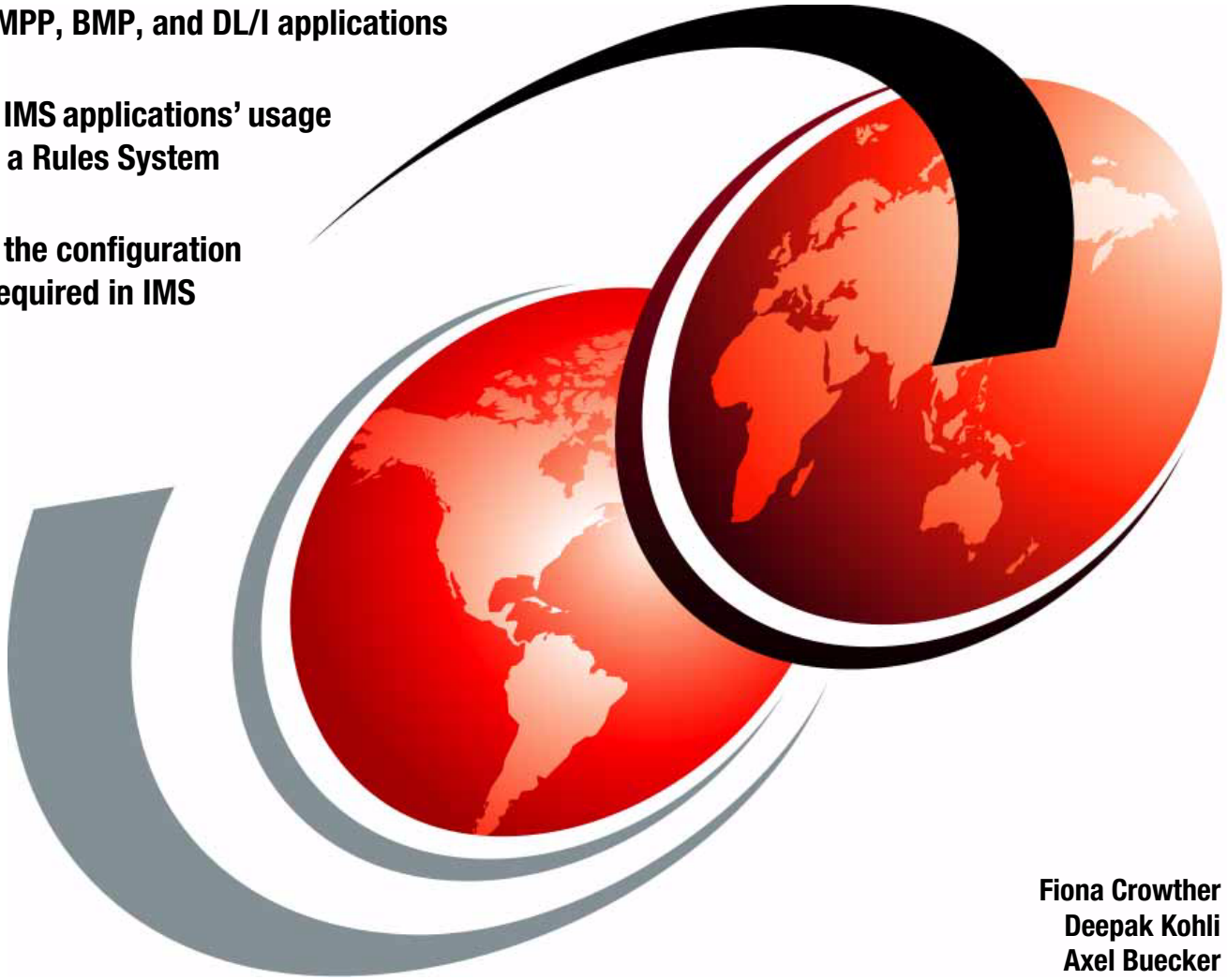
IBM® WebSphere®

Using IBM Operational Decision Manager

IMS COBOL BMP, COBOL DLIBATCH, and COBOL MPP

- Provides a step-by-step guide for calling ODM from IMS MPP, BMP, and DL/I applications
- Discusses IMS applications' usage of ODM as a Rules System
- Considers the configuration changes required in IMS



Fiona Crowther
Deepak Kohli
Axel Buecker

Redpaper

ibm.com/redbooks

International Technical Support Organization

Using IBM Operational Decision Manager: IMS COBOL BMP, COBOL DLIBATCH, and COBOL MPP

June 2013

REDP-4997-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (June 2013)

This edition applies to IBM Operational Decision Manager (ODM) Version 8.0.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®
DB2®
IBM®
IMS™
Orchestrate®
Redbooks®
Redpaper™
Redbooks (logo) ®
VTAM®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

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

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM® Operational Decision Manager (ODM) is an implementation of a Business Rule Management System (BRMS). It enables you to create, manage, test, and govern business rules and events. You can store these in a central repository where multiple individuals and software products can access them.

IBM ODM Version 8.0 provides support for IBM IMS™ COBOL programs. This IBM Redpaper™ publication walks you through a step-by-step approach for using IBM ODM for rules management from an IMS COBOL MPP, BMP, or DL/IBATCH program.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Austin Center.

Fiona Crowther is part of the development team for IBM ODM on IBM z/OS® in Hursley, UK. She has a Masters degree in Information Systems from the Robert Gordon University in Aberdeen, Scotland, and has worked as a Software Engineer in IBM for 16 years. She moved to Hursley in 2000, where she has worked on various products including IBM WebSphere® Message Broker, WebSphere Enterprise Service Bus, and WebSphere Service Registry and Repository.

Deepak Kohli is a Senior Software Engineer with IBM IMS development at the IBM Silicon Valley Lab. He has a Masters Degree in Computer Science from New York University. Since 1979, Deepak has worked on various products, such as IMS, IBM CICS®, IBM DB2®, IBM VTAM®, TCP/IP, WebSphere Application Server, and IBM Workload Deployer (IWD). Currently, Deepak is part of the IMS product management group providing product direction. He has also been part of the IMS service-oriented architecture (SOA) core team helping clients to enable their IMS systems for SOA. Deepak has also provided IMS Level 2 technical support. Deepak is a regular speaker at various conferences worldwide.

Axel Buecker is a Certified Consulting Software IT Specialist at the ITSO, Austin Center. He writes extensively and teaches IBM classes worldwide about areas of software security architecture and network computing technologies. He has a degree in Computer Science from the University of Bremen, Germany. He has 26 years of experience in a variety of areas related to workstation and systems management, network computing, and e-business solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.

