputs = new IlrParameterMap();
    context.setParameters(inputs);

    // Initialize the working memory
    // context.insert(object);

```

**Hint**

You can find the code snippets in the `<LabfilesDir>\code\execute_rules_locally.txt` file.

- To define the Borrower object:

```

training_loan.Borrower borrower = new
training_loan.Borrower("John", "Doe", training_loan.DateUtil.makeDate(1968,
java.util.Calendar.MAY, 12), "123456789");
borrower.setZipCode("91320");
borrower.setCreditScore(600);
borrower.setYearlyIncome(100000);
inputs.setParameter("borrower", borrower);

```

- To define the Loan object:

```

training_loan.Loan loan = new
training_loan.Loan(DateUtil.makeDate(2015,
java.util.Calendar.JUNE, 1), 72, 100000, 0.7d);
inputs.setParameter("loan", loan);

```

**Important**

These code snippets are not identical to the code snippets that are required to initialize the same ruleset parameter with a launch configuration.

- ___ 3. Add the following `import` statement to import statements for `SimpleRuleEngineRunner` to make sure that the class compiles:

```
import training_loan.DateUtil;
```

The `DateUtil` class is defined in the `loan-xom` Java project that serves as the basis for the `loan-rules` project, and is automatically added to the path of the runner application.

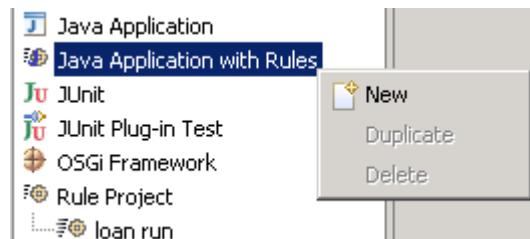
- ___ 4. Save your work.

2.4. Executing the rules locally by using the runner application

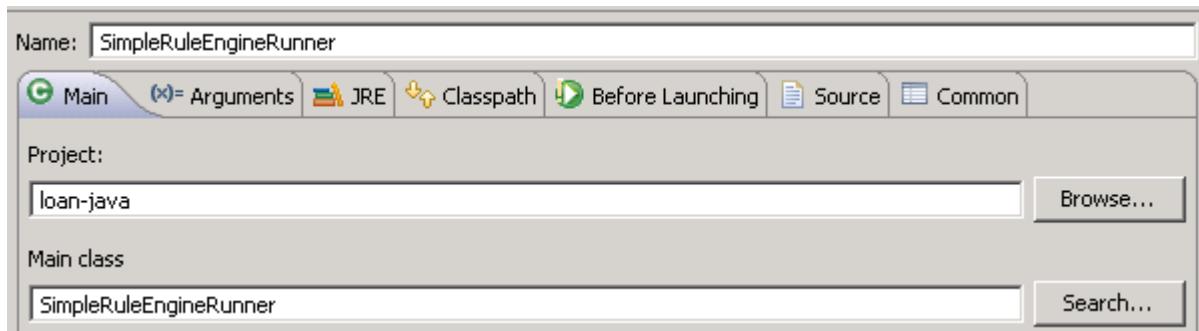
In this section, you execute the rule artifacts by using the runner application.

- ___ 1. Create a run configuration for the Java project.

- ___ a. Click **Run > Run configurations**.
- ___ b. Right-click **Java Application with Rules** and click **New**.



- ___ c. For the configuration name, type: `loan java`
- ___ d. In the **Project** field, click **Browse**, select `loan-java`, and click **OK**.
- ___ e. In the **Main class** field, click **Search**.
- ___ f. In the Choose Main Type window, select `SimpleRuleEngineRunner` and click **OK**.



- ___ g. Click **Apply**.
- ___ 2. Run the new configuration by clicking **Run**.

- ___ 3. Verify that the traces available in the Console view are the same traces as when you executed the ruleset with the launch configuration.

See "Running the launch configuration" on page 12-5.

End of exercise

Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 12: Executing rules locally > 02-answer**.

The first part of the exercise looked at how to execute rules locally by using a launch configuration in Rule Designer. The second part of the exercise looked at how to execute rules locally by using a Java application.

Exercise 13.Debugging a ruleset

What this exercise is about

This exercise teaches you how to debug a ruleset in Rule Designer.

What you should be able to do

After completing this exercise, you should be able to:

- Debug a ruleset
- Set breakpoints on an action rule and on a task in a ruleflow
- Inspect objects in the working memory or rule instances in the agenda
- Use the various views of the Debug perspective

Introduction

In many cases, executing rules locally in Rule Designer is not enough for you to find out why the ruleset does not create the expected decision. In this exercise, you debug a ruleset with the Rule Debugger.

The exercise includes these tasks:

- Section 1, "Running the debug launch configuration"
- Section 2, "Debugging with breakpoints"
- Section 3, "Debugging with breakpoints in the ruleflow"
- Section 4, "Resolving the problem"

The steps that are described here are extracted from the ***Debugging a ruleset*** tutorial, which is distributed with the product documentation. During the exercise, you do not fix the errors that you find. However, you can consult the tutorial for more details if you want instructions on how to fix the errors.

Requirements

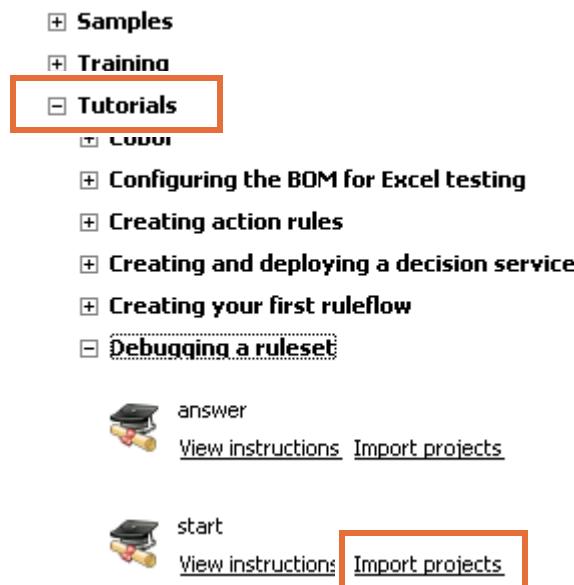
There are no specific requirements for this exercise.

Section 1. Running the debug launch configuration

In this section, you use a launch configuration to debug the ruleset execution.

1.1. Setting up your environment for this exercise

- ___ 1. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\debug`
- ___ 2. Close the Welcome view and switch to the Samples Console perspective.
 - ___ a. Click the **Open Perspective** icon in the upper-right part of the Eclipse window.
 - ___ b. Select **Samples Console**, and click **OK**.
- ___ 3. Select **Rule Designer > Tutorials > Debugging a ruleset > start > Import projects**.



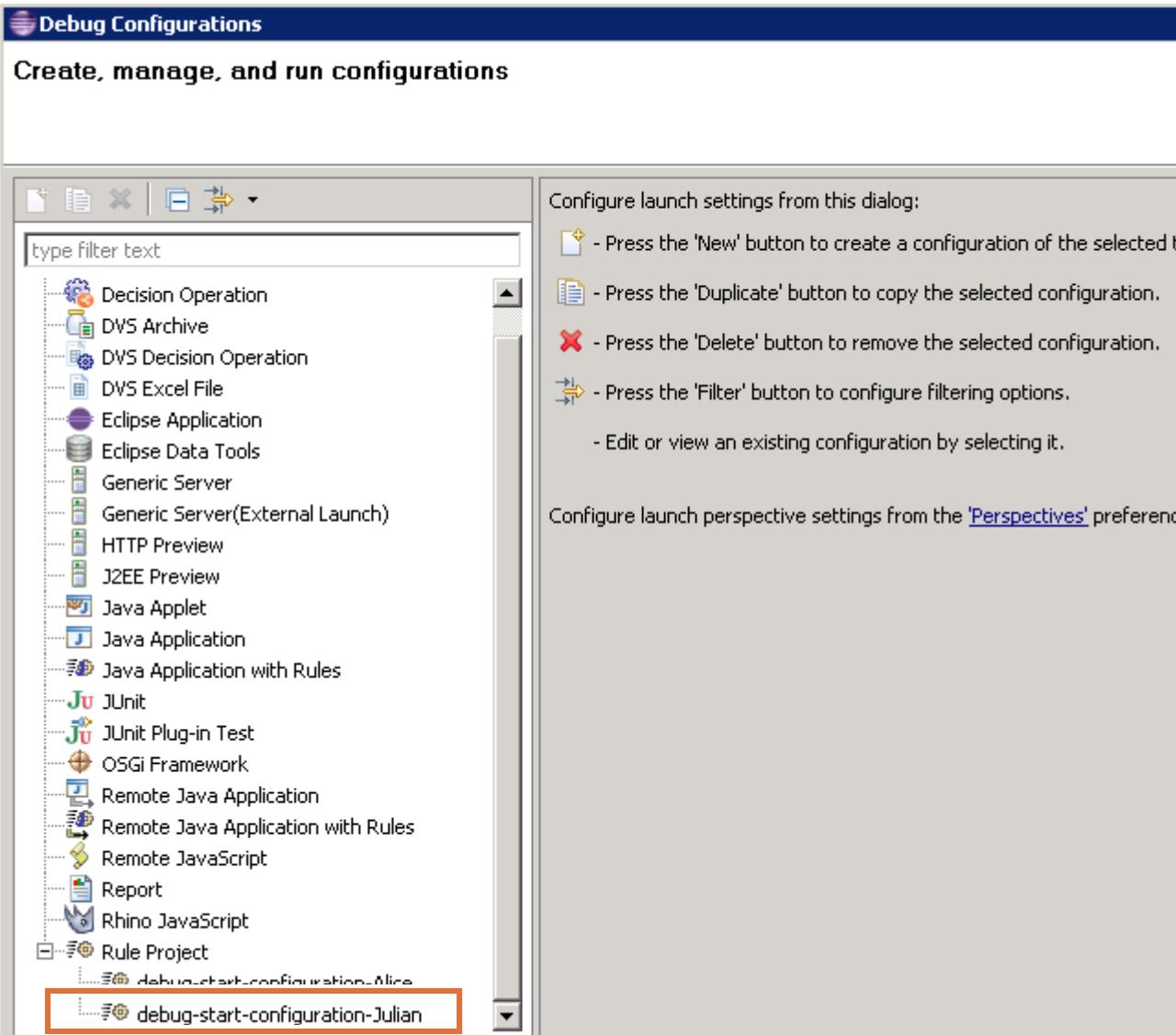
The Rule perspective opens with these projects:

- carrental-xom
 - debug-rules-start
- ___ 4. Close the Help view.

1.2. Running the debug configuration

- ___ 1. From the menu, click **Run > Debug Configurations**.

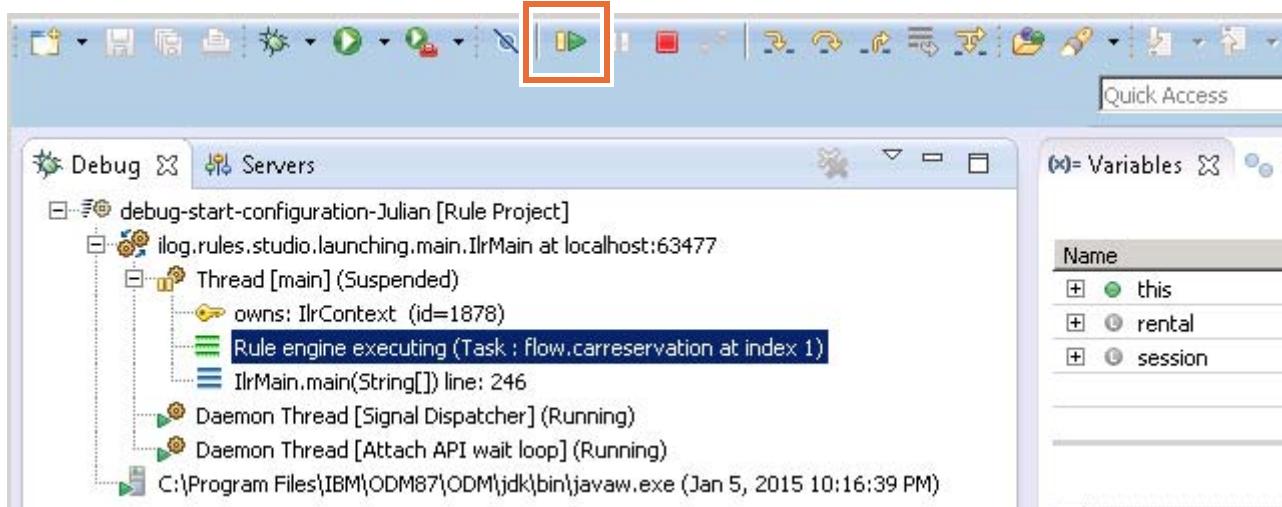
2. In the Debug Configurations window, expand **Rule Project** and select the **debug-start-configuration-Julian** launch configuration.



The **debug-start-configuration** is a launch configuration of type Rule Project that is configured to start the Rule Debugger.

3. Click **Debug**.
 4. When prompted to switch to the Debug perspective, click **Yes**.

- ___ 5. On the toolbar In the Debug perspective, click **Resume** (or press F8) to execute the application.



- ___ 6. Look at the results in the Console view.

As you scroll through the results, notice that the prices for the special offers are set to 0.00.

1.3. Setting breakpoints in the rules

To identify the reason for these results, you must watch the rules that are related to special offers in the `pricing` package.

- ___ 1. Switch back to the Rule perspective.

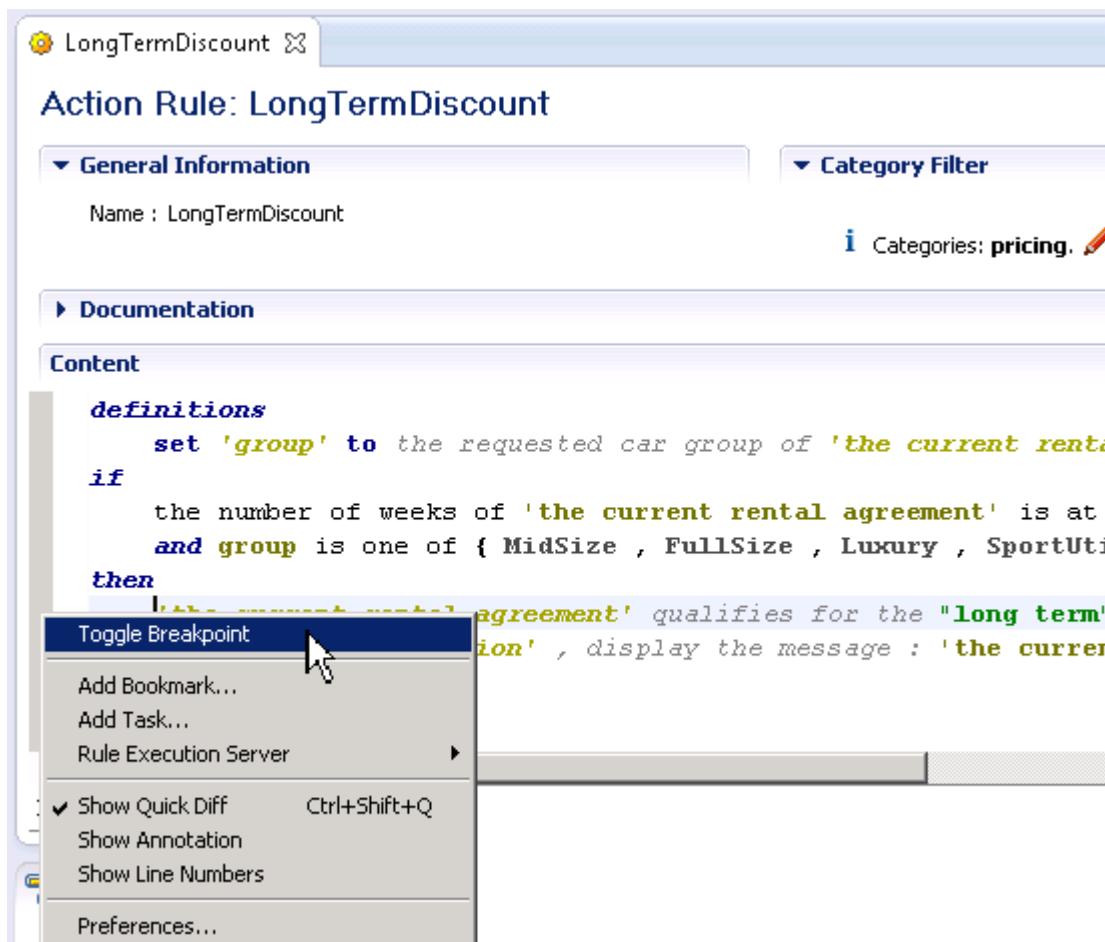


- ___ 2. In Rule Explorer, expand `debug-rules-start > rules > pricing.qualifyFor` and open the `LongTermDiscount` rule with the Intellirule editor.

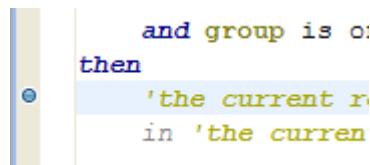


Double-clicking an action rule opens it in the most recently used rule editor, either the Guided editor or the Intellirule editor. To make sure that you open an action rule with the Intellirule editor, right-click it and click **Open With > Intellirule Editor**.

3. In the Intellirule editor, right-click the shaded border in the Content section beside the first line of the `then` statement, and click **Toggle Breakpoint**.



A breakpoint is now visible on the shaded border in the Content section of this rule.



4. Add a similar breakpoint on the first action statements in these rules:

- `pricing.qualifyFor.SpringSeason` rule
- `pricing.price.LongTermDiscount` rule

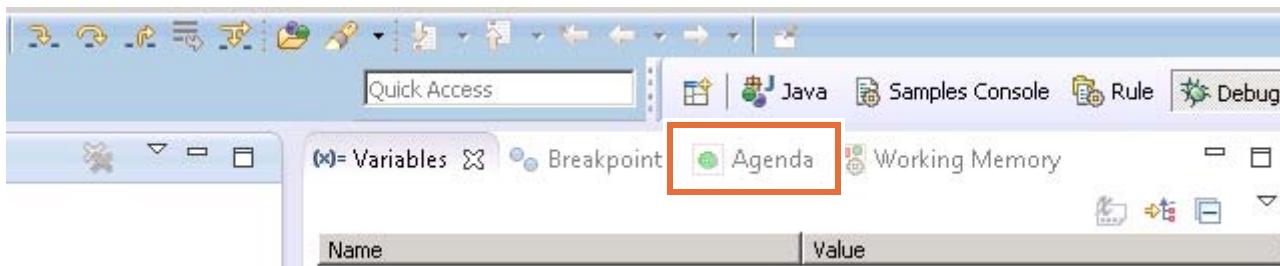
By adding these breakpoints, you ask execution to stop when it reaches these points so that you can check what is happening in the agenda.

Section 2. Debugging with breakpoints

- 1. Restart the debugging session (as you did in Section 1, "Running the debug launch configuration," on page 13-2).
 - a. From the menu, click **Run > Debug Configurations**.
 - b. In the Debug Configurations window, expand **Rule Project** and select the `debug-start-configuration-Julian` launch configuration.
 - c. Click **Debug**.
 - d. Click **Yes** when prompted to switch to the Debug perspective.

2.1. Inspecting the agenda

- 1. In the Debug perspective, click the **Agenda** tab so that the Agenda view is visible.



- 2. On the toolbar, click **Resume** (or press F8) to start the execution of the application.
- 3. When the execution stops, look at the Agenda view.

| Name | Value |
|--|---|
| <code>pricing.qualifyFor.LongTermDiscount</code> | <code>IlAgendaRuleInfo (id=1935)</code> |
| <code>pricing.qualifyFor.SpringSeason</code> | <code>IlAgendaRuleInfo (id=1835)</code> |

You can see that the rules that are currently instantiated in the Agenda are:

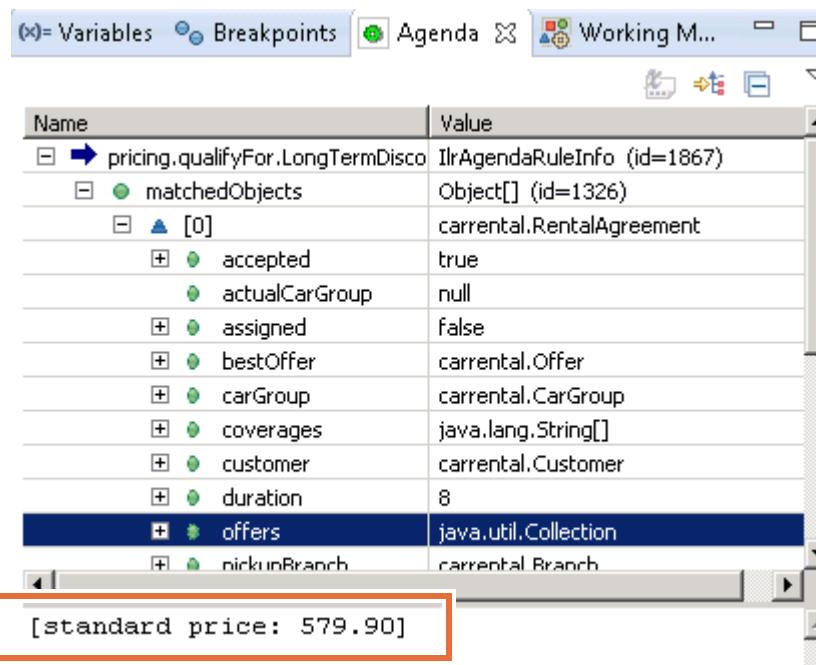
- `pricing.qualifyFor.LongTermDiscount`
- `pricing.qualifyFor.SpringSeason`

The `qualifyFor.LongTermDiscount` is marked with an arrow, meaning that execution is stopped at the breakpoint in this rule.

However, the `pricing.price.LongTermDiscount` rule is not listed, although it is supposed to run every time the `pricing.qualifyFor.LongTermDiscount` rule is true.

- 4. Expand `pricing.qualifyFor.LongTermDiscount > matchedObjects` for `carrental.RentalAgreement`.

5. Inspect the value of `offers` by clicking `offers=java.util.Collection` and note the current value in the detail pane.

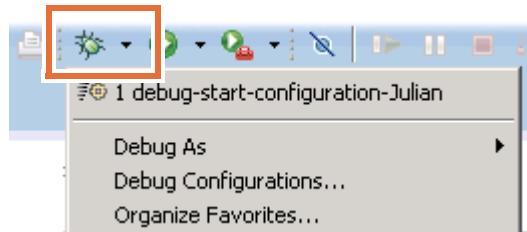


6. Click **Resume** (or press F8) to resume debugging until execution stops at the next breakpoint, and again note which rules were put in the agenda, and which one is being fired.
Execution stops in the `pricing.qualifyFor.SpringSeason` rule.
Normally, the `pricing.price.LongTermDiscount` rule is supposed to run before the `pricing.qualifyFor.SpringSeason` rule.
7. In the Agenda view, again check the value of `offers=java.util.Collection`.
The detail pane now displays the price for the long-term offer, which is 0.00. This price is incorrect.
8. Click **Terminate** to stop debugging.

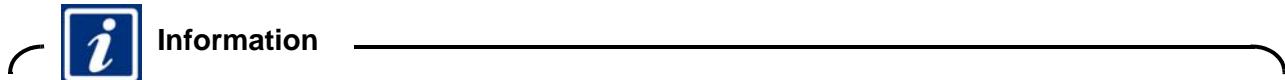
Section 3. Debugging with breakpoints in the ruleflow

In this section, you add breakpoints in the ruleflow of the rule project.

- ___ 1. Switch to the Rule perspective.
- ___ 2. In Rule Explorer, expand **debug-rules-start > rules > flow** and double-click the carreservation ruleflow in the `flow` package.
- ___ 3. Set breakpoints on the `eligibility` and `pricing` tasks.
 - ___ a. Select the `eligibility` task, and right-click it to select **Toggle Breakpoint**.
 - ___ b. Add another breakpoint on the `pricing` task.
- ___ 4. Switch to the Debug perspective again, and click the **Debug** icon on the toolbar to restart debugging with the `debug-start-configuration-Julian` configuration.



- ___ 5. When the execution stops on the `eligibility` task, open the **Variables** view to find the `rental` variable.
- ___ 6. Click `rental` to see which customer's rental agreement is the current value of the `rental` variable.
The rental agreement for Julian Bayles should be listed in the detail pane.
- ___ 7. Resume debugging (F8) until the execution stops.



With the current rule project, the special offers are always equal to 0. This situation happens because:

- The default value for each offer is set to 0.
- The `pricing.price.*` rules set the offers.

When the `qualifyFor` rules determine that the rental agreement qualifies, it adds that offer to the rental agreement.

The problem is that the rules do not update the `RentalAgreement` object, and the rule engine cannot see the offer to evaluate it. As a result the application does not run the rules that compute the price.

In the BOM, the **Update object state** property of the `addOffer` method of the `RentalAgreement` class is not set. As a consequence, the rule engine is not notified of the update of the rental agreement. Although the RetePlus algorithm is used on the Pricing task, the rule engine does not reevaluate the rule instances to execute. The rule engine thus does not add the appropriate special offer price rule to the agenda.

To obtain special offers, you must make sure that the **Update object state** check box of the `addOffer` method of the `RentalAgreement` class is selected in the BOM editor.



Section 4. Resolving the problem

In this section, you resolve the issue.

- ___ 1. Switch to the Rule perspective.
- ___ 2. In Rule Explorer, expand **debug-rules-start > bom > model > carrental > RentalAgreement**, and double-click the `addOffer` method to open it in the BOM editor.
- ___ 3. On the Member page for `addOffer`, select **Update object state**.

Member addOffer (class: carrental.RentalAgreement)

General Information

Name: `addOffer`

Type: `void`

Class: `carrental.RentalAgreement`

Static Final
 Deprecated Update object state

Member Verbaliza

✖ Remove the verba

✚ Create an action

Action : "a renta

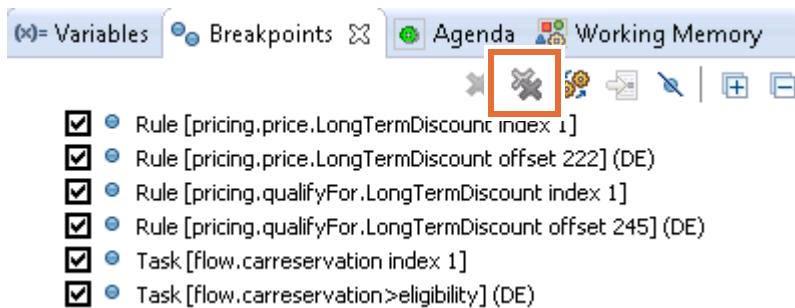
Template: `{this}`

- ___ 4. Save your work.
- ___ 5. Return to the Debug perspective and run the debug tools again.

If you debug with the Agenda view open, you see that the `pricing.price.LongTermDiscount` rule now shows up in the Agenda as expected. The behavior is resolved from Activity 2.1, "Inspecting the agenda," on page 13-6.

In the detail pane, you see that the special offers are now correctly included in the final price.

- ___ 6. After you finish debugging, open the Breakpoints view and clear the breakpoints by clicking the **Remove All Breakpoints** icon.



- ___ 7. Click **Yes** when prompted to confirm removal of the breakpoints.

End of exercise

Exercise review and wrap-up

This exercise looked at how to debug a ruleset with the Rule Debugger. You saw how to set breakpoints on an action rule and on a task in a ruleflow, and you inspected objects in the working memory and rule instances in the agenda.

To see the solution projects for this exercise, switch to a new workspace and import the project **Rule Designer > Tutorials > Debugging a ruleset > answer > Import projects**.



Note

You can use the Rule Debugger with a Java client application. You can also create a launch configuration for this purpose, which is then of type **Java Application with Rules**, and have it configured to start the Rule Debugger.

Exercise 14. Enabling tests and simulations

What this exercise is about

This exercise teaches you how to set up testing and simulation functionality for business users.

What you should be able to do

After completing this exercise, you should be able to:

- Validate the BOM and generate scenario file templates in Excel format
- Customize scenario file templates
- Create virtual attributes to test collections of complex objects
- Set up a test server and run tests in Rule Designer to validate remote conditions

Introduction

In this exercise, you set up Decision Validation Services on a remote server, and you run tests in Rule Designer to validate the remote testing environment.

As developers, your role with Decision Validation Services is mainly to enable business users to run tests and simulations with minimal dependence on you. This exercise provides an overview of the tasks that are involved, including:

- Section 1, "Validating the rule project"
 - Choosing a constructor
 - Editing argument names for column headings in the template
- Section 2, "Generating the scenario file template"
- Section 3, "Populating the template and testing the scenario"
- Section 4, "Customizing input for the scenario file template"
- Section 5, "Running tests remotely by deploying the XOM"
- Section 6, "Creating a DVS project to enable remote tests"
- Section 7, "Running tests on the remote server"
- Section 8, "Repackaging the SSP in a DVS Project"
- Section 9, "Testing the redeployed SSP"

Requirements

There are no specific requirements for this exercise.

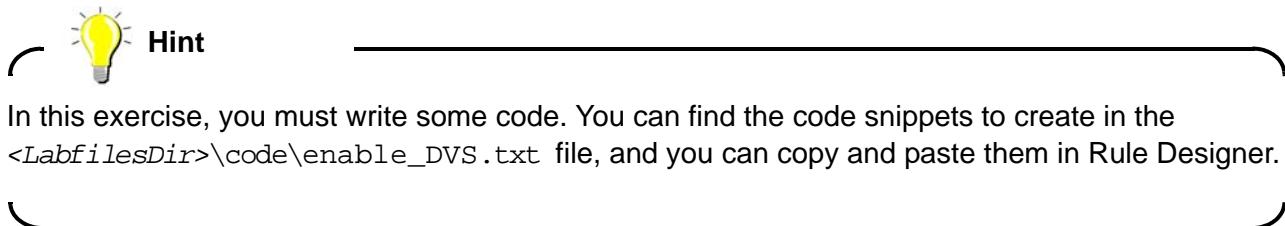
Section 1. Validating the rule project

To use the default Excel format for your scenarios, you must validate that the rule projects to be tested can correctly generate the appropriate columns in the Excel scenario file template.

You use the DVS Project Validation view when you validate your rule project. This view raises error and warning messages if the BOM or the input parameters of the rule project must be modified.

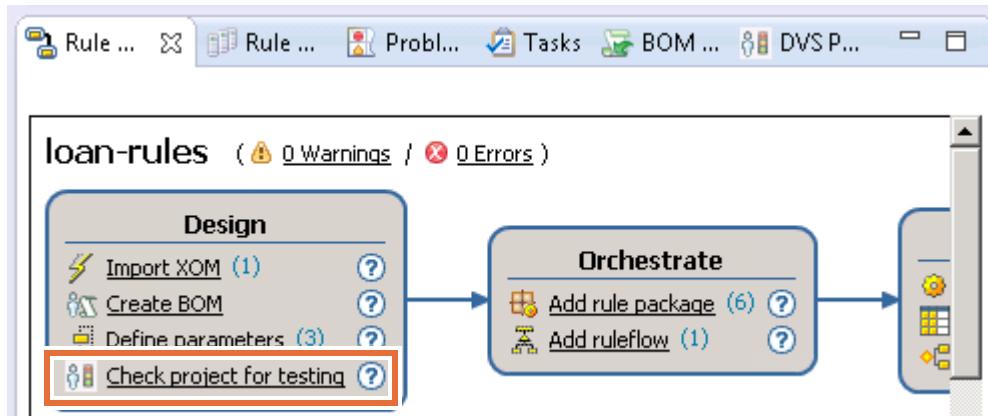
1.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace:
 - a. From the **File** menu, click **Switch Workspace > Other**.
 - b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\enable_dvs`
 - c. Close the Welcome view and switch to the Samples Console perspective.
- 2. Close the Welcome view and switch to the Sample Console perspective.
- 3. Select **Rule Designer > Training > Ex 14: Enabling tests and simulations > 01-start > Import projects**.
- 4. Close the Help view.



1.2. Validating the rule project

- 1. In Rule Explorer, make sure the `loan-rules` project is selected so that the Rule Project Map opens.
- 2. On the Rule Project Map, click **Check project for testing**.



**Note**

You can also check the project by right-clicking the `loan-rules` project in Rule Explorer, and clicking **Decision validation service > Check Project**.

- 3. In the DVS Project Validation view, select the error to open and read the Solution Description.

| ID | BOM Element |
|------------|--------------------------|
| GBRTV0003E | training_loan.Borrower[] |

Solution Description:

input parameter borrowers of type training_loan.Borrower[]: cannot find a Decision Validation Services constructor for the business object model class. If there are no constructors, create one. If there are several constructors in the business object model class, you must specify a constructor by selecting the "DVS constructor" option in the business object model

An error is raised because no class constructor was defined or specified as the Decision Validation Services constructor.

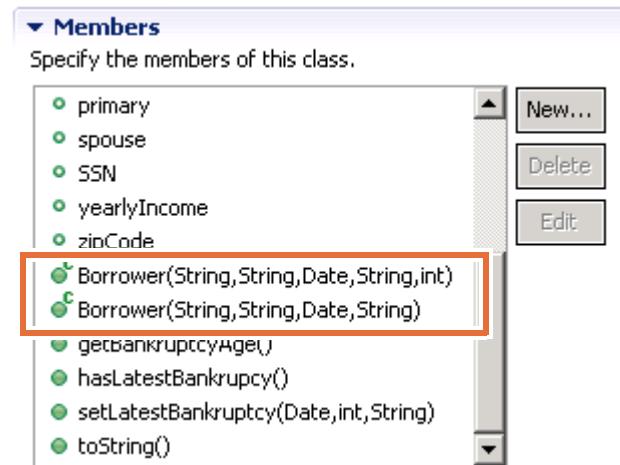
1.3. Choosing a constructor

BOM classes that are used to define ruleset input parameters must have at least one constructor. One of these constructors must be specified as the constructor for Decision Validation Services. The columns of the Microsoft Excel scenario file template then correspond to the argument of this BOM class constructor.

To choose the correct constructor:

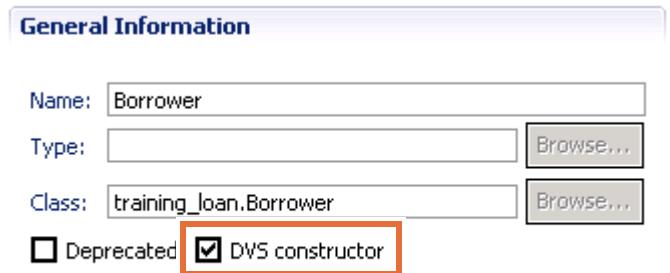
- 1. In the Rule Explorer, expand **loan-rules > bom > model > training_loan** and double-click **Borrower** to open the BOM editor and fix the problem.

The Class page opens for the `Borrower` class. Scroll through the Members list and notice that `Borrower` has two constructors.



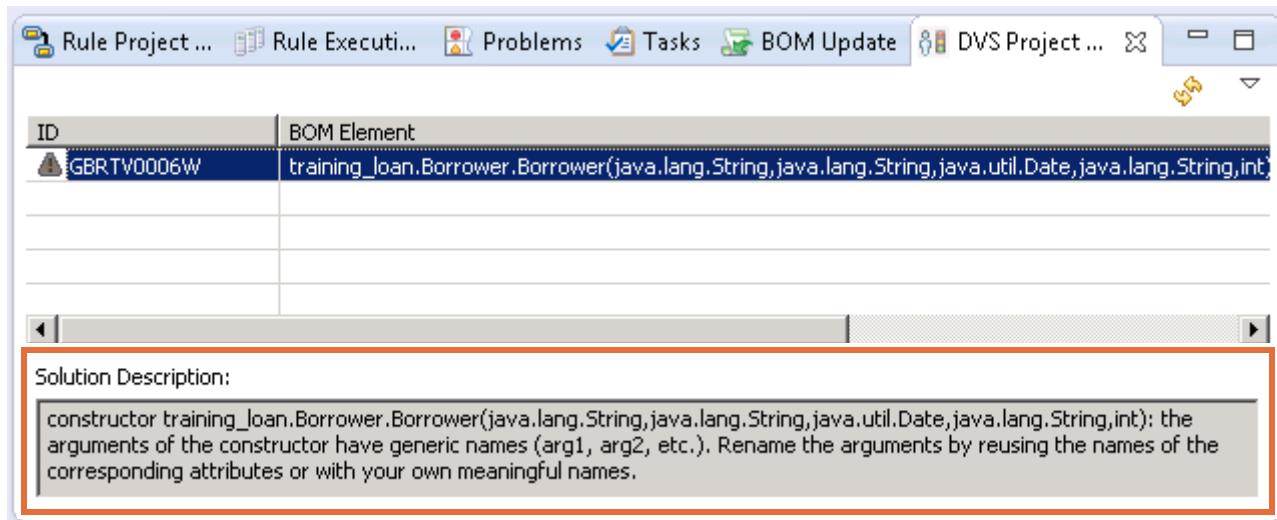
When there are several constructors for the same class, you must specify which one to use for the template.

- ___ 2. In the Members section of the `Borrower` class page, double-click **`Borrower(String, String, Date, String, int)`**, which is the first constructor.
- ___ 3. In the Member page, select the **DVS constructor** check box.



- ___ 4. Save your work.
- ___ 5. Check the project again, either from the Rule Project Map or in Rule Explorer, by right-clicking `loan-rules` and clicking **Decision Validation Services > Check Project**.

This time, the DVS Project Validation view displays a warning message, indicating that the constructor arguments use generic names that you must rename with meaningful names.



The arguments (arg1, arg2, arg3, arg4, and arg5) correspond to the arguments of the selected constructor:

`Borrower (String, String, Date, String, int)`

- 6. Double-click the warning message to open the Borrower page and edit the constructor argument names.

Arguments
Edit the arguments of this member.

| Name | Type | Domain |
|------|------------------|--------|
| arg1 | java.lang.String | |
| arg2 | java.lang.String | |
| arg3 | java.util.Date | |
| arg4 | java.lang.String | |
| arg5 | int | |

1.4. Editing argument names for column headings in the template

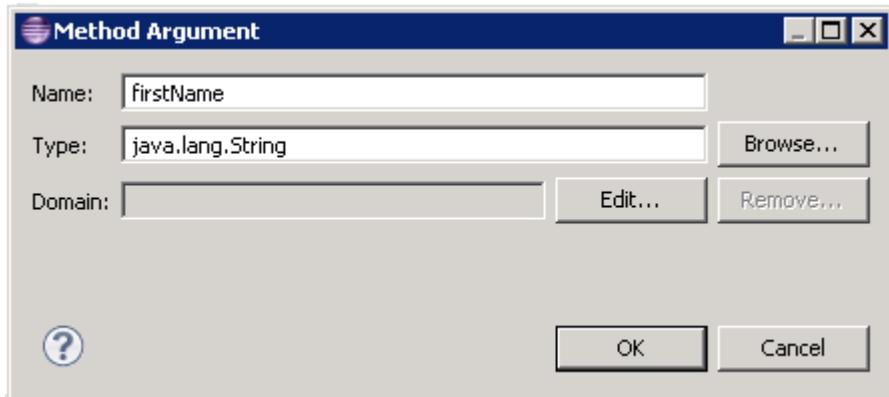
The argument names become the column headings in the scenario file template. You should edit the argument names to make them meaningful for the business users who use the scenario file.

When you rename the generic arguments, make sure that the names match the corresponding class attribute names. For example, if `arg1` sets the value of the `firstName` attribute, then rename `arg1` as: `firstName`

To rename the constructor arguments for column headings:

- 1. In the Arguments section of the `Borrower` class page, double-click `arg1`.

- __ 2. In the **Name** field, change `arg1` to `firstName` and click **OK**.



The name must match the corresponding attribute name in the `Borrower` class.

When the template is generated, the column headings display the verbalized name of the corresponding attribute or argument. For example, the generated template column for the `firstName` attribute uses the heading: `first name`

- __ 3. Edit the names of the remaining arguments to these attribute names:

- `secondName`
- `birthDate`
- `ssn`
- `creditScore`

- __ 4. Save your work.

▼ Arguments
Edit the arguments of this member.

| Name | Type | Domain |
|--------------------------|-------------------------------|--------|
| <code>firstName</code> | <code>java.lang.String</code> | |
| <code>secondName</code> | <code>java.lang.String</code> | |
| <code>birthDate</code> | <code>java.util.Date</code> | |
| <code>ssn</code> | <code>java.lang.String</code> | |
| <code>creditScore</code> | <code>int</code> | |
| | | |
| | | |
| | | |

- __ 5. Check the project again (**Decision Validation Services > Check Project**).

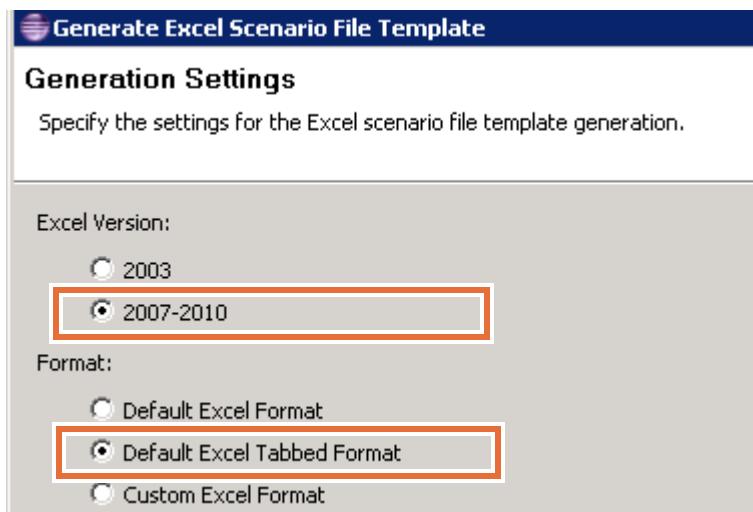
The project now has no warnings.

Section 2. Generating the scenario file template

Decision Validation Services uses Microsoft Excel 2007 or Microsoft Excel 2010 as the default format for scenario file templates. You can populate the template manually or by using data from a database.

To generate the Excel scenario file template:

- ___ 1. In the Rule Explorer, right-click `loan-rules` and click **Decision Validation Services > Generate Excel Scenario File Template**.
- ___ 2. In the Generate Excel Scenario File Template window, select the following values.
 - ___ a. In the Rule Project page, make sure that `loan-rules` is selected and click **Next**.
 - ___ b. In the Generation Settings page, make sure that:
 - **2007-2010** is selected as the **Excel version** option.
 - **Default Excel Tabbed Format** is selected as the **Format** option.



- ___ c. Leave the remaining fields in the Generation Settings page unchanged, and click **Next**.

- __ d. In the Expected Results page, expand **the loan report**, which is one of the output ruleset parameters, and select the attributes that you want to test:
- approved
 - corporate score
 - messages

| Element | Operator |
|---|--------------------|
| <input checked="" type="checkbox"/> the loan report | |
| <input checked="" type="checkbox"/> approved | equals |
| <input checked="" type="checkbox"/> hnrrowers | |
| <input checked="" type="checkbox"/> corporate score | equals |
| <input type="checkbox"/> insurance | |
| <input type="checkbox"/> insurance rate | |
| <input type="checkbox"/> insurance required | |
| <input type="checkbox"/> loan | |
| <input type="checkbox"/> loan grade | |
| <input type="checkbox"/> message | |
| <input checked="" type="checkbox"/> messages | equals (unordered) |
| | valid data |



Warning

Make sure that you select the **messages** attribute, with the final “s”, not the **message** field.

These attributes are transformed into columns on the **Expected Results** sheet of the generated scenario file template. You must then complete these columns with the values of these attributes that you expect to be produced at run time. The values that you enter are then compared with the actual execution results at test time to give the test results.



Information

For each attribute, you can select the operator that is used for the test. The possible operators depend on the type of the attribute. The default operator is the equality operator, meaning that the test succeeds if the value entered in the `testsuite.xlsx` equals the value that is obtained at run time. You might use other operators such as does not equal, is greater than, is lower than, or contains.

- __ e. Keep the default operators, and click **Next**
 __ f. In the Expected Execution Details page, click **Finish**.

The `testsuite.xlsx` file becomes available in the `loan-rules` project in the Rule Explorer.

- __ 3. In the Rule Explorer, right-click `testsuite.xlsx` and click **Open With > System Editor** to open it with Microsoft Excel Viewer.

-
- ___ 4. Explore the testsuite.xlsx scenario file template:
 - ___ a. Note the sheets that are generated: the **Scenarios** sheet, the **Expected Results** sheet, and a separate sheet with columns for the `Borrower` class attributes.
 - ___ b. Notice that the column headings on each of the sheets are the verbalized names of the classes and attributes.
 - ___ c. Close the test suite when you are finished.

Section 3. Populating the template and testing the scenario

The scenario file template can be used to store up to thousands of scenarios. For tests and simulations that require a large number of scenarios, you can import the data from a database.



Note

For this exercise, you use a prepopulated scenario file.

If you have access to Microsoft Excel on your computer, you can modify the scenario file template directly instead of using the prepopulated file.

3.1. Viewing the prepopulated scenario file

- ___ 1. In the Rule Explorer, expand the `data` folder in the `loan-rules` project.
- ___ 2. Right-click the `testsuite-populated1.xlsx` file, and click **Open With > System Editor** to open it with Microsoft Excel Viewer.
A set of values and expected results are provided for one scenario. In the **Scenarios** sheet, notice that the `monthlyRepayment` and the `yearlyInterestRate` columns are empty. The values for these attributes are calculated as a result of rule execution, so the columns remain empty.
- ___ 3. Close the `testsuite-populated1.xlsx` file.

3.2. Running the tests with the populated scenario file

- ___ 1. From the menu, click **Run > Run Configurations**.
- ___ 2. In the Run Configurations wizard, double-click **DVS Excel File** to create a configuration.
- ___ 3. In the **Name** field of the configuration, enter: `loan local DVS`
- ___ 4. On the **Excel File** tab, select the following values.
 - ___ a. For the **Excel File** field, click **Browse** to select `loan-rules > data > testsuite-populated1.xlsx`, and click **OK**.
 - ___ b. For the **Classic Rule Project** field, click **Browse** and select `loan-rules`.
 - ___ c. For the **HTML Report** field, enter:
`\loan-rules\data\report1`
- ___ 5. On the **DVS Configuration** tab, make sure that the **Local execution** option is selected.
- ___ 6. Click **Apply** and click **Run**.

The Console view shows the execution progress, and you should see the following traces after execution ends:

```
--- Output for scenario 'Scenario 1' :  
Borrower( firstName: John lastName: Doe born: Dec 1, 1965 SSN: 111-11-1111  
isPrimary: true zip code: 93210 income: 60000 credit score: 700)  
Loan( amount: 100000 starting: July 1, 2014 duration: 72 month LTV: 70% rate:  
5.7% monthly repayment: 1,643.16)  
Report( validData: true approved: true score: 870 grade: B insuranceRequired:  
true insuranceRate: 2% message: Low risk loan  
Congratulations! Your loan has been approved  
)  
Execution finished  
Starting generation of DVS HTML report now  
Finished generating DVS HTML report
```

- 7. In the Rule Explorer, right-click **loan-rules > data > report1.html** and click **Open With > Web Browser** to see the successful test results.



Hint

If you do not see the `report1.html` file, refresh the **loan-rules > data** folder in Rule Explorer by selecting the `data` folder and pressing F5.

- 8. When you are finished, close the report window.

Section 4. Customizing input for the scenario file template

Because of the relationship between the BOM and the scenario file, customizing the scenario file template requires modifying the BOM.

4.1. Removing columns from the template

When preparing the BOM to generate a template, you can exclude BOM members that are not useful as input to tests or simulations. For example, attributes that are based on calculated values should not be included as input columns because they do not have input values.

To remove columns:

- 1. In the Rule Explorer, expand **loan-rules > bom > model > training_loan**, and double-click **Loan** to open it in the BOM editor.
- 2. In the **Members** section of the **Class** page, double-click **yearlyInterestRate** to open it in the Member page.
In the Member page for the **yearlyInterestRate** attribute, you can see that **Ignore for DVS** is not selected, meaning that this attribute is included in the scenario file template.
- 3. Select **Ignore for DVS**.

The screenshot shows the IBM ODM BOM Editor interface. The title bar says "Member yearlyInterestRate (class: training_loan.Loan)". The left pane has a "General Information" tab with fields for Name (yearlyInterestRate), Type (double), Class (training_loan.Loan), and checkboxes for Read/Write (selected), Static, Deprecated, and Update object state. The "Ignore for DVS" checkbox is checked and highlighted with a red border. The right pane has sections for "Member Verbalization" (with options to remove verbalization, create navigation or action phrases, or edit subjects), "Navigation : 'the yearly interest rate of a lo" (with a template field containing "{yearly interest rate} of {this}"), and "Action : 'set the yearly interest rate of a lo" (with a template field containing "set the yearly interest rate of {this} to").

- 4. Repeat these steps to ignore the **monthlyRepayment** attribute of the **Loan** class.
 - 5. Save your work.
- No input columns are created for these attributes in the **Scenarios** sheet of the template. However, when you generate the template, you can still choose to include these attributes in the **Expected Results** sheet to test that their values are calculated correctly as a result of rule execution.

4.2. Creating a virtual attribute to test complex objects



Requirements

The business users want test results to indicate whether the borrower included in the loan report is the primary borrower. The `training_loan.Borrower` BOM member has a Boolean attribute called: `primary`

However, this `primary` attribute is not directly accessible from the `training_loan.Report` BOM member, which is returned as an output ruleset parameter.

To handle this request, you create a virtual attribute in the `Report` class that makes the `Borrower.primary` attribute visible in the test results.

To see the `Borrower.primary` attribute:

- 1. In the Rule Explorer, expand **loan-rules > bom > model > training_loan** and double-click **Borrower** to open it in the BOM editor.
- 2. Notice that the `primary` attribute is listed in the **Members** section of the Class page.

To see the `Report.borrowers` attribute:

- 1. In the Rule Explorer, expand **loan-rules > bom > model > training_loan** and double-click **Report** to open it in the BOM editor.
- 2. In the **Members** section of the Class page, double-click the `borrowers` attribute to open it.

Member `borrowers` (class: `training_loan.Report`)

General Information

Name: `borrowers`

Type: `training_loan.Borrower[]`

[Browse...](#)

Class: `training_loan.Report`

[Browse...](#)

Read/Write Read Only Write Only

Static Final

Deprecated Update object state

Ignore for DVS

- 3. In the Member page, click **Browse** next to the **Type** field, and notice that `borrowers` is an array of Borrower objects.

Decision Validation Services does not provide direct access to attributes of a complex type, such as collections of complex objects. To access the primary attribute for each Borrower class in the `training_loan.Report.borrowers` collection, you must create a virtual attribute.

To create the virtual attribute in the `Report` class:

- 1. Go back to the **Class** tab for the `Report` class in the BOM editor.

2. Create a virtual read-only attribute for the `Report` class:
- In the **Members** section, click **New**.
The New Member window opens.
 - Make sure that the **Attribute** option is selected.
 - In the **Name** field, enter: `borrowerPrimary`
 - In the **Type** field, enter: `string`
 - Click **Finish**.
- The `borrowerPrimary` attribute is now listed in the Members section.
3. Double-click **borrowerPrimary** to define the BOM-to-XOM mapping for the new virtual attribute.
- On the Member page, select **Read Only**.
 - In the Member Verbalization section, click **Create a default verbalization**.
 - Click **Edit the subject used in phrases**.
 - Change the verbalization in the **Singular** field from `borrower primary` to `primary borrower` and click **OK**.
 - In the **BOM to XOM Mapping** section, type the following code in the **Getter** field:

```
StringBuilder builder = new StringBuilder();
for(int i = 0; i < this.borrowers.length; i++) {
    builder.append(this.borrowers[i].primary + ",");
}
return builder.toString();
```

**Hint**

You can find this code snippet in the `<LabfilesDir>\code\enable_DVS.txt` file.

**Note**

Notice the comma “,” that is added within the loop to separate the values for the different elements in the array. For each `borrower`, the appended text is either “true,” or “false.”. The returned String is thus a series of Boolean values that are followed with a comma.

4. Save your work.

4.3. Regenerating the scenario file template

Now that you customized the BOM, regenerate the scenario file template.

- 1. In the Rule Explorer, right-click **loan-rules** and click **Decision Validation Services > Generate Excel Scenario File Template**.
- 2. In the Generate Excel Scenario File Template wizard:
 - a. In the Rule Project page, make sure that **loan-rules** is selected and click **Next**.
 - b. In the Generation Settings page, set the following options, and click **Next**.
 - Select **2007-2010** as the **Excel version** option.
 - Select **Default Excel Tabbed Format** as the **Format** option.
 - In the **Excel Scenario File Name** field, enter:
`/loan-rules/testsuite2.xlsx`
 - c. In the Expected Results page, expand **the loan report**, and select the following criteria.
 - **approved**
 - **corporate score**
 - **messages**
 - The new **primary borrower** attribute

Expected Results

Select the columns to include in the Expected Results sheet

| Element | Operator |
|--|--------------------|
| <input checked="" type="checkbox"/> the loan report | |
| <input checked="" type="checkbox"/> approved | equals |
| <input checked="" type="checkbox"/> corporate score | equals |
| <input type="checkbox"/> insurance | |
| <input type="checkbox"/> insurance rate | |
| <input type="checkbox"/> insurance required | |
| <input type="checkbox"/> loan | |
| <input type="checkbox"/> loan grade | |
| <input type="checkbox"/> messages | equals (unordered) |
| <input checked="" type="checkbox"/> primary borrower | equals |

- d. Click **Finish**.
- 3. In Rule Explorer, in the **loan-rules** project, right-click the **testsuite2.xlsx** file, and click **Open With > System Editor** to open it with Microsoft Excel Viewer.
- 4. Notice that the columns **monthly repayment** and **yearly interest rate** are no longer visible in the Scenarios spreadsheet.

Your new template is ready to be populated with scenario values and expected results.



Information

Again, for this exercise, you use the prepopulated scenario file that is called `testsuite-populated2.xlsx`, which is in the `data` folder. A set of values and expected results are provided for one scenario.

If you open the populated file, notice that the value given for the column `the primary borrower of the loan report equals` is “true,” with the final comma.

- 5. Close the `testsuite2.xlsx` file.

4.4. Testing the customized scenario file

- 1. Run the DVS Excel file launch configuration with the prepopulated scenario:
 - a. From the **Run** menu, click **Run Configurations**.
 - b. In the **Run Configurations** wizard, select **DVS Excel File > loan local DVS**.
 - c. In the **Excel file** field, browse to select **loan-rules > data > testsuite-populated2.xlsx** and click **OK**.
 - d. Change the report name in the **HTML Report** field to:
`\loan-rules\data\report2`
 - e. Click **Apply** to save your changes, and then click **Run**.
 - f. After execution, refresh the `data` folder in Rule Explorer (by pressing F5).

- ___ g. Open the updated `report2.html` file by right-clicking it and clicking **Open With > Web Browser**.

| Name | Success Rate | Tests | Failures | Errors | Message |
|------------|--------------|-------|----------|--------|---------|
| Scenario 1 | 100% | 4 | 0 | 0 | |

Details by Scenario

Scenario 1

| Name | Success | Failure | Error | Message |
|--|---------|---------|-------|--|
| the loan report is approved equals ✓ | | | | The observed value true is the expected value. |
| the corporate score in the loan report equals | ✓ | | | The observed value 870 is the expected value. |
| the messages of the loan report equals (unordered) | ✓ | | | The set of values [Low risk loan, Congratulations! Your loan has been approved] equals (unordered) the expected values [Low risk loan, Congratulations! Your loan has been approved] |
| the primary borrower of the loan report equals | ✓ | | | The observed value true, is the expected value. |

The report shows 100% success and proves that the value for the virtual `borrowerPrimary` attribute was accessed successfully.

- ___ 2. When you are finished viewing the report, you can close the report window and any remaining windows that are open in Rule Designer.

Section 5. Running tests remotely by deploying the XOM

In this section, you deploy the XOM that is required to execute the ruleset. You then run the tests remotely by using the Scenario Service Provider (SSP). After you deploy the XOM to Rule Execution Server, you can test your ruleset remotely without repackaging the SSP with the XOM and then redeploying the SSP.



Information

In this exercise, you are introduced to the notion of XOM deployment to Rule Execution Server. You learn more about XOM deployment in Unit 12, *Deploying rules and XOMs*.

To deploy the XOM to Rule Execution Server:

1. If the sample server is not already running, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.



Warning

Make sure that the sample server is started before continuing with the next steps.

2. In Rule Explorer, right-click the `loan-rules` project and click **Decision Validation Services > Deploy XOM for Remote DVS Testing**.

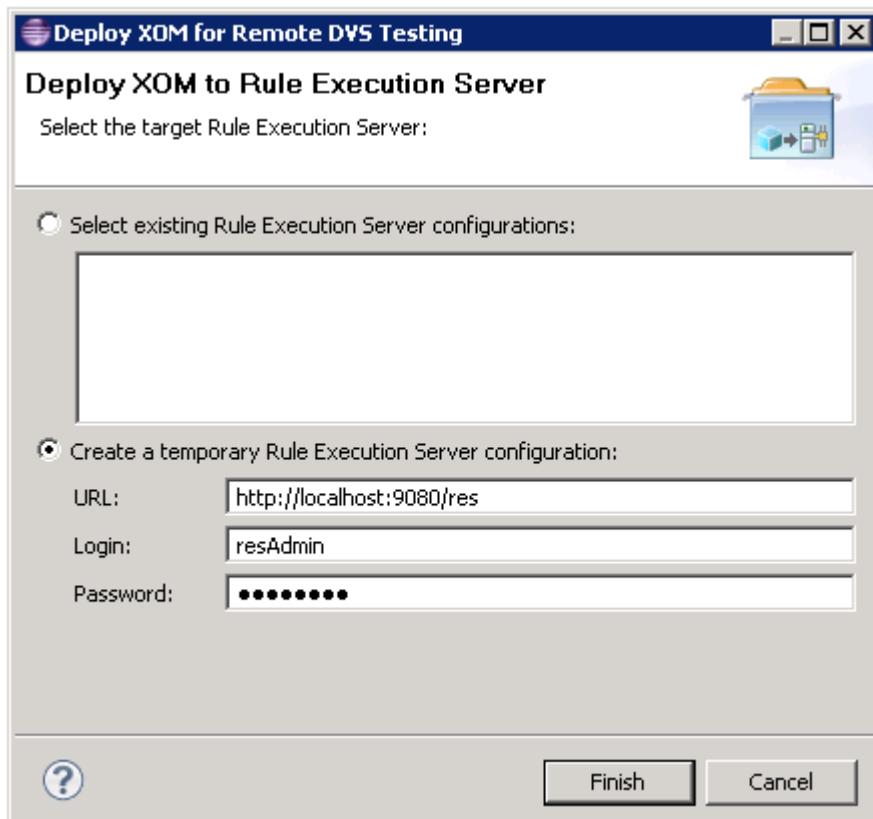


Troubleshooting

If you receive a Java notification prompt, you can select **Do not show this message again** and click **OK**.



The “Deploy XOM for Remote DVS Testing” window opens.



- 3. Make sure that the configuration settings are correct:

- **Create a temporary Rule Execution Server configuration** is selected
- **URL:** http://localhost:9080/res
- **Login:** resAdmin
- **Password:** resAdmin



Note

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

- 4. Click **Finish** to connect to Rule Execution Server, and notice the information in the Console view:

Following XOM libraries have been deployed:
resuri://loan-xom.zip/1.0



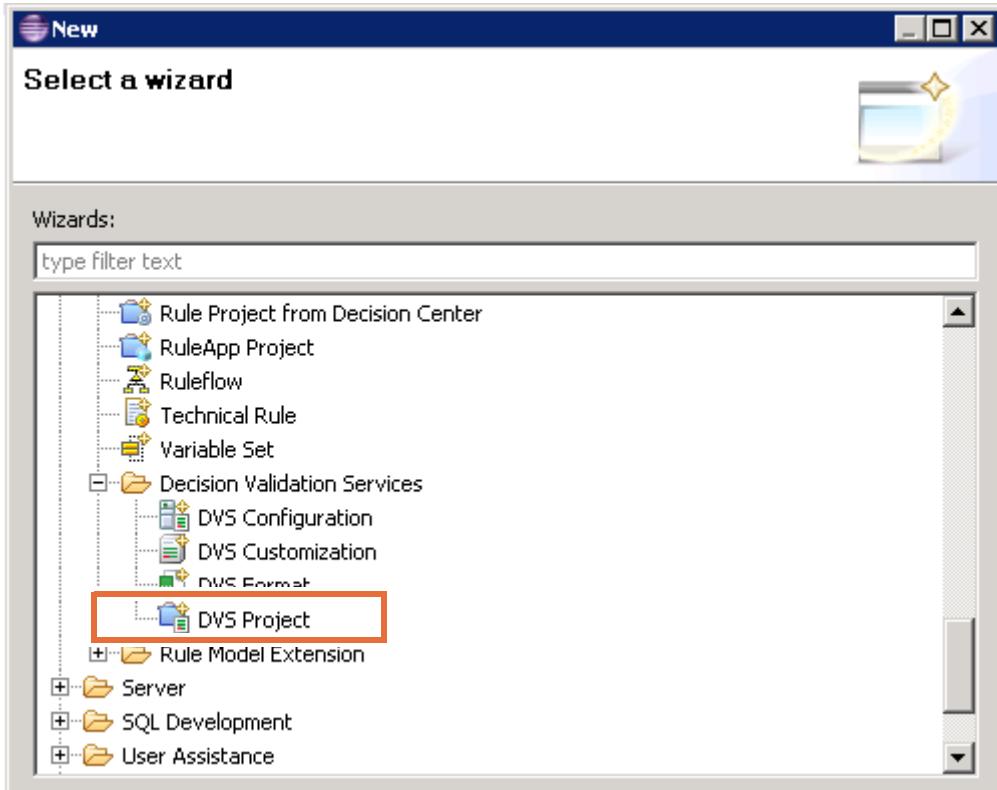
Information

The URI to the deployed XOM is identified as: resuri://loan-xom.zip/1.0

Section 6. Creating a DVS project to enable remote tests

In this section, you provide the ability for you or business users to run tests and simulations remotely. You do so by defining a DVS project that specifies how to access Rule Execution Server, as follows.

- 1. In the Rule Explorer, right-click anywhere to open the menu, and click **New > Other**.
- 2. Expand **Rule Designer > Decision Validation Services**, select **DVS Project**, and click **Next**.



- 3. In the DVS Project page, leave the name as **DVS Project** and click **Next**.
- 4. In the Customization Name page, keep **Customization** as the default name and click **Finish**.

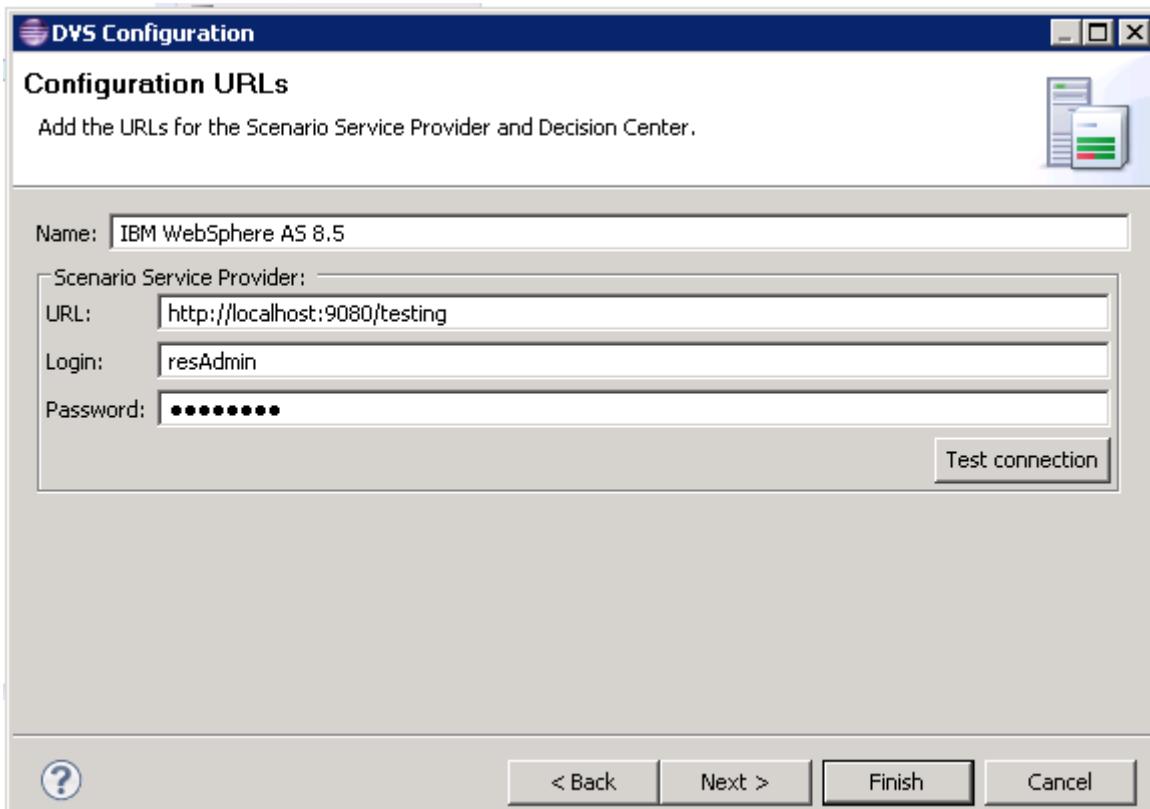
The **DVS Project** is now visible in the Rule Explorer, and contains an empty customization called **Customization.sspc**.

The **Customization.sspc** file opens in the DVS Customization editor.

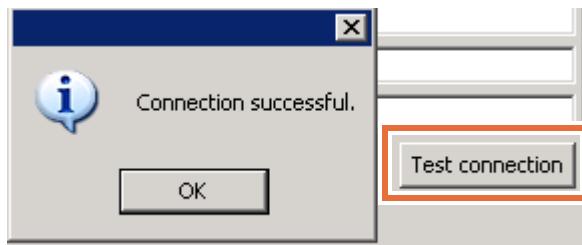
- 5. Edit the customization to create a configuration that defines the Rule Execution Server that is used to execute the remote tests.
 - a. In the Configurations section on the **Overview** tab, click **Create** to create a configuration.
 - b. In the “Configure environment” window, select **IBM WebSphere AS 8.5**, and click **Next**.

- ___ c. In the Configuration URLs page, keep the default values:

- **Name:** IBM WebSphere AS 8.5
- **URL:** http://localhost:9080/testing
- **Login:** resAdmin
- **Password:** resAdmin



- ___ d. Click **Test connection** to verify that the entered values are correct, and click **OK** if the connection is successful.



Make sure that you use the correct port for your environment. If the connection test fails, correct the problem and try again until the connection succeeds.

- ___ e. Click **Next** to view the Decision Validation Service Archive page.

Take some time to look at this page. Because you do not require a specific archive, leave all fields unchanged to use the default archive.

- ___ f. Click **Next** to view the Decision Center Application Archive page.

Take some time to look at this page. Again, because you do not require a specific archive, leave all fields unchanged to use the default archive.

- ___ g. Click **Finish**.

In the Rule Explorer, you can now see the Decision Validation Services configuration that is called `IBM WebSphere AS 8.5.smc` in the DVS Project. This DVS configuration also opens in the DVS Configuration editor.

- __ h. Save all your work (Ctrl+Shift+S), including the `Customization.sspc` file.

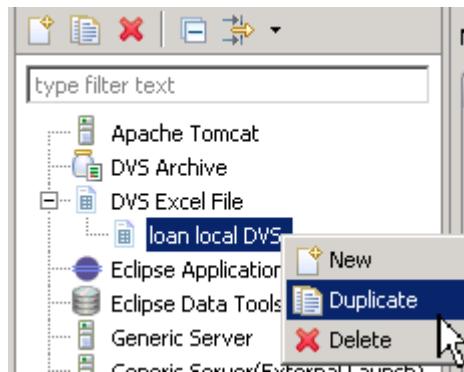
Section 7. Running tests on the remote server

In this section, after you deployed the XOM and created the appropriate DVS project, you can now run the tests remotely from Rule Designer. You can copy the configuration that you created for the previous run with some changes to enable remote testing of the same scenarios.

- 1. From the **Run** menu, click **Run Configurations**.

The Run Configurations window opens.

- 2. Copy your **loan local DVS** configuration by right-clicking **loan local DVS** and clicking **Duplicate**.



A duplicate configuration is created, as **loan local DVS (1)**.

- 3. Rename the duplicate configuration by entering the following text in the **Name** field:

loan remote DVS

- 4. On the **Excel File** tab, change the **HTML Report** field to:

/loan-rules/data/report-remote.html

- 5. On the **DVS Configuration** tab, complete the execution details:

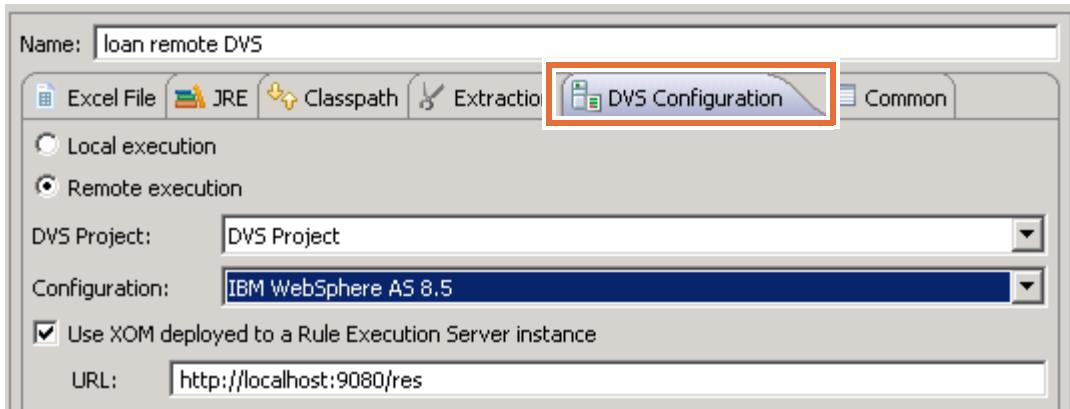
- a. Select **Remote execution**.

- b. In the **DVS Project** field, make sure that **DVS Project** is selected.

- c. In the **Configuration** field, select **IBM WebSphere AS 8.5**.

The **URL** field is automatically set to the URL of the Rule Execution Server that you entered in the previous section.

- __ d. Make sure that **Use XOM deployed to Rule Execution Server** is selected.



- __ 6. Click **Apply** to save your changes to the file.
- __ 7. Click **Run**.
- __ 8. After execution finishes, view the report.
- __ a. Refresh the `loan-rules > data` folder in Rule Explorer (by selecting the `data` folder and pressing F5) to see the newly created `report-remote.html` file.
- __ b. To open the report, right-click the `report-remote.html` file in the `data` folder, and click **Open With > Web Browser**.

The report shows that the test executed successfully. The **Execution** line indicates that the execution was remotely done on the testing server, so you now know that the remote testing environment works.

| Summary | |
|--------------------------|--|
| Execution | Remote on <code>http://localhost:9080/testing</code> |
| Decimal Precision | 2 digits are used |
| Scenarios | 1 |
| Tests | 4 |
| Success Rate | 100% |
| Failures | 0 |
| Errors | 0 |

- __ 9. Close the report.

Section 8. Repackaging the SSP in a DVS Project

In this section, you repackage the SSP to include the XOM, by using the DVS project. You then deploy it. Finally, you run tests remotely by using the deployed SSP.

8.1. Repackage the SSP

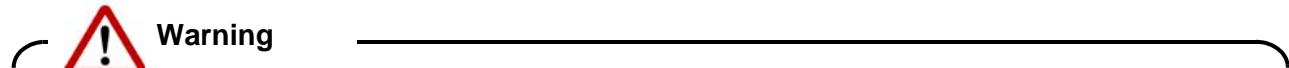
- ___ 1. In the Rule Explorer, right-click **loan-rules** and click **Decision Validation Services > Repackage XOM for Remote DVS Testing**.
- ___ 2. In the Rule Project page, make sure that **loan-rules** is selected and click **Next**.
- ___ 3. In the **DVS Properties** page, verify these settings, and then click **Finish**:
 - **DVS Project:** DVS Project
 - **Customization:** Customization
 - **Configuration:** IBM WebSphere AS 8.5

When the repackaging is completed, the “Repackaging status” window opens. This message indicates that the `jrules-ssp-WAS85.ear` file is updated in the **DVS Project** folder in your workspace.

- ___ 4. Click **OK** to close the “Repackaging status” window.
- ___ 5. In the Rule Explorer, expand **DVS Project > applicationservers > WebSphere85** to see the new `jrules-ssp-WAS85.ear` file.

8.2. Deploying the SSP to the application server

- ___ 1. Open and sign in to the WebSphere Application Server administrative console.
 - ___ a. Click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Administrative console**.



When you start the WebSphere Application Server administrative console, you might have a message that indicates that there is a problem with the security certificate of this website. Ignore this message, and click **Continue to this website** to proceed.

You might also see a JSP error. You can ignore this error.

- ___ b. For **User ID**, enter: `odm`
- ___ c. For **Password**, enter: `odm`
- ___ d. Click **Log in**.

**Warning**

If you get a certificate warning, you can ignore it and continue to the web page.

You might also see a JSP error when the console opens. You can ignore this error.

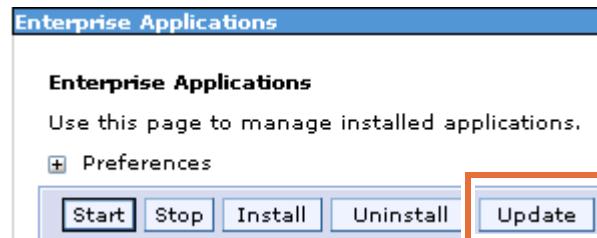
- 2. In the left pane, expand **Applications > Application Types**, and click **WebSphere enterprise applications**.



- 3. In the Enterprise Applications page, select the check box that corresponds to the **jrules-ssp-WAS85** application.



- 4. Click **Update** at the top of the Enterprise Applications page.



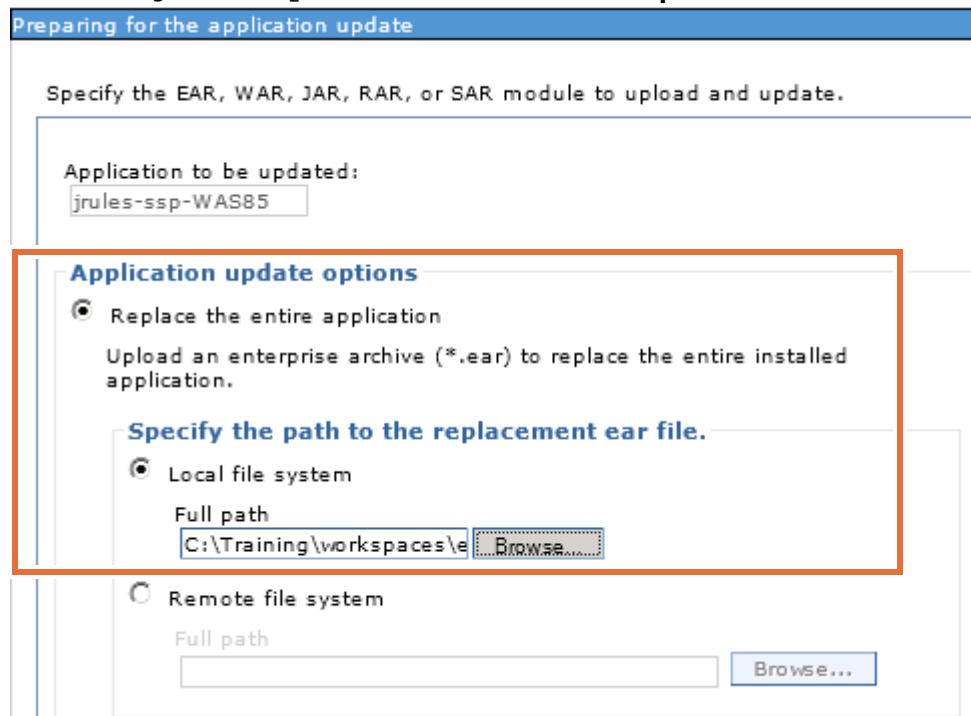
The “Preparing for the application update” page opens.

- 5. In the “Preparing for the application update” page, specify the path to the replacement EAR file:

- a. In the “Application update options” section, select **Replace the entire application**.
- b. Click **Browse** next to the **Local file system** field, and locate the SSP EAR file that you created, which is in your workspace folder (`enable_DVS`) in the `<LabfilesDir>\workspaces` directory.

`<LabfilesDir>\workspaces\enable_DVS\DVSP Project\applicationservers\`
`WebSphere85`

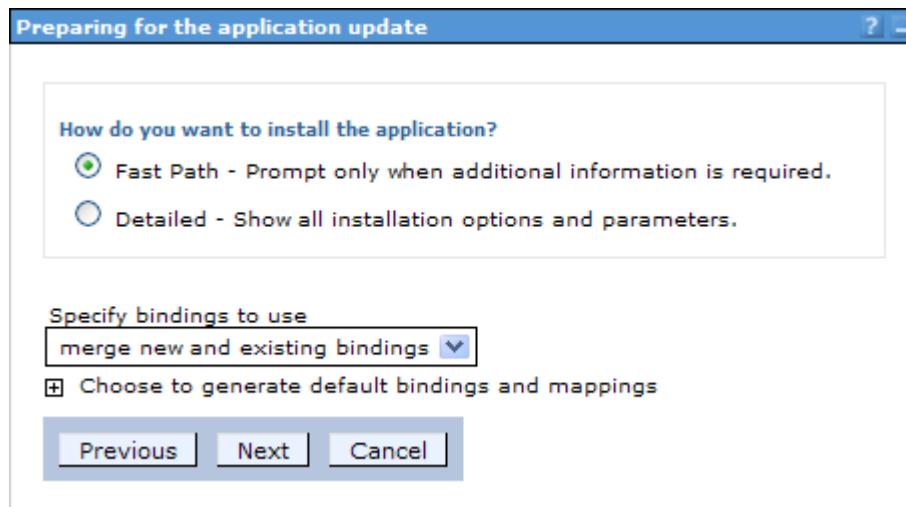
- ___ c. Select the `jrules-ssp-WAS85.ear` file and click **Open**.



- ___ d. Click **Next** at the bottom of the page.



- ___ e. In the “How do you want to install the application” section, keep the default **Fast Path** option selected, and click **Next**.



The “Select installation options” page opens.

- ___ f. Click **Next** to skip the “Select installation options” page.

The “Map modules to servers” page opens.

- ___ g. Click **Next** to skip the “Map modules to servers” page.

The Summary page opens.

- __ h. In the Summary page, click **Finish** to redeploy the SSP.

Redeploying the SSP might take a few minutes. When the application deployment finishes, you should see a message that prompts you to save changes.

Changes have been made to your local configuration. You can:

- [Save](#) directly to the master configuration.
- [Review](#) changes before saving or discarding.

Application jrules-ssp-WAS85 installed successfully.

To start the application, first save changes to the master configuration.

The application might not be immediately available while being started on all servers.

Changes have been made to your local configuration. You can:

- [Save](#) directly to the master configuration.
- [Review](#) changes before saving or discarding.

To work with installed applications, click the "Manage Applications" link.

[Manage Applications](#)

- __ i. Click **Save** to save the master configuration.
- __ j. Click **Logout** and close the administrative console.

Section 9. Testing the redeployed SSP

To make sure that the remote conditions are working properly, run remote tests from Rule Designer now.

- ___ 1. In Rule Designer, from the **Run** menu, click **Run Configurations**.
The Run Configurations window opens.
- ___ 2. Expand **DVS Excel File** and click **loan remote DVS** if it is not already open.
- ___ 3. On the **Excel File** tab and change the **HTML Report** field to:
`/loan-rules/data/report-remote-ssp.html`
- ___ 4. On the **DVS Configuration** tab:
 - ___ a. Select the **Remote execution** option.
 - ___ b. In the **DVS Project** field, make sure that **DVS Project** is selected.
 - ___ c. In the **Configuration** field, select **IBM WebSphere AS 8.5**.
 - ___ d. Clear the **Use XOM deployed to Rule Execution Server** check box.



Information

By clearing this option, you force the rule execution to use the XOM that was deployed with the SSP.

- ___ e. Click **Apply** to save the configuration.
- ___ f. Click **Run**.
- ___ 5. After execution finishes, refresh the Rule Explorer (F5) to see the newly created `report-remote-ssp.html` file in the `data` folder.
- ___ 6. To open the report, right-click `report-remote-ssp.html` and click **Open With > Web Browser**.

Summary

| | |
|--------------------------|--|
| Execution | Remote on <code>http://localhost:9080/testing</code> |
| Decimal Precision | 2 digits are used |
| Scenarios | 1 |
| Tests | 4 |
| Success Rate | 100% |
| Failures | 0 |
| Errors | 0 |

The report shows the correct results, and the **Execution** line indicates that the redeployed SSP was used for test execution:

Remote on `http://localhost:9080/testing`



Warning

Classes in a XOM that is packaged in the SSP take precedence over classes with the same name in a XOM that is deployed in the Rule Execution Server.

If you want to modify the XOM for your tests, you must either:

- Repackage the SSP with the modified XOM, and redeploy the SSP, or
- Deploy an SSP that does not embed the XOM, such as the SSP provided by default with the product installation, and then deploy the modified XOM to Rule Execution Server

End of exercise

Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 14: Enabling tests and simulations > 02-answer**.

The first part of the exercise looked at how to set up the testing environment for business users by preparing the BOM and creating a customized scenario file template. You saw how to set up Decision Validation Services locally in Rule Designer.

The second part of the exercise looked at how you can set up the test environment for business users to run tests with Decision Validation Services remotely from Decision Center.

After the redeployed SSP is accessible on a remote server, business users can run tests from Decision Center. Final preparation of testing and simulation functionality is done in Decision Center, either by you or by the business analysts. If the rule project is not already published to Decision Center, publishing would be the next step. Otherwise, the only remaining step is to make sure that the test suite is available for the business users.

While scenario file templates can be generated either in Rule Designer or Decision Center, customization to the BOM must be done in Rule Designer. For this reason, you are probably going to work regularly with the business users to support testing and simulation functionality.



Exercise 15. Managing deployment

What this exercise is about

This exercise teaches you how to deploy rules and XOMs for managed execution with Rule Execution Server.

What you should be able to do

After completing this exercise, you should be able to:

- Create a RuleApp
- Define a RuleApp and ruleset properties
- Create a Rule Execution Server Configuration project
- Create deployment configurations
- Use Rule Designer to deploy RuleApps to Rule Execution Server
- Deploy the XOM for its management in Rule Execution Server

Introduction

After finalizing the content of the rule project, you are ready to deploy it to the execution environment in Rule Execution Server.

In this exercise, you create a RuleApp to package your ruleset for deployment to Rule Execution Server. You learn how to deploy from Rule Designer and how to manage the XOM in Rule Execution Server.

This exercise is divided into these sections:

- Section 1, "Creating a RuleApp in Rule Designer"
- Section 2, "Creating deployment configurations"
- Section 3, "Deploying a RuleApp from Rule Designer"
- Section 4, "Deploying the managed XOM from Rule Designer"

Requirements

There are no specific requirements for this exercise.

Section 1. Creating a RuleApp in Rule Designer

In this part of the exercise, you create a RuleApp to package your ruleset for deployment to Rule Execution Server. You create a RuleApp project in Rule Designer, and you also learn how to add and remove RuleApp properties and ruleset properties in Rule Designer.

1.1. Setting up your environment for this exercise

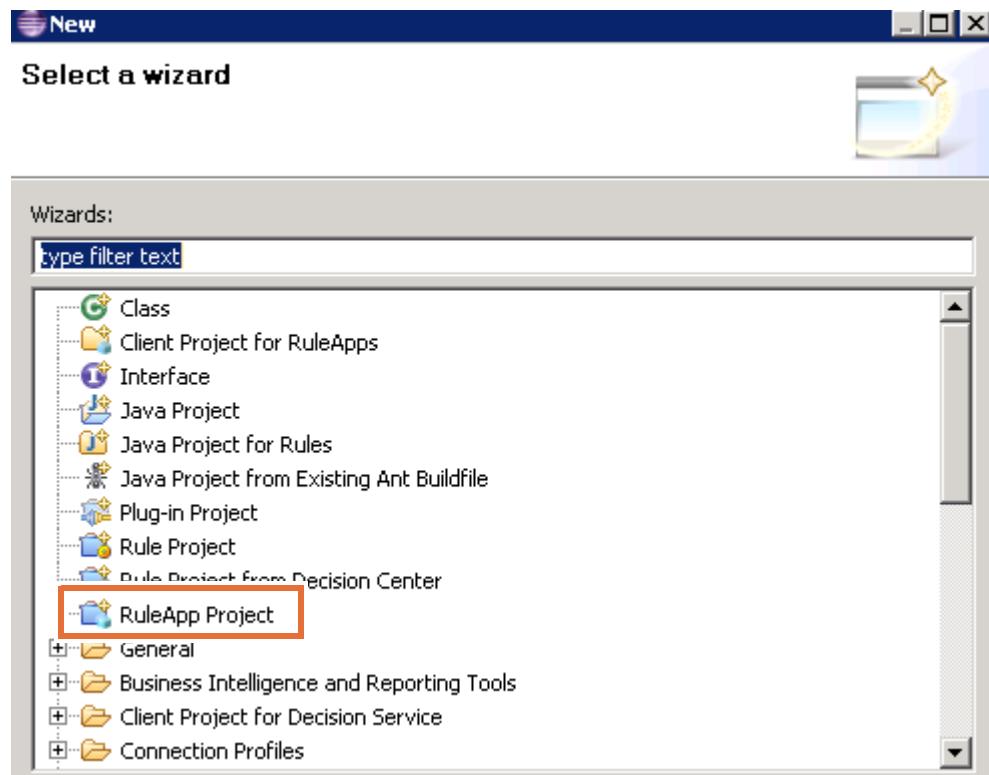
- ___ 1. If the sample server is not started, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.
Starting the server might take several minutes.
- ___ 2. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\deployment`
- ___ 3. Close the Welcome view and switch to the Samples Console perspective.
- ___ 4. Select **Rule Designer > Training > Ex 15: Managing deployment > 01-start > Import projects**.
- ___ 5. Close the Help view.

1.2. Creating a RuleApp

Create the RuleApp project that is required to deploy your ruleset to Rule Execution Server from Rule Designer.

- ___ 1. Create a RuleApp project.
 - ___ a. From the **File** menu, click **New > Other**.

- ___ b. Select the **RuleApp Project** wizard and click **Next**.



- ___ c. In the **Project name** field, enter `loan-RuleApp` and click **Next**.
 ___ d. On the **Rule Projects** tab of the Add Ruleset Archives page, click **Add**.
 The Ruleset Archives window opens.
 ___ e. In the Ruleset Archives window, select **loan-rules** and click **OK**.



Note

In this exercise, you use the **Ruleset Archives** window only to select the rule projects upon which your RuleApp is based.

However, in some cases, you can also use the **Ruleset Archives** tab to browse your environment and select the ruleset archive files that you want to add to your RuleApp.

- ___ f. Click **Finish**.
 ___ 2. In the Rule Explorer, expand `loan-RuleApp > runtime` to see the generated file.
 The `loanrules10.jar` file that is in the `runtime` directory is the ruleset archive that corresponds to the `loan-rules` project.

**Questions**

Where are the input ruleset parameters of the `loan-rules` project packaged: within the RuleApp archive or the ruleset archive?

**Hint**

You can explore the content of the ruleset archive (JAR file) by pressing F2 and changing the extension to `.zip` and explore it with the Windows Explore function. See Section 3.3, "Exploring the content of the ruleset archive," on page 11-17 in Exercise 11, "Queries and ruleset extraction".

If you change the extension of the ruleset archive file, make sure that you rename it back to `.jar` before you proceed.

Answer

The ruleset parameters are visible in the `descriptor.xml` file, which is located under the `META-INF` directory of the ruleset archive.