Thanks to the following people for their contributions to this project:

Chris Backhouse, Jaime Buxton, Mark Hiscock
IBM

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Obtain more information about the residency program, browse the residency index, and apply online at:

http://ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  http://ibm.com/redbooks
- Send your comments in an email to:

  http://redbooks@us.ibm.com
- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Using IBM Operational Decision Manager to support rules management

IBM Operational Decision Manager (ODM) 8.0 provides support for rules management for IMS COBOL message processing program (MPP), batch message processing program (BMP), and DLIBATCH programs. This IBM Redpaper publication provides a step-by-step approach for using ODM for rules management for each of these types of programs.

In the current environment, it is necessary for systems to change quickly and effectively to keep up-to-date with constant change. The uncertainty of the current economic climate and the complexity of the business environment mean that business policies are changed more often than ever before. But keeping the software up-to-date with these changes is difficult. In many cases, the decision making is buried deep within the code, and only a long product change lifecycle can change them.

If you use a *business rules and events engine*, you can separate the policies from the software itself. By keeping the decision-making details separate, you can change them more conveniently, without modifying the software itself. This way, many different software modules can access the same rules, so there is less risk of contradicting policies. It is no wonder that such Business Rule Management Systems (BRMS) have become popular.

IBM ODM for z/OS V8.0 provides a rules engine that can be accessed directly from IMS MPP, BMP, and DLIBATCH programs, so when you use IMS you benefit directly from the advantages of using such a system.

The use of business terms in the definition of rules in ODM means that business analysts can change the rules directly without having to go to IT for the changes, further speeding up the process of changing rules. Decision tables enable you to implement more complex rules, still outside the main application.

You can run ODM either stand-alone on z/OS or within a WebSphere Application Server environment, in which case it is accessed via WebSphere z/OS Optimized Local Adapters (WOLA).

Figure 1-1 shows how IMS and ODM run on the same logical partition (LPAR).

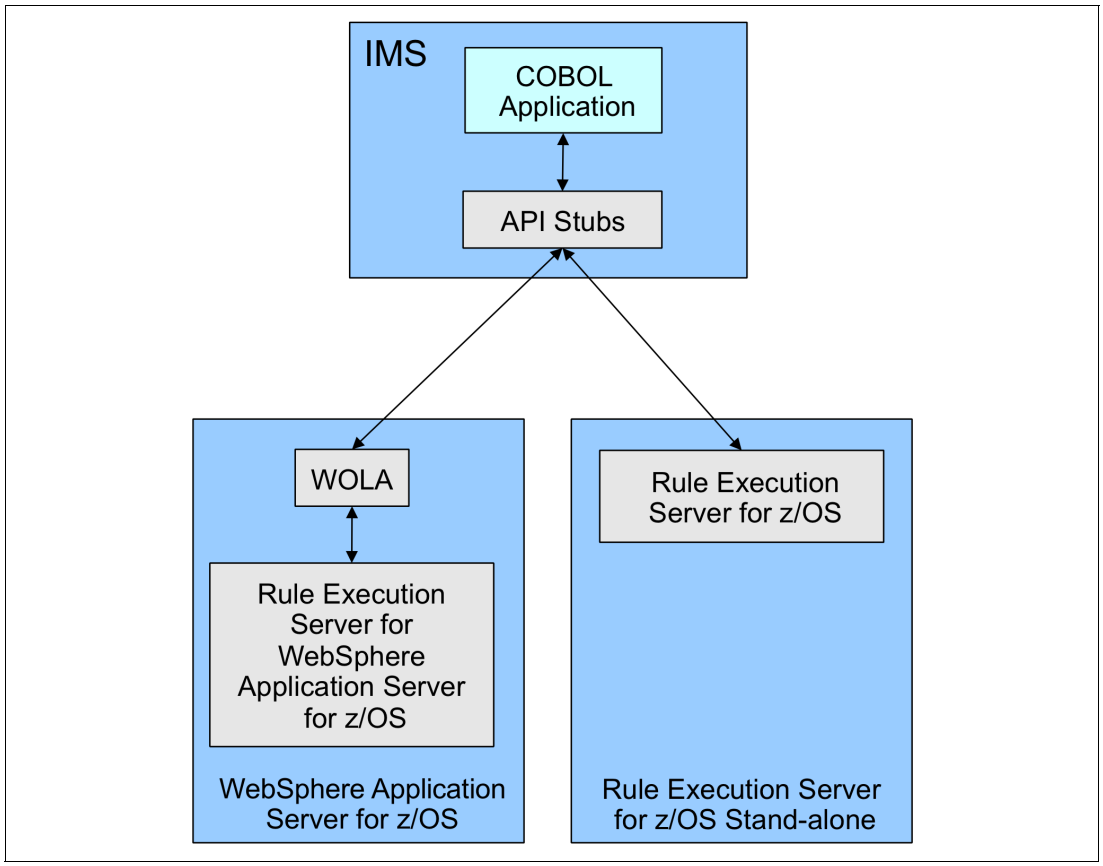


Figure 1-1 IMS connection via the API stubs to the Rule Execution Server for z/OS

This paper explains how an IMS application can use ODM as a rules system to separate the business policies from the main application.

2

Setting up rules in ODM

Begin by considering exactly how to translate a *business policy* into a set of rules that can be stored inside IBM Operational Decision Manager (ODM).

You create all rules in the Rules Designer GUI, which is shipped as part of ODM. The Rules Designer works on an Eclipse platform, and you employ it to create rules that your application uses. You then deploy these rules to the Rules Server.

In this chapter, you learn how to create simple rules in the Rules Designer.

You need to understand these two concepts before learning more about rules:

- Business Object Model (BOM)

  The BOM represents a model of the core concepts of a business, such as a *loan* or a *borrower*, and their logical connections. You explain this in business terms, and it represents the objects to the business user for use in rule authoring.

- Execution Object Model (XOM)

  The XOM represents a model that the runtime implementation uses to execute the rules. It references the application objects and data, and is the runtime implementation of the BOM.

In the following example, you create an XOM from a COBOL copybook, and from this you create a BOM to represent the content of this copybook in business-oriented language. This helps you create a rule. This section presents an overview of the scenario for this chapter.

This section discusses rules and the example scenario:

- Rules

  There are many ways of defining rules. They can be as simple as:

  ```
  if balance < 0 then account_status = overdrawn
  ```

  Rules can also be much more complicated, involving multiple factors and multiple statuses. For example, consider a choice of interest amounts, depending on the type and balance of an account, as shown in Table 2-1 on page 4.

Table 2-1 Complex rule example

Age of borrower	Loan < annual income	Annual income < loan <= 3x annual income	3x annual income < loan
<=21	Approved	Rejected	Rejected
21-60	Approved	Approved	Approved
60+	Approved	Approved	Rejected

This kind of rule might be better developed as a decision table or a decision tree. The construction of such a rule is slightly more complicated. Appendix E, “Further rules definitions” on page 61 contains an example of these constructions.

- Example scenario

  A borrower wants to take out a loan. Whether the loan is approved is based on various factors:

  - The company refuses to lend more than $1 million to any customer.
  - The customer must be 65 years of age or younger.
  - The loan repayment must be less than or equal to 30% of the annual income of the borrower.
  - The income of the borrower after tax must be $24,000 or greater.

  This example demonstrates how you create a project containing these rules ready for use by ODM.

2.1 Creating a rule project

To create a rule project, follow these steps:

1. In the Rule Designer, click **New** → **Rule Project**. Then, select **Standard Rule Project** and click **Next**.
2. In the Project name field, type `loan-decision-rules`, as shown in Figure 2-1.

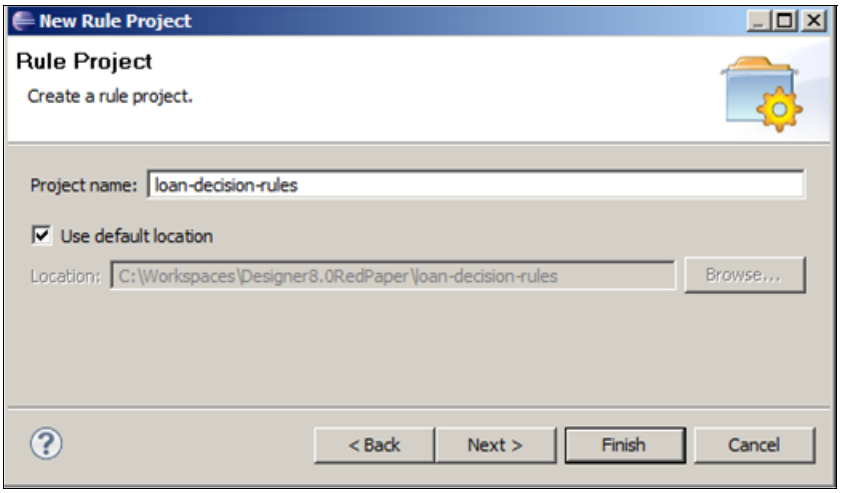


Figure 2-1 Create a rule project

3. Click **Finish** to create the rule project in the Rule Designer.
4. Examine the Rule Explorer window. It currently only contains empty folders, as shown in Figure 2-2.

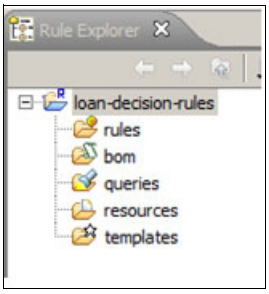


Figure 2-2 Rule Explorer overview

2.2 Create COBOL XOM from a COBOL Copybook

Before you create a BOM, you must create a COBOL Executable Object Model (XOM). Follow these steps:

1. In the Design part of the Rule Project Map tab, click **Import XOM**, as shown in Figure 2-3.

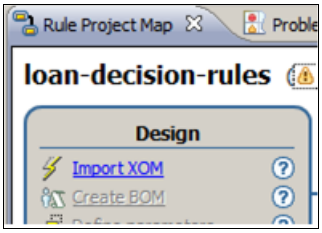


Figure 2-3 Locate the Import XOM task

2. On the Import XOM page, choose **COBOL Execution Object Model**, as shown in Figure 2-4. Click **OK**.

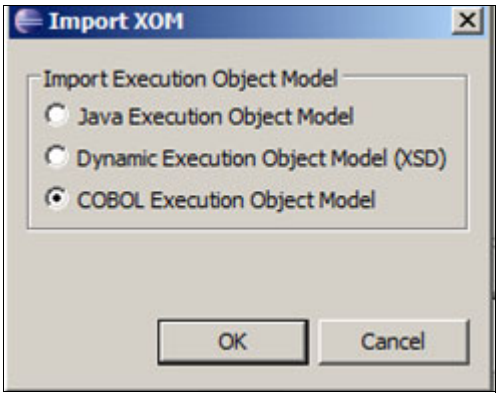


Figure 2-4 Import XOM

3. On the Properties for loan-decision-rules page, select **COBOL Execution Object**, and click **Add,** as shown in Figure 2-5.