```
<?xml version="1.0" encoding="UTF-8"?>
<desc:ruleset-descriptor xmlns:sig="http://www.ilog.com/products/xml/schemas/sign
xmlns:desc="http://www.ilog.com/products/xml/schemas/ruleset_descriptor_1.0">
    <desc:ruleset-name>ruleset</desc:ruleset-name>
    <desc:isbusiness>true</desc:isbusiness>
    <desc:businessxmlserviceenable>false</desc:businessxmlserviceenable>
    <desc:generation-date>2013-05-08T11:01:53Z</desc:generation-date>
    - <desc:signature-declaration>
        - <sig:parameter>
            <sig:name>borrower</sig:name>
            <sig:direction>in</sig:direction>
            <sig:type xom-type="training_loan.Borrower" kind="native" bom-type="traini
            </sig:parameter>
        - <sig:parameter>
            <sig:name>loan</sig:name>
            <sig:direction>inout</sig:direction>
            <sig:type xom-type="training_loan.Loan" kind="native" bom-type="trainin
            </sig:parameter>
        - <sig:parameter>
            <sig:name>report</sig:name>
            <sig:direction>out</sig:direction>
            <sig:type xom-type="training_loan.Report" kind="native" bom-type="traini
            </sig:parameter>
```

1.3. Adding a RuleApp property to a RuleApp

In this section, you define the `ruleapp.interceptor.classname` RuleApp property for the RuleApp of the `loan-RuleApp` project.

- 1. Right-click the `archive.xml` file in the `loan-RuleApp` project and click **Open With > RuleApp Editor** to open the `loan-RuleApp` RuleApp.
- 2. On the **Overview** tab, in the RuleApp Properties section, click **New** to add a RuleApp property.

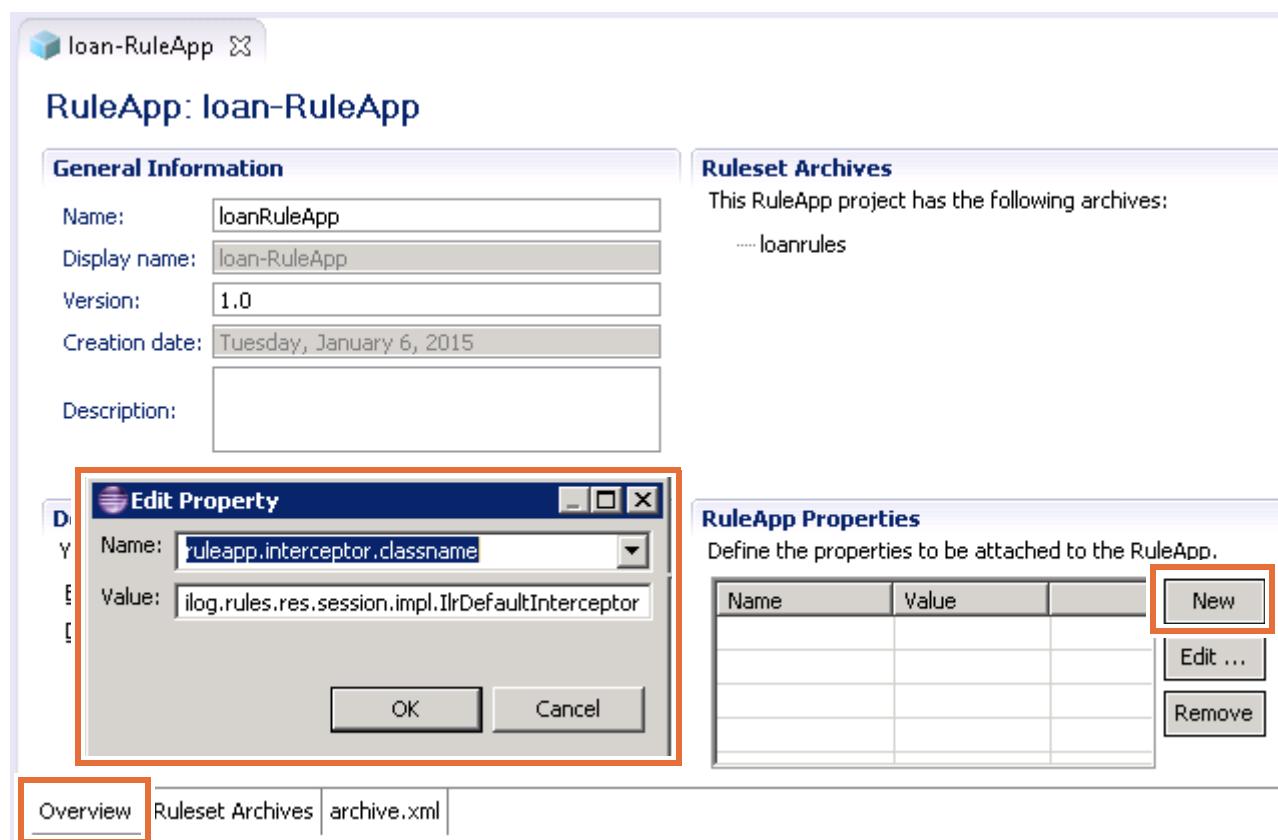
This step is for you to explore the RuleApp editor only. You delete the property after you see how it is used.

- a. In the list next to the **Name** field, select the `ruleapp.interceptor.classname` property.

The **Value** field is automatically set by default to:

`ilog.rules.res.session.impl.IlrDefaultInterceptor`

For this exercise, leave this default value unchanged.



- b. Click **OK**.
 - c. Save your work.
- 3. In the RuleApp editor, open the `archive.xml` tab for the `loan-RuleApp` to look at the added XML code.



Questions

Do you see where the new RuleApp property is added?

Answer

Look for this code:

```
<ruleapp-property>
  <ruleapp-property-name>ruleapp.interceptor.classname</ruleapp-property-name>
  <ruleapp-property-value>ilog.rules.res.session.impl.IlrDefaultInterceptor
  </ruleapp-property-value>
</ruleapp-property>
```

- ___ 4. Return to the **Overview** tab, and delete the `ruleapp.interceptor.classname` RuleApp property.
 - ___ a. Select the `ruleapp.interceptor.classname` RuleApp property in the RuleApp Properties section on the Overview tab.
 - ___ b. Click **Remove**.
 - ___ c. Save your work.

The XML definition of the RuleApp property is no longer visible in the XML code of the RuleApp on the **archive.xml** tab.

1.4. Adding a ruleset property to a ruleset

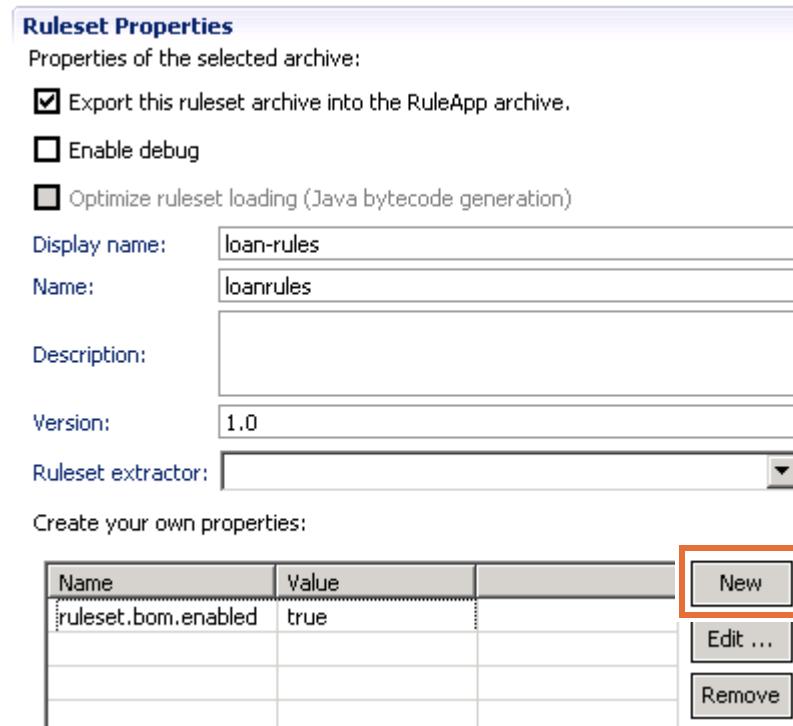
In this section, you add the `rulesetSEQUENTIAL.trace.enabled` ruleset property to the `loanrules` ruleset of the `loan-RuleApp` RuleApp.

- ___ 1. In the RuleApp editor, click the **Ruleset Archives** tab of the `loan-RuleApp` RuleApp to list all its rulesets.
- ___ 2. In the Ruleset Archives pane, select **loanrules**.

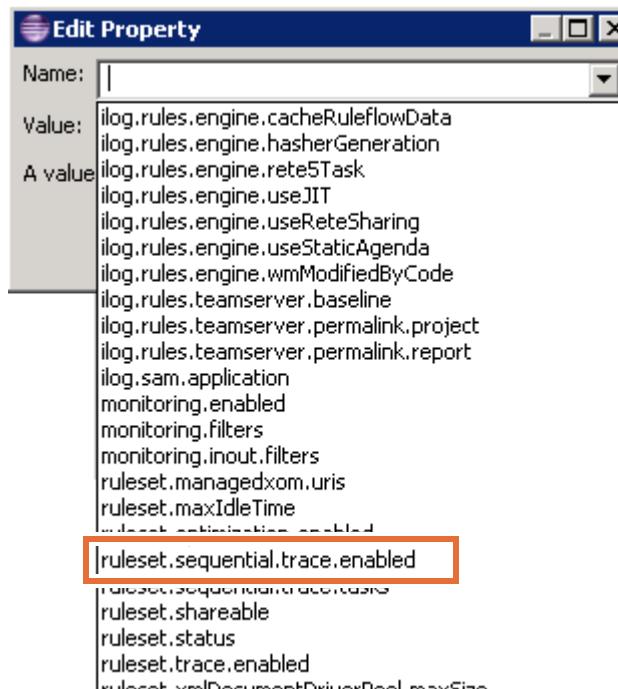
The Ruleset Properties section of the tab now contains data that relates to this ruleset.

3. In the Ruleset Properties section, add a property:

- a. Click **New**.



- b. In the Edit Property window, look at the names of possible ruleset properties in the list that is next to the **Name** field, and find `ruleset.sequential.trace.enabled`.



The names in the list are names of predefined ruleset properties. Each ruleset property has a specific role, which is described in the product documentation.

- ___ c. Select the `rulesetSEQUENTIALtraceenabled` ruleset property and set its **Value** to: true
 - ___ d. Click **OK** to close the Edit Property window.
 - ___ e. Save your work.
- ___ 4. Open the **archive.xml** tab to find the added XML code, and verify that the XML definition of the ruleset property is as follows:

```
<ruleset-property>
  <ruleset-property-name>rulesetSEQUENTIALtraceenabled</ruleset-property
  -name>
  <ruleset-property-value>true</ruleset-property-value>
</ruleset-property>
```

- ___ 5. Delete the `rulesetSEQUENTIALtraceenabled` property.

You do not need this property for the rest of this exercise.

- ___ a. Select `rulesetSEQUENTIALtraceenabled` in the Ruleset Properties section of the **Ruleset Archives** tab.
 - ___ b. Click **Remove**.
- ___ 6. Save your work.

The XML definition of the ruleset property is no longer visible in the XML code of the ruleset archive.



Warning

In Rule Designer, you cannot rename a RuleApp property or a ruleset property. If you want to change the name of a RuleApp property or of a ruleset property, you must first remove this property, and add one with the appropriate name.

- ___ 7. Close the RuleApp editor for `loan-RuleApp`.

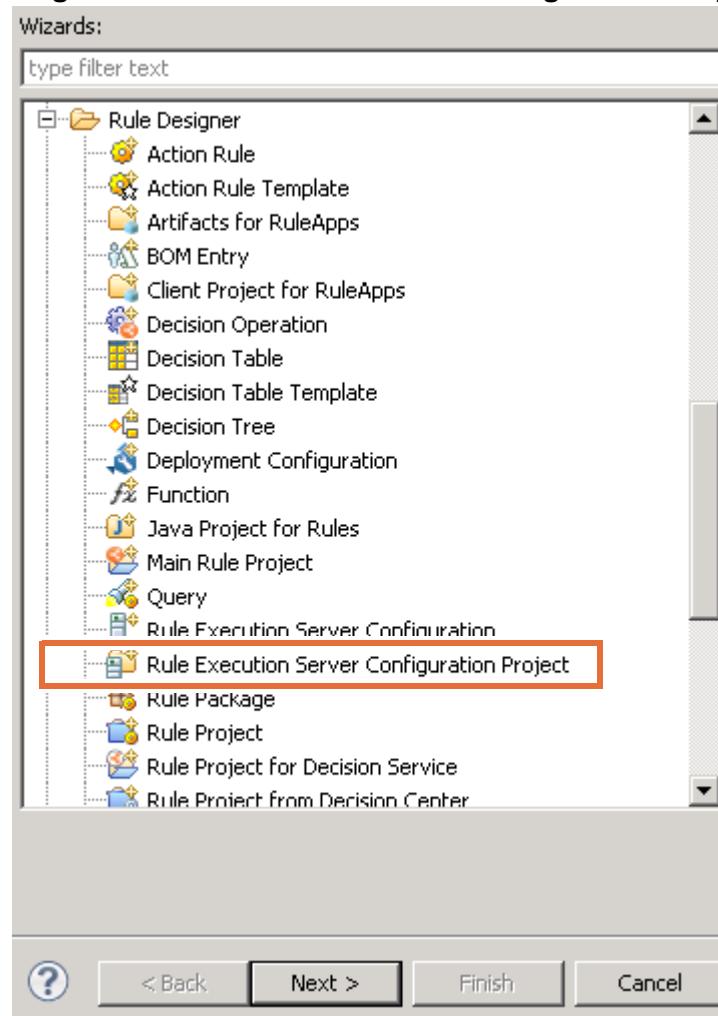
Section 2. Creating deployment configurations

In this section, you create Rule Execution Server configurations to manage deployments to Rule Execution Server.

2.1. Creating a Java SE-based Rule Execution Server configuration

In this section, you start by creating a Rule Execution Server configuration for a Java SE-based Rule Execution Server.

- 1. From the **File** menu, click **New > Other**.
- 2. Select **Rule Designer > Rule Execution Server Configuration Project**, and click **Next**.



- 3. In the New Rule Execution Server Configuration Project page, in the **Project name** field, enter: `loan-RESconfigs`
- 4. Make sure that **Use default location** is selected and click **Next**.
- 5. In the Configure Rule Execution Server page, clear **an application server** to disable all the listed application servers, and click **Next**.

- ___ 6. In the Configuration Name page, keep Java SE configuration as the default value of the **Name** field, and click **Next**.
- ___ 7. In the RuleApp Deployment page, make sure that the **to a file system** option is selected, keep `res_data` as the default value of the **Destination directory** field.
- ___ 8. Click **Finish**.
You created the `loan-RESconfigs` Rule Execution Server configuration project, which contains the new Java SE configuration Rule Execution Server configuration.
- ___ 9. In Rule Explorer, expand the `loan-RESconfigs` project to see the `Java SE configuration.esc` file.

2.2. Updating a Rule Execution Server configuration

In this section, you work with the Rule Execution Server Configuration editor to learn how to update a Rule Execution Server configuration.

- ___ 1. If the `Java SE configuration.esc` file is not open, double-click it to open it with the Rule Execution Server Configuration editor.
Alternatively, you can right-click the `Java SE configuration` Rule Execution Server configuration and click **Open With > Rule Execution Server Configuration Editor**.
- ___ 2. In the Rule Execution Server Configuration editor, explore the fields of the `Java SE configuration` Rule Execution Server configuration.
 - ___ a. Select **an application server** to explore the Server Information section in the Rule Execution Server configuration.
 - ___ b. Select **Rule Execution Server console deployed** to explore the Rule Execution Server console section in the Rule Execution Server configuration.



Questions

Can you figure out the purpose of the Server information section and of the Rule Execution Server console section?

Answer:

- With the Server information section in the Rule Execution Server configuration, you can specify the type of Rule Execution Server that you want to deploy to. You can also specify the directories where the server files are installed, and where to deploy the RuleApps to.
- With the Rule Execution Server console section, you can:
 - Specify the URL, name, and password that are required to connect Rule Designer to Rule Execution Server.

- Test the connection between Rule Designer and Rule Execution Server.
- Start the Rule Execution Server console in your web browser.

- c. Clear **Rule Execution Server console deployed** and **an application server**, and save your work.

2.3. Creating a WebSphere Application Server-based Rule Execution Server configuration

In this section, consider that Rule Execution Server is deployed to IBM WebSphere Application Server. You create a Rule Execution Server configuration that is required when you want to deploy the RuleApp for its embedded execution with such a deployment of Rule Execution Server.

- 1. From the **File** menu, click **New > Other**.
- 2. In the New wizard, expand **Rule Designer**, select **Rule Execution Server Configuration**, and click **Next**.

Rule Designer opens the “Add New Configuration to a Project” page, for you to select the Rule Execution Server configuration project as the location to persist the new Rule Execution Server configuration.



Note

This time, you already have the `loan-RESconfigs` Rule Execution Server configuration project available for selection. Do not click **Create a new configuration project** to create one.

- 3. In the “Add New Configuration to a Project” page, select the `loan-RESconfigs` Rule Execution Server configuration project, and click **Next**.
- 4. In the Configure Rule Execution Server page, make sure that **an application server** is selected, select **IBM WebSphere AS 8.5**, and click **Next**.
- 5. In the Server Configuration page, set the fields of the configuration.

- a. Keep the default value of the **Name** field as `IBM WebSphere AS 8.5`.

- b. In the **Installation directory** field, you can either click **Browse** and browse to the installation folder of WebSphere Application Server V8.5, or you can type the path directly:

`C:\Program Files\IBM\ODM87\WAS\AppServer`

If you are using another exercise environment, ask your instructor for the installation folder of WebSphere Application Server V8.5. This installation folder is called `ODM87\WAS\AppServer` and should be in the same folder as the installation folder of Operational Decision Manager V8.7.

- c. Leave the **Deployment directory** field empty.

The value of this field is not required if Rule Execution Server is hosted in WebSphere Application Server, as is the case in this course.

- ___ d. Make sure that the **URL** field is set to `http://localhost:9080/res`
Make sure that you use the correct port for your environment.
- ___ e. In the **Login** and **Password** fields, enter: `resAdmin`
- ___ f. Leave the **Client application URL** field empty.



Note

You do not have to click **Test Connection** now to verify that your settings are correct. You test this connection later in this exercise.

- ___ g. Click **Next**.

The RuleApp Deployment page opens.

- ___ 6. In the RuleApp Deployment page, select **to the Rule Execution Server console** and click **Finish**.

The `IBM WebSphere AS 8.5.esc` Rule Execution Server Configuration is created in the `loan-RESconfigs` Rule Execution Server Configuration project.

- ___ 7. Expand the `loan-RESconfigs` Rule Execution Server Configuration project and verify that it contains two Rule Execution Server configurations:
 - `IBM WebSphere AS 8.5.esc`
 - `Java SE configuration.esc`



Note

The Rule Execution Server name, the Rule Execution Server console URL, and the client application URL are stored in a `<servername>.esc` file in the Rule Execution Server configuration project directory. The login and password are stored in a `configurations.xml` file in the `.metadata\plugins\ilog.rules.studio.res\` directory in your workspace.

Section 3. Deploying a RuleApp from Rule Designer

In this section, you deploy a RuleApp for its managed execution in Rule Execution Server.

3.1. Deploying the RuleApp from the RuleApp itself

In this section, you deploy the RuleApp from the RuleApp itself.

- 1. In Rule Explorer, right-click the `loan-RuleApp` project, and click **RuleApp > Deploy**.

The “Deploy a RuleApp to Rule Execution Server” window opens. You can now deploy the RuleApp to one or more Rule Execution Servers.



Note

With this window, you can select several Rule Execution Server configurations to deploy the same RuleApp with the same version to several Rule Execution Servers in a single click. On this page, you can also create a Rule Execution Server configuration to deploy your RuleApp.

If you already prepared the Rule Execution Server configuration (as in this exercise), you do not have to specify configuration parameters again each time you deploy to the same Rule Execution Server from Rule Designer.

- 2. Click **Next** to keep the default deployment type.

In the “Deploy a RuleApp to Rule Execution Server” page, you can define the policy to update the RuleApp version, or ruleset version. In your enterprise, a version management policy must be defined and applied.



Troubleshooting

If you receive a Java version notification warning, click **Do not show this message again**, and click **OK**.

- 3. Select the **Java SE configuration** and clear the **Deploy XOM of rule projects and archives contained in the RuleApp** option.

By selecting the **Java SE configuration** Rule Execution Server configuration, you indicate to Rule Designer that you are going to deploy your RuleApp to the Rule Execution Server in Java SE.

You clear the **Deploy XOM of rule projects and archives contained in the RuleApp** option because XOM deployment is not supported for the Java SE configurations. You learn how to deploy the XOM in a later section of this exercise.

- 4. Click **Finish**, and if prompted to save, click **Yes**.

As specified in the Java SE configuration.esc configuration, the RuleApp is now deployed in the res_data folder under <InstallDir>\ODM.

In the Console view, the traces are as follows:

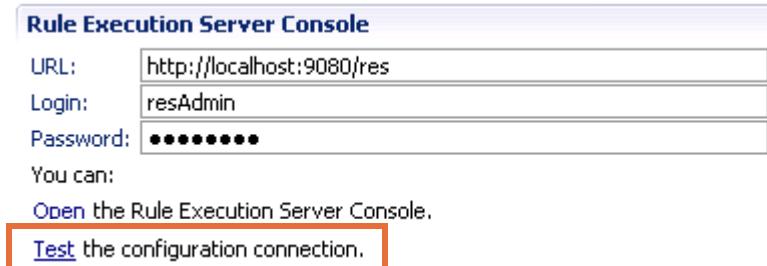
The "loan-RuleApp" RuleApp project was successfully deployed on the "Java SE configuration" configuration.

```
/loanRuleApp/1.0 -> /loanRuleApp/1.0: Element added  
/loanRuleApp/1.0/loanrules/1.0 -> /loanRuleApp/1.0/loanrules/1.0:  
Element added
```

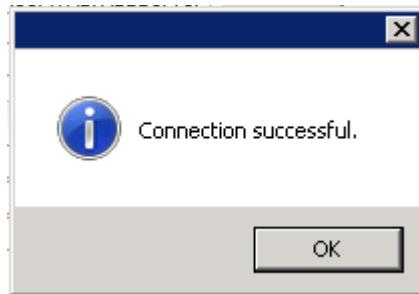
3.2. Deploying a RuleApp from the Rule Execution Server configuration

In this section, you deploy the RuleApp to Rule Execution Server deployed on IBM WebSphere Application Server 8.5, by using the associated Rule Execution Server configuration.

- ___ 1. In the loan-RESconfigs Rule Execution Server configuration project, double-click the IBM WebSphere AS 8.5.esc Rule Execution Server configuration to open it.
- ___ 2. In the **Rule Execution Server console** section, make sure that the **Login** and **Password** fields are set to resAdmin and save your work.
- ___ 3. Make sure that Rule Execution Server is started.
 - ___ a. If the sample server is not started, click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.
 - ___ b. To test the connection from Rule Designer to Rule Execution Server, click **Test the configuration connection** in the **Rule Execution Server console** section.



A window opens to indicate whether the connection is successful.



If the connection test fails, verify the values of the **URL**, **Login**, and **Password** fields, and use the received error message to correct the problem. Make sure that you use the correct port for your environment. Then, try again until the connection succeeds.

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

- ___ 4. Deploy the loan-RuleApp RuleApp by using the IBM WebSphere AS 8.5.esc Rule Execution Server configuration.
- ___ a. In the Deployment section of the IBM WebSphere AS 8.5.esc Rule Execution Server configuration, click **Deploy**.



The “Deploy RuleApp(s) on one Rule Execution Server” page opens. It is similar to the “Deploy a RuleApp to Rule Execution Server” page that you saw in the previous step. However, here, your selection applies to *all* the RuleApps that you select in the next page.

- ___ b. Click **Next** to keep the default deployment type.
- ___ c. In the “Deploy RuleApp(s) on one Rule Execution Server” page, select **loan-RuleApp** and clear the **Deploy XOM of rule projects and archives contained in the RuleApp** option.

For this part of the exercise, you do not deploy the XOM from the RuleApp. You later redeploy the RuleApp with the XOM to learn the differences.

- ___ d. Click **Finish**.
- ___ e. Look at the new traces in the Console view in Rule Designer:

The "loan-RuleApp" RuleApp project was successfully deployed on the "IBM WebSphere AS 8.5" configuration.