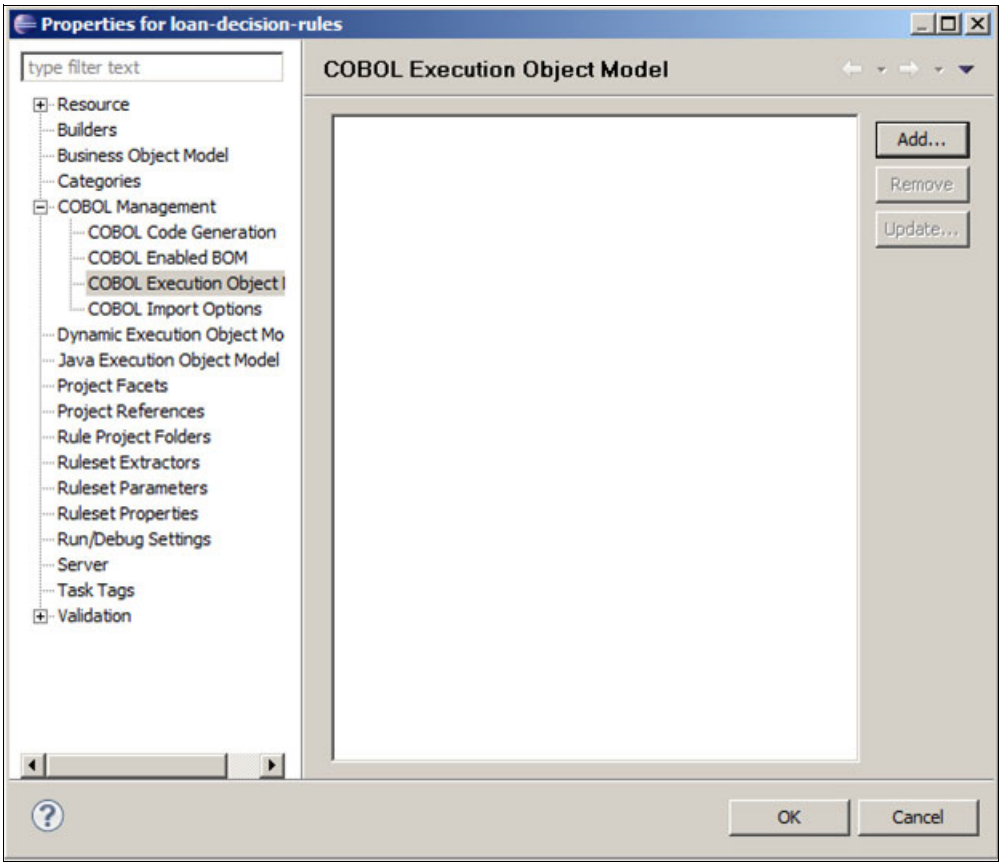


Figure 2-5 Properties for the loan-decision-rules

4. In the Import COBOL XOM dialog, click **Add**, as shown in Figure 2-6.

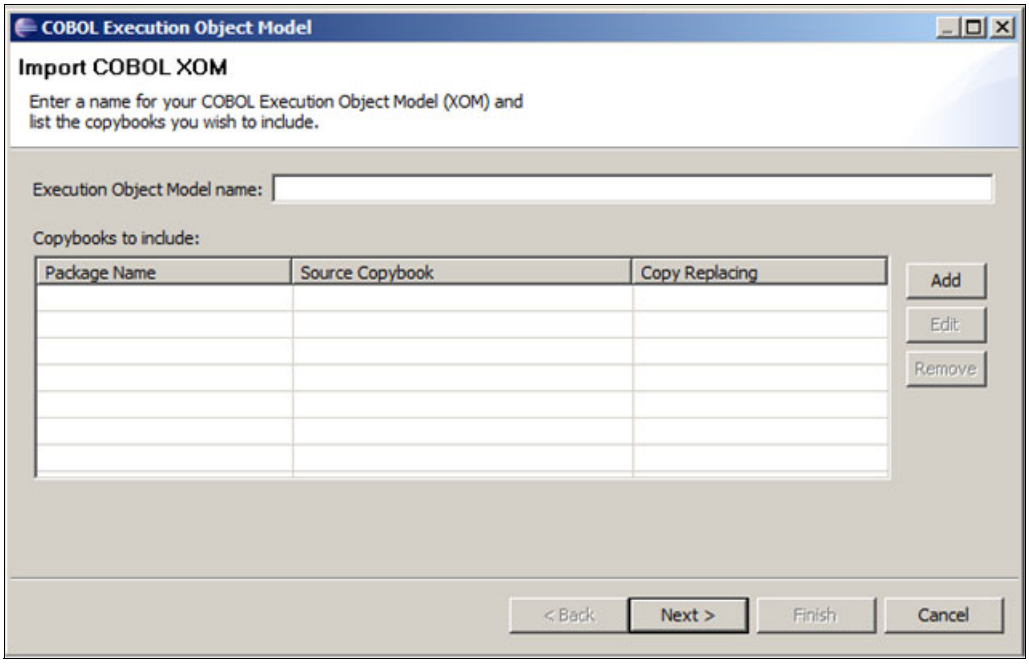


Figure 2-6 Add a copybook

5. Select **File system** and click **Browse**. Navigate to the copybook that you want to import. Replace the Package name it selects with `requestDetails`, as shown in Figure 2-7. Click **OK**.

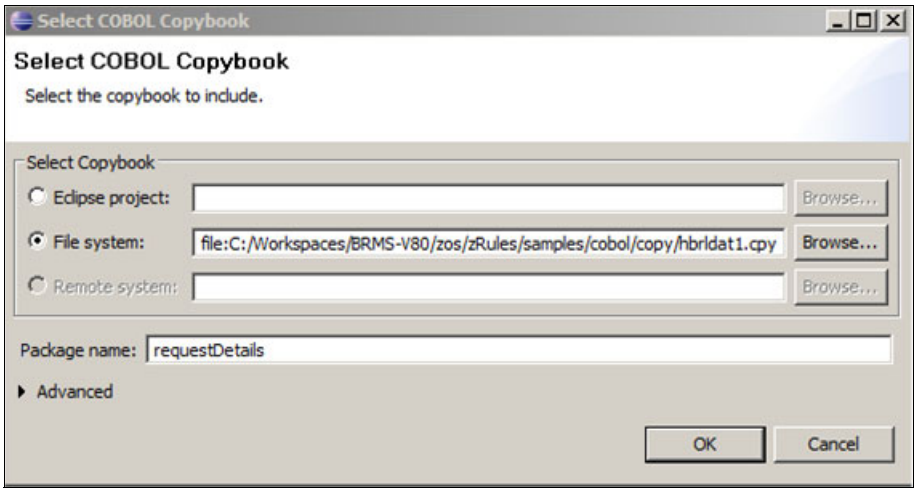


Figure 2-7 Select the copybook file

6. In the Import COBOL XOM panel, enter `requestDetails` for the Execution Object Model name, as shown in Figure 2-8. Click **Next**.

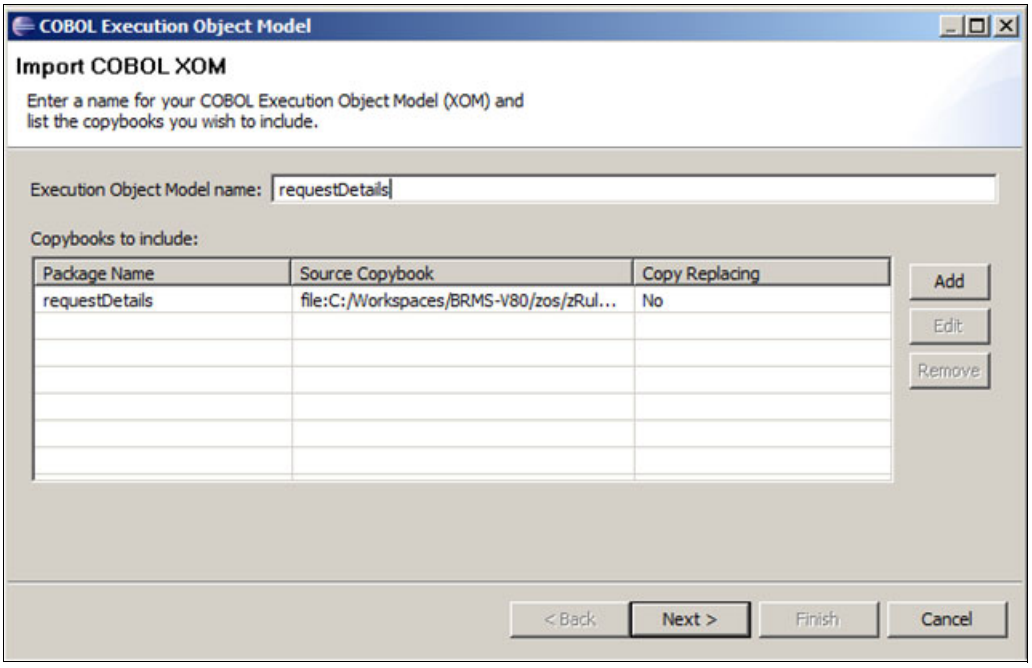


Figure 2-8 Provide a name for the COBOL XOM

7. Figure 2-9 shows a summary of the default Java types and business object model (BOM) attributes that are derived from each COBOL item in the copybook.

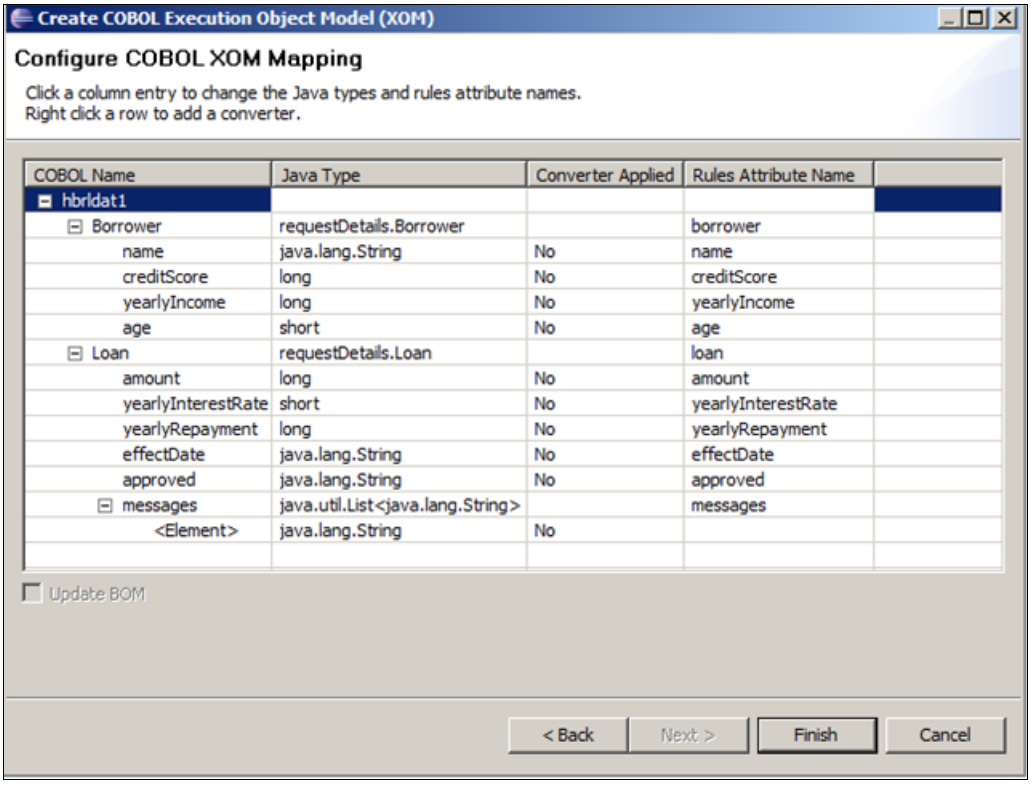


Figure 2-9 Summary panel

8. Click **Finish** to create the COBOL XOM. This takes you back to the Properties dialog shown in Figure 2-10.

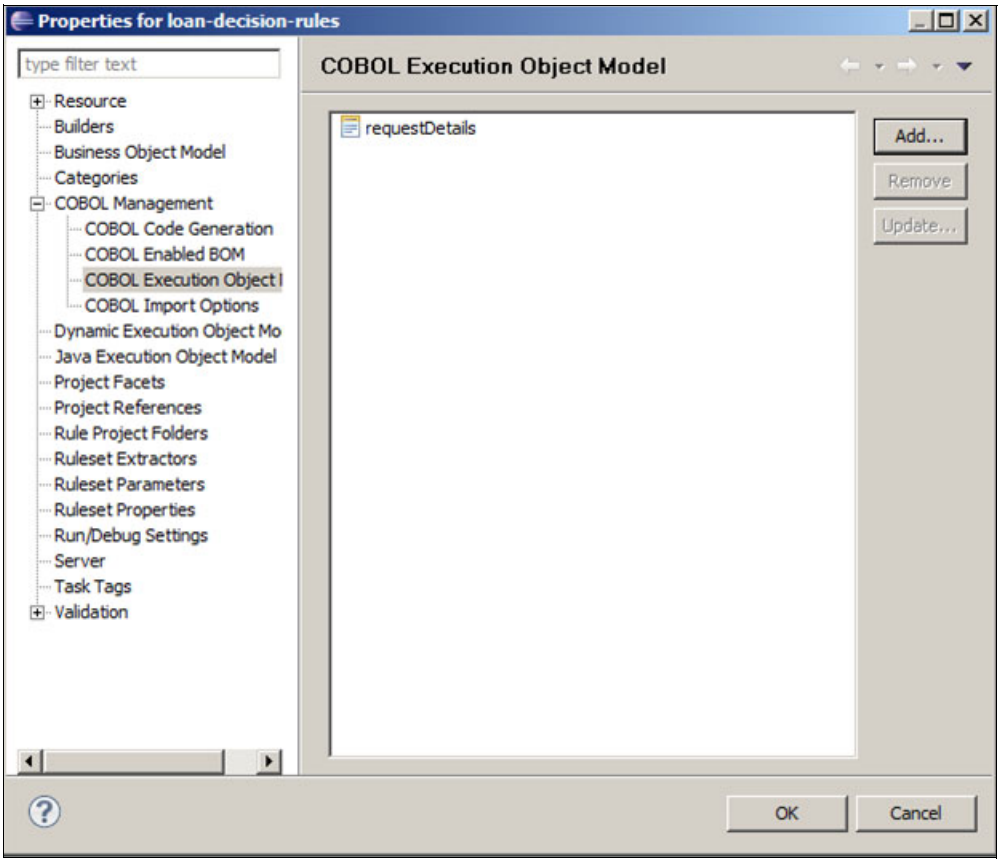


Figure 2-10 Properties for loan-decision-rules with the new COBOL XOM

9. Click **OK** to close the Properties for loan-decision-rules window.

The following artifacts are created:

- The `requestDetails-marshaller.jar` marshaller jar
- The `CobolXomConfig.xml` COBOL XOM configuration file
- The requestDetails project containing the defined structures in Java format, as shown in Figure 2-11 on page 11.