```
/loanRuleApp/1.0 -> /loanRuleApp/1.0: Element added
/loanRuleApp/1.0/loanrules/1.0 -> /loanRuleApp/1.0/loanrules/1.0: Element added
```



Questions

Can you figure out what the traces mean? Can you relate them to the ruleset path or to the managed XOM?

Answer

The hyphen (-) character is not an authorized character in the name of a RuleApp or of a ruleset.

- The RuleApp defined with the `loan-RuleApp` project is thus deployed as a `loanRuleApp` file to Rule Execution Server. The same principle applies to the name of the deployed ruleset, which is modified as `loanrules`.

- The `/loanRuleApp/1.0/loanrules/1.0` ruleset path corresponds to the ruleset path of the `loanrules` ruleset, version 1.0, within the RuleApp `loanRuleApp`, version 1.0.



Important

This `/loanRuleApp/1.0/loanrules/1.0` ruleset path, without the hyphen character, is what client applications must use to request the execution of the `loan-rules` ruleset.



Questions

In this section, you started Rule Execution Server before deployment. Why were you not required to start it when you deployed the RuleApp with the Java SE Rule Execution Server configuration?

Answer

You did not start Rule Execution Server because that configuration deployed the RuleApp to the `<InstallDir>\ODM\res_data` folder of your computer and only required access to that local folder.

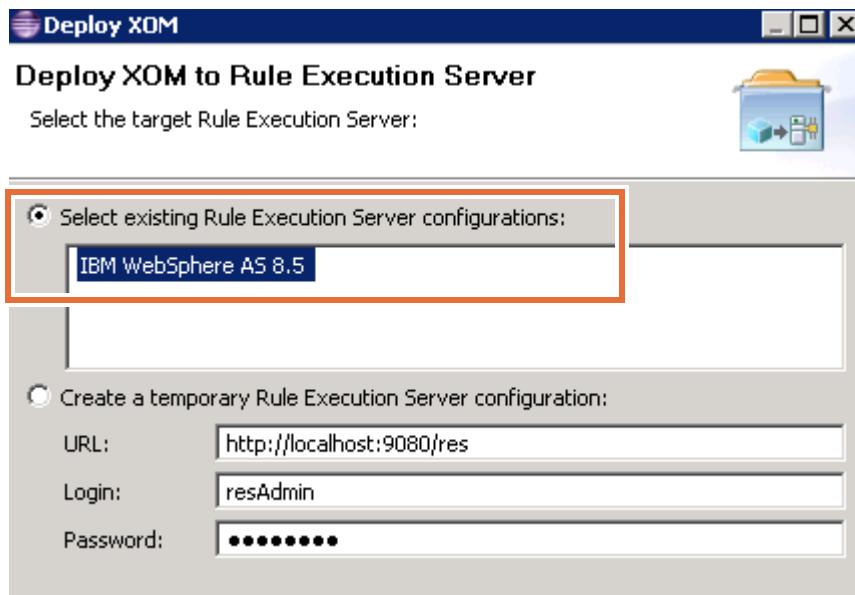
Section 4. Deploying the managed XOM from Rule Designer

In this part of the exercise, you deploy the XOM that the `loan-rules` project uses. Deploying the XOM to Rule Execution Server allows you to access and manage the XOM through the Rule Execution Server console.

4.1. Deploying the managed XOM from the rule project

In this section, you deploy the XOM directly from the rule project.

- ___ 1. Back in your Rule Designer workspace, in Rule Explorer, verify that the `resources` folder in the `loan-rules` project is empty.
Because you did not yet deploy the XOM, Rule Designer did not create the `deployment.xml` file in the `resources` folder.
- ___ 2. Deploy the XOM that the `loan-rules` project uses.
 - ___ a. Right-click the `loan-rules` project, and click **Rule Execution Server > Deploy XOM**.
 - ___ b. In the “Deploy XOM to Rule Execution Server” page, select the **Select existing Rule Execution Server configurations** option, and select **IBM WebSphere AS 8.5**.



- ___ c. Click **Finish**.
- ___ 3. Look at the traces in the Console view in Rule Designer:

Following XOM libraries have been deployed:
`resuri://loan-xom.zip/2.0`



Note

Depending on the order in which you completed the exercises, the version number for this deployment might be different.

- ___ 4. Expand **loan-rules > resources > META-INF** and double-click the `deployment.xml` file to open it.
- ___ 5. In the `deployment.xml` file, expand the nodes to find the definition of the URI for this XOM, and verify that its value matches the URI in the Console view:

`resuri://loan-xom.zip/2.0`

| Node | Content |
|---------------------------------|--|
| <code>?=? xml</code> | <code>version="1.0" encoding="UTF-8"</code> |
| <code>project-deployment</code> | |
| <code>xmlns</code> | <code>http://www.ibm.com/rules/ruleproject/deployment</code> |
| <code>xmlns:xsi</code> | <code>http://www.w3.org/2001/XMLSchema-instance</code> |
| <code>config</code> | |
| <code>serverURL</code> | <code>http://localhost:9080/res</code> |
| <code>vom</code> | |
| <code>uri</code> | |
| <code>value</code> | <code>resuri://loan-xom.zip/2.0</code> |



Note

Depending on the number of times you try deploying the XOM, you might see a different version number in your results. The main point of this exercise is to see that the version number in the `deployment.xml` file matches the result in the Console view, and to recognize how you can use version numbers to manage deployments.

- ___ 6. In the next steps, you change the XOM, redeploy it, and verify that the version of the deployed XOM is incremented.
 - ___ a. Expand **loan-xom > src > training_loan** and double-click the `Test.java` class to edit it by commenting the final line of the `main` method:


```
// System.out.println(r1);
```
 - ___ b. Save the `Test` class.

Your workspace should build automatically after saving if **Build Automatically** is selected on the **Project** menu.
- ___ 7. Redeploy the XOM from the `loan-rules` project.
 - ___ a. Right-click the `loan-rules` project and click **Rule Execution Server > Deploy XOM**.
 - ___ b. Select the **Select existing rule Execution Server configurations** option, and select **IBM WebSphere AS 8.5**.
 - ___ c. Click **Finish**.
 - ___ d. Look at the traces in the Console view in Rule Designer, and verify that the new traces show an incremented version number for this XOM.

Because the XOM changed, its checksum also changed, as did the version of the managed XOM in Rule Execution Server.

**Note**

Depending on whether the XOM was deployed earlier during exploration of the Rule Execution Server, you might have a different version number. The main point of this exercise is to recognize how you can use version numbers to manage deployments.

4.2. Deploying a RuleApp associated with a managed XOM

In this section, you deploy the RuleApp again. But this time, you also indicate the managed XOM that is associated with this RuleApp.

- 1. Undo the changes in the XOM that you made in Step 6 on page 15-18.
 - a. In the `loan-xom` project, open the `Test` class and uncomment the final line of the `main` method:

```
System.out.println(r1);
```

 - b. Save the `Test` class and close the file.
 - c. Wait for your workspace to rebuild.

**Important**

Because the XOM is changed, you must make sure that the `deployment.xml` file in the `loan-rules` project is up to date with the appropriate XOM version before you deploy the RuleApp.

- 2. Update the `deployment.xml` file in the `loan-rules` project.
 - a. Redeploy the XOM from the `loan-rules` project (as in Step 2 on page 15-17) by right-clicking the `loan-rules` project and clicking **Rule Execution Server > Deploy XOM**.
 - b. Look at the traces in the Console view in Rule Designer, and verify that they are now as follows:
 Following XOM libraries have been deployed:
`resuri://loan-xom.zip/2.0`
- 3. Reopen the `deployment.xml` file, and verify that the URI reverted to the previous version.

**Questions**

Although you deployed the XOM again, the URI indicates an earlier version. Why?

Answer

The version in the URI of the managed XOM depends on the checksum that Rule Execution Server computes for this XOM. Because you restored the XOM to its initial state, the deployed XOM has the same checksum as the earlier version of the XOM.

As a consequence, Rule Execution Server considers that you deployed that XOM version again; so that URI is used.

- ___ 4. Deploy the `loan-RuleApp` RuleApp by using the `IBM WebSphere AS 8.5.esc` Rule Execution Server configuration.
 - ___ a. In Rule Explorer, expand **loan-RESconfigs** and double-click **IBM WebSphere AS 8.5.esc**.
 - ___ b. In the `IBM WebSphere AS 8.5.esc` Rule Execution Server configuration, scroll down to the **Deployment** section and click **Deploy**.

Deployment

You can:

[Deploy](#) one or more RuleApp(s) to this Rule Execution Server.

The “Deploy RuleApp(s) on one Rule Execution Server” page opens.

- ___ c. Keep the default choice to increment the RuleApp major version, and click **Next**.
- ___ d. In the “Deploy RuleApp(s) on one Rule Execution Server” page, select **loan-RuleApp**.
- ___ e. Make sure that **Deploy XOM of rule projects and archives contained in the RuleApp** is selected.

Unlike the previous RuleApp deployment, you now select this option to deploy the RuleApp and make sure that it is associated with its XOM in Rule Execution Server.

- ___ f. Click **Finish**.
- ___ 5. Look at the new traces in the Console view in Rule Designer and notice that the major version of the RuleApp is incremented.

The "loan-RuleApp" RuleApp project was successfully deployed on the "IBM WebSphere AS 8.5" configuration.

```
/loanRuleApp/1.0 -> /loanRuleApp/2.0: Version changed and element added
/loanRuleApp/1.0/loanrules/1.0 -> /loanRuleApp/2.0/loanrules/1.0:
Version changed and element added
XOM Deployed : resuri://loan-xom.zip/2.0
```

**Note**

Depending on the number of times you tried deploying the RuleApp during this exercise, you might see a different version number in your results. The main point of this exercise is to recognize how you can use version numbers to manage deployments.

**Stop**

During Exercise 16, "Exploring the Rule Execution Server console", you learn how to monitor deployed RuleApps and XOMs in the Rule Execution Server by using the Rule Execution Server console.

End of exercise

Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 15: Managing deployment > 02-answer**.

The first part of the exercise looked at how to create a RuleApp to package a ruleset for deployment to Rule Execution Server. You also learned how to add and remove a RuleApp property and a ruleset property. The next parts of the exercise showed you how to deploy a RuleApp for managed execution in Rule Execution Server from both Rule Designer and Decision Center. Finally, you saw how to deploy a managed XOM from Rule Designer and from Decision Center.



Note

Exercise 16, "Exploring the Rule Execution Server console" depends on the deployments from this exercise.

If you want to redo this exercise after completing Exercise 16, "Exploring the Rule Execution Server console", make sure that you remove all the RuleApps that you deployed:

- Delete the contents of the `<InstallDir>\ODM\res_data` folder.
- Delete all the RuleApps in Rule Execution Server. You can use the Rule Execution Server console to delete RuleApps, as you learn later in this course.
- On the **Configure** tab in Decision Center, delete all the RuleApps for the `loan-rules` project.
- Delete the `loan-rules` project from Decision Center as described in Exercise 2, "Setting up rule projects".

Exercise 16.Exploring the Rule Execution Server console

What this exercise is about

This exercise teaches you how to work with the Rule Execution Server console.

What you should be able to do

After completing this exercise, you should be able to:

- Work with Rule Execution Server console tools
- Manage RuleApps and rulesets through the Rule Execution Server console

Introduction

After you deploy RuleApps and XOMs to Rule Execution Server, you can manage them through the Rule Execution Server console.

In this exercise, you explore the Rule Execution Server console environment and features.

The exercise is divided into these sections:

- Section 1, "Exploring the Rule Execution Server console"
- Section 2, "Exploring the deployed RuleApps"
- Section 3, "Working with deployed resources"
- Section 4, "Exploring the Diagnostics and Server Info tabs"
- Section 6, "Managing RuleApps and rulesets"
- Section 7, "Testing a deployed ruleset"
- Section 8, "Testing a ruleset with no associated managed XOM"

Requirements

You should complete Exercise 15, "Managing deployment" before starting this exercise.

Section 1. Exploring the Rule Execution Server console

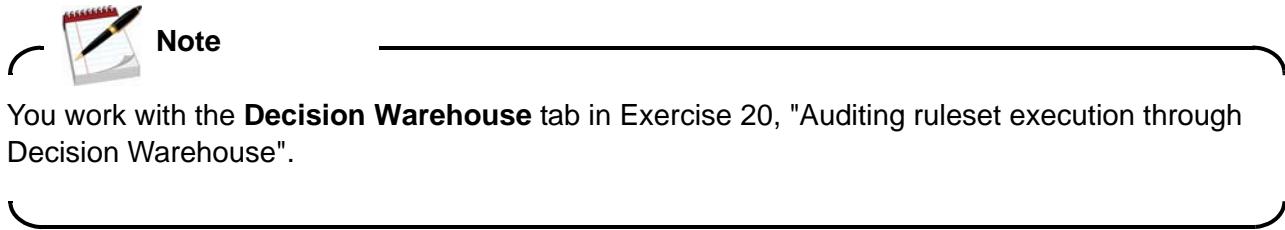
Make sure that you complete Exercise 15, "Managing deployment" before you start this exercise.

1.1. Setting up your environment for this exercise

- ___ 1. If the sample server is not started, start it now by double-clicking the shortcut on the desktop. You can also start the server by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.
Starting the sample server might take several minutes.
- ___ 2. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\explore_RES`
- ___ 3. Close the Welcome view and switch to the Samples Console perspective.
- ___ 4. Select **Rule Designer > Training > Ex 16: Exploring Rule Execution Server console > 01-start > Import projects**.
- ___ 5. Close the Help view.

1.2. Exploring the Rule Execution Server console pages

- ___ 1. Open Rule Execution Server console.
 - ___ a. Double-click the shortcut on the desktop or click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Rule Execution Server console**.
 - ___ b. In both the **User Name** and **Password** fields, enter: `resAdmin`
- ___ 2. Note the tabs: **Home**, **Explorer**, **Decision Warehouse**, **Diagnostics**, **Server Info**, and **REST API**.



- ___ 3. Click the **Explorer** tab.

The Navigator pane gives access to the elements that you can manage in the Rule Execution Server console:

- The **RuleApps** link gives access to the RuleApps View pane, where you can manage the deployed RuleApps.

- The **Resources** link gives access to the Resources View pane, where you can manage the managed resources.
- The **Libraries** link gives access to the Libraries View pane, where you can manage the managed libraries.
- The **Service Information** link gives access to the Transparent Decision Services View pane, where you can manage the created decision services.

By default, the Explorer page opens with the RuleApps View pane visible, and you can see the RuleApps that you deployed in Exercise 15, "Managing deployment".

Section 2. Exploring the deployed RuleApps

In this section, you explore the RuleApps deployed in Rule Execution Server, and relate them to what you did in Exercise 15, "Managing deployment".



Note

Depending on the number of deployments that you tried before doing this exercise, the version numbers that you see in the Rule Execution Server console might differ from the versions in these instructions.

To explore the RuleApp deployed from Rule Designer:

- 1. In the Navigator pane, expand **RuleApps**.
You are looking for the RuleApp that you deployed in Exercise 15, "Managing deployment".
- 2. In the Navigator pane, click **loanRuleApp/2.0** to open your most recently deployed RuleApp, and inspect the RuleApp View pane.

| Name | Value |
|----------------------|----------------------------------|
| Name | loanRuleApp |
| Version | 2.0 |
| Creation Date | Jan 6, 2015 1:56:47 AM GMT-08:00 |
| Display Name | loan-RuleApp |
| Description | |

The **Display Name** field indicates `loan-RuleApp`. This display name is the name for the RuleApp project in Rule Designer, from where you deployed this RuleApp.

- 3. Click the **loanrules** ruleset to open it, and explore its properties.

The Ruleset View pane opens, where you can see the `borrower`, `loan`, and `report` ruleset parameters that are defined for this ruleset.

- 4. Click **Show Managed URIs**.

You can see that this ruleset is associated with a deployed XOM. The URI corresponds to the XOM that you deployed in Exercise 15, "Managing deployment".

You work with this XOM in Section 3, "Working with deployed resources".

Section 3. Working with deployed resources

In this section, you explore the XOM resources that are deployed in Rule Execution Server, and relate them to what you did in Exercise 15, "Managing deployment".

- 1. In the Navigator pane, click **Resources**.
- 2. In the Resources View, click the **loan-xom.zip** link for the XOM that you deployed.
- 3. Click **Show references from Rulesets** to see the list of rulesets that reference this XOM.

You see the reference points to the `loanrules` rulesets from the last `loanRuleApp` version that you deployed from Rule Designer.

Section 4. Exploring the Diagnostics and Server Info tabs

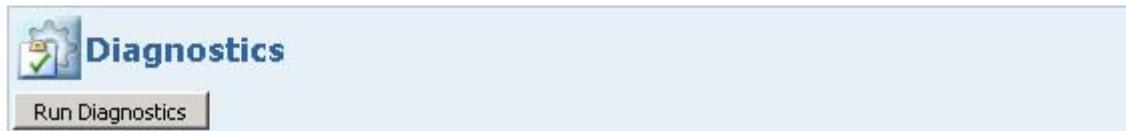
In this section, you explore the Diagnostics and Server Info pages.

__ 1. Explore the Diagnostics page.

___ a. Click the **Diagnostics** tab.



Diagnostics View



___ b. Click **Run Diagnostics**, and click **Expand All** to view the results.

__ 2. Explore the Server Info page.

___ a. Click the **Server Info** tab.



___ b. Notice the location of the log file, the logging level, and the server that is being used for the execution unit.

Section 5. Exploring the REST API tab

The REST API tool is a web application that you can use to test the management capabilities of REST resources and their associated parameters. REST resources include RuleApps, rulesets, XOM libraries, and XOM resources.

- 1. In the Rule Execution Server console, click the **REST API** tab to display the test tool.



- 2. Explore the **REST API** tab by clicking the tab for each resource and noting which methods are available:

- /ruleapps
- /rulesets
- /libraries
- /xoms

REST API tool

You can use this test tool to test the management REST API. To test the ruleset execution REST API, REST API WADL file: </res/apiauth/v1/DecisionServer.wadl>

/ruleapps **/rulesets** **/libraries** **/xoms**

GET /ruleapps
getRuleApps Returns all the RuleApps contained in the repository.

GET /ruleapps?count=true
getCountOfRuleApps Counts the number of elements in this list.

POST /ruleapps
deployRuleAppArchive Deploys a RuleApp archive in the repository, based on the merge request body.

POST /ruleapps
addRuleApp Adds a new RuleApp in the repository. The RuleApp representation is passed in the request body, the response body contains a specific error description and the HTTP status 202 is returned.



Note

You work with the REST API in Exercise 21, "Working with the REST API".

Section 6. Managing RuleApps and rulesets

In this section, you learn the basics of RuleApp and ruleset management with Rule Execution Server console. You can use the Rule Execution Server console to create, edit, and delete RuleApps and rulesets.

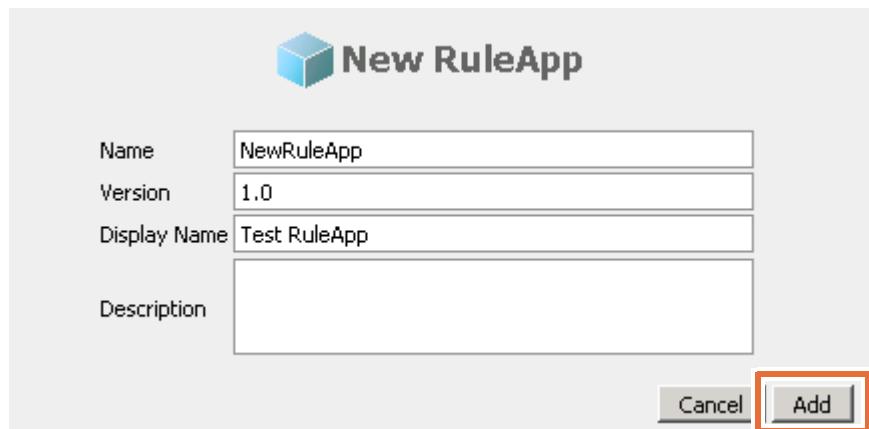
6.1. Creating a RuleApp

- __ 1. Click the **Explorer** tab.
- __ 2. In the RuleApps View, click **Add RuleApp**.

RuleApps View



- __ 3. In New RuleApp pane, leave the **Name** field set to: NewRuleApp
- __ 4. In the **Version** field, keep: 1.0
- __ 5. In the **Display Name** field, enter: Test RuleApp
- __ 6. Click **Add**.



The RuleApp View now shows the properties of the NewRuleApp/1.0 RuleApp.

6.2. Adding a ruleset to your RuleApp

- __ 1. Add a ruleset to the new NewRuleApp.
 - __ a. In the RuleApp View pane, click **Add Ruleset**.

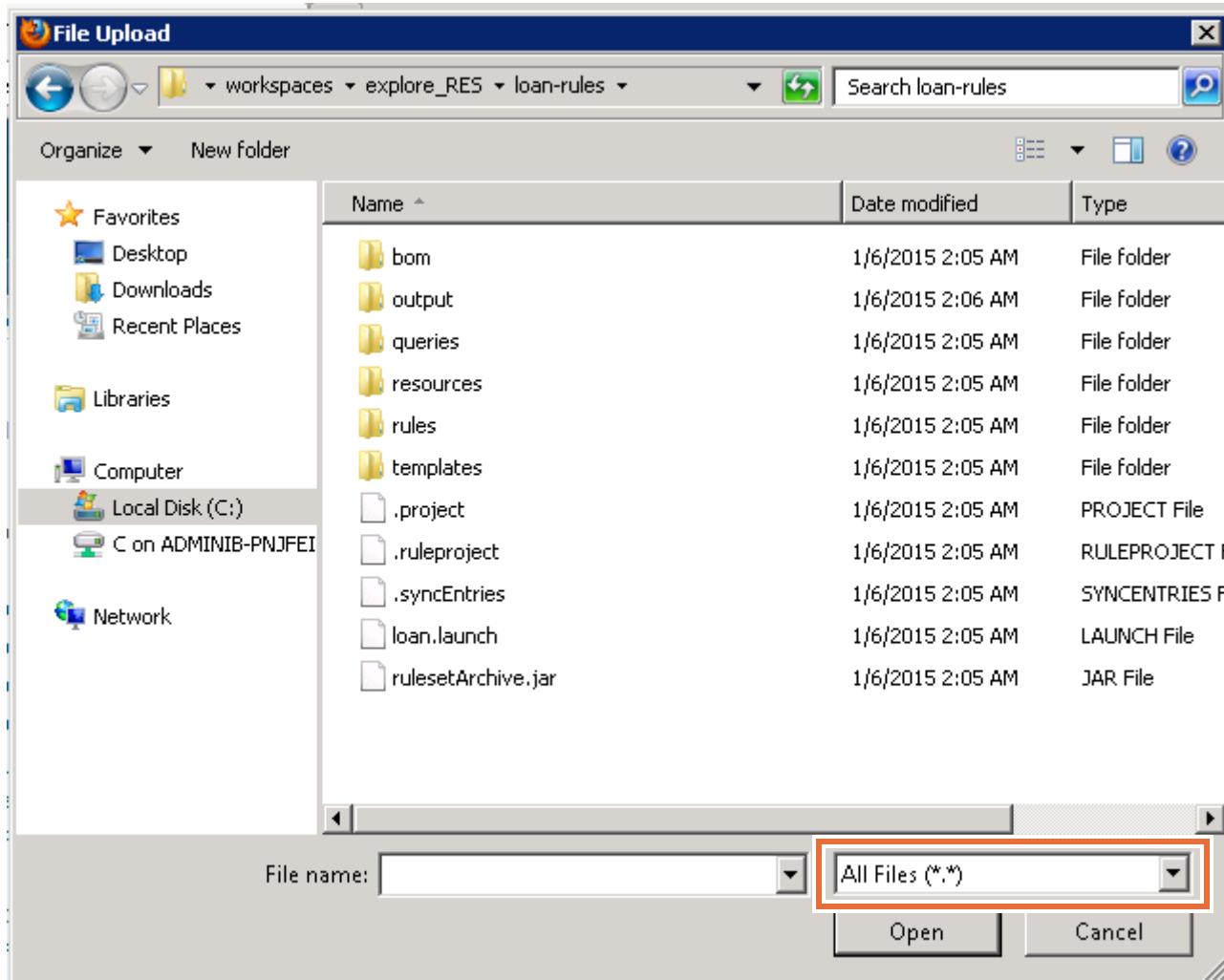
RuleApp View



The New Ruleset pane opens.

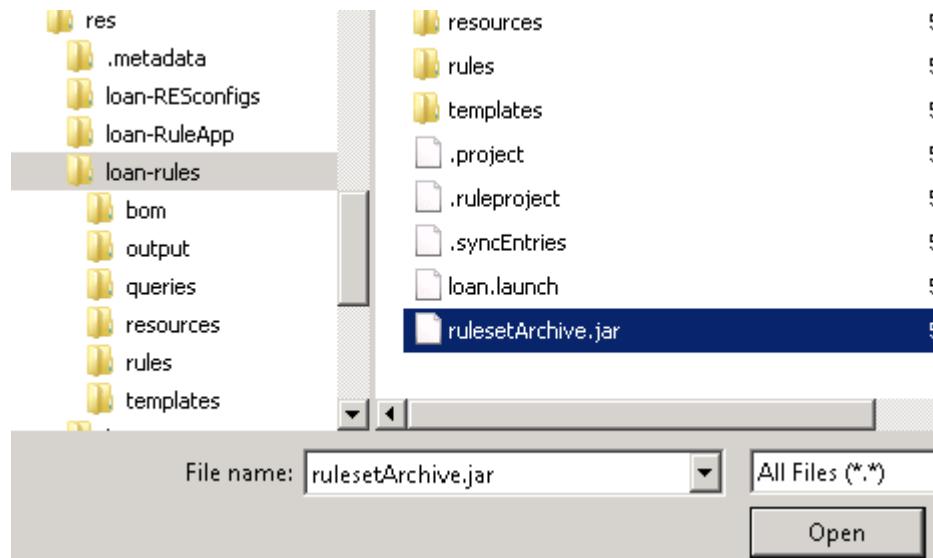
- __ b. In the **Name** field, keep: NewRuleset

- ___ c. In the **Version** field, keep: 1.0
- ___ d. In the **Display Name** field, enter: Test Ruleset
- ___ e. Click **Browse** next to **Local path of Ruleset Archive** and go to the following folder in your current Rule Designer workspace directory:
`<LabfilesDir>\workspaces\workspace_name\loan-rules\`
- ___ f. Make sure that **All Files (*.*)** is selected for the file type to search for.



The Ruleset Archive is a .jar file, and is not visible with the default file type setting.

- __ g. Select the `rulesetArchive.jar` file, and click **Open**.



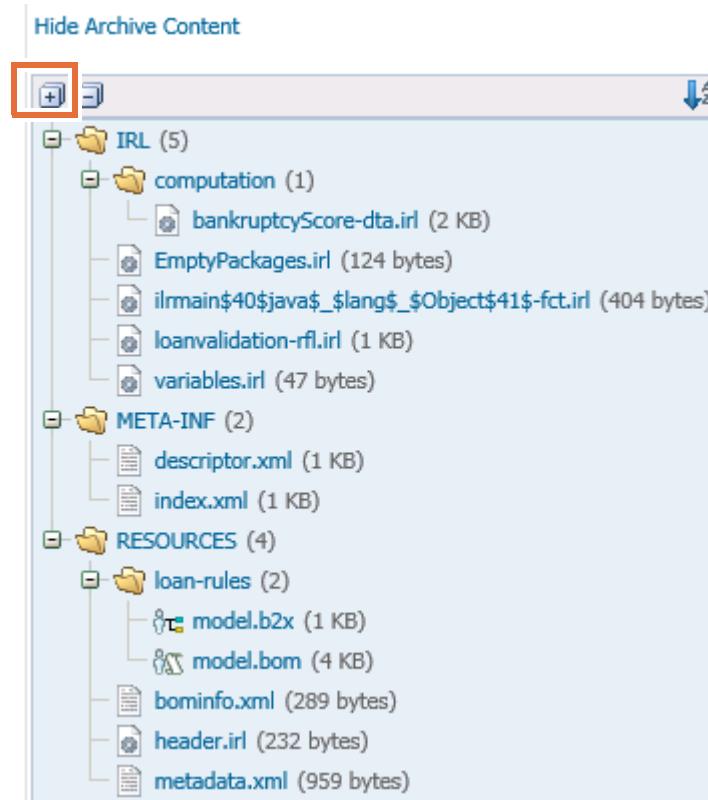
- __ h. In the New Ruleset page, leave the other fields unchanged, and click **Add**.

The `NewRuleApp/1.0/NewRuleset/1.0` ruleset is now created.

- __ 2. In the Ruleset View page for the `NewRuleApp/1.0/NewRuleset/1.0` ruleset, scroll down the page and click **Show Archive Content**.



- __ 3. Expand the folders to explore the content of the ruleset archive.



- The **IRL** folder contains the IRL version of all the rule artifacts in the ruleset.
- The **META-INF** folder contains the `descriptor.xml` and the `index.xml` files.
 - The `descriptor.xml` file contains the properties of the ruleset (name, ruleset signature, version).
 - The `index.xml` file contains the indexes of the ruleset, that is, the names of the different files that constitute this ruleset.
- The **RESOURCES** folder contains the definition of the business object model, and the definition of the rule artifact properties.

6.3. Adding a managed XOM to your ruleset

- 1. On the toolbar of the Ruleset View page, click **Add Managed URI**.

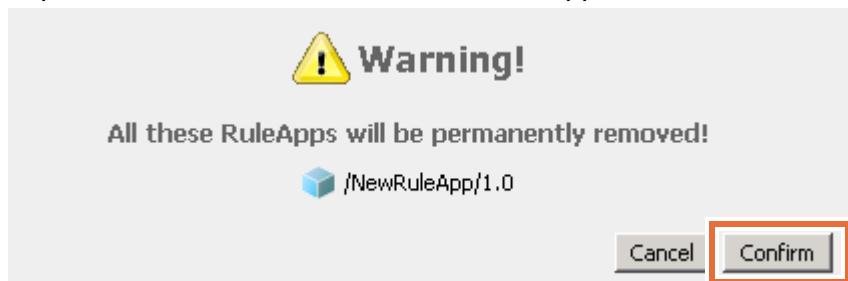


The Add Managed URI window opens with two tabs: **Latest version** and **Specific version**.

- 2. On the **Latest version** tab of the Add Managed URI window, select **loan-xom.zip** and click **Add**.
- 3. In the Ruleset View page, click **Show Managed URIs**.
 The URI `resuri://loan-xom.zip` is now visible in the list of managed XOMs associated with the ruleset.
- 4. View the references, as you did in Section 3, "Working with deployed resources".
 — a. In the Navigator page, click **Resources**.
 — b. In the Resource View, click the most recent **loan-xom.zip**.
 — c. Click **Show references from Rulesets** to see the rulesets that depend on this XOM.
 The reference points to your new test ruleset.

6.4. Deleting the RuleApp

- 1. In the Navigator pane, click **RuleApps** to open the list of RuleApps in the Ruleapps View.
- 2. In the list of RuleApps, select **NewRuleApp** and click **Remove**.
- 3. When prompted to confirm the removal of the RuleApp, click **Confirm**.



This RuleApp is not required for the rest of the course.

Section 7. Testing a deployed ruleset

In this section, you test the `loanrules` ruleset directly in Rule Execution Server console.



Hint

In this exercise, you must write some code. You can use the code snippets in the `<LabfilesDir>\code\test_in_res_console.txt` file, and copy and paste them in Rule Designer.

7.1. Testing the ruleset

- 1. In the Navigator pane, expand RuleApps, and select the most recently deployed `loanRuleApp` (version 2.0).
- 2. Click the `loanrules` ruleset.

The Ruleset View page opens for the selected `loanrules` ruleset.

- 3. On the toolbar, click **Test Ruleset**.

Ruleset View

The Test Ruleset View page opens and lists the input ruleset parameters of the `loanrules` ruleset: `borrower` and `loan`

| Direction | Name | Kind | XOM Type | Value |
|--------------------------|-----------------------|--------|-------------------------------------|--|
| <input type="checkbox"/> | <code>borrower</code> | native | <code>training_loan.Borrower</code> | <pre>import training_loan.*; Borrower borrower = null;</pre> |
| <input type="checkbox"/> | <code>loan</code> | native | <code>training_loan.Loan</code> | <pre>import training_loan.*; Loan loan = null;</pre> |

Execute Task
Task Name

To test your ruleset, you must provide values to these input ruleset parameters. You can either edit the associated Java code or provide a Java class that defines the ruleset parameter.

- 4. In the Test Ruleset View pane, define the values of the input ruleset parameters.
 - a. Click **Edit mode** next to the `borrower` ruleset parameter.

You can now write code in the Value section of the `borrower` ruleset parameter.

The  **Visualization mode** is also available for you to save your changes.

- b. In the **Value** field of the `borrower` ruleset parameter, enter the following code:

```
import training_loan.*;
import java.util.Calendar;

Borrower borrower = new Borrower("John", "Doe", DateUtil.makeDate(1968,
    Calendar.MAY, 12), "123456789");
borrower.setCreditScore(600);
borrower.setYearlyIncome(100000);
borrower.setLatestBankruptcy(DateUtil.makeDate(1990,Calendar.JANUARY,01),
    7, "Unemployment");
```



Hint

You can find this code snippet in the `<LabfilesDir>\code\test_in_res_console.txt` file.



Note

The code that you require to define an input ruleset parameter is like Java code. It is similar to the code that you can find in the `Test` class of the `loan-xom` project to define the `Borrower` `b1` object. However, in the Rule Execution Server console:

- The name of the instance must be the ruleset parameter name. In the present case, it must be: `Borrower borrower`
- You need to write only the lines of code that create and initialize the instance of the appropriate class, and have the required `import` statements precede them. You do not need to write a complete class or a complete method.
- You must write only the lines of code that create and initialize the instance of the appropriate class, and have them preceded with the required import statements

- c. Click  **Visualization mode** to save your change.
- d. Click  **Edit mode** next to the `loan` ruleset parameter.
- e. In the **Value** field of the `loan` input ruleset parameter, enter the following code:

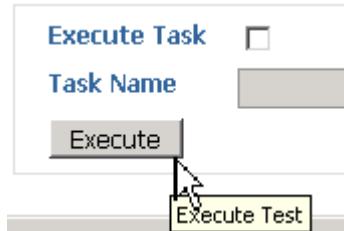
```
import training_loan.*;
import java.util.Calendar;
```

```
Loan loan = new Loan(DateUtil.makeDate(2014, Calendar.JUNE, 1), 60,
    100000, 0.70);
```

**Hint**

You can find this code snippet in the `<LabfilesDir>\code\test_in_res_console.txt` file.

- __ f. Click **Visualization mode** to save your change.
- __ 5. Click **Execute**.



The Execution result pane opens and displays the following items:

- The executed canonical ruleset path
- The execution duration
- The number of rules (total, fired, not fired)
- The rules that were fired (for example, validation.borrower.checkZipcode)
- The number of tasks (total, executed, not executed)
- The tasks that were executed, for example:

```
loanvalidation
loanvalidation>initResult
loanvalidation>validation
```
- The output traces, for example:

```
Borrower( firstName: John lastName: Doe born: May 12, 1968 SSN:
123-45-6789 income: 100000 credit score: 600 (bankruptcy on: Jan 1, 1990
for: Unemployment filed under chapter: 7))
Loan( amount: 100000 starting: Jun 1, 2014 duration: 60 month LTV: 70%)
Report( validData: false approved: false score: 0 message: The borrower's
Zip Code should have 5 digits
)
```

The Output Parameters pane is also open and shows the values of the output ruleset parameters.

Section 8. Testing a ruleset with no associated managed XOM

Testing a ruleset in Rule Execution Server console is possible only if this ruleset has an associated managed XOM. In this optional section, you test a ruleset with no associated managed XOM to learn the type of error message that you can have.



Stop

This section of the exercise uses the `loanrules/1.0` of the `loanRuleApp/1.0` that you deployed in "Deploying a RuleApp from the Rule Execution Server configuration" on page 15-14 in Exercise 15, "Managing deployment". This ruleset does not have an associated XOM.

If this ruleset is no longer available, you can use another ruleset that does not have an associated XOM. You might also use a ruleset that is associated with a managed XOM, and edit this ruleset to delete this association. In the Ruleset View for this ruleset, click **Show Managed URIs**, select all URIs, and click **Remove**.

Test the `loanrules` ruleset when it does not have an associated managed XOM in the Rule Execution Server console.

- 1. In the Rule Execution Server console, click the **Explorer** tab and expand **RuleApps** in the Navigator pane.
- 2. In the Navigator pane, in the list of RuleApps, expand the `loanRuleApp/1.0` and click its `loanrules` ruleset.

The Ruleset View pane opens for the selected `loanrules` ruleset.

To verify that this ruleset does not have an associated XOM, you can click **Show Managed URIs** and note the "0" which indicates that no XOM URIs are associated with the ruleset.



- 3. On the toolbar of the Ruleset View, click **Test Ruleset**.

The Test Ruleset View pane opens, with the default code to create `null` instances of the `Borrower` and of the `Loan` classes.

For this exercise, you do not need to define values for the input ruleset parameters in the Test Ruleset View pane.

- 4. In the Test Ruleset View pane, click **Execute**.

An error occurs and you see the error traces displayed:

- 5. Search for this line in the error traces:

```
Problem occurred loading translation : training_loan.Borrower : Cannot find
execution class "training_loan.Borrower" for translating business class
"training_loan.Borrower"
```

This problem occurs when Rule Execution Server console tries to find the XOM and load the translation for the “training_loan.Borrower” statement in the code that initializes the borrower ruleset parameter. Because no XOM is associated with this ruleset, translation fails.

- ___ 6. Sign out and close the Rule Execution Server console.

End of exercise

Exercise review and wrap-up

This exercise looked at how you can work with the Rule Execution Server console. You explored the different pages of the Rule Execution Server console. You also learned how to manage the RuleApps, the rulesets, and the managed XOMs.

Exercise 17. Executing rules with Rule Execution Server in Java SE

What this exercise is about

This exercise teaches you how to execute rules with Rule Execution Server in Java SE.

What you should be able to do

After completing this exercise, you should be able to:

- Create a client application that requests ruleset execution with Rule Execution Server in Java SE
- Describe and use the Rule Execution Server API that is required for ruleset execution with Rule Execution Server in Java SE

Introduction

You deployed the rule project as a RuleApp for its managed execution with Rule Execution Server.

In this exercise, you create a Java SE client application to run rules that are packaged as a RuleApp, and explore the required Rule Execution Server API.

This exercise is divided into these sections:

- Section 1, "Enabling ruleset decision traces"
- Section 2, "Creating the client project for a RuleApp"
- Section 3, "Exploring the generated client application project"
- Section 4, "Completing and running the client project for a RuleApp"

Requirements

There are no specific requirements for this exercise.

Section 1. Enabling ruleset decision traces

In this section, you enable the full decision traces for your ruleset by defining the `rulesetSEQUENTIALTRACE_ENABLED` ruleset property and setting its value to `true`. You define this ruleset property before you create the Java SE client application.

1.1. Setting up your environment for this exercise

- 1. In Rule Designer, you can continue with the workspace that you used for Exercise 16, "Exploring the Rule Execution Server console", or switch to a new workspace:
 - a. From the **File** menu, click **Switch Workspace > Other**.
 - b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\java_SE`
- 2. Close the Welcome view and switch to the Samples Console perspective.
- 3. Select **Rule Designer > Training > Ex 17: Executing rules in Java SE > 01-start > Import projects**.
- 4. Close the Help pane.

In the workspace, you have:

- The `loan-rules` project
- The `loan-RuleApp` project
- The `loan-RESconfigs` project, with the Rule Execution Server configurations that you require to deploy your RuleApp to Rule Execution Server
- The `loan-xom` project



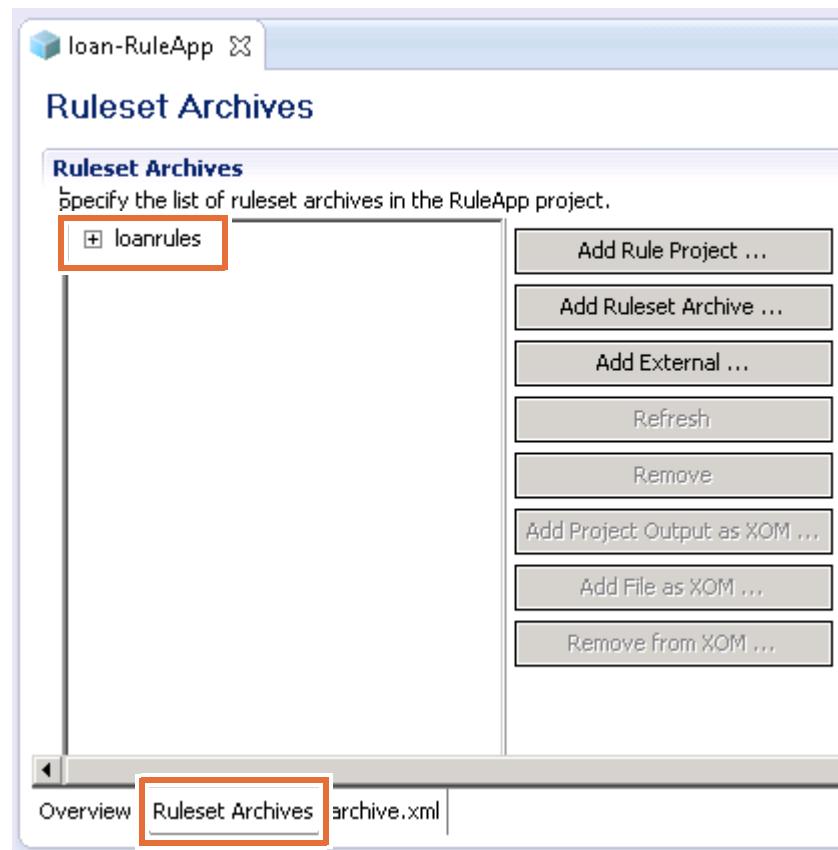
Hint

In this exercise, you must write some code. You can use the code snippets in the `<LabfilesDir>\code\execute_with_res.txt` file, and you can copy and paste them in Rule Designer.

1.2. Enabling the tracing ruleset property

- 1. In Rule Explorer, expand the `loan-RuleApp` project and double-click `archive.xml` to open the file in the RuleApp editor.

- ___ 2. Open the **Ruleset Archives** tab of the editor to see the ruleset archives defined in the RuleApp.



- ___ 3. In the Ruleset Archives section, select the `loanrules` ruleset.
- ___ 4. Enable decision monitoring on the ruleset by setting the `rulesetSEQUENTIALTRACEenabled` property.
- In the **Create your own properties** table of the **Ruleset Properties** section, click **New** to add a ruleset property.
 - In the Edit Property window, you can either type directly or select the `rulesetSEQUENTIALTRACEenabled` ruleset property from the list
 - Set **Value** to: `true`
 - Click **OK**.
- ___ 5. Save the `loan-RuleApp` (Ctrl+S).



Information

You use the `rulesetSEQUENTIALTRACEenabled` ruleset property because your ruleset contains rule tasks that use the sequential execution mode.

Section 2. Creating the client project for a RuleApp

In this section, after defining the ruleset property, you create the Java SE client project for your RuleApp with Rule Designer.

- ___ 1. Right-click anywhere in Rule Explorer, click **New > Project > Client Project for RuleApps**, and click **Next**.
- ___ 2. In the Client Project for RuleApps page, expand **Rule Execution Server** to make sure the **Plain Old Java Object (POJO)** template is selected, and click **Next**.

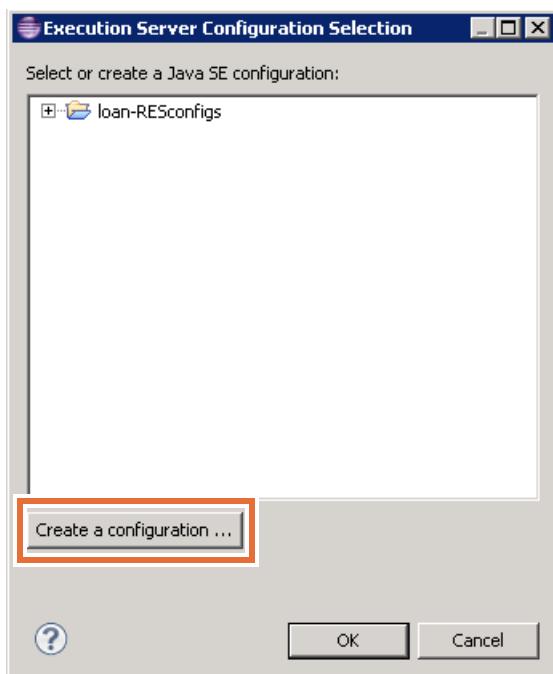


Information

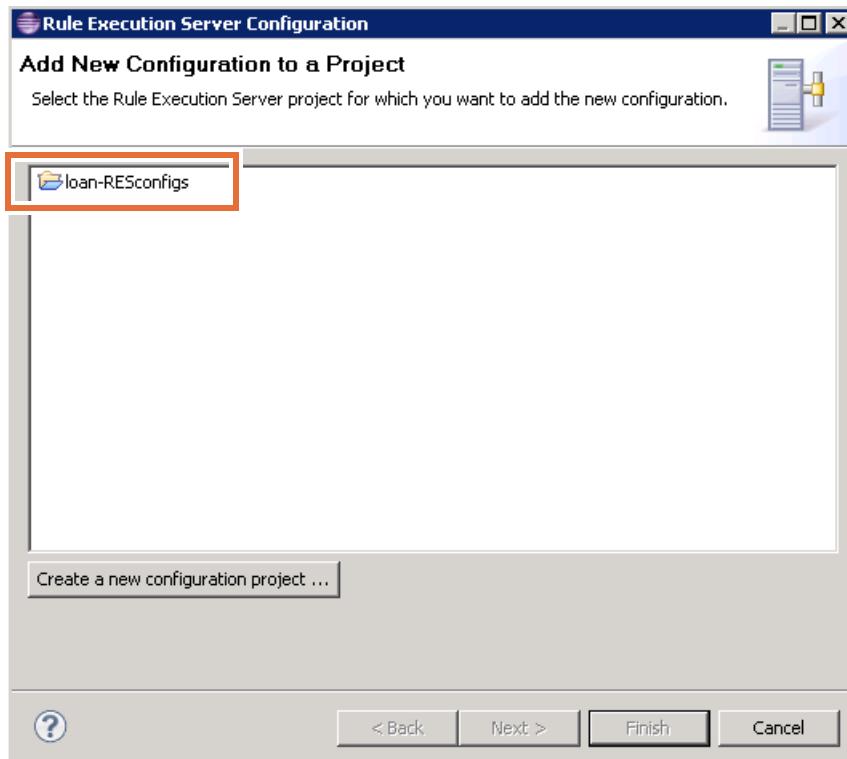
The client application that you are about to create uses the *Java SE* rule session, which in turn is based on POJO (plain old Java objects). This client application is not based on the POJO rule session.

- ___ 3. In the Client Project Name page, in the **Project name** field, enter `loan-resclient` and click **Next**.
- ___ 4. In the RuleApp Project page, select **loan-RuleApp** and click **Next**.
- ___ 5. In the Ruleset Archive page, make sure that **loan-rules** is selected as the ruleset archive, and click **Next**.
- ___ 6. Click **Next** to skip the page for initializing the ruleset parameters.
You initialize the ruleset parameters later in this exercise.
- ___ 7. Click **Next** to skip the page for defining an alternative ruleflow.
- ___ 8. Define the POJO properties.
 - ___ a. Keep the proposed values for the **Interface name** field and for the **Implementation name** field.
 - ___ b. Make sure that **Generate an interface for the POJO** is selected.

- ___ c. For the **Rule Execution Server configuration** field, click **Browse** and click **Create a configuration**.



- ___ d. Select loan-RESconfigs and click **Next**.



- ___ e. Keep the default name for the Java SE configuration and click **Next**.
- ___ f. With "to a file system" selected, leave the default path for the **Destination directory** field, and click **Finish**.
- ___ g. Click **OK**.

- ___ 9. Click **Next**.
- ___ 10. In the Execution Options page, specify the trace filters.
 - ___ a. Select **Generate a trace with the following data** to generate traces for the execution.
 - ___ b. Keep the default selection of traces.
- ___ 11. Click **Finish**.
Rule Designer generates the client application, which is now visible in the Rule Explorer.
- ___ 12. Close the **loan-RuleApp** editor.

Section 3. Exploring the generated client application project

In this section, you explore the generated client application.

- 1. In the Rule Explorer, expand the newly created `loan-resclient` project.
- 2. Expand the `res_data` directory of the client project and verify that `loanRuleApp` was generated in this directory.
- 3. Expand `loan-resclient > src > loanRuleApp` and double-click `LoanresClientRunnerImpl.java` to open the class file and explore the generated Java code.

In particular, notice the `executeloanrules` method.



Information

The following list describes some (not all) of the lines in the `src/LoanresClientRunnerImpl` class file.

- The following line creates the factory that is required to obtain a rule session:


```
IlrJ2SESessionFactory factory = getFactory();
```
- The following line creates the request that is sent to Rule Execution Server to execute the ruleset:


```
IlrSessionRequest request = factory.createRequest();
```
- The following line sets the ruleset that is executed. The ruleset is defined with a non-canonical ruleset path to ensure that its most recent version is executed:


```
request.setRulesetPath(IlrPath.parsePath( "/loanRuleApp/loanrules" ));
```
- The following line sets the data that is specific to the calling application. This data can be associated to a rule session and retrieved during several stages of the rule execution workflow.


```
request.setUserData( "loanRuleApp.LoanresClientRunnerImpl.executeloanrules" );
```
- The following line enables the decision traces:


```
request.setTraceEnabled(true);
```

Rule Designer generates an `IlrSessionRequest.setTraceEnabled(true)` call in the generated client application code because you selected **Generate a trace with the following data**.
- The lines that start with `request.getTraceFilter().setInfo` set the filters on the decision traces.



Example

```
request.getTraceFilter().setInfoExecutionDate(true);
```

The lines that set filters depend on which trace options you selected during the client application creation.

- The following lines define the input ruleset parameters. Notice how the ruleset parameters are defined by using their names:

```
request.setInputParameter("borrower", borrower);
request.setInputParameter("loan", loan);
```

- The following line defines the response. This response is set when the client application requests the ruleset execution (see next).

```
IlrSessionResponse response;
```

- The following line defines the stateless session that is required to execute the ruleset:

```
IlrStatelessSession session = factory.createStatelessSession();
```

- Rule Designer creates a series of intermediate classes, as helpers, such as the ExecutionHook class. These classes are not detailed in this course. For more information, see the product documentation.

- The following line asks for the ruleset execution with Rule Execution Server:

```
response = session.execute(request);
```

- The following lines retrieve the output ruleset parameters from the response:

```
loanrulesresult.loan = (training_loan.Loan)
    response.getOutputParameters().get("loan");
loanrulesresult.report = (training_loan.Report)
    response.getOutputParameters().get("report");
```

- The following line retrieves the execution output from the response:

```
loanrulesresult.outputString =
    response.getRulesetExecutionOutput();
```

- The following line retrieves the Decision ID from the response:

```
loanrulesresult.executionId = response.getExecutionId();
```

- The lines that contain `response.getRulesetExecutionTrace().getExecution` retrieve the Decision ID from the response, for example:

```
loanrulesresult.setInfoExecutionDate(
    response.getRulesetExecutionTrace().getExecutionDate());
```

In many cases, adding the `IlrSessionRequest.setTraceEnabled(true)` call is enough to have ruleset decision traces enabled. However, if the ruleset contains rule tasks that do not use the RetePlus execution mode, you must also define the

`ruleset.sequential.trace.enabled` property and set its value to `true` to get the full ruleset decision traces.

**Note**

If you define the `ruleset.sequential.trace.enabled` ruleset property only after you deploy the ruleset as a RuleApp with the wizard, you must redeploy the RuleApp to get the full ruleset decision traces.

Section 4. Completing and running the client project for a RuleApp

Rule Designer has no idea of the values that you require for the input ruleset parameters or of the objects in the working memory that your client application requires. In this section, you complete the client application by adding the required code to define these objects, as follows:

- ___ 1. Expand the **loan-resclient > src > loanRuleApp** directory, and double-click the **Main.java** file to open it.
- ___ 2. In the **Main.java** file, go to the **main(String[] args)** method, and add the required code to initialize the **borrower** and **loan** objects.



Hint

You can find this code snippet in the **<LabfilesDir>\code\execute_with_res.txt** file in the section that starts with:

--- Java SE

- ___ a. Add these **import** statements to the **loanRuleApp** package just before the **Main** class:

```
import java.util.Calendar;
import training_loan.Borrower;
import training_loan.DateUtil;
import training_loan.Loan;

package loanRuleApp;

import java.util.Calendar;
import training_loan.Borrower;
import training_loan.DateUtil;
import training_loan.Loan;

public class Main {
```

- ___ b. Replace the following line:

```
training_loan.Borrower borrowerloanrules = null;
```

With the following lines:

```
training_loan.Borrower borrowerloanrules = new Borrower("John",
    "Doe", DateUtil.makeDate(1968, Calendar.MAY, 12), "123456789");
borrowerloanrules.setZipCode("91320");
borrowerloanrules.setCreditScore(600);
borrowerloanrules.setYearlyIncome(100000);
```

- ___ c. Replace the following line:

```
training_loan.Loan loanloanrules = null;
```

With the following lines:

```
training_loan.Loan loanloanrules = new Loan(DateUtil.makeDate(2014,  
    Calendar.JUNE, 1), 72, 100000, 0.7d);
```

- __ d. Save your work.



Note

You must use the `Borrower` class setter methods to set the attributes of the `borrowerloanrules` object because you do not have direct access to the `Borrower` class fields. For example, write `setZipCode("91320")` instead of `zipCode = "91320"`.

- __ 3. Switch to the Java perspective, expand **loan-resclient > src > loanRuleApp**, right-click **Main.java**, and click **Run As > Java Application with Rules**.

You see a series of traces in the Console view, including lines that show the loan was approved:

...

Congratulations! Your loan has been approved

...

You can ignore the other trace messages.

End of exercise

Exercise review and wrap-up

This exercise looked at how to create a Java SE client application and package it as a RuleApp to run rules. You also explored the required Rule Execution Server API.

Exercise 18. Executing rules with Rule Execution Server in Java EE

What this exercise is about

This exercise teaches you how to execute rules with Rule Execution Server in Java EE.

What you should be able to do

After completing this exercise, you should be able to:

- Use the basic Rule Execution Server API to request ruleset execution with Rule Execution Server in Java EE

Introduction

You deployed the rule project as a RuleApp for managed execution with Rule Execution Server deployed in WebSphere Application Server. Next, you create a web client application that uses your ruleset.

In this exercise, you create the web client application that is based on the API for Rule Execution Server in Java EE. After you finish, you can compare your work with the Java client application that you created in Exercise 17, "Executing rules with Rule Execution Server in Java SE".

This exercise includes these sections:

- Section 1, "Building the web application"
- Section 2, "Deploying the web application to WebSphere Application Server"
- Section 3, "Executing the web application with the RuleApp deployed"

Requirements

This exercise uses the `loanRuleApp`, which must be deployed to Rule Execution Server. You deployed this RuleApp in Exercise 15, "Managing deployment".



Stop

If you deleted the `loanRuleApp` RuleApp from Rule Execution Server for any reason, you must redeploy it, along with its managed XOM, to Rule Execution Server.

You can use your workspace from Exercise 15, "Managing deployment" which should include these projects:

- The `loan-rules` project
- The `loanRuleApp` that is defined in the `loan-RuleApp` project
- The `IBM WebSphere AS 8.5.esc` configuration that is defined in the `loan-RESconfigs` project

To redeploy, follow the instructions in "Deploying a RuleApp associated with a managed XOM" on page 15-19.

Section 1. Building the web application



Requirements

Business analysts ask you to create a web application that requests the execution of the `loanrules` ruleset that is packaged in the `loanRuleApp` RuleApp. They give you the `loan-webapp` project, which is a skeleton of a client web application that requests the execution of business rules execution from a JSP page. You must complete, deploy, and execute this client application.

In this section, you examine and finalize the `loan-webapp` project.

1.1. Setting up your environment for this exercise

- 1. If the sample server is not started, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.
Starting the server might take several minutes.
- 2. In Rule Designer, switch to a new workspace:
 - a. From the **File** menu, click **Switch Workspace > Other**.
 - b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\java_EE`
- 3. Close the Welcome view and switch to the Samples Console perspective.
- 4. Select **Rule Designer > Training > Ex 18: Executing rules in Java EE > 01-start > Import projects**.
- 5. Close the Help pane.

You have a series of projects available in the Rule Explorer:

- `loan-webapp`

This project is a client web application with a minimal graphical user interface, from which you can request business rule execution.

- `loan-xom`
- `loan-rules`
- `loan-RuleApp`
- `loan-RESconfigs`



Hint

In this exercise, you must write some code. You can use the code snippets in the `<LabfilesDir>\code\execute_with_res.txt` file, and copy and paste them in Rule Designer.

1.2. Finalizing the client application code to call your rules for execution

As you learned during the lectures, the Rule Execution Server API follows a uniform pattern. The only difference between Java SE and Java EE is in the rule session factory that is used. In a Java EE-based client application, the `IlrPOJOSessionFactory` must be used instead of the `IlrJ2SESessionFactory`, which is used in a Java SE context.

- 1. In the Rule Explorer, expand **loan-webapp > src > WebContent** and open the `executerules.jsp` file with the JSP editor in Rule Designer.



Hint

To edit the JSP page, either double-click it, or right-click it and click **Open With > JSP editor**.

- 2. In the `executerules.jsp` file, search for the lines with dots ... and replace these dots with the appropriate calls.

- a. Create an `IlrPOJOSessionFactory` factory by replacing dots ... after the following line:

```
// Create session factory
```

With:

```
factory = new IlrPOJOSessionFactory();
```

- b. Create a stateless rule session from the factory by replacing dots ... after the following line:

```
// Create rule session
```

With:

```
ruleSession = factory.createStatelessSession();
```



Questions

Find the following line:

```
IlrPath path = new IlrPath("...", "...");
```

How should you replace the dots to set the `IlrPath` correctly?

Answer

The `IlrPath` class defines the ruleset path that is used to execute the ruleset. You might use a canonical ruleset path, that is, use the RuleApp version and the ruleset version. However, in most cases and in this exercise, you use the non-canonical ruleset path to make sure that Rule

Execution Server uses the most recently deployed versions of the RuleApp and ruleset at run time.

- ___ c. Define the non-canonical ruleset path by replacing the following line:

```
IlrPath path = new IlrPath("...", "...");
```

With:

```
IlrPath path = new IlrPath("loanRuleApp", "loanrules");
```



Important

The rule project is called `loan-rules`, and its ruleset is deployed by using a RuleApp project that is called `loan-RuleApp`. However, the deployed RuleApp is called `loanRuleApp` (no hyphen), and its ruleset is called `loanrules` (no hyphen). To define the ruleset path, you must use these later names, which are known in Rule Execution Server (no hyphen).

- ___ 3. Save your work (Ctrl+Shift+S) and close the JSP editor.

1.3. Building the application for deployment

- ___ 1. Create the web application in the `loan-webapp` project by using the `build.xml` Ant file.
 - ___ a. In the `loan-webapp` project, double-click **common.xml** to open the file.
 - ___ b. Find the `wodm.home` property and make sure that it is equal to `<InstallDir>\ODM` (for example, `"C:/Program Files/IBM/ODM87/ODM"`) and save the `common.xml` file if required.
 - ___ c. Close the `common.xml` file.
 - ___ d. Right-click **build.xml** and click **Run As > Ant Build...**



Warning

Make sure that you choose the **Ant Build...** option with the Ellipsis (...).

The Edit Configuration window opens.

- ___ e. In the Edit Configuration window, make sure that **loanvalidation.ear [default]** is selected and click **Run**.

The Edit Configuration window closes.

The `loanvalidation.ear` target of the `build.xml` file runs and creates the `loanvalidation.ear` application file in the `loan-webapp\build` folder.

- ___ 2. Verify that the web application is created.
 - ___ a. In the Rule Explorer, select the `loan-webapp` project and press **F5** to refresh the content.
 - ___ b. Expand the `loan-webapp` project and verify that the `build` folder is now visible.
 - ___ c. Verify that the `build` folder contains the `loan-xom.jar` file and the `loanvalidation.ear` file.
 - The `loan-xom.jar` file is the Java archive file for the XOM. This file is an intermediate artifact that is required to build the `loanvalidation.ear` file.
 - The `loanvalidation.ear` file is the file that you must deploy to WebSphere Application Server for its execution.

Section 2. Deploying the web application to WebSphere Application Server

In this section, you deploy the `loanvalidation.ear` file to WebSphere Application Server for its execution. Then, you verify the deployment of the `LoanValidation` application in the WebSphere Application Server administrative console.

2.1. Deploying the web application

- 1. Right-click the `build.xml` file and click **Run As > Ant Build...**

Make sure to choose the Ant Build option with the Ellipsis (...). The **Edit Configuration** window opens.

- 2. In the **Edit Configuration** window:

- Clear **loanvalidation.ear [default]**.
- Select **deploy**.
- Click **Apply** and click **Run**.

The **Edit Configuration** window closes.

The `deploy` target of the `build.xml` file runs and deploys the `loanvalidation.ear` application file into WebSphere Application Server.

Deploying the application might take some time, and successfully completes when you can see the following information in the Console view of Rule Designer:

```
LoanValidation was successfully installed  
BUILD SUCCESSFUL
```

2.2. Verifying application deployment to the application server

- 1. Open and sign in to the WebSphere Application Server administrative console.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Administrative console**.



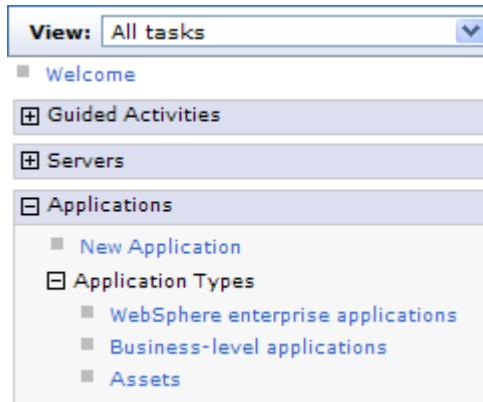
Warning

When you start the WebSphere Application Server administrative console, you might have a message that indicates that there is a problem with the security certificate of this website. Ignore this message, and click **Continue to this website** to proceed.

You might also have a JSP error. You can ignore the error for this exercise.

- b. In the **User ID** and **Password** fields, enter: `odm`
- c. Click **Log in**.

- ___ 2. In the left pane, expand **Applications > Application Types**, and click **WebSphere enterprise applications**.



- ___ 3. In the “Enterprise application” pane, verify that the `LoanValidation` application is present, and that its application status is started, which is marked with a green arrow.

This screenshot shows the "Enterprise Applications" page. At the top, there is a toolbar with buttons for "Start", "Stop", "Install", "Uninstall", "Update", "Rollout Update", "Remove File", "Export", "Export DDL", and "Edit". Below the toolbar is a toolbar with icons for "Select", "New", "Delete", "Copy", "Move Up", and "Move Down". The main area has three tabs: "Select", "Name" (which is currently selected), and "Application Status". A message "You can administer the following resources:" is displayed above a table. The table has two columns: "Name" and "Status". It contains one row for the "LoanValidation" application, which is marked as "Started" (indicated by a green arrow icon).

| Name | Status |
|--------------------------------|---------|
| LoanValidation | Started |

- ___ 4. Click **Logout** to exit the WebSphere Application Server Console.

Section 3. Executing the web application with the RuleApp deployed

In this section, you execute the web application.

- 1. Open a web browser at the following web address:

`http://localhost:9080/LoanValidation`

- 2. Look at three different scenarios in the first page:

- The first scenario uses a loan amount of 100,000. The loan is approved.
- The second scenario uses an amount of 200,000. The loan is rejected because the debt-to-income ratio is too high.
- The third scenario has an SSN with letters instead of digits, and a birth year of 1768. The loan is rejected because of the invalid input data.

- 3. Click **Start loan validation** at the top of the page.

The form to enter the borrower information opens.

- 4. Enter the borrower information in the form, according to the scenario of your choice, or leave the default values; and click **Next Step**.

The form to enter the loan information opens.

- 5. Enter the loan information in the form, according to the scenario of your choice, or leave the default values; and click **Calculate loan**.

The **Loan Report** page opens with the result of the execution of your ruleset.

End of exercise

Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 18: Executing rules in Java EE > 02-answer**.



Important

If you are not using the VMware image that is provided with this course, make sure that the `wodm.home` property in the `common.xml` file in the solution project is equal to: `<InstallDir>\ODM`

The `loanvalidation.ear` file is not present in the `build` folder of the `loan-webapp` project. You must re-create it by using the `build.xml` file, as explained in this exercise.

This exercise looked at how to create and deploy a web client application to execute business rules with Rule Execution Server in a Java EE environment. You also explored the required Rule Execution Server API.

Exercise 19. Executing rules as a hosted transparent decision service (HTDS)

What this exercise is about

This exercise teaches you how to execute rules as a hosted transparent decision service (HTDS).

What you should be able to do

After completing this exercise, you should be able to:

- Deploy an HTDS that can be used in a service-oriented architecture (SOA)
- Create a client application that uses an HTDS
- Monitor HTDS execution statistics in the Rule Execution Server console

Introduction

In this exercise, you review all the steps that are required to use hosted transparent decision services in an SOA environment.

After you deploy the ruleset as a RuleApp to Rule Execution Server, you create an HTDS from this ruleset. You then create a web service client application that uses the HTDS. After you run the application, you monitor the execution results in Rule Execution Server console by using the statistics that are associated with the HTDS.

This exercise is divided into these sections:

- Section 1, "Deploying the RuleApp to Rule Execution Server"
- Section 2, "Getting the HTDS WSDL file from the deployed ruleset"
- Section 3, "Creating the client project"
- Section 4, "Finalizing and executing the web service client application"
- Section 5, "Viewing decision service statistics"

Requirements

There are no specific requirements for this exercise.

Section 1. Deploying the RuleApp to Rule Execution Server

In this section, you deploy the RuleApp to Rule Execution Server.

1.1. Setting up your environment for this exercise

- ___ 1. If the sample server is not running, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.
- ___ 2. In Rule Designer, switch to a new workspace:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\htds`
- ___ 3. Close the Welcome view and switch to the Samples Console perspective.
- ___ 4. Select **Rule Designer > Training > Ex 19: Executing rules as a hosted transparent decision service > 01-start > Import projects**.
- ___ 5. Close the Help pane.

Your workspace now contains the `htds-rules` project, and the associated XOM and RuleApp projects. It also contains a Rule Execution Server configuration that is required to deploy the RuleApp to Rule Execution Server.



Hint

In this exercise, you must write some code. You can use the code snippets in the `<LabfilesDir>\code\execute_web_service.txt` file, and copy and paste them in Rule Designer.

1.2. Deploying the RuleApp

- ___ 1. In Rule Designer, expand the `htds-RESConfigs` folder and double-click the **IBM WebSphere AS 8.5** Rule Execution Server configuration to open it in the editor.
- ___ 2. Set the Rule Execution Server console **Login** and **Password** fields to: `resAdmin`
- ___ 3. Save the configuration (Ctrl+S).
- ___ 4. Right-click the `htds-rules` project, and click **Rule Execution Server > Deploy XOM**.



Troubleshooting

If you receive a Java version notification warning, click **Do not show this message again**, and click **OK**.

-
- ___ 5. In the “Deploy XOM to Rule Execution Server” page, make sure that the **Select existing Rule Execution Server configurations** option is selected, select the **IBM WebSphere AS 8.5** Rule Execution Server configuration, and then click **Finish**.

The traces in the Console view in Rule Designer are as follows:

Following XOM libraries have been deployed:

```
resuri://htds-xom.zip/1.0
```

- ___ 6. Deploy the **htds-RuleApp** RuleApp by using the **IBM WebSphere AS 8.5.esc** Rule Execution Server configuration.
 - ___ a. In the **IBM WebSphere AS 8.5.esc** Rule Execution Server configuration, click **Deploy** in the Deployment section.
 - ___ b. In the “Deploy RuleApp(s) on one Rule Execution Server” page, keep the default choice to increment the RuleApp major version, and click **Next**.
 - ___ c. In the “Deploy RuleApp(s) on one Rule Execution Server” page, select **htds-RuleApp** to deploy this RuleApp.
 - ___ d. Make sure the **Deploy XOM of rule projects and archives contained in the RuleApp** option is selected, and click **Finish**.

The traces in the Console view in Rule Designer give the version of the deployed RuleApp, and the URI to the associated managed XOM.

The “htds-RuleApp” RuleApp project was successfully deployed on the “IBM WebSphere AS 8.5” configuration.