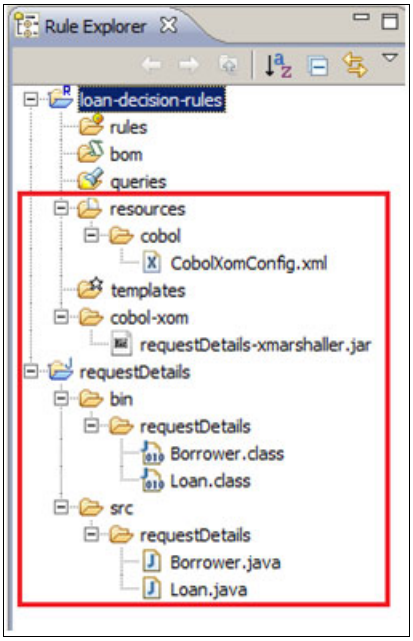


Figure 2-11 Java artifacts for the project

2.3 Creating a business object model from the JAVA XOM

To create a business object model from the JAVA XOM, follow these steps:

1. In the Design part of the Rule Project Map tab, click **Create BOM**, as shown in Figure 2-12.

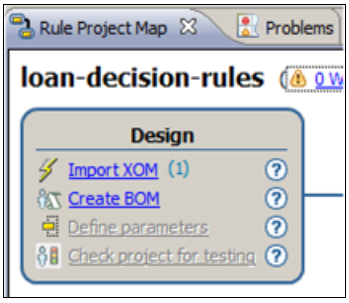


Figure 2-12 Locate the Create BOM task

2. In the New BOM Entry wizard, in the Name field, accept the default name **model** for the BOM entry. Ensure that the **Create a BOM entry from a XOM** option is selected, as shown in Figure 2-13. Click **Next** to continue.

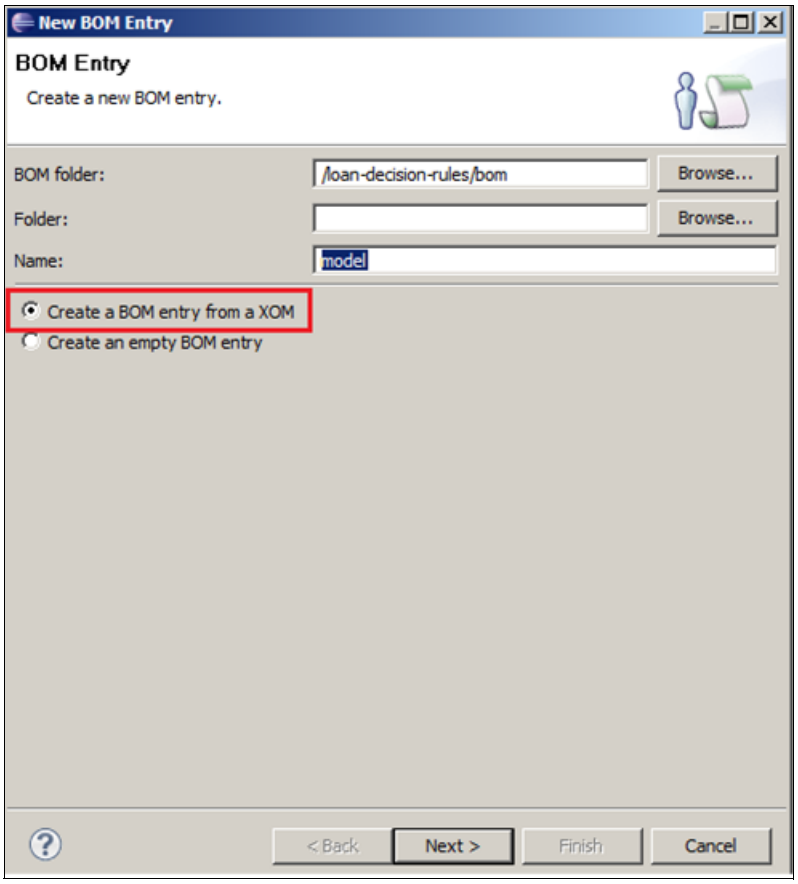


Figure 2-13 Begin the New BOM Entry wizard

3. On the Browse XOM page, in the Choose a XOM Element list, select **platform:/requestDetails**, as shown in Figure 2-14.

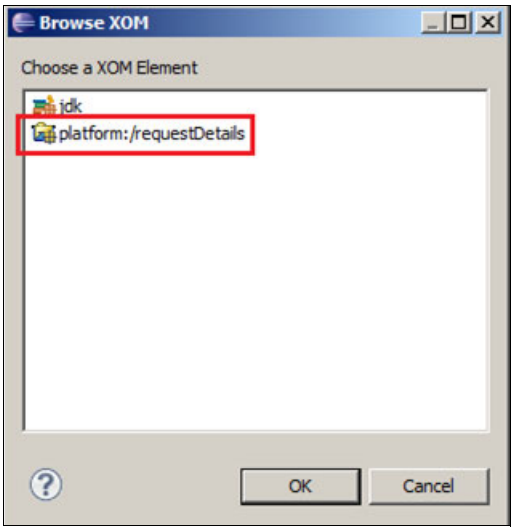


Figure 2-14 Choose a XOM Element

4. In the Select classes list of the New BOM Entry window shown in Figure 2-15, select the **requestDetails** package. When you select the package, you automatically select all of the classes that it contains.

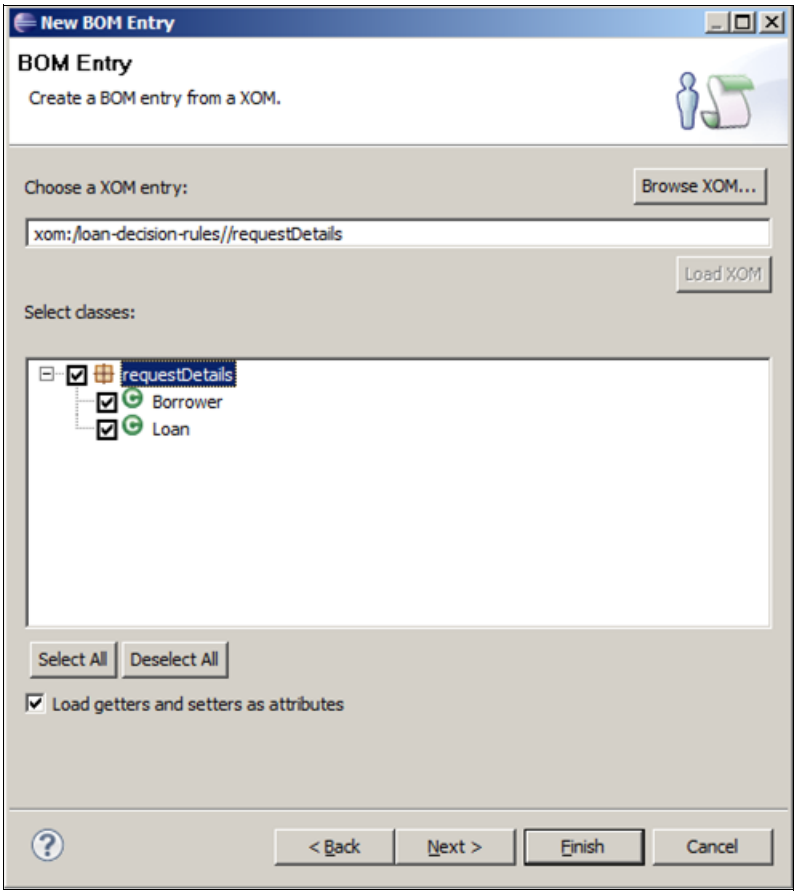


Figure 2-15 Select classes

5. Now, click **Finish**. In the Rule Explorer view, the bom folder contains a new BOM entry model, as shown in Figure 2-16.

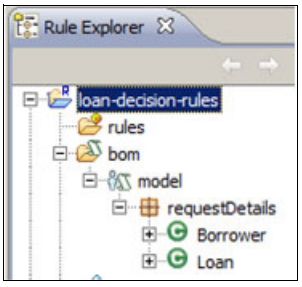


Figure 2-16 New BOM entry

6. Examine the generated BOM and its verbalization:
   a. In the Rule Explorer view shown in Figure 2-17 on page 14, double-click **bom** → **model** to open the BOM editor.

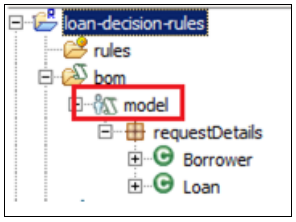


Figure 2-17 Double-click model

b. In the BOM editor, expand the **requestDetails** package to view the generated BOM, as shown in Figure 2-18.

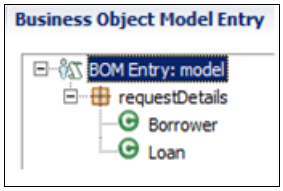


Figure 2-18 Expanded requestDetails view

c. Double-click the **Borrower** class to view the default class verbalization, which is shown in Figure 2-19.

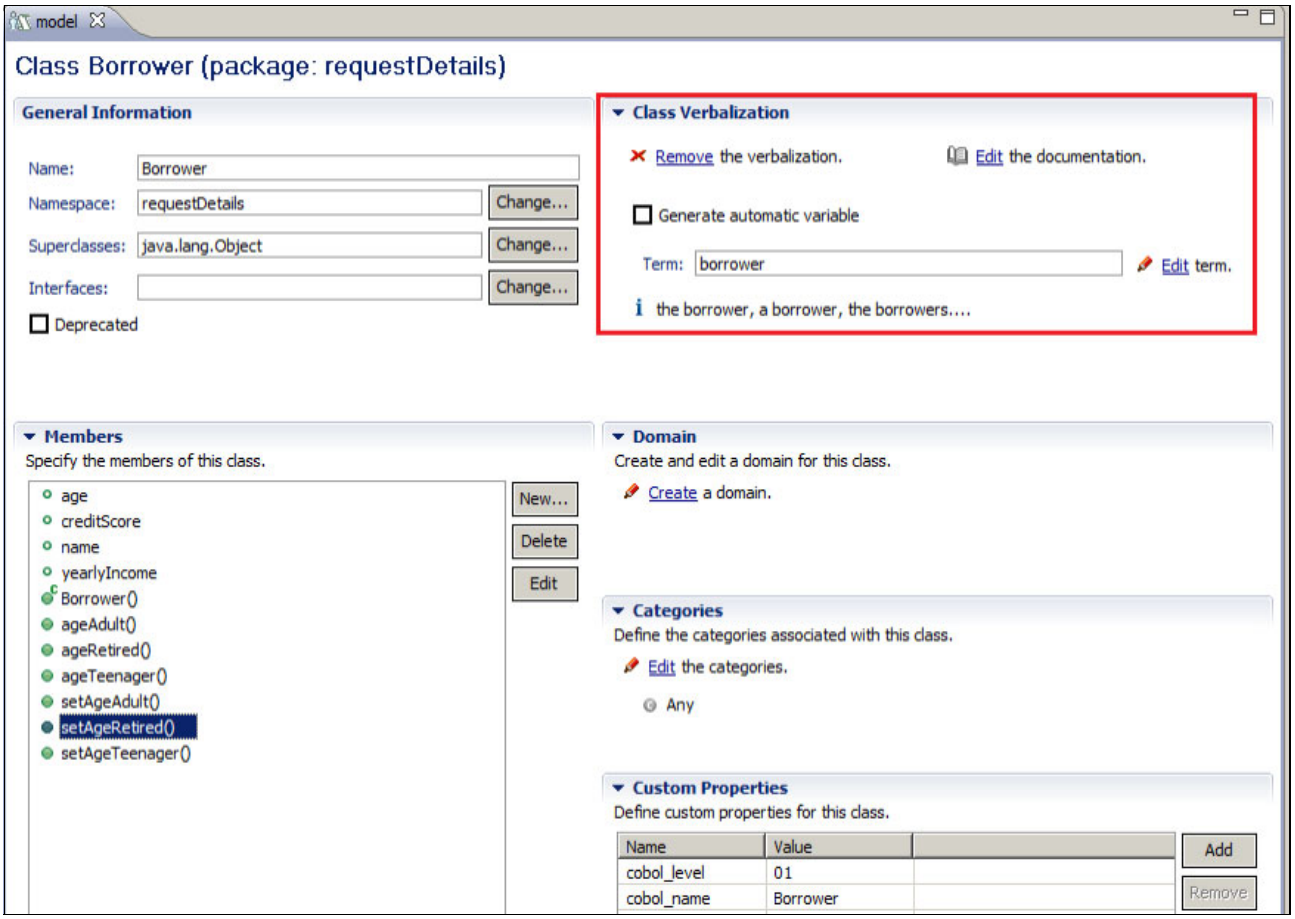


Figure 2-19 Class verbalization for borrower

2.4 Declaring ruleset parameters

Ruleset parameters provide the means to exchange data between a COBOL application and the rule application. You define ruleset parameters by name, type, and direction.