```
/htdsRuleApp/1.0 -> /htdsRuleApp/1.0: Element added  
/htdsRuleApp/1.0/htdsrules/1.0 -> /htdsRuleApp/1.0/htdsrules/1.0:  
Element added  
XOM Deployed : resuri://htds-xom.zip/1.0
```

Section 2. Getting the HTDS WSDL file from the deployed ruleset

In this section, you retrieve the HTDS WSDL file, which is the definition of the web service that is associated with the deployed ruleset.

Now that the `htdsRuleApp` is deployed with its associated managed XOM in Rule Execution Server, open the Rule Execution Server console to retrieve the corresponding decision service, and its WSDL interface.

- 1. Open the Rule Execution Server console.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Rule Execution Server console**.
 - b. In both the **User Name** field and the **Password** field, enter: `resAdmin`
- 2. Click the **Explorer** tab and in the list of RuleApps, click the `htdsRuleApp` that you deployed to open the RuleApp View page with the details of this RuleApp.
- 3. Click the `htdsrules` ruleset of `htdsRuleApp`.

The Ruleset View page opens with the details of this ruleset.
- 4. On the Ruleset View page, click **Show HTDS Options**.



You see a series of options, which you can edit if you have specific requirements in your enterprise. For this exercise, you use the default settings.

- 5. Click **Retrieve HTDS Description File**.

[Hide HTDS Options](#)

 **Hosted Transparent Decision Services (HTDS) Options**

| | |
|---------------------------------------|---|
| Location | http://localhost:9080/DecisionService  |
| Web service endpoint | http://localhost:9080/DecisionService  |
| Target namespace (SOAP only) | http://www.ibm.com/rules/decisionservice/HtdsRuleApp/Htdsrules  |
| Parameter target namespace | http://www.ibm.com/rules/decisionservice/HtdsRuleApp/Htdsrules/param  |
| Retrieve HTDS Description File | |

This link is also available in the top menu of the Ruleset View page.

- 6. In the Retrieve HTDS Description File pane, specify the required options.
 - a. Select these options:
 - **Latest ruleset version**
 - **Latest RuleApp version**

You select **Latest ruleset version**, **Latest RuleApp version**, or both, to create a WSDL file that either uses a canonical ruleset path or not. In the present case, you select both check boxes to create a WSDL file that is based on a non-canonical

ruleset path (where both versions are ignored) to execute the most recent RuleApp and ruleset versions.

- **Decision trace information**

Select **Decision trace information** to set filters in the request, and later retrieve decision traces in the response.

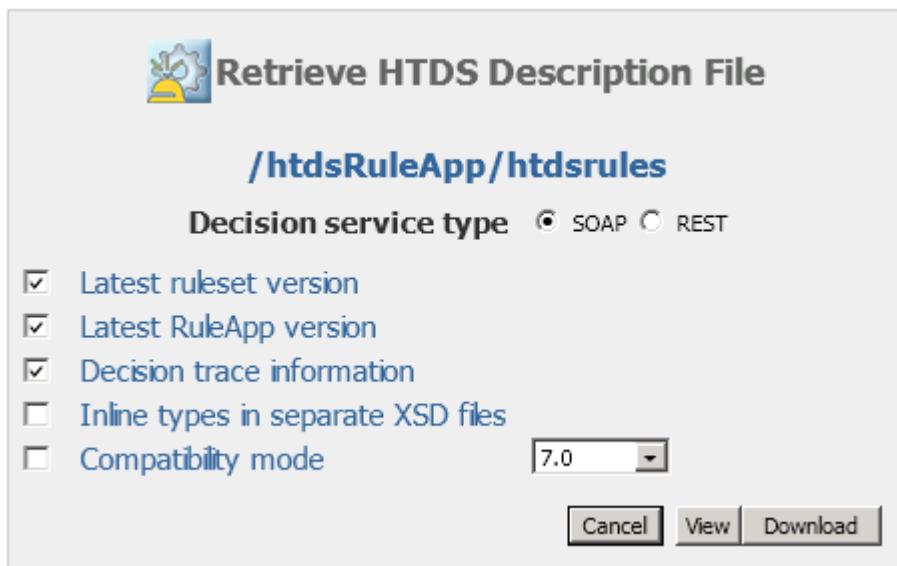
- ___ b. Leave the other check boxes as not selected.

- **Inline types in separate XSD files**

You select or clear this check box, depending on whether you have models with the same name and the same namespace, and on how you want to manage their potential conflict. See the documentation for more details. In this course, there are no risks of conflict, so clear this check box.

- **Compatibility mode**

You select or clear this check box, depending on whether you want to create a WSDL file that is compatible with a previous version of Operational Decision Manager. If you select it, you must also select the appropriate version. In this course, you do not try to be compatible with any previous version of the product, so clear this check box.

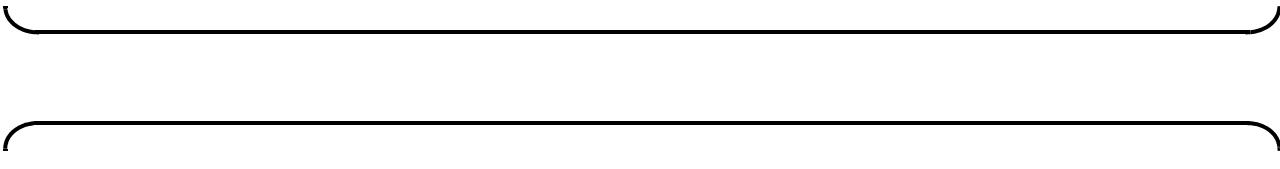


- ___ 7. Click **View** to see the contents of the WSDL file.

Questions

Can you answer the following questions?

- ___ a. What is the signature of this decision service? Can you relate it to the signature of the deployed ruleset and to the XSD used to define your XOM?
- ___ b. Where are the RuleApp version and the ruleset version defined?

- ___ c. What web address do you use if you want to execute the most recent activated ruleset that is deployed to Rule Execution Server, with decision traces enabled?
- 

Answers

- The signature of the WSDL file is the same as the signature of the ruleset. In the present case, there are two ruleset parameters: a `Borrower` and a `Loan`.

This WSDL file is strongly typed and the definition of the ruleset parameters is included in the `<wsdl:types>` element of the WSDL file.

- The RuleApp version and the ruleset version are given in the ruleset path in the `<wsdl:service>` element of the WSDL file.
- You can read the WSDL file web address in the **Address** field of the web browser:

`http://localhost:9080/DecisionService/ws/htdsRuleApp/
htdsrules?WSDL&trace=true`

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

To make sure that you always execute the most recent activated ruleset that is deployed in Rule Execution Server, you use a non-canonical ruleset path in the web address.

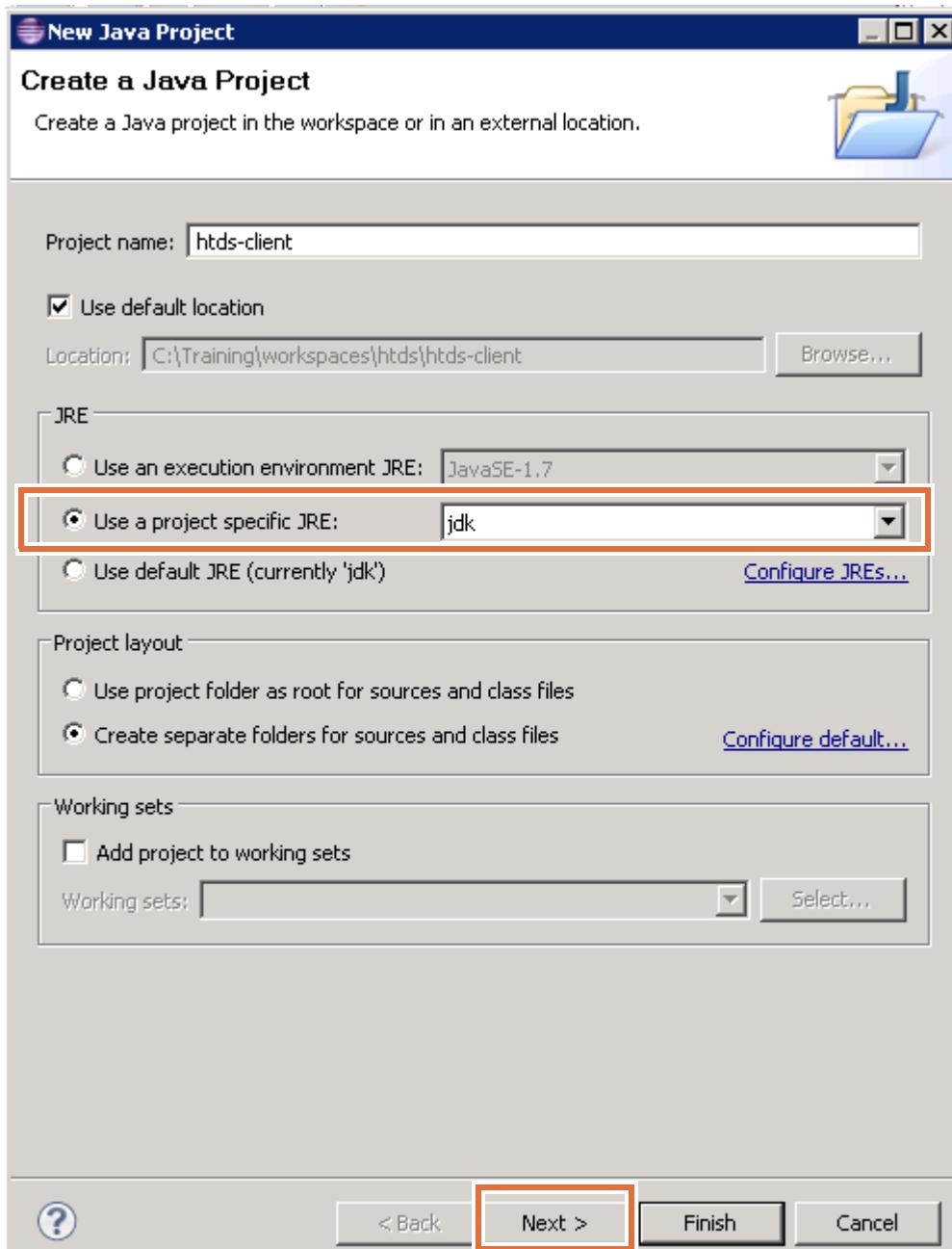


Section 3. Creating the client project

In this section, you create a web service client project to request the ruleset execution as a web service.

- ___ 1. In Rule Designer, switch to the Java perspective.
- ___ 2. Right-click anywhere in the Package Explorer view, and click **New > Java Project**.
The New Java Project window opens.
- ___ 3. Define the Java project in the “Create a Java Project” pane.
 - ___ a. In the **Project name** field, enter: htds-client

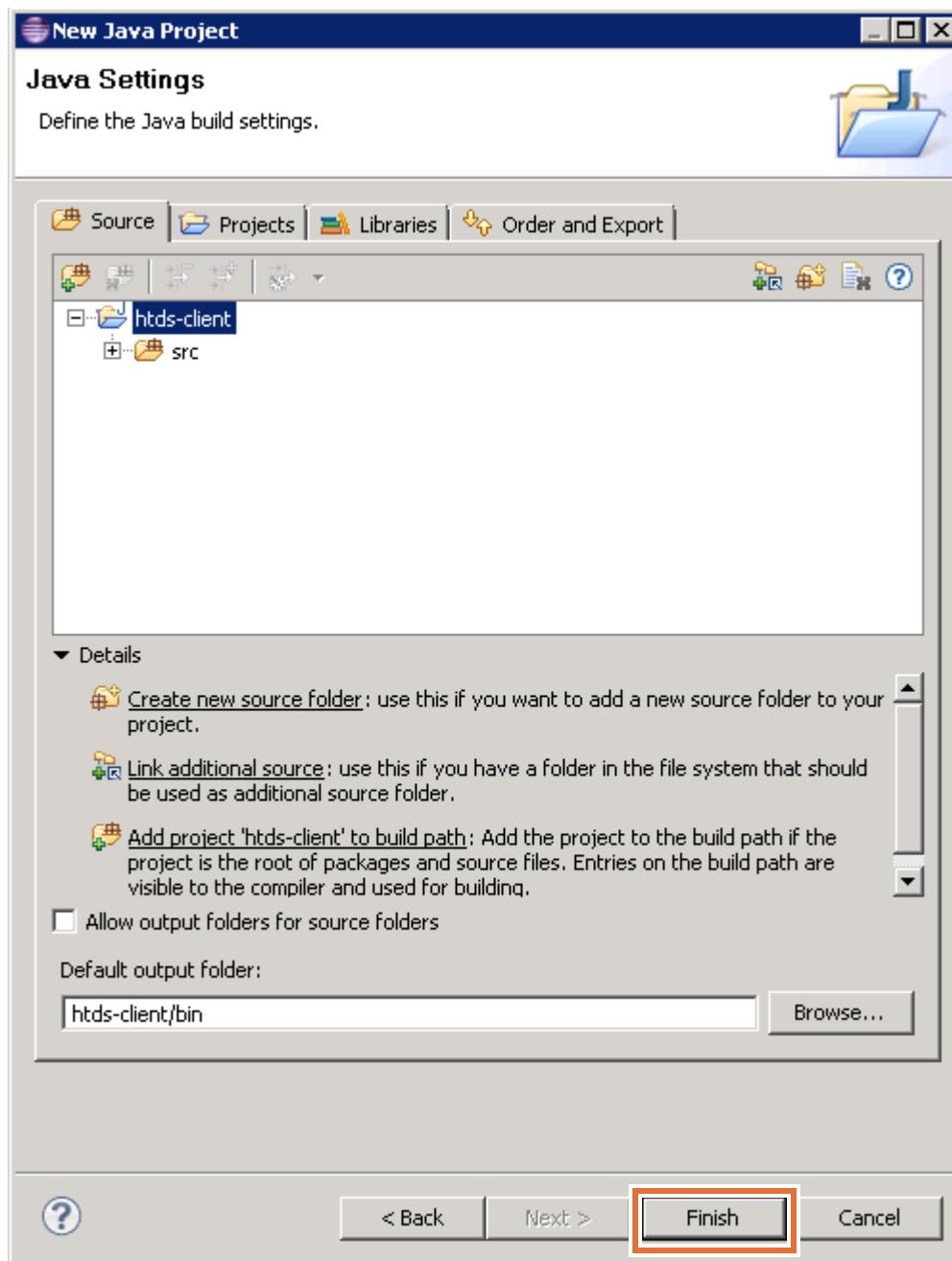
- __ b. In the JRE section, select the **Use project specific JRE** option, and make sure that `jdk` is selected.



- __ c. Leave other fields unchanged and click **Next**.

The Java Settings pane opens.

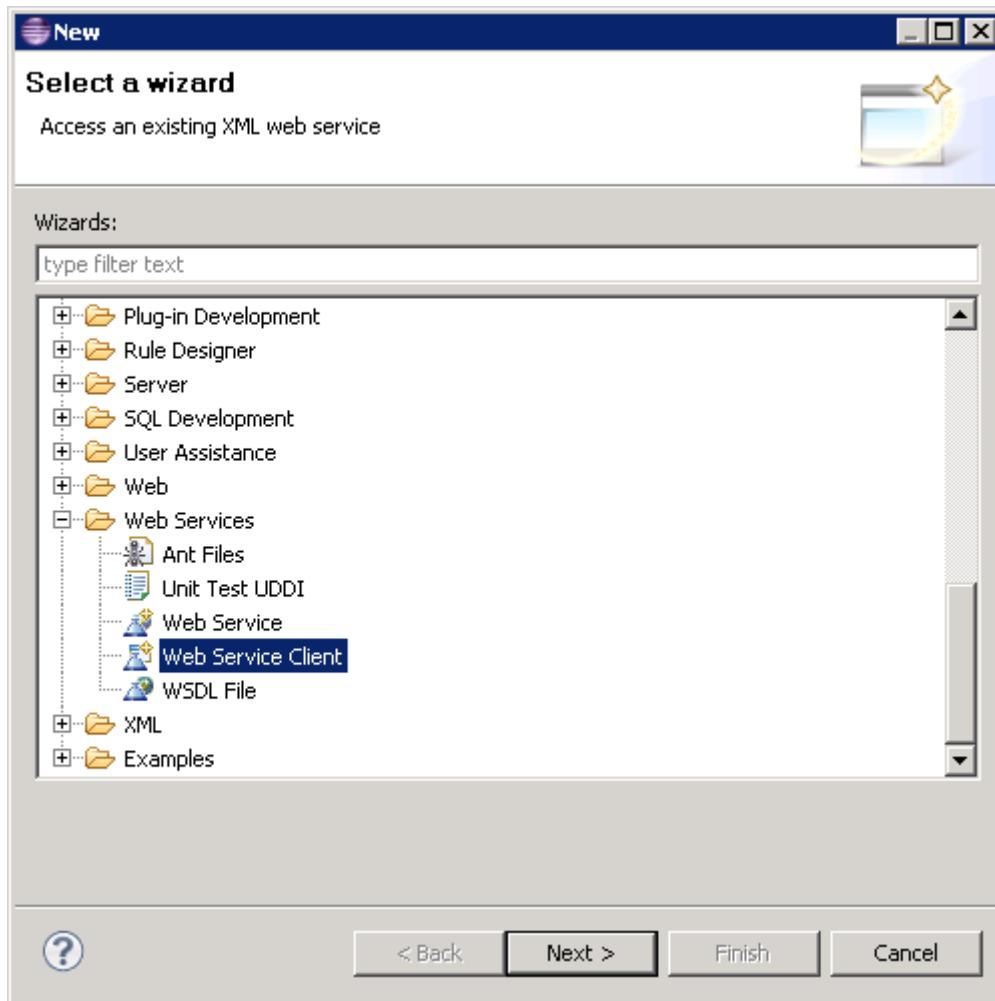
4. In the Java Settings pane, keep all default values and click **Finish**.



The htds-client project is now visible in the Package Explorer.

You now add a web service client to this project.

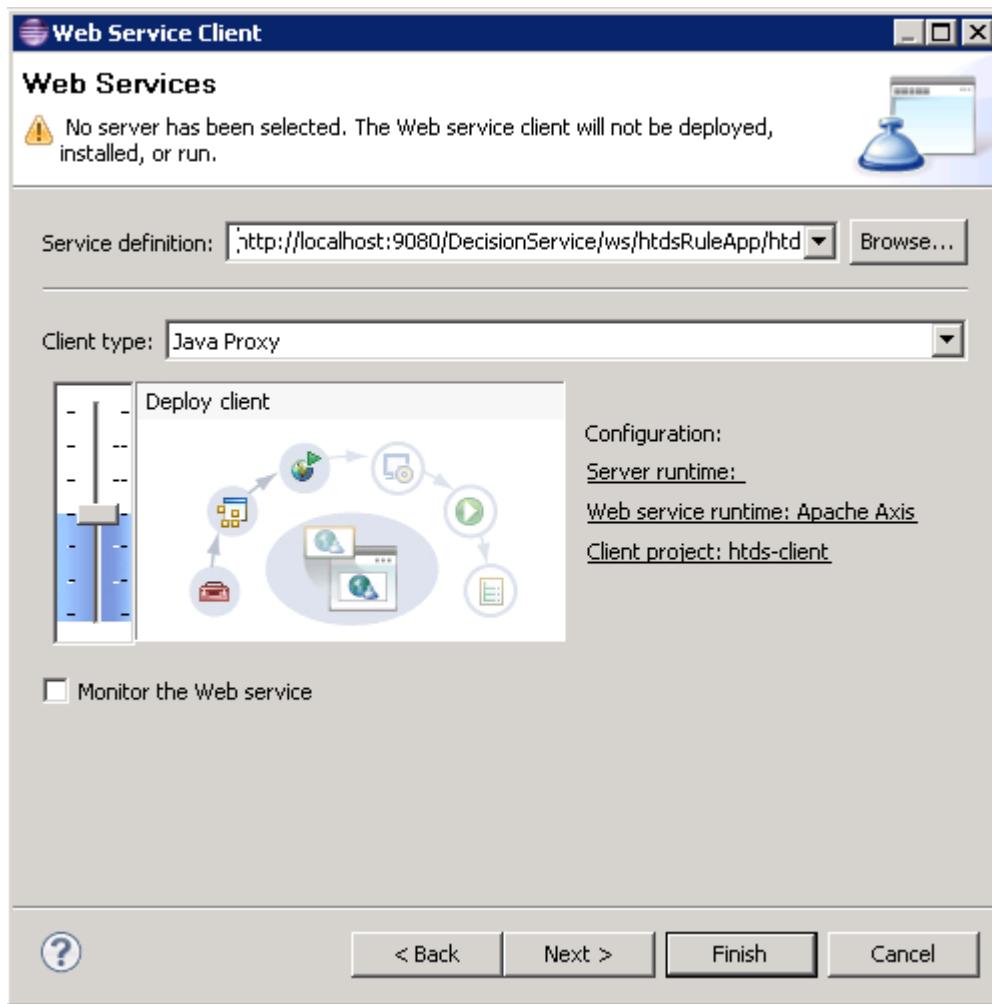
- ___ 5. In the Package Explorer, right-click the `htds-client` project, click **New > Other > Web Services > Web Service Client**, and click **Next**.



The Web Service Client window opens.

- ___ 6. In the Web Services page, define the web service.
- ___ a. In the **Service definition** field, enter the URL of the WSDL file:
`http://localhost:9080/DecisionService/ws/htdsRuleApp/htdsrules?WSDL&trace=true`
Make sure that you use the correct port for your environment.
- ___ b. In the **Client Type** field, keep **Java Proxy**.

- ___ c. Leave other fields unchanged.



- ___ d. Click **Next**.
- ___ e. In the Web Service Proxy Page page, make sure that the **Output folder** field is equal to /htds-client/src and click **Finish**.

A full web service client application is generated in the `src` directory of the `htds-client` project.

Section 4. Finalizing and executing the web service client application

In this section, you complete the code of the web service client application, and then execute this application.

4.1. Exploring the web service client application

Explore the web service client application in the **htds-client > src** folder and notice these points:

- The `com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules` package defines the Java classes that are used in the client application. These Java classes mimic the XOM classes of the `htds-xom` project for serialization and deserialization purposes.
- The classes in the `com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules.param` package define the ruleset parameters, including `Borrower` and `Loan`.

These classes wrap the classes that are defined in the `com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules` package.

To define the value of a ruleset parameter, you must:

- First, create an instance of a class that is defined in the `com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules` package.
- Wrap this instance in an instance of the wrapper class that is defined in the `com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules.param` package.

Guided instructions for this task are provided in the next steps.

- The `com.ibm.www.rules.decisionservice.trace` package and the `com.ibm.www.rules.decisionservice.tracefilter` package define the Java classes that are used in the client application to manage decision traces.

4.2. Writing the code that executes your ruleset as a web service



Hint

You can find this code snippet in the `<LabfilesDir>\code\execute_web_service.txt` file.

- 1. Create the public `Main` class for your web service client project.
 - a. In the Package Explorer, expand the **htds-client > src** folder.
 - b. Create the stub for this `Main` class by right-clicking the `src` directory and clicking **New > Class**.
 - c. In the **Name** field of the Java Class pane, enter: `Main`

-
- ___ d. In the method stubs section, select the **public static void main (String[] args)** check box and leave the other fields unchanged.
 - ___ e. Click **Finish**.
- ___ 2. Set the correct `import` statements for the `Main` class.
- ```
import com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules.*;
import com.ibm.www.rules.decisionservice.trace.ArtifactInformation;
import com.ibm.www.rules.decisionservice.tracefilter.TraceFilter;
```
- \_\_\_ 3. In the `main` method, replace:
- ```
// TODO Auto-generated method stub
```

With the following code:

```
try {
    String decisionID = null;
    TraceFilter decisionTraceFilter = new TraceFilter();
    decisionTraceFilter.setAll(true);
    int creditScore = 600;
    String name = "Joe";
    int yearlyIncome = 80000;
    Borrower borrower = new Borrower(creditScore, name, yearlyIncome);
    com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules.param.Borrower
    borrower_param = new
    com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules.param.Borrower(borrower
    );
    int amount = 500000;
    boolean approved = true;
    int duration = 240;
    String[] messages = new String[0];
    double yearlyInterestRate = 0.05;
    int yearlyRepayment = 20000;
    Loan loan = new Loan(amount, approved, duration, messages, yearlyInterestRate,
    yearlyRepayment);
    com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules.param.Loan loan_param =
    new
    com.ibm.www.rules.decisionservice.HtdsRuleApp.Htdsrules.param.Loan(loan);
    HtdsrulesRequest htdsrulesRequest = new HtdsrulesRequest(decisionID,
    decisionTraceFilter, borrower_param, loan_param);
    HtdsrulesResponse response = new
    HtdsrulesDecisionServiceProxy().htdsrules(htdsrulesRequest);
    // Print elements of the response:
    System.out.println("Decision ID      :" + response.getDecisionID());
    System.out.println("Output string   :" +
    response.getDecisionTrace().getOutputString());
    System.out.println("Ruleset path    :" +
    response.getDecisionTrace().getExecutedRulesetPath());
    System.out.println("Loan is approved:" +
    response.getLoan().getLoan().isApproved());
    System.out.println("Messages        :" );
    messages = response.getLoan().getLoan().getMessages();
    if (messages != null) {
        for (int i = 0; i < messages.length; i++) {
            System.out.println("* Messages#" + i + ":" + messages[i]);
        }
    }
    // Print elements of the execution:
    System.out.println("Rules in the ruleset:");
    ArtifactInformation[] ai = response.getDecisionTrace().getRules();
    for (int i = 0; i < ai.length; i++) {
```

```

        System.out.println("* Rule#" + i + ":" + ai[i].getBusinessName());
    }
System.out.println("Executed task in the ruleset:");
ArtifactInformation[] fti = response.getDecisionTrace().getTasksExecuted();
for (int i = 0; i < fti.length; i++) {
    System.out.println("* Executed task #" + i + ":" + fti[i].getBusinessName());
}
System.out.println("Executed rules:");
ArtifactInformation[] fai = response.getDecisionTrace().getRulesFired();
for (int i = 0; i < fai.length; i++) {
    System.out.println("* Executed rule #" + i + ":" + fai[i].getBusinessName());
}
} catch (Exception e) {
e.printStackTrace();
}

```



Note

In this exercise, the Decision ID used to create the `DecisionServiceRequest` is set to null at first. Rule Execution Server thus sets and returns this ID.

- 4. Save your work.

The `Main` class compiles without errors.



Note

You can ignore the warnings in the Problems view.

4.3. Running the Main class

- 1. Right-click the `Main` class in the Project Explorer view, and click **Run As > Java application**.

In the Console view, you see a message that describes which rules were executed.

Decision ID :<a value that uniquely identifies this execution>
Output string :true []

Ruleset path :/htdsRuleApp/1.0/htdsrules/1.0
Loan is approved:true
Messages :
Rules in the ruleset:
...

**Note**

You can ignore the warnings in the console.

Because of the values of the ruleset parameters, no action rule is fired, the loan is approved, and the Messages are empty.

2. In the Main class, change the amount of the loan request to 2,000,000.

```
public static void main(String[] args) {
    try {
        String decisionID = null;
        TraceFilter decisionTraceFilter = new TraceFilter();
        decisionTraceFilter.setAll(true);
        int creditScore = 600;
        String name = "Joe";
        int yearlyIncome = 80000;
        Borrower borrower = new Borrower(creditScore, name, yearlyIncome);
        com.ibm.bpm.rules.decsionservice.HtdsRuleApp.Htdsrules.param.Borr
        int amount = 2000000;
        boolean approved = true;
        int duration = 240;
        String[] messages = new String[0];
        double yearlyInterestRate = 0.05;
        int yearlyRepayment = 20000;
```

3. Save the Main class and run it again to compare the results.

Decision ID :<a value that uniquely identifies this execution>
Output string :false [The loan cannot exceed 1,000,000]

Ruleset path :/htdsRuleApp/1.0/htdsrules/1.0
Loan is approved:false
Messages :
* Messages#0: The loan cannot exceed 1,000,000
Rules in the ruleset:
...

Because the loan amount exceeds 1,000,000, the maximum amount action rule in the validation rule package executes, the loan is not approved, and the Messages indicate the reason for this decision.

Notice that the Decision ID changed. This value uniquely identifies the decision that is taken.

Section 5. Viewing decision service statistics

In this section, you examine the statistics that are associated with the decision service that you created. Statistics are available in Rule Execution Server console.

- 1. Go to the Rule Execution Server console.
- 2. In the Navigator pane on the **Explorer** tab, click **Service Information**.

The Transparent Decision Services View opens. You can see the `HTDS/htdsRuleApp/htdsrules` decision service.

It might be necessary to click **Refresh** in the Transparent Decision Services View to see this HTDS in the list of decision services.

Transparent Decision Service Information View

Transparent Decision Service Information

Total Number of Transparent Decision Services 1

- 3. Click the `HTDS/htdsRuleApp/htdsrules` decision service to view its details.
- You can click this decision service either in the Navigator pane, or in the Transparent Decision Service View pane.
- The Transparent Decision Service View pane shows the details of the `HTDS/htdsRuleApp/htdsrules` decision service.

Transparent Decision Service View

| Server Name | Status | Executed Canonical Rules |
|---|--------|------------------------------|
| SamplesCell - SamplesNode - SamplesServer | ✓ | /htdsRuleApp/1.0/htdsrules/1 |

4. Click  **View Statistics** to view the statistics that are associated with the executions of this transparent decision service.

The Transparent Decision Service Statistics View pane opens for the HTDS/htdsRuleApp/htdsrules decision service.

In this pane, you can verify the executed canonical ruleset path, which is based on the versions of the ruleset and of the RuleApp that you deployed. You can also see the statistics about the executions of this decision service.

 **HTDS/htdsRuleApp/htdsrules**

Executed Canonical Ruleset Path /htdsRuleApp/1.0/htdsrules/1.0

| Server | Statistics | | | | | | | | | | | | | | | | | | | | | | |
|---|--|--------|-------|--------------|---|-----------------|---------------|-------|---|-----------------|-----|-------------------|--------|----------------|---|----------------|-----|--------------------------|---|----------------------|-----------------------------------|---------------------|-----------------------------------|
| SamplesCell - SamplesNode - SamplesServer | <table border="1"><thead><tr><th>Metric</th><th>Value</th></tr></thead><tbody><tr><td>Errors Count</td><td>0</td></tr><tr><td>Error Last Date</td><td>Not Available</td></tr><tr><td>Count</td><td>2</td></tr><tr><td>Total Time (ms)</td><td>109</td></tr><tr><td>Average Time (ms)</td><td>54.500</td></tr><tr><td>Min. Time (ms)</td><td>0</td></tr><tr><td>Max. Time (ms)</td><td>109</td></tr><tr><td>Last Execution Time (ms)</td><td>0</td></tr><tr><td>First Execution Date</td><td>Jan 11, 2015 6:03:35 PM GMT-08:00</td></tr><tr><td>Last Execution Date</td><td>Jan 11, 2015 6:06:02 PM GMT-08:00</td></tr></tbody></table> | Metric | Value | Errors Count | 0 | Error Last Date | Not Available | Count | 2 | Total Time (ms) | 109 | Average Time (ms) | 54.500 | Min. Time (ms) | 0 | Max. Time (ms) | 109 | Last Execution Time (ms) | 0 | First Execution Date | Jan 11, 2015 6:03:35 PM GMT-08:00 | Last Execution Date | Jan 11, 2015 6:06:02 PM GMT-08:00 |
| Metric | Value | | | | | | | | | | | | | | | | | | | | | | |
| Errors Count | 0 | | | | | | | | | | | | | | | | | | | | | | |
| Error Last Date | Not Available | | | | | | | | | | | | | | | | | | | | | | |
| Count | 2 | | | | | | | | | | | | | | | | | | | | | | |
| Total Time (ms) | 109 | | | | | | | | | | | | | | | | | | | | | | |
| Average Time (ms) | 54.500 | | | | | | | | | | | | | | | | | | | | | | |
| Min. Time (ms) | 0 | | | | | | | | | | | | | | | | | | | | | | |
| Max. Time (ms) | 109 | | | | | | | | | | | | | | | | | | | | | | |
| Last Execution Time (ms) | 0 | | | | | | | | | | | | | | | | | | | | | | |
| First Execution Date | Jan 11, 2015 6:03:35 PM GMT-08:00 | | | | | | | | | | | | | | | | | | | | | | |
| Last Execution Date | Jan 11, 2015 6:06:02 PM GMT-08:00 | | | | | | | | | | | | | | | | | | | | | | |



If you execute the decision service again, you must click  **Refresh** to refresh the statistics that are associated with its executions.

End of exercise

Exercise review and wrap-up

To see the solution to this exercise, switch to a new workspace and import the project **Ex 19: Executing rules as decision services > 02-answer** in the Java perspective.

This exercise looked at how you can use rule-based decision services in an SOA environment. You first deployed the RuleApp to Rule Execution Server and retrieved the HTDS WSDL file, which is the definition of the web service that is associated with the deployed ruleset. Next, you created a web service client application that executes your rules as a web service. Finally, you examined the statistics that are associated with the decision service that you created.

Exercise 20. Auditing ruleset execution through Decision Warehouse

What this exercise is about

This exercise describes how to enable monitoring of ruleset execution and how to audit execution traces in Decision Warehouse.

What you should be able to do

After completing this exercise, you should be able to:

- Enable monitoring for ruleset execution
- Retrieve decision traces through Decision Warehouse
- Optimize Decision Warehouse
- Delete trace data from Decision Warehouse

Introduction

After the rules are deployed in the execution environment, you must be able to audit which rules were executed to determine the decision. In this exercise, you learn how to enable monitoring for a ruleset and deploy it from Rule Designer. Next, you test rule execution to generate rule execution traces that you can review in Decision Warehouse.

You also configure monitoring properties and deploy from Decision Center to see how execution traces in Decision Warehouse can link you directly to the corresponding rule artifacts in Decision Center.

The exercise includes these sections:

- Section 1, "Enabling ruleset monitoring"
- Section 2, "Retrieving decision traces in Rule Execution Server console"
- Section 3, "Optimizing Decision Warehouse"
- Section 4, "Deleting trace information from the database"

Requirements

This exercise uses the web application that you deployed during Exercise 15, "Managing deployment". You should complete these exercises before proceeding:

- Exercise 16, "Exploring the Rule Execution Server console"
- Exercise 18, "Executing rules with Rule Execution Server in Java EE"

Section 1. Enabling ruleset monitoring

You can set properties on your ruleset to capture execution traces and determine the behavior of the ruleset during execution.

When you enable monitoring, you can retrieve information such as:

- How many tasks were executed
- How many rules were executed
- What events took place in the working memory
- Which version of the ruleset was executed

You set monitoring properties at ruleset level. You can set this ruleset property in the Rule Execution Server console by selecting the ruleset and clicking **Add Property**. By default, all the information is retrieved, except working memory events.

In this section, you enable monitoring properties on your ruleset, redeploy it, and then execute it to generate traces. After execution, you review the traces in Decision Warehouse.

1.1. Setting up your environment for this exercise

This exercise uses the same project that you used during Exercise 16, "Exploring the Rule Execution Server console".

- 1. If the sample server is not started, start it now by clicking **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.
- 2. In Rule Designer, switch to a new workspace:
 - a. From the **File** menu, click **Switch Workspace > Other**.
 - b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\decision_warehouse`
- 3. Close the Welcome view and switch to the Samples Console perspective.
- 4. Select **Rule Designer > Training > Ex 20: Auditing ruleset execution > 01-start > Import projects**.
- 5. Close the Help view.

1.2. Enabling monitoring

In this section, you use Rule Designer to set these properties on the ruleset:

- `monitoring.enabled` with its value set to `true`
- `rulesetSEQUENTIAL.trace.enabled` with its value set to `true`

You use the `rulesetSEQUENTIAL.trace.enabled` property when the ruleset contains tasks that use the sequential or the Fastpath execution modes.

To enable monitoring of your ruleset:

- ___ 1. In Rule Explorer, expand the **loan-RuleApp** project and double-click the `archive.xml` file to open it in the RuleApp editor.
- ___ 2. Click the **Ruleset Archives** tab of the RuleApp editor and select **loanrules** in the Ruleset Archives pane.

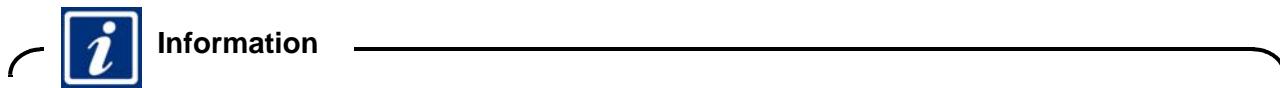
After you select the `loanrules` ruleset, the ruleset data opens in the Ruleset Properties section of the editor so you can add or modify properties.

The screenshot shows the RuleApp editor interface. On the left, the 'Ruleset Archives' pane displays a list of ruleset archives, with 'loanrules' selected and highlighted by a red box. On the right, the 'Ruleset Properties' pane shows various configuration options for the selected archive. A second red box highlights the 'Create your own properties:' section, which contains a table with one row: Name = ruleset.bom.enabled and Value = true.

| Name | Value |
|---------------------|-------|
| ruleset.bom.enabled | true |

The `ruleset.bom.enabled` property is already enabled.

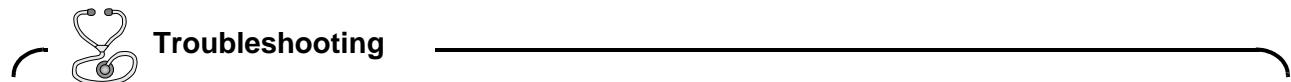
- ___ 3. In the “Create your own properties” section, define the `ruleset.sequential.trace.enabled` ruleset property.
 - ___ a. In the “Create your own properties” section, click **New**.
 - ___ b. In the Edit Property window, select the `ruleset.sequential.trace.enabled` ruleset property and set its **Value** to: `true`
 - ___ c. Click **OK** to close the Edit Property window.
- ___ 4. Repeat Step 3 to create the `monitoring.enabled` ruleset property and set its **Value** to: `true`
- ___ 5. Save your work.



1.3. Deploying the updated RuleApp from Rule Designer

In this section, you redeploy the RuleApp from Rule Designer to Rule Execution Server to make sure that the deployed RuleApp has the correct properties defined.

- 1. In Rule Explorer, expand the `loan-RESconfigs` folder, and double-click `IBM WebSphere AS 8.5.esc` to open the configuration.
- 2. Set the login details for the configuration.
 - a. In the **Rule Execution Server console** section, set the **Login** and **Password** fields to: `resAdmin`
 - b. Save the `IBM WebSphere AS 8.5.esc` configuration with your login information.



If you do not save the login information, deployment fails.

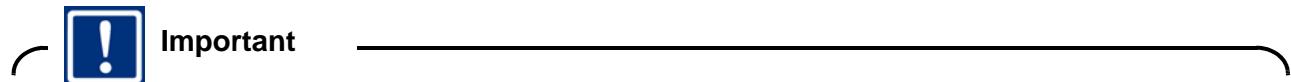
- 3. Deploy the RuleApp to Rule Execution Server.
 - a. In the Deployment section of the `IBM WebSphere AS 8.5.esc` configuration, click **Deploy**.



If you receive a Java version notification warning, click **Do not show this message again**, and click **OK**.

- b. Keep the default version policy and click **Next**.
- c. Select **loan-RuleApp** and click **Finish** to deploy the `loanRuleApp` and the XOM.
- 4. Look at the traces that are displayed in the Console view and make a note of the value of the major version of this `loanRuleApp`.

You must know the version of this RuleApp in a later step.



You use this `loanRuleApp` version later in "Retrieving decision traces in Rule Execution Server console" on page 20-7.

1.4. Running the client application

In this section, you run the Loan Validation application that you created in Exercise 18, "Executing rules with Rule Execution Server in Java EE" to generate some traces. Now that there are defined

monitoring properties on your ruleset, the monitoring properties capture decision trace data when the client application calls that ruleset for execution. The data is stored in Decision Warehouse.

- __ 1. Open the client application in a web browser at the following URL:

`http://localhost:9080/LoanValidation`

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

- __ 2. Click **Start Loan Validation** and run the application with the default values.

Next, you work in Rule Execution Server console to view the execution traces that you generated.

Section 2. Retrieving decision traces in Rule Execution Server console

In this section, you work in Rule Execution Server console. You examine the RuleApp that you deployed from Rule Designer. You then explore the decision traces that are associated with the execution of the ruleset that is packaged in this RuleApp.

- ___ 1. If the Rule Execution Server console is not already open, sign in as an administrator.
 - ___ a. Open a web browser at the following web address:
`http://localhost:9080/res`
 - ___ b. In the **Username** and **Password** fields, enter `resAdmin` and click **Sign In**.



Troubleshooting

In some cases, Rule Execution Server or the Loan Validation application might fail to load when you type the URL in a browser.

If you experience this problem, closing and reopening the browser can resolve the issue. Otherwise, close your browsers again and try starting Rule Execution Server console from the **Start** menu.

- ___ 2. In the Rule Execution Server console, examine the RuleApp that you deployed from Rule Designer.

- ___ a. Click the **Explorer** tab.
- ___ b. In the RuleApps View page, click the **loanRuleApp** version that you deployed in "Deploying the updated RuleApp from Rule Designer" on page 20-5.

You can see in the details of this RuleApp that the value of its **Display Name** field is:

`loan-RuleApp`

- ___ c. In the Ruleset(s) section, click the **loanrules** ruleset for this RuleApp.

The Ruleset View page opens for the `loanrules` ruleset. You can see in the details of this ruleset that the value of its **Display Name** field is:

`loan-rules`

- ___ d. Click **Show Properties** for this ruleset.

You can see the ruleset properties that you explicitly defined in Rule Designer for this ruleset:

- `monitoring.enabled`
- `ruleset.bom.enabled`
- `rulesetSEQUENTIAL.trace.enabled`

- ___ 3. In the Rule Execution Server console, click the **Decision Warehouse** tab.

The **Decision Warehouse** tab includes these tasks:

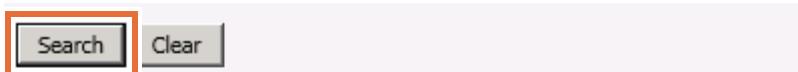
- **Search Decisions:** To define filters on the data source to find only the events or decisions of interest.
 - **Persistence Properties:** To select the data source for Decision Warehouse to query during your web session. Decision Warehouse is installed with a default data source. You can add more sources, such as a historical database.
- When your web session expires, the active data source reverts to the default data source.
- **Clear Decisions:** To delete existing decisions from the Decision Warehouse.

By default, the Search Decisions page opens. You can switch between these pages with the “Select a task” left pane of the **Decision Warehouse** tab.



- ___ 4. In the Search Decisions page, leave all fields empty and click **Search**.
- ___ 5. Scroll down the page to see the results.

All decisions that are stored in Decision Warehouse are listed because you did not specify any search filters.



| 3 Decision(s) found | | | | |
|---------------------------------------|---------------------|--------------------------------|-----------------------|---------------------------------------|
| Decision ID | Date | Ruleset Version | Number of rules fired | Decision Trace |
| c445073c-b073-49d3-91ab-ec49072fa7680 | 2015-01-14 13:11:23 | /loanRuleApp/4.0/loanrules/1.0 | 9 | View Decision details |

- ___ 6. Click the **View Decision details** link in the **Decision Trace** column.

The Decision Trace page opens, and shows the details of the execution.

- 7. Look at the fields in the Execution Details section to relate them to the ruleset execution.

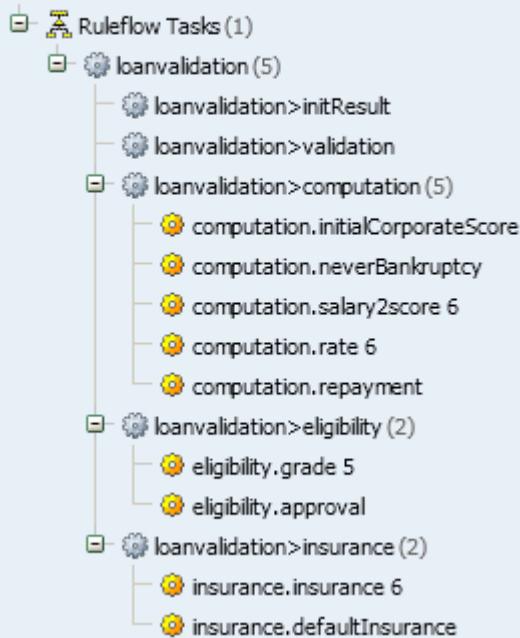
Execution Details

| | |
|---------------------------------|---------------------------------------|
| Decision ID: | c445073c-b073-49d3-91ab-ec49072fa7680 |
| Date: | 2015-01-14 13:11:23 |
| Executed ruleset path: | /loanRuleApp/4.0/loanrules/1.0 |
| Engine type: | cre |
| Processing Time (ms) | 31 |
| Number of rules fired | 9 |
| Number of tasks executed | 6 |

Your RuleApp version and statistics might differ from this screen capture.

- 8. Expand the tree in the **Decision Trace** section.

Decision Trace



Questions

Do you understand the trace?

Answer

Each node corresponds to a rule task or an executed rule artifact, such as a fired action rule, a row in a decision table, or a function. For a row in a decision table, the rank of the row is given. For example, `computation.rate 6` is the sixth row in the `computation.grade` decision table.

-
- ___ 9. Close the Decision Trace window.



Note

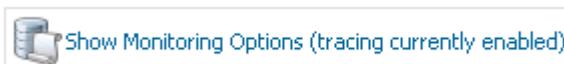
The Execution Details section in Decision Warehouse does not provide you with links that directly access the rule artifacts that were involved in the taken decision. However, when you deploy from Decision Center, the rules that are listed in the decision trace are links to the actual rule artifacts in Decision Center.

Section 3. Optimizing Decision Warehouse

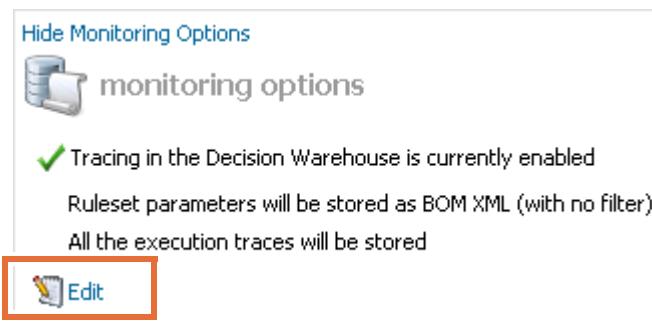
When the trace is enabled, Decision Warehouse captures all execution trace data. You can optimize Decision Warehouse by filtering which parts of the ruleset execution trace you monitor and store, and by turning off BOM serialization.

3.1. Working with monitoring options

- 1. In Rule Execution Server console, click the **Explorer** tab and expand RuleApps in the Navigator pane to open the RuleApp that you deployed.
- 2. Open the Ruleset View page for the `loanrules` ruleset of your RuleApp.
This ruleset is the same one that you worked with during Section 2, "Retrieving decision traces in Rule Execution Server console," on page 20-7 to set the ruleset properties.
- 3. Click **Show Properties** to reopen the list of ruleset properties, and note the `monitoring.enabled` ruleset property, which should be set to `true`.
- 4. Click **Show Monitoring Options**.



- 5. Identify the connection between the **Enable tracing in Decision Warehouse** option and the `monitoring.enabled` ruleset property.
 - a. Click **Edit** to open the monitoring options.



- b. Notice that **Enable tracing in Decision Warehouse** is selected.
- c. Clear **Enable tracing in Decision Warehouse** and click **Save**.
- d. Refresh Rule Execution Server console by pressing F5.



Troubleshooting

If you see a prompt from the web browser to confirm whether you want to resend information, click **Retry**.

- ___ e. Look in the properties section to verify that the `monitoring.enabled` ruleset property is now set to `false`.

| Select All | Name | Value |
|--------------------------|--------------------|-------|
| <input type="checkbox"/> | monitoring.enabled | false |
| <input type="checkbox"/> | monitoring.filters | |

- ___ 6. Enable decision tracing again.

- ___ a. Reopen the **monitoring options** section and click the **Edit** icon.

Hide Monitoring Options

monitoring options

Tracing in Decision Warehouse is currently disabled

- ___ b. Select **Enable tracing in Decision Warehouse**.

- ___ c. Click **Save**.

3.2. Specifying filters on the trace data

To reduce the information that is stored in an execution trace in Decision Warehouse, you:

- Apply the `monitoring.filters` property to the ruleset
- Select which filters to set so that you can refine the data that is stored

You can set the `monitoring.filters` property filter values in Rule Designer or Decision Center before you deploy a ruleset. You can also set this property on the deployed ruleset in Rule Execution Server console, which you do next.

- ___ 1. In the **monitoring options** section, click **Edit** to reopen the monitoring properties.
- ___ 2. In the **Select the execution traces to store in the Decision Warehouse** list, notice that several traces are selected.

The selected traces are listed as values of the `monitoring.filters` ruleset property.

| | |
|--------------------|--|
| monitoring.enabled | true |
| monitoring.filters | INFO_EXECUTION_DATE=true,INFO_EXECUTION_DURATION=true,INFO_TOTAL_T |

- ___ 3. Select *all* the remaining execution traces in the list and click **Save**.
- ___ 4. Notice that the `monitoring.filters` property disappears from the list of properties.
Selecting all the optional filters is equivalent to turning off the filter property.
- ___ 5. Click **Edit** to reopen the monitoring options and clear all the selected traces, and then click **Save**.

- ___ 6. Notice that the `monitoring.filters` property is again in the list of properties, but the value is empty.

| | | | |
|--------------------------|--|---------------------|-------------------------------|
| <input type="checkbox"/> | | monitoring.enabled | <input type="checkbox"/> true |
| <input type="checkbox"/> | | monitoring.filters | <input type="checkbox"/> |
| <input type="checkbox"/> | | ruleset.bom.enabled | <input type="checkbox"/> true |

- ___ 7. Turn on the default monitoring filters again.
- ___ a. Click **Edit** to reopen the monitoring options, clear **Enable tracing in Decision Warehouse** and click **Save**.
 - ___ b. Again, click **Edit** to reopen the monitoring options, select **Enable tracing in Decision Warehouse** and click **Save**.

3.3. Removing BOM serialization

When the `ruleset.bom.enabled` property is set to `true`, it converts the list of objects in the ruleset parameters into a memory buffer by using BOM serialization.



Important

For large amounts of input and output data, BOM serialization can result in poor performance.

To optimize performance, you can turn off BOM serialization.

- ___ 1. If the properties section for this ruleset is closed, click **Show Properties**, and click the **Edit** icon for the `ruleset.bom.enabled` ruleset property.
- ___ 2. Set `ruleset.bom.enabled` to `false` and click the **Save** icon.



Note

You can edit only one property at a time. If you try to edit the values of more than one property, only one of the values is saved.

- ___ 3. In the **monitoring options** section, click **Edit** to reopen the monitoring options.

- ___ 4. Notice that **Native format** is the option for storing the values of ruleset parameters.

When BOM serialization is turned off, the BOM objects that are passed as parameters are stored either in XML for dynamic XOMs or in a string representation for Java XOMs.

- ___ 5. Rerun the Loan Validation application with the default values to generate a new decision trace in Decision Warehouse.

If the browser is still open, click **Try again**. Otherwise, open a browser at this URL:

<http://localhost:9080/LoanValidation>

- ___ 6. In Rule Execution Server console, open the **Decision Warehouse** tab, and click **Search** to get the latest traces.
- ___ 7. Click **View Decision details** for your latest trace and notice the format for the input and output parameters.

Input Parameters

```
borrower = Borrower( firstName: John lastName: Doe born: May 12, 1968 SSN: 123-45-6789 zip code: 06560 income: 100000 credit score: 600)  
loan = Loan( amount: 100000 starting: Jan 14, 2015 duration: 72 month LTV: 70%)
```

Output Parameters

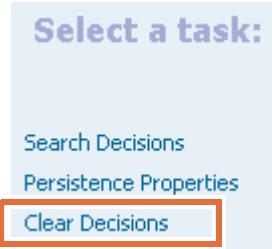
```
report = Report( validData: true approved: true score: 820 grade: B insuranceRequired: true insuranceRate: 2%  
message: Low risk loan Congratulations! Your loan has been approved )  
loan = Loan( amount: 100000 starting: Jan 14, 2015 duration: 72 month LTV: 70% rate: 5.7% monthly repayment:  
1,643.16)
```

- ___ 8. Click **View Decision details** for the previous trace and take a moment to compare the formats of the input and output parameters.
- ___ 9. Close the Decision Trace windows.

Section 4. Deleting trace information from the database

You can delete trace information from the Decision Warehouse database by specifying the ruleset paths or execution dates.

- 1. In the Rule Execution Server console, click the **Decision Warehouse** tab.
- 2. Click **Search Decisions > Search** and look at the list of decisions to see RuleApp version numbers and dates.
- 3. In the “Select a task” pane, click **Clear Decisions**.



- 4. In the Clear Decisions page, you can either leave all fields empty to delete all traces, or specify which traces to delete, according to the RuleApp versions and dates that you saw listed.

For example, if you have traces for `loanRuleApp` version 2.0, specify that version to delete.

- a. In the **Executed ruleset path** field, type: `loanRuleApp/2.0`



Information

You can leave this field empty to erase all traces. You can also specify a partial or complete ruleset path to limit which traces are erased.

- b. In the **From Date** field, click the calendar icon to specify a date, such as yesterday's date, or leave the field blank.

- ___ c. In the **To Date** field, click the calendar icon to specify a date, such as today's date, or leave the field blank.

Clear Decisions

Use the following fields to remove decision traces:

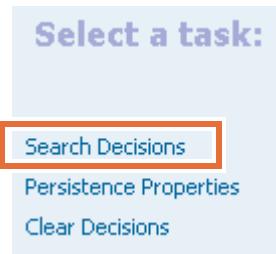
Executed ruleset path: loanRuleApp/4.0

From date: 1/13/2015

To date:

Clear

- ___ 5. Click **Clear** to delete the specified traces.
A warning message prompts you to confirm the deletion.
- ___ 6. Click **Confirm** to delete the traces.
- ___ 7. To view the remaining traces, you can click **Search Decisions** and click **Search** to verify the deletion.



End of exercise

Exercise review and wrap-up

This exercise looked at how you can use Decision Warehouse to audit ruleset executions.

Exercise 21. Working with the REST API

What this exercise is about

This exercise describes how to use the REST API for ruleset execution and resource management.

What you should be able to do

After completing this exercise, you should be able to:

- Use the REST API to test ruleset execution and manage RuleApp resources

Introduction

In this exercise, you learn how to work with the REST API.

The exercise includes these sections:

- Section 1, "Using REST services to test ruleset execution"
- Section 2, "Using REST services to deploy and execute rules"

Requirements

You should complete this exercise before proceeding:

- Exercise 16, "Exploring the Rule Execution Server console"



Hint

In this exercise, you write some code. You can use the `<LabfilesDir>\code\rest.txt` file to copy and paste the code snippets.

Section 1. Using REST services to test ruleset execution

To access REST API test tool for ruleset execution, you first open the Ruleset View page of Rule Execution Server console.

To test the ruleset execution for a ruleset:

- ___ 1. If Rule Execution Server console is closed, open it and sign in with `resAdmin` for the user name and password.
- ___ 2. Click the **Explorer** tab.
- ___ 3. In the Navigator pane, expand **RuleApps** and expand **miniloanruleapp**.
- ___ 4. Click the latest `miniloanrules` ruleset (which should be version 1.1) to open the Ruleset View page.
- ___ 5. Click **Retrieve HTDS Description File** on the toolbar.



- ___ 6. Select the **REST** option and click **Test**.

Decision service type SOAP REST

Latest ruleset version
 Latest RuleApp version
 Decision trace information
 Inline types in separate XSD files

Test



Note

If the **Latest ruleset version** and other options on this page are selected, you can ignore them as they do not affect the REST test tool.

The test tool opens in a new browser tab. You can modify the execution request to set the ruleset parameter values with test data.

7. Change these values for the borrower and loan attributes:

- **creditScore** to: 600
- **yearlyIncome**: 80000
- **amount**: 2000000
- **duration**: 240
- **yearlyInterestRate**: 0.05
- **approved**: true

The XML request should match this code:

```
<par:Request xmlns:par="http://www.ibm.com/rules/decisionservice/
Miniloanruleapp/Miniloanrules/param">
    <!--Optional:-->
    <par:DecisionID>string</par:DecisionID>
    <!--Optional:-->
    <par:borrower>
        <creditScore>600</creditScore>
        <yearlyIncome>80000</yearlyIncome>
    </par:borrower>
    <!--Optional:-->
    <par:loan>
        <amount>2000000</amount>
        <approved>true </approved>
        <duration>240</duration>
        <yearlyInterestRate>.05</yearlyInterestRate>
    </par:loan>
</par:Request>
```

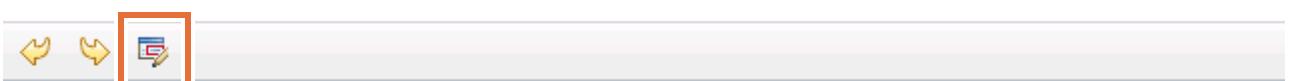


Hint

You can find this code snippet in the `<LabfilesDir>\code\rest.txt` file.

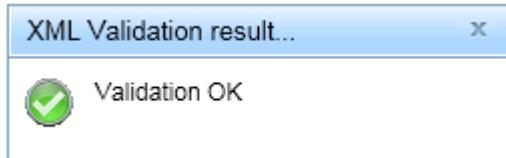
8. Click the validation icon to confirm that the XML request is correctly formatted.

Execution Request:



```
1 <par:Request xmlns:par="http://www.ibm.com/rules/decisionservice/Miniloanruleapp/Miniloanrules/param">
2     <!--Optional:-->
3     <par:DecisionID>string</par:DecisionID>
4     <!--Optional:-->
5     <par:borrower>
6         <creditScore>600</creditScore>
7         <yearlyIncome>80000</yearlyIncome>
8     </par:borrower>
```

The XML Validation result opens to show you whether the request is validated.



If you make an error in the text, the validation tool shows you where the error is so that you can correct it. Error messages are returned in JSON format.

___ 9. Close the validation result and click **Execute Request**.

___ 10. Scroll down to see the server response.

The result includes the `approved` attribute, which returns `false` for this test.

___ 11. Change the amount for loan to another lower value, such as 20000, and execute the request again; and check to see that the `approved` attribute returns `true`.

___ 12. Close the REST Service browser.

Section 2. Using REST services to deploy and execute rules



Information

This part of the exercise is based on the REST API sample that is provided with the product.

2.1. Setting up your environment for this exercise

- 1. In Rule Designer, switch to a new workspace:
 - a. From the **File** menu, click **Switch Workspace > Other**.
 - b. In the **Workspace Launcher** window, enter the path:
`<LabfilesDir>\workspaces\rest`
- 2. Close the Welcome view and switch to the Sample Console perspective.
The Samples and Tutorials page opens.
- 3. In the **Rule Execution Server** section, expand **Samples** to find the sample **REST API in Java**.



REST API in Java

[View instructions](#) [View sample commands](#) [Import projects](#)

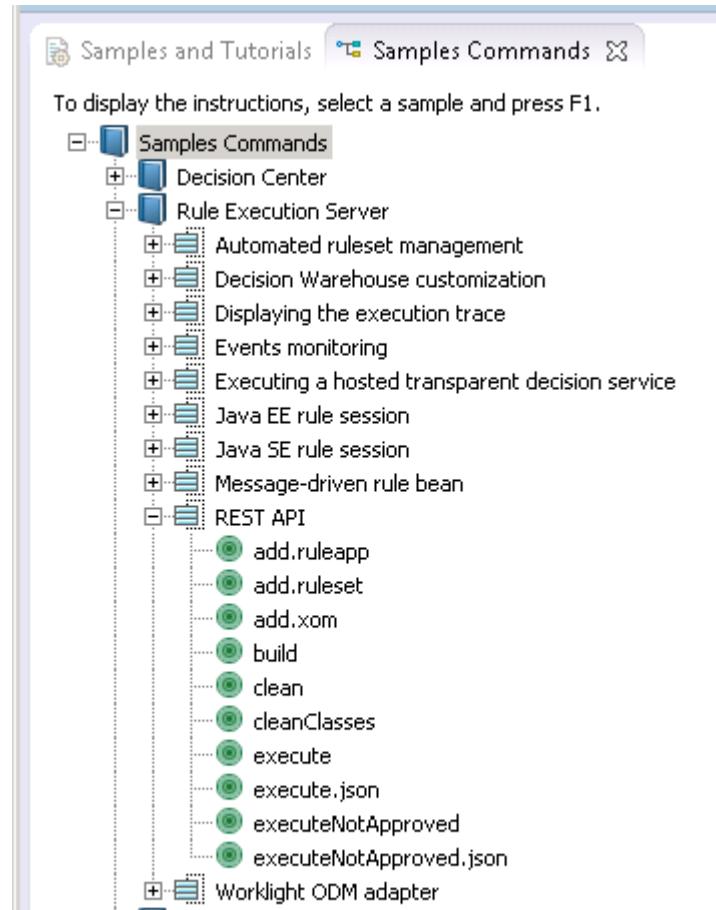
- 4. In the **REST API in Java** section, click **View sample commands** to open the Samples Commands page.



REST API in Java

[View instructions](#) [View sample commands](#) [Import projects](#)

The **Samples Commands** tab opens to the REST API sample commands.



- ___ 5. Go back to the Samples and Tutorials page and click **Import projects**.



A new workspace opens in the Java perspective with the projects that are used in this sample. You can expand the projects to view the code that was used and to better understand this sample.

2.2. Deploying the RuleApp resources

- ___ 1. Switch back to the Samples Console perspective.
 ___ 2. In the Samples Commands page, double-click the `add.ruleapp` command.

The Console view opens so you can see the results of this command, which compiles the code and completes these tasks:

- Checks whether the scenario can run
- Lists all the RuleApps
- Adds a RuleApp
- Lists all the RuleApps again to show the new one

- Modifies the RuleApp
 - Shows the modified RuleApp
- ___ 3. Verify the results of adding a RuleApp in the Rule Execution Server console.
- ___ a. Go to the browser that is running Rule Execution Server console and click the **Explorer** tab.
You might need to refresh the view (F5).
- ___ b. Notice in the RuleApps View that you can now see the newly added `myRuleApp` RuleApp.
- ___ c. Click the new RuleApp to see its description:
`My REST modified RuleApp`
The RuleApp does not contain a ruleset.
- ___ 4. Go back to the Samples Commands page in Rule Designer, and double-click the `add.xom` command.
The command deploys the XOM.
- ___ 5. Return to the **Explorer** tab in Rule Execution Server console, and expand **Resources** to see the newly added resource: `myxom.zip/1.0`
This XOM is not referenced by any rulesets.
- ___ 6. Back in the Samples Commands page, double-click the `add.ruleset` command.
- ___ 7. Return to the **Explorer** tab in Rule Execution Server console, and expand **RuleApps > myRuleApp/1.0** in the Navigator pane to see that `myRuleApp/1.0` now contains a ruleset: `myRuleset`.
- ___ 8. Click **myRuleset** to open it in the Ruleset View and see its description:
`My REST ruleset`

2.3. Testing ruleset execution

- ___ 1. Go back to the Samples Commands page in Rule Designer, and double-click the `execute` command.
The command compiles the code and processes as follows:
- Checks whether the scenario can run.
 - Passes the ruleset parameters in XML format and executes the ruleset to apply for a loan for an amount of 5000.
- In the Console view, when you see the `BUILD SUCCESSFUL` message, scroll up to see the XML response with the **approved** field of the loan output parameter set to `true`.
- ___ 2. In the Samples Commands page, double-click the `executeNotApproved` command.
This time, the amount of the requested loan is 500000 and the loan is not approved. The Console view shows the **approved** field set to `false`.

Next, because the execution trace is enabled on the ruleset, you can view the execution trace in Decision Warehouse.

2.4. Viewing execution traces in Decision Warehouse

- ___ 1. Go back to the Rule Execution Server console and click the **Decision Warehouse** tab.
- ___ 2. On the Search Decisions page, click **Search** to search for the latest decisions.

You should see two decisions that are identified as **idApproved** and **idNotApproved**. Decision **idNotApproved** rejects the loan as the result of the execution of one rule.

2.5. Cleaning the scenario

- ___ 1. Go back to the Samples Commands page in Rule Designer and double-click the `clean` command.

After you see the `BUILD SUCCESSFUL` message in the Console view of Rule Designer, scroll up to see the list of all the RuleApps, which no longer includes `myRuleApp`. You also see the list of all the resources, which no longer includes `myxom.zip`.

- ___ 2. Reopen the Rule Execution Server console and click the **Explorer** tab.

The RuleApps View does not list `myRuleApp` and the Resources View does not list `myxom.zip`.

End of exercise

Exercise review and wrap-up

The first part of the exercise looked at how you can access the REST API and generate a WADL representation of the REST API and its documentation. You also learned how to use REST services to view and manage deployed resources, and test ruleset execution.

Exercise 22. Working with decision services

What this exercise is about

This exercise shows you how to work with a decision service in Rule Designer and Decision Center with the Decision Governance Framework.

What you should be able to do

After completing this exercise, you should be able to:

- Create a decision service
- Define decision operations
- Define deployment configurations
- Manage a decision service in the Business console from release creation to deployment

Introduction

In this exercise, you work with a decision service, starting in Rule Designer. You also see how the decision service lifecycle is governed through development, testing, and deployment in Decision Center through the Decision Governance Framework.

- Section 1, "Creating decision service rule projects"
- Section 2, "Creating a main decision service rule project"
- Section 3, "Creating decision operations for the decision service"
- Section 4, "Creating and using a deployment configuration"
- Section 5, "Publishing from Rule Designer to Decision Center"
- Section 6, "Using the Decision Governance Framework in Business console"

Requirements

There are no specific requirements for this exercise.



Hint

In this exercise, you write some code. You can use the code snippets in the `<LabfilesDir>\code\decision_service.txt` file, and copy and paste them in Rule Designer.

Section 1. Creating decision service rule projects

In this exercise, you create the initial elements that are required to set up a business rule application, including a rule project, the associated XOM and BOM, and the vocabulary.

You also create a decision operation. Each operation must define a ruleset and a signature: its input and output parameters. A ruleset combines business rules into an executable container that is deployed to a target server, and the parameters are used to exchange data between a ruleset and a calling application.

Before starting, you must clean the databases that are used by Decision Center and Rule Execution Server.

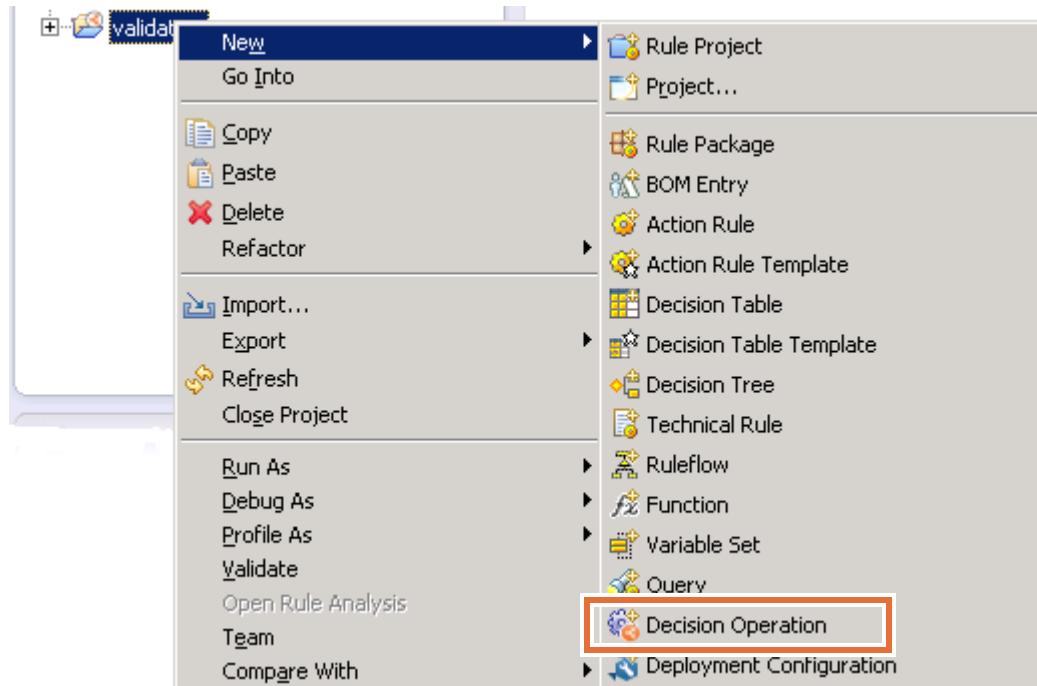
1.1. Setting up your environment for this exercise

- ___ 1. If the sample server is not running, start it now by double-clicking the shortcut on your desktop or by clicking **All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Start server**.
- ___ 2. To avoid synchronization conflicts during this exercise, delete the existing decision service from the Decision Center database.
 - ___ a. Open Decision Center Enterprise console by clicking the shortcut on your desktop or by clicking **All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Decision Center Enterprise console**.
 - ___ b. Sign in to the Enterprise console with `rtsAdmin` as the user name and password.
 - ___ c. On the **Home** tab, select **Work on decision service** and click the **Configure** tab.
 - ___ d. Under the **Administration** section, click **Erase Current Decision Service**.
 - ___ e. When prompted to confirm deletion of the decision service and included projects, click **Yes**.
 - ___ f. Sign out of the Enterprise console.
- ___ 3. To avoid deployment issues during this exercise, remove the `loanvalidation-xom` resources from the Rule Execution Server database.
 - ___ a. Open Rule Execution Server console by double-clicking the shortcut on your desktop or by clicking **All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Rule Execution Server console**.
 - ___ b. Sign in to the console with `resAdmin` as the user name and password.
 - ___ c. Click the **Explorer** tab and click **Resources**.
 - ___ d. Select **loanvalidation-xom.zip**, and click **Remove**.
 - ___ e. When prompted to confirm removal of this resource, click **Confirm**.
- ___ 4. If Rule Designer is closed, open it by clicking **All Programs > IBM > Operational Decision Manager V8.7 > Rule Designer**.

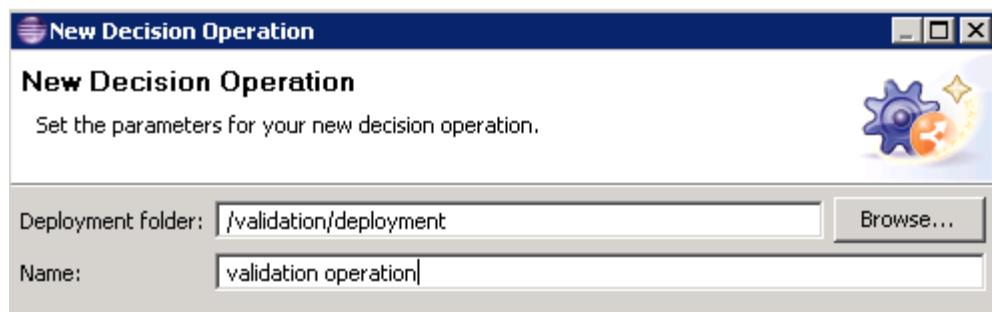
- ___ 5. In Rule Designer, switch to a new workspace by following these steps:
 - ___ a. From the **File** menu, click **Switch Workspace > Other**.
 - ___ b. In the **Workspace Launcher** window, enter the path:
C:\labfiles\workspaces\decision_service
 - ___ 6. Close the Welcome view and switch to the Samples Console perspective.
 - ___ a. Click the Open Perspective icon in the upper-right part of the Eclipse window.
 - ___ b. Select **Samples Console**, and click **OK**.
 - ___ 7. Select **Rule Designer > Training > Ex 22: Working on decision services > 01-start > Import projects**.
- The Rule perspective opens.
- ___ 8. Close the Help view.

1.2. Creating a decision operation

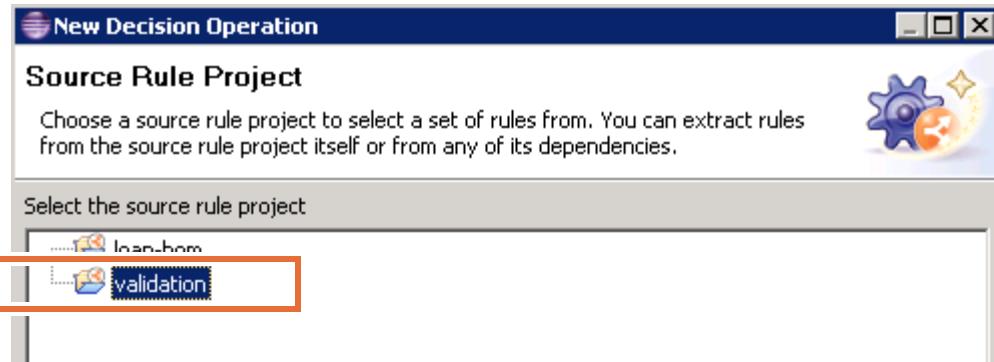
- ___ 1. In Rule Explorer, right-click **validation** and click **New > Decision Operation**.



- ___ 2. Enter validation operation as the name of the operation.



- ___ 3. Click **Next** and select **validation**.



- ___ 4. Click **Finish**. The Decision Operation editor opens.

Two things to note about the decision operation:

- The operation does not use a ruleflow.
- No query is selected.

- ___ 5. Define the input and output parameters to the decision operation to exchange information with a calling application.
- ___ a. Open the **Signature** tab (alternatively, click **Define** under the Signature section on the Overview page).

- ___ b. Expand **loan-bom > loanvalidation-parameters**.

- ___ c. Drag **borrower** to **Input Parameters**, and drag **report** to **Output Parameters**.

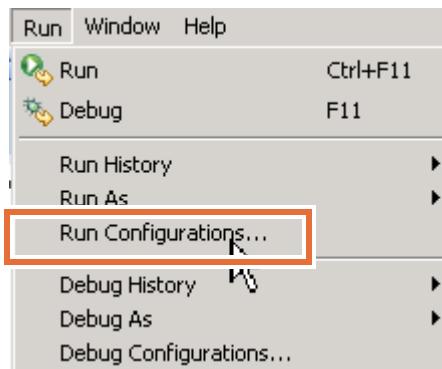
Decision Operation Signature - validation operation

The screenshot shows the 'Decision Operation Signature' interface. On the left, there's a tree view under 'Eligible variables' showing 'ds loan-bom' with a child 'loanvalidation-parameters' containing 'borrower' and 'report'. A toolbar below the tree has 'Refresh' and 'Add as ruleset parameter' buttons. The right side is divided into three sections: 'Input Parameters' (highlighted with a red box), 'Input - Output Parameters' (unhighlighted), and 'Output Parameters' (highlighted with a red box). The 'Input Parameters' section contains one row: Parameter name 'borrower', Verbalization 'the borrower', and Type 'loan'. The 'Output Parameters' section contains one row: Parameter name 'report', Verbalization 'the loan report', and Type 'loan'.

- ___ 6. Press Ctrl+S to save your work.

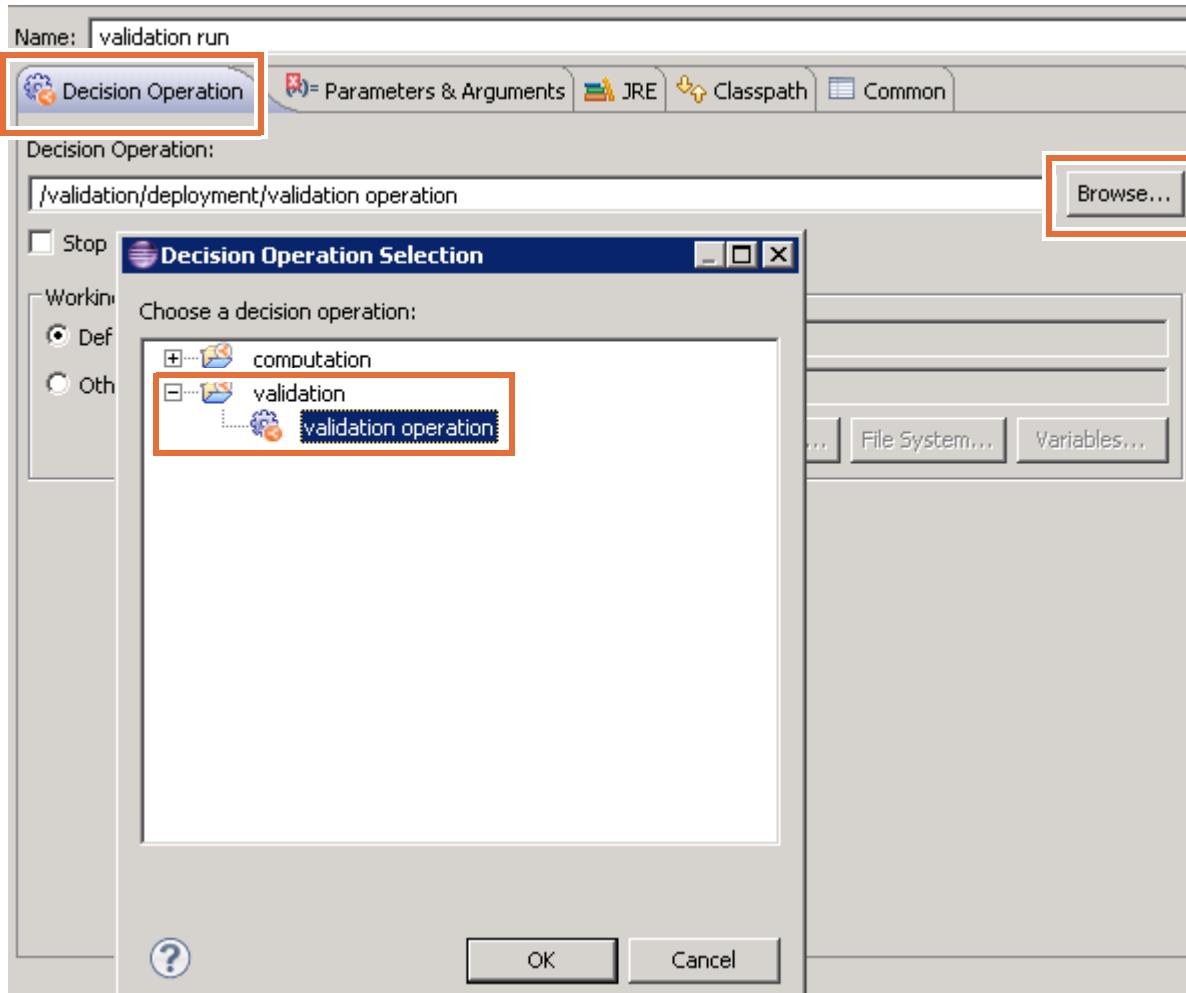
1.3. Creating a run configuration for the decision operation

- ___ 1. From the **Run** menu, click **Run Configurations**.



- ___ 2. In the list of configurations, right-click **Decision Operation** and click **New**.
 ___ 3. Enter the name **validation run** as the name of the run configuration.

- ___ 4. In the **Decision Operation** field of the **Decision Operation** tab, click **Browse**, click **validation operation**, and click **OK**.



- ___ 5. Click **Apply** to commit your changes.
 ___ 6. Define the parameter value for the input parameter.
 ___ a. Click the **Parameters & Arguments** tab.



- ___ b. Select **borrower** and click **Edit Value** to define the parameters.

- c. Select **Function body**, and replace the code with the following text:

```
java.util.Date birthDate = loan.DateUtil.makeDate(1950,1,1);
loan.Borrower borrower = new
loan.Borrower("Smith", "John", birthDate, "123121234");
borrower.zipCode = "12345";
borrower.creditScore = 200;
borrower.yearlyIncome = 20000;
return borrower;
```



Hint

You can use the code snippets in the `<LabfilesDir>\code\decision_service.txt` file, and copy and paste them in Rule Designer.

- d. Click **OK**

7. Click **Apply** to apply your changes.
 8. Click **Run** to run the configuration.

You can see the decision operation results in the console:

```
The borrower's SSN is well formatted.
The borrower's Zip Code is valid.
The borrower's age is valid.
The borrower's SSN area number belongs to an authorized area.
The borrower's name is not empty.
```

A screenshot of a Java console window titled 'Console'. The window shows the output of a validation run. The text in the console is as follows:

```
<terminated> validation run [Decision Operation] C:\Program Files\IBM\ODM86\ODM\jdk\bin\javaw.exe {
| The borrower's SSN is well formatted.
| The borrower's Zip Code is valid.
| The borrower's age is valid.
| The borrower's SSN area number belongs to an authorized area.
| The borrower's name is not empty.
```

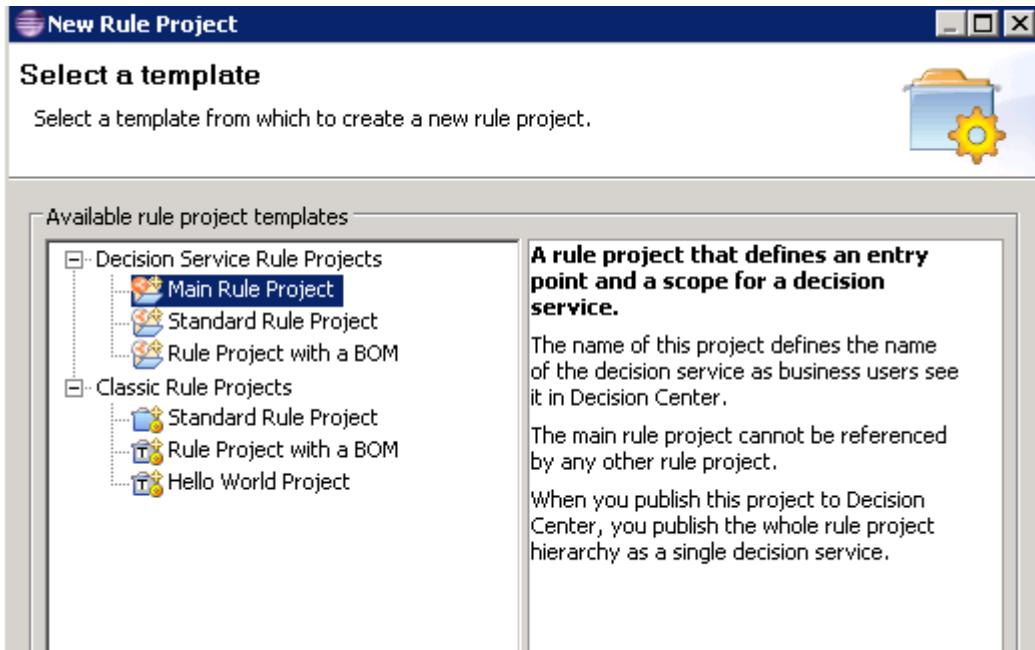
9. Close the validation operation window.

Section 2. Creating a main decision service rule project

In this section, you define the main rule project as the top-level project for the decision service.

— 1. Create the main rule project for the decision service:

- a. Click **File > New > Rule Project**.
- b. Select **Main Rule Project**, and click **Next**.

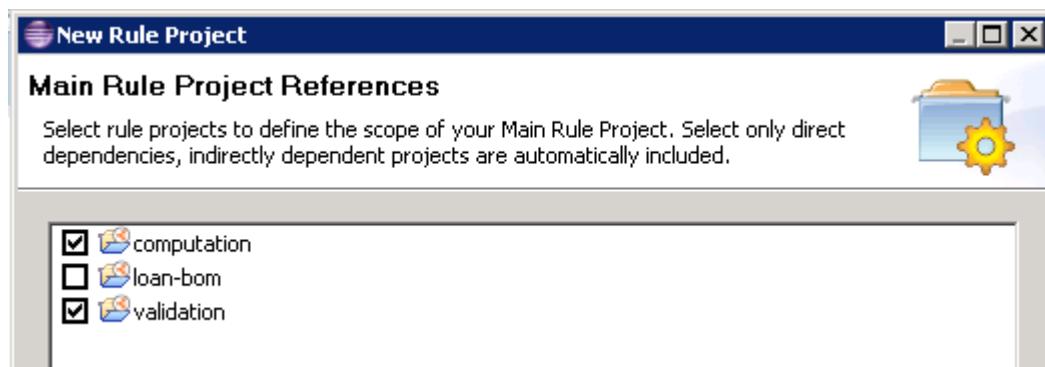


— c. Enter **Loan service** as the name of the main rule project.

— d. Click **Next**.

— e. Select the following rule projects to be referenced:

- computation
- validation



— f. Click **Finish**.

The project opens in the Rule Explorer.

— 2. Set the **Loan service** project properties.

- a. Right-click **Loan service**, and click **Properties**.

- ___ b. Click **Decision Service** and make sure that **Main Rule Project** is selected.



Information

After you create a decision service main rule project, you can later change its type to become a standard rule project. You can also upgrade a standard rule project to become the main rule project. Changing a project can require changes to the way projects reference each other, and can affect the deployment and synchronization of the project.

- ___ c. Click **Project References** and make sure that **computation** and **validation** are selected.
 ___ d. Click **Rule Engine** and make sure that **Classic rule engine** is selected.



Information

All the projects in the decision service must use the same rule engine.

- ___ e. Click **Java Execution Object Model** and select **loanvalidation-xom**.
 ___ f. Click **OK**.
 ___ 3. Import business rules that determine the insurance requirements and pricing for the loan.
 ___ a. Right-click **Loan service > rules**, and click **Import**.
 ___ b. Select **General > File System**, and click **Next**.
 ___ c. Click **Browse** and go to **<LabfilesDir>/code/decision service/loan-service-rules**, and click **OK**.

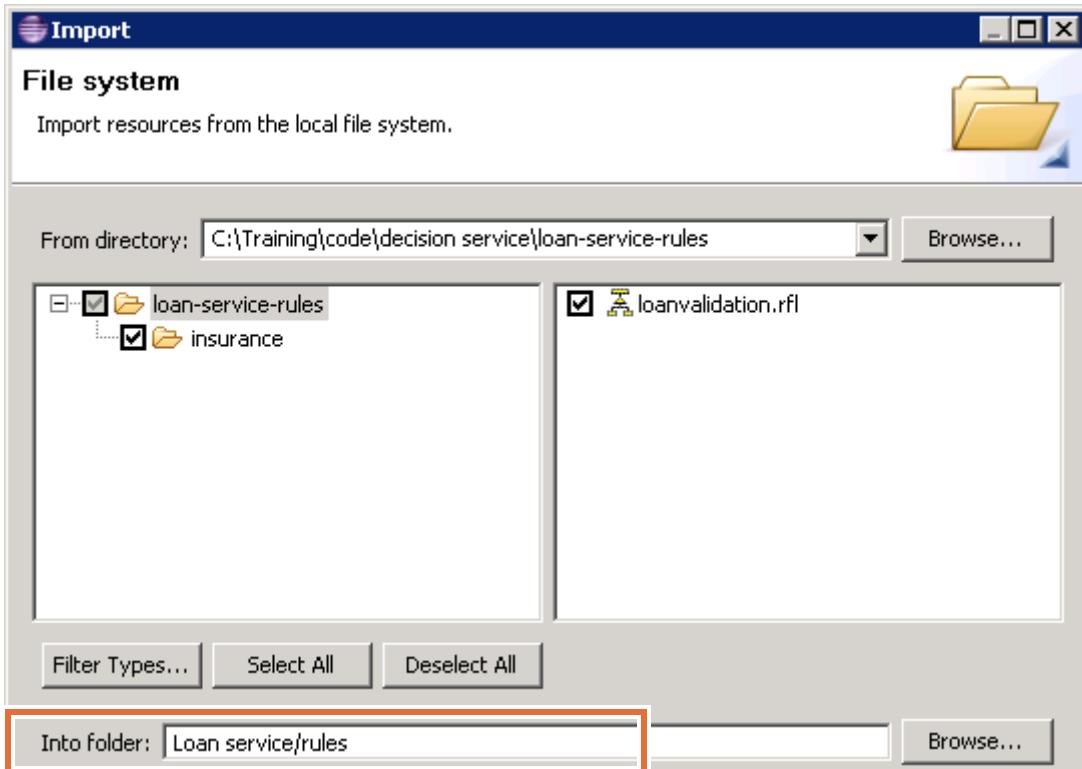


Note

By default, **<LabfilesDir>** is: C:\labfiles

- ___ d. Expand **loan-service-rules** and select **insurance** and **loanvalidation.rfl**.

- __ e. Make sure that the **Into folder** field points to Loan service/rules.



- __ f. Click **Finish** to import the rules.

Section 3. Creating decision operations for the decision service

In this task, you create a decision operation for the full decision service.

A decision service can contain more than one decision operation. You can use more than one decision operation when you deploy from a decision service.

3.1. Creating a decision operation

- ___ 1. Define the decision operation for the main rule project.
 - ___ a. Right-click **Loan service > deployment** and click **New > Decision Operation**.
 - ___ b. Enter **all rules** operation as the name of the decision operation, and click **Next**.
 - ___ c. Select **Loan service** and click **Next**.
 - ___ d. Expand **Loan service > rules** and select the **loanvalidation.rfi** ruleflow, and click **Finish**.

- ___ 2. Define the operation signature.
 - ___ a. Click the **Signature** tab (or click **Define** in the Signature section).
 - ___ b. In the **Eligible variables** section:
 - Expand **loan-bom > loanvalidation-parameters**.
 - Drag **borrower** to **Input Parameter**.
 - Drag **report** to **Output Parameters**.
 - ___ c. Expand **computation > loan variables** and drag **loan** to **Input-Output Parameters**.

| Parameter name | Verbalization |
|----------------|---------------|
| borrower | the borrower |

| Parameter name | Verbalization |
|----------------|---------------|
| loan | the loan |

| Parameter name | Verbalization |
|----------------|-----------------|
| report | the loan report |

- __ d. Save your work (Ctrl+S).

**Note**

If you do not save your work after defining the ruleset signature, these parameters are not available as part of the decision operation when you prepare the run configuration.

- __ 3. Create run configurations for the decision operations:

- __ a. From the **Run** menu, click **Run Configurations**.
- __ b. Right-click **Decision Operation** and click **New**.
- __ c. For the configuration name, enter: `all rules run`.
- __ d. For the **Decision Operation** field, click **Browse** and select **Loan service > all rules operation**, and then click **OK**.
- __ e. Click the **Parameters & Arguments** tab and define the function body for the parameter values as follows:

- borrower:

```
java.util.Date birthDate = loan.DateUtil.makeDate(1950,1,1);
loan.Borrower borrower = new
loan.Borrower("Smith", "John", birthDate, "123121234");
borrower.zipCode = "12345";
borrower.creditScore = 200;
borrower.yearlyIncome = 20000;
return borrower;
```

- loan:

```
java.util.Date loanDate = new java.util.Date();
loan.Loan loan = new loan.Loan(loanDate, 48, 100000, 1.2);
return loan;
```

**Hint**

You can use the code snippets in the `<LabfilesDir>\code\decision_service.txt` file, and copy and paste them in Rule Designer.

- __ f. Click **Apply**, and then click **Run**.

The Console shows the following results:

The borrower's age is valid.

The borrower's name is not empty.

The borrower's SSN area number belongs to an authorized area.

The borrower's SSN is well formatted.

The borrower's Zip Code is valid.

monthly repayment calculated.

Borrower: First name Smith Last name John born Feb 1, 1950 SSN 123-12-1234

- Zipcode 12345

- Yearly income 20,000

- Credit Score 200

Loan: Amount 100,000 Starting Jan 14, 2015 Duration 48 month(es) Loan to value 120%

- Rate 0.055

- Monthly repayment 2,325.65

Report: Valid data true Approved false score 300

- Grade C

- Message Average risk loan

Too big Debt/Income ratio: 1.40

We are sorry. Your loan has not been approved

The decision operations reference rules from all the projects that are associated with the decision service. For example, the **checkName** rule is in the **validation** project, and returns the result: "The borrower's name is not empty."

- 4. Close the all rules operation window.

Section 4. Creating and using a deployment configuration

You create a deployment configuration for the decision service and use it to deploy rules to a target server.

In this task, you create a deployment configuration for the decision service. Then, you use the configuration to deploy a RuleApp that contains the rulesets that are defined by the decision operations. After deployment, you test the rulesets by using the Rule Execution Server console on the target server.

4.1. Creating a deployment configuration

- ___ 1. In Rule Explorer, expand the **loan-bom** project, and in the **resources** folder, delete the **META-INF** folder.
- ___ 2. Click **Yes** to confirm the deletion.



Important

When you publish a decision service to Decision Center for testing from Business console, the XOM resource must be associated with the decision service so that Rule Execution Server can access it. The `deployment.xml` file in the `resources/META-INF` folder tells Rule Execution Server where to look for the XOM. To make sure that the `deployment.xml` file is created in the main decision service project, you associated the XOM to the `Loan service` project during Section 2, "Creating a main decision service rule project".

When you deploy, the `deployment.xml` file is created for the `Loan service` project and re-created in the `loan-bom` project.

-
- ___ 3. Right-click **Loan service** and click **New > Deployment Configuration**.
 - ___ 4. Enter `main` deployment configuration as the name of the deployment configuration, and click **Finish**.

The Deployment Overview opens and you see the available tabs for this editor.

Deployment, Overview - main deployment configuration The deploym... operation.

General

Name: main deployment configuration

Type: Production Nonproduction Deploy the XOM: Yes No

Description:

RuleApp

Configure the RuleApp to be created upon deployment.

RuleApp Name: main_deployment_configuration

RuleApp Base Version: 1.0

RuleApp Properties:

| Name | Value |
|------|-------|
| | |

Target Servers

Define target servers for RuleApp deployment:

Deployment

[Proceed to RuleApp deployment.](#)

Overview Decision Operations Target Servers Source

- ___ 5. Click the **Decision Operations** tab (or you can click **Include decision operations**).
- ___ 6. Click the plus sign under **Configured Decision Operations**.

Configured Decision Operations

Select and configure decision operations for RuleApp deployment:

- ___ 7. Click **Select existing decision operations**.
- ___ 8. Expand **Loan service** and select **all rules operation**.
- ___ 9. Click **Finish**.

By default, the ruleset version policy is set to **Increment minor version numbers**, which retains the major version number and updates the minor version number. The version policy makes the new version available and retains previous versions so that you can switch to another version in the Rule Execution Server console.

4.2. Setting the target server

- ___ 1. Click the **Overview** tab.

- __ 2. In the **Deploy the XOM** section, make sure that **Yes** is selected to deploy the XOM with the decision service.

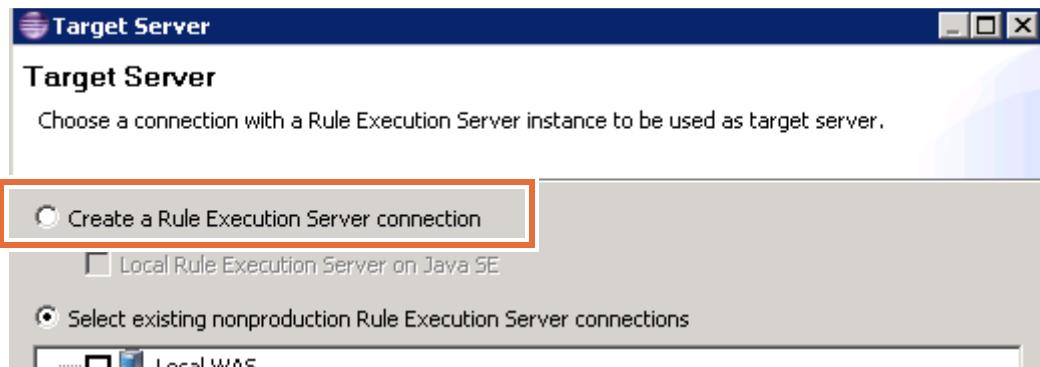


- __ 3. Click the **Target Servers** tab (or click the **Define target servers** link).
- __ 4. In the **Deployment Locations** section, click the plus sign.



In the Target Server window, you can define a new server or use the predefined servers. For this exercise, you use the predefined sample server.

- __ 5. In the Target Server window, make sure that **Create a Rule Execution Server Connection** is selected and click **Next**.



- __ 6. Enter the following information:

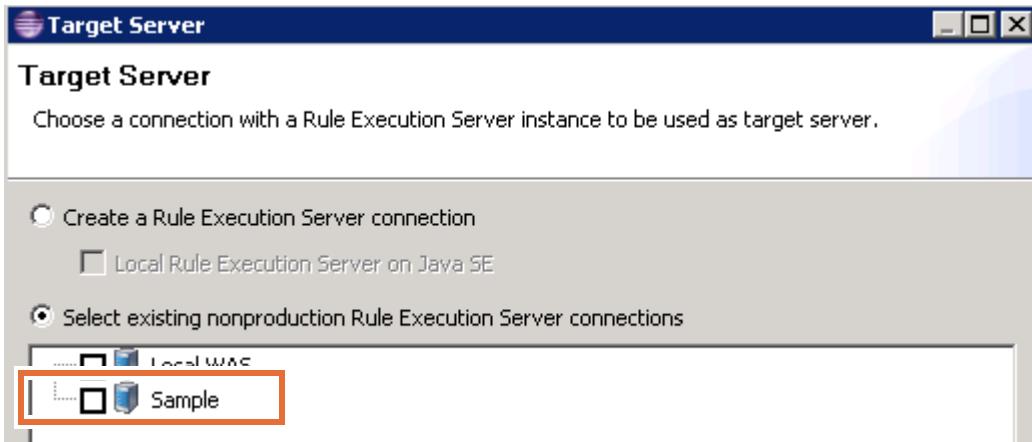
- **Name:** Sample
- **URL:** <http://localhost:9080/res>
- **User name:** resAdmin
- **Password:** resAdmin

- __ 7. Select **Save my login information**, and click **Finish**.



Troubleshooting

If the Sample server is already defined, you can select it directly by clicking **Select existing nonproduction Rule Execution Server connections** and selecting **Sample**.



- ___ 8. Save your work (Ctrl+Shift+S).

4.3. Deploying the decision service

The deployment configuration is set to deploy the decision service to the Operational Decision Manager sample server.

- ___ 1. Click the **Overview** tab.
- ___ 2. Click **Proceed to RuleApp deployment**.

Deployment

[Proceed to RuleApp deployment...](#)



Troubleshooting

If you receive a Java version notification warning, click **Do not show this message again**, and click **OK**.

- ___ 3. Review the Deployment Summary, which lists the operations to be deployed, the target server, and the version policy, and click **Next**.
- ___ 4. Review the Credentials table, which lists the target server and login information, and click **Next**.
- ___ 5. Review the Version Summary, which shows the rulesets to be packaged in the RuleApp archive for deployment, and click **Finish**.

The progress of the deployment is shown at the bottom of editor.

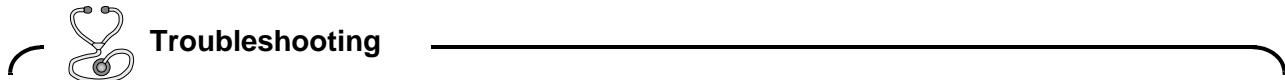
A report opens to show the results of the deployment. It lists the items that were deployed, and it states whether the deployment was successful.

- ___ 6. Close the Deployment Report and the main deployment configuration windows.

4.4. Deploying the XOM for testing

To make sure that the XOM is accessible when you run tests from the Decision Center Business console, you can deploy the XOM directly for this purpose.

- ___ 1. Right-click **Loan service** and click **Decision Validation Services > Deploy XOM for testing from the Business console**.



If you are prompted to choose a configuration, use the main deployment configuration and click **Next**.

- ___ 2. Review the XOM deployment summary and click **Next**.
- ___ 3. Verify the credentials and click **Finish**.

The resources folder of the Loan service project now includes the `deployment.xml` file.

4.5. Verifying deployment in Rule Execution Server console

- ___ 1. View the decision service in the Rule Execution Server console:
 - ___ a. If the Rule Execution Server console is not open, open it by double-clicking the shortcut on your desktop. You can also open Rule Execution Server console by going to **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Rule Execution Server Console**.
 - ___ b. Sign in with the following information:
 - **User name:** resAdmin
 - **Password:** resAdmin
 - ___ c. View the deployed RuleApp
 - ___ d. Click the **Explorer** tab to view the RuleApps.
 - ___ e. Click **main_deployment_configuration**.

The RuleApp View opens in the RuleApp View and lists the `all_rules_operation` ruleset version 1.0.

- ___ 2. Test the ruleset in the Rule Execution Server console:
 - ___ a. Click **all_rules_operation** to open it in the Ruleset View.
 - ___ b. On the toolbar, click **Test Ruleset**.

The Test Ruleset View displays the input parameters.

- ___ c. Change the values for the input parameters to the values listed here by clicking the **Edit** icon and saving your changes by clicking the Save icon.

- borrower:

```
java.util.Date birthDate = loan.DateUtil.makeDate(1950,1,1);
loan.Borrower borrower = new
loan.Borrower("Smith", "John", birthDate, "123121234");
borrower.zipCode = "12345";
borrower.creditScore = 200;
borrower.yearlyIncome = 20000;
```

- loan:

```
java.util.Date loanDate = new java.util.Date();
loan.Loan loan = new loan.Loan(loanDate, 48, 100000, 1.2);
```



Hint

You can use the code snippets in the `<LabfilesDir>\code\decision_service.txt` file, and copy and paste them in Rule Designer.

- ___ 3. Click **Execute** to test the ruleset.

After execution finishes, the results are shown at the bottom of the Test Ruleset View. If the rulesets run correctly, they produce the same results that are shown in Section 3, "Creating decision operations for the decision service".

Section 5. Publishing from Rule Designer to Decision Center

In this part of the exercise, you publish the rule project from Rule Designer to Decision Center.

- __ 1. Go back to Rule Designer, and publish the **Loan service** project to Decision Center.
 - __ a. In the Rule Explorer, right-click the **Loan service** project and click **Decision Center > Connect**.
 - __ b. Enter the connection information:
 - **URL:** `http://localhost:9080/teamserver`
 - **User name:** `rtsAdmin`
 - **Password:** `rtsAdmin`



Note

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

- __ c. Click **Connect**.
- __ 2. Click **Next**.
- __ 3. In the Synchronization Settings, select **Use Decision Governance Framework** and click **Next**.

The framework setting creates a new decision service in the Decision Center Business console Library. The project contains an initial release in the **Releases** tab and a main branch in the **Branches** tab.

If you do not select this option, a decision service is created with only a main branch in the **Branches** tab and no initial release in the **Releases** tab.

The Decision Service Dependent Projects page lists the projects that are referenced by the main **Loan service** project.



Note

All the dependent projects must be published to Decision Center to create an initial release. In subsequent synchronizations, you can publish individual projects.

- __ 4. Click **Finish**.
- __ 5. You should not have synchronization errors so you can click **No** when prompted to switch to the Team Synchronizing perspective, and close the Synchronization “no changes found” window.



Troubleshooting

If you have synchronization issues on the loan-bom project, you can ignore them for this exercise. Your decision service should be correctly deployed.

- 6. Close Rule Designer.

Section 6. Using the Decision Governance Framework in Business console

In this section, you take on the roles of Paul the project owner, Abu the tester, and Bea the rule author. By using different roles, you can see how different business users manage the decision service within the Decision Governance Framework. You see the management of decision service lifecycle from creation of a release through to deployment.

6.1. Project management: Paul

- 1. Open Decision Center Business console.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.7 > Sample server > Decision Center Business console** or type the following URL into a browser:
`http://localhost:9080/decisioncenter`



Troubleshooting

Make sure that you use Mozilla Firefox as the browser.

- b. Sign in by using `Paul` as both the **User name** and the **Password**.



Note

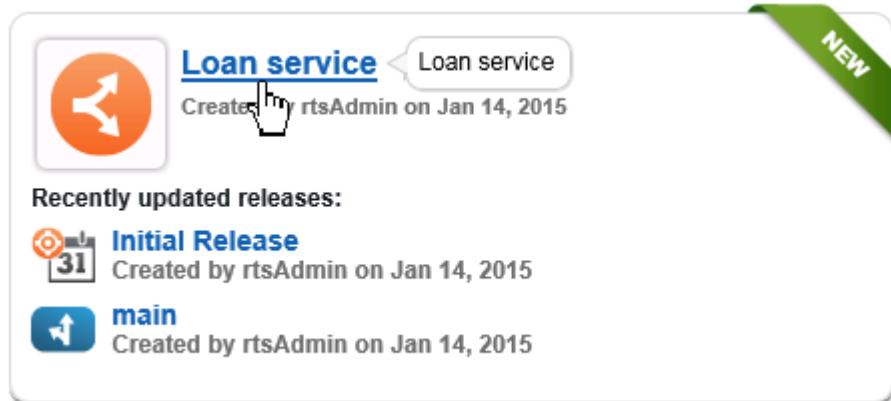
If you are already signed in as `rtsAdmin`, make sure you sign out, and sign back in as Paul.

- 2. Click **Library**, and note that **Loan service** is listed on the Decision Services page.

The screenshot shows the Decision Center interface. At the top, there is a navigation bar with tabs: HOME, LIBRARY (which is highlighted in yellow), and WORK. Below the navigation bar, there are two main sections: 'Decision Services' and 'Classic Rule Projects'. Under 'Decision Services', there is a search bar with fields for 'Date' and 'Name'. A card for a 'Loan service' is listed, featuring an orange icon with a white arrow, the text 'Loan service', and the date 'Created by rtsAdmin on Jan 14, 2015'. A green diagonal badge labeled 'NEW' is positioned in the top right corner of the card.

6.2. Creating a release: Paul

- ___ 1. Hover your mouse on **Loan service** so the Releases section opens, and notice the existing releases.



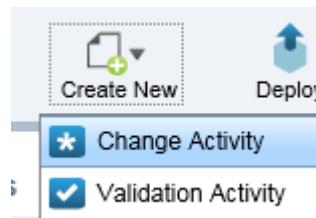
- ___ 2. Click **Loan service** to open the **Releases** page.
- ___ 3. On the **Releases** page, click the plus sign to create a new release.
- ___ 4. In the “Create a Release” window, define the release details.
 - ___ a. Name your release: Training Release
 - ___ b. Set **Owner** to **Paul**.
 - ___ c. Set **Approvers** to **Paul**.
 - ___ d. Click **Create**.

The release page opens to the **Activities** tab.

As the release owner, Paul is responsible for creating activities through which edits can be made to the release.

6.3. Creating a Change Activity: Paul

- ___ 1. In the Training Release page, with the **Activities** tab open, click the **Create New** icon on the toolbar and choose **Change Activity**.



- ___ 2. Define the change activity details in the creation window:
 - ___ a. Enter the activity name: Update minimum age
 - ___ b. Set **Owner** and **Approvers** to **Paul**.
 - ___ c. Set **Authors** to **Bea**.

- __ d. Click **Create**.

The screenshot shows a dialog box titled "Create a Change Activity". At the top, there is a text input field labeled "Give your activity a name:" containing the placeholder "Update minimum age". Below this is a rich text editor toolbar with various styling options like bold (B), italic (I), underline (U), and alignment tools. A large text area for entering activity details is present but empty. Underneath, there is a section for specifying a due date, with a text input field showing "Jan 28, 2015" and a calendar icon. Further down, there are sections for defining participants: "Owner" set to "Paul", "Approvers" also set to "Paul" (with green and red plus/minus buttons), and "Authors" set to "Bea" (also with green and red plus/minus buttons). At the bottom right of the dialog is a blue "Create" button.

6.4. Creating a Validation Activity: Paul

- __ 1. Return to the **Activities** tab of the release page, click the **Create New** icon on the toolbar and choose **Validation Activity**.
- __ 2. In the validation activity creation window, define the activity details.
 - __ a. Enter the activity name: Test minimum age
 - __ b. Set **Owner** and **Approvers** to **Paul**.
 - __ c. Set **Testers** to **Abu**.
 - __ d. Click **Create**.
- __ 3. Sign out as Paul.

6.5. Validation: Abu

- ___ 1. Sign in as Abu, with Abu as the user name and password.
- ___ 2. Click **Work** on the console menu to see the activities that are assigned.
- ___ 3. Click the **Test minimum age** link to open the activity page.



My activities

The following items require your attention.

The screenshot shows a list of activities under the heading 'Due Later (1)'. One item, 'Test minimum age', is highlighted with a red box. Below it, the text 'In Loan service > Training Release' is visible. To the right, there is a status bar showing 'Due Date: Jan 28, 2015' and 'Owner: Paul'.

6.6. Generating your own scenario file

You can generate the scenario file in the Business console.

- ___ 1. Click the **Generate Scenario File** icon to create a test suite.

The screenshot shows the 'Test Suites' screen. At the top, there are tabs for 'Test Suites' and 'Test Plans'. Below them, a button labeled 'Generate Scenario File' is visible. A large green plus sign icon is highlighted with a red box. The table below lists test suites with columns for 'Name', 'Operation', and 'Status'.

| Name | Operation | Status |
|------|-----------|--------|
| | | |

2. In the **Operation** field, select **all rules operation** and click **OK**.

Select an operation

validation operation

loanvalidation-rulesDecision

all rules operation

all rules operation

Stored in: Loan service

Input Parameters: the borrower

Input/Output Parameters: the loan

Output Parameters: the loan report

Rules scoped to: Loan service

Ruleset Name: all_rules_operation

Main Ruleflow: loanvalidation

Business Rule Content: Validator: Default Validator

In the scenario file, you can select which BOM members to test and define expected results. In the **Tests** section, you choose at least one member, and the operator to use.

 **Note**

You must choose at least one parameter from the list.

3. Expand the **loan report**, you can select **approved** and the **equals** operator.

***Tests:**

| Field | Operator |
|--|---------------------------------------|
| <input type="checkbox"/> the loan | |
| <input type="checkbox"/> the loan report | |
| <input type="checkbox"/> borrower | |
| <input type="checkbox"/> loan | |
| <input checked="" type="checkbox"/> approved | <input type="button" value="equals"/> |

In your scenario, you can then set the expected result to true or false.

The choice for **Expected execution details to include in scenario file** section is optional.

Generally, you are encouraged to select more fields in both the Tests and Expected execution details sections to generate different scenario files. This way, you can better examine and understand how to use the scenario file to test rules and rule execution.

- 4. Click the **Download** icon on the upper-right side of toolbar, and when prompted, save the file to a temporary location, such as the desktop.



Note

If you are using Internet Explorer, the file might be saved by default to the **Downloads** folder, depending on which preferences settings were defined.

- 5. Open the file in Excel Viewer to see which fields you would need to complete with values.
- 6. Close the file.

For this exercise, you use a prepopulated scenario file.

6.7. Creating the test suite: Abu

- 1. Return to the **Tests** page by clicking the **Test minimum age** breadcrumb.
- 2. On the Test Suites page, click the plus sign (+) to create a test suite.

- 3. If you are prompted to select an operation, select **all rules operation** and click **OK**. However, your previous selection of the all rules operation for the scenario file might be reused for creation of this test suite.
The test suite opens.
- 4. In the **Name** field, change the name to **Test minimum age**.

- ___ 5. In the File to use section, click **Choose**.

File to use:
None selected [Choose...](#)

- ___ 6. Browse to the <LabfilesDir>/code/decision service folder, select the abutest.xlsx file, and click **Open**.

This scenario file is populated with two scenarios specific to test the checkAge rule: one for approval and one for rejection.

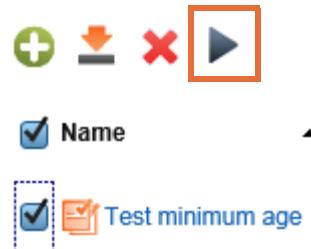
- ___ 7. Click **Save**.

The screenshot shows the IBM Decision Center interface. At the top, there are tabs for 'Decision Center', 'HOME', 'LIBRARY' (which is selected), and 'WORK'. The title bar indicates 'Loan service > Training Release > Test minimum age > New Test Suite'. On the right side, there are buttons for 'Save', 'Save as' (which is highlighted with a red box), and 'Run'. The main form contains fields for 'Name' (set to 'Test minimum age'), 'Operation' (set to 'all rules operation'), 'Server' (set to 'Sample'), and 'Decimal Precision' (set to 'All decimals'). Under the 'Scenarios' tab, 'File format' is set to 'Excel 2007 (tabbed)' and 'File to use' is set to 'abutest.xlsx' with a '(Updated)' status. Under the 'Report' tab, 'Report name' is set to 'Report'. A section titled 'Expected execution details to include in report:' lists various options like 'The number of rules fired', 'The number of executed ruleflow tasks', etc., with several checkboxes checked. At the bottom left, there is a back arrow.

- ___ 8. Click **Create New Version**.

You can select the test suite and see that the **Run** icon is enabled in the toolbar. You can also hover your mouse beside the test suite name to access the **Run** icon.

- 9. Run the test suite.
- a. Select the test suite and click the **Run** icon.



- b. When the “Run Test suite” confirmation window opens, click **OK**.
 - 10. When the test finishes, click the **Report** link to view the results.
- The report shows a 50% success rate, which confirms that changes to the `checkAge` rule are required. Abu will rerun his tests after the rules are edited.
- 11. Log out as Abu.

6.8. Rule authoring: Bea

For this part of the exercise, you do a simple rule change.

- 1. Sign in as Bea and switch to the **Work** page.
- The change activity is shown under Bea’s “My activities” page on the **Work** tab.
- 2. Click **Update minimum age** to load the activity page.
 - 3. Edit the minimum age in the `checkAge` rule to 21.
- a. On the **Rules** tab of the activity page, switch to **validation** in the **Project** list.

| Rules | Tests | Simulations | Deployments | Snapshots |
|------------------------------|-----------------------|-------------|-------------|-----------|
| Project: Loan service | Name Folder Filter: | | | |
| computation | | | | |
| Loan service | | | | |
| loan-bom | | | | |
| validation | | | | |

- b. Expand **validation > borrower** and click **checkAge** to open the rule.
- c. On the toolbar, click **Edit**.
- d. Change the minimum age from 0 to 21, and click **Save**.
- e. Click **Create New Version**.

- ___ 4. Return to the activities page by clicking the **Update minimum age** breadcrumb.



Bea can test her change by creating her own test suite.

- ___ 5. Click the **Tests** tab and create and run a test suite by following the steps in Section 6.7, "Creating the test suite: Abu".

To avoid file conflicts, use a different name for the test suite, such as: Test minimum age - Bea

The test is 100% successful, so the rule edit was correct. Bea first marks the activity as complete.

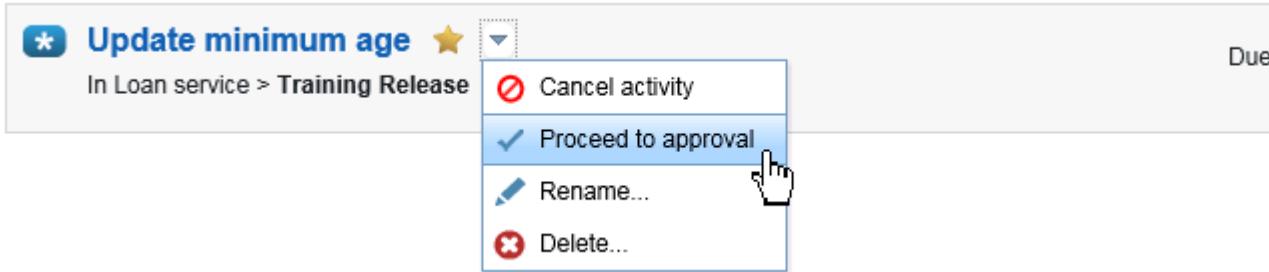
- ___ 6. Return to the "Update minimum age" page by clicking the breadcrumb.
 ___ 7. On the **Change Activity** pane, in the **Authors** section, click **Working** and select **Finish working**.

Bea's status changes to **Finished**.

- ___ 8. Sign out as Bea.

6.9. Approving the change activity: Paul

- ___ 1. Sign in as Paul, and switch to the **Work** tab.
 ___ 2. Click the **Update minimum age** activity.
 ___ 3. On the **Reports** tab of the **Tests** page, Paul can look at the report from the latest test run or rerun the test himself to verify that Bea's change was implemented correctly.
 ___ 4. Return to the **Work** page.
 ___ 5. Click the action menu beside the **Update minimum age** activity link, and click **Proceed to approval**.



- ___ 6. Click the activity link to open the Change Activity pane.

The **Status** field is set to **Ready for Approval**.

- ___ 7. In the Approvers section, beside Paul's name, click **Not Reviewed** and select **Approve changes**.

The screenshot shows the 'Change Activity' pane. At the top, it says 'Status Ready for Approval' and 'Owner Paul'. Below that, there's a section titled 'Approvers' which lists 'Paul' with a dropdown menu. The dropdown menu has 'Not Reviewed' (which is currently selected and highlighted with a blue background) and 'Approve changes' (which is highlighted with a red border). There's also a 'Reject changes' option. Below the approvers section, there's another section titled 'Authors' which lists 'Bea' with a status of 'Finished' and a green checkmark icon.

- ___ 8. When prompted to confirm, click **Approve**.

In the Change Activity pane, the **Status** is now set to **Complete**.

- ___ 9. Return to the release page to see the status of activities. The validation activity is now ready to be completed.
 ___ 10. Log out as Paul.

6.10. Validation: Abu

- ___ 1. Sign in as Abu, with `Abu` as the user name and password.
 ___ 2. Go to the **Work** tab to see the activities that are assigned.
 ___ 3. Click the **Test minimum age** link to open the activity page.
 ___ 4. Rerun the test suite that you created at the beginning of this validation activity in Section 6.7, "Creating the test suite: Abu".

This time, the status is green and the report shows 100% success as expected. This activity is ready to be completed.

- ___ 5. Return to the "Test minimum age" activity page by clicking the breadcrumb.
 ___ 6. In the Validation Activity pane, in the **Testers** section, change Abu's status from **Working** to **Finish Working**.
 ___ 7. Log out as Abu.

6.11. Approving the activity and deploy: Paul

- ___ 1. Sign in as Paul, and switch to the **Work** tab.
 ___ 2. Click the action menu beside the **Test minimum age** activity link, and select **Proceed to approval**.

- ___ 3. Approve the Validation Activity as you did in section Section 6.9, "Approving the change activity: Paul".

6.12. Completing the release: Paul

To complete the release:

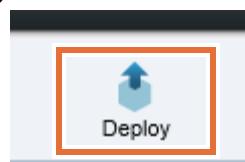
- ___ 1. Return to the **Training Release** page.
- ___ 2. In the **Release** pane, in the **Status** field, click **In Progress** and select **Proceed to approval**.
The status changes to **Ready for Approval**.
- ___ 3. In the **Approvers** section, click the list beside Paul's name, click **Not Reviewed**, and click **Approve changes**.
- ___ 4. Click **Approve** to confirm approval of the release.
The release status now changes to **Complete**.

6.13. Deploying the Training Release: Paul

Now that the release is completed, Paul deploys it to production.

- ___ 1. Return to the **Training Release** page and click the **Deployments** tab.
For this exercise, you use the main deployment configuration that you created in Rule Designer.
- ___ 2. Click the main deployment configuration to open it and view the deployment details.
The deployment configuration includes the following tabs:
 - The **General** tab, which lists basic information about the deployment configuration.
 - The **Operations** tab, which lists the decision operations to deploy. Here, **all rules operation** is selected for deployment.
 - The **Targets** tab, which lists 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 can use to choose the groups that can see and use this deployment configuration.
 - The **Deployment Snapshot** tab, which defines whether a snapshot is taken at the moment of deployment.

- ___ 3. On the toolbar, click **Deploy**.



- ___ 4. In the Deploy main window, review the configuration details and click **Deploy**.
- ___ 5. Click **OK** to close the Deployment status window.

6. Click the **Reports** tab to open the list of deployment reports.

The screenshot shows the IBM Decision Service interface. At the top, there's a header bar with 'Loan service >' and a date '31'. To the right are buttons for 'Take Snapshot' (with a camera icon) and 'Deploy' (with an upward arrow icon). Below the header is a navigation bar with tabs: Activities, Rules, Tests, Simulations, Deployments, and Snaps. The 'Deployments' tab is currently active. Underneath the navigation bar, there are two buttons: 'Configurations' and 'Reports', with 'Reports' being highlighted by a red box.

7. Click the report link for your deployment to view the summary report of your deployment.

Notice that a deployment snapshot was created and you can click the link to open it. The **Redeploy** icon beside 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 report for 'Report 2015-01-14_11-22-27'. The report is titled 'Summary'. It contains several sections with deployment details:

- Deployment Content:** Training Release In Loan service (Success)
- Configuration name:** main deployment configuration
- Operations Deployed:** * all rules operation
- Target Servers:** * Sample
- Deployment snapshot:** main_deployment_configuration_2015-01-15T07_22_1 (highlighted by a red box)

8. Close the report.

End of exercise

Exercise review and wrap-up

This exercise looked at how you work with a decision service, starting in Rule Designer. You also saw how the decision service lifecycle is governed through development, testing, and deployment in Decision Center through the Decision Governance Framework.

To see the final project in Rule Designer before deployment and publishing to Decision Center, switch to a new workspace and import the project **Ex 22: Working with decision services > 02-answer**.

You might see errors on the deployment configuration if the login information is not saved for the target server. To eliminate errors, redefine the target server with the same name, Sample, and this information.

- **URL:** `http://localhost:9080/res`
- **User name:** `resAdmin`
- **Password:** `resAdmin`

IBM
®