In this sample, you create ruleset parameters for the borrower and loan classes. The IN direction is used for the borrower, and the IN_OUT direction is used for the loan because it contains both input and output parameters. The output parameters are populated by the rule.

You cannot use the OUT parameter direction with the Rule Execution Server for z/OS because COBOL programs do not support memory allocation dynamically.

Follow these steps:

1. In the Design part of the Rule Project Map tab, click **Define parameters**, as shown in Figure 2-20.

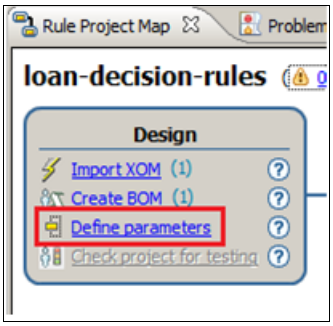


Figure 2-20 Locate the Define parameters task

2. On the Ruleset Parameters page, ensure that **Enable type check for COBOL XOM** is selected.
3. To define a request parameter, click **Add**. Then, change the following default values:
   a. In the Name column, type `borrower`.
   b. In the Type column, click the ellipsis **(...)** button on the right of the cell and choose **borrower**. The requestDetails.borrower entry is entered in the cell automatically.
   c. In the Direction column, choose the **IN** direction.
   d. In the Verbalization column, type `the borrower`.
4. To define the response parameters, click **Add**. Then, change the following default values, as shown in Figure 2-21 on page 16:
   a. In the Name column, type `loan`.
   b. In the Type column, choose **loan**. The requestDetails.loan entry is added to the cell automatically.
   c. In the Direction column, choose the **IN_OUT** direction.
   d. In the Verbalization column, type `the loan`.

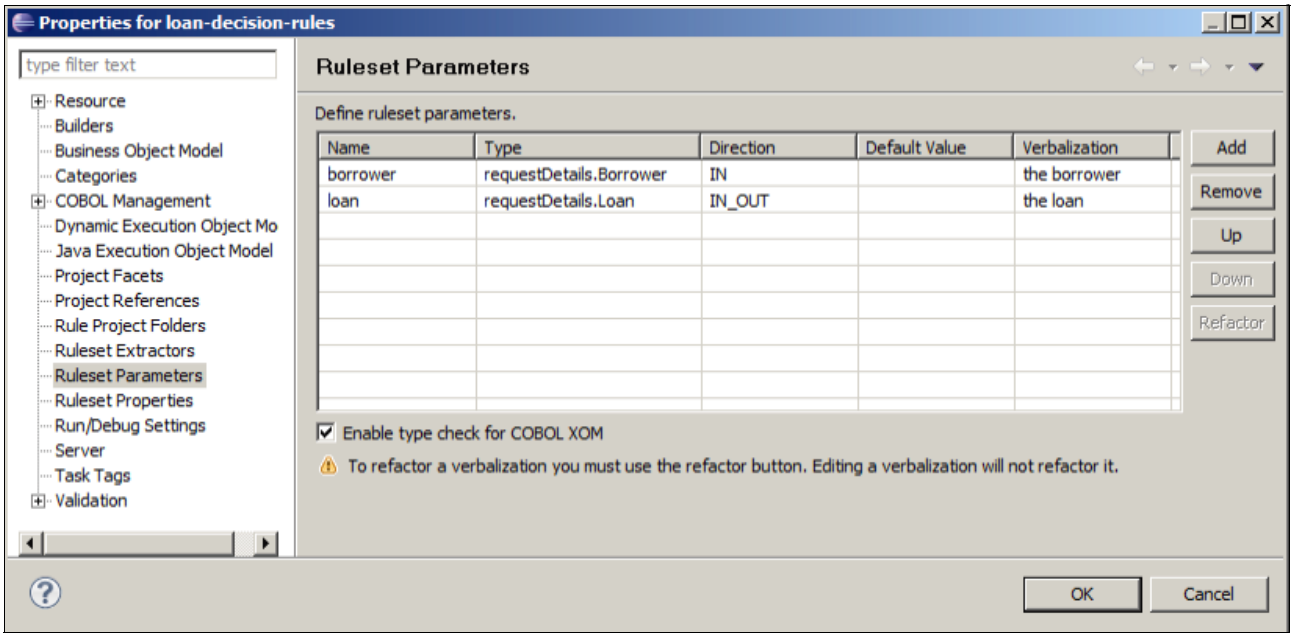


Figure 2-21 Properties for loan-decision-rules

5. Click **OK** to close the Properties for loan-decision-rules dialog box.

2.5 Adding BOM methods and map them to the XOM

The BOM methods are used to specify conditions and actions in your rules. You create methods in the Rule Designer. When you add methods to the BOM, you use BOM-to-XOM mapping in the BOM editor to implement the method.

You cannot map the BOM method to a Java XOM method because you must not change the XOM.

In this example, you create a method to reject a loan request for a configurable reason.

2.5.1 Adding the reject methods

Follow these steps to add the reject methods:

1. In the Rules Explorer view, expand the model package, and then double-click the **Loan** class, as shown in Figure 2-22.

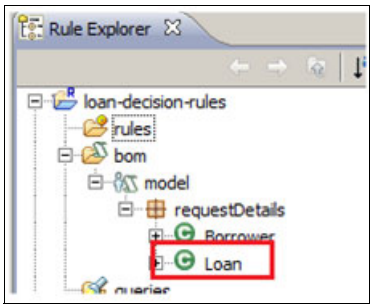


Figure 2-22 Navigate to the Class Loan page

2. On the Class Loan page of the BOM editor, to the right of the Members section, click **New**, as shown in Figure 2-23.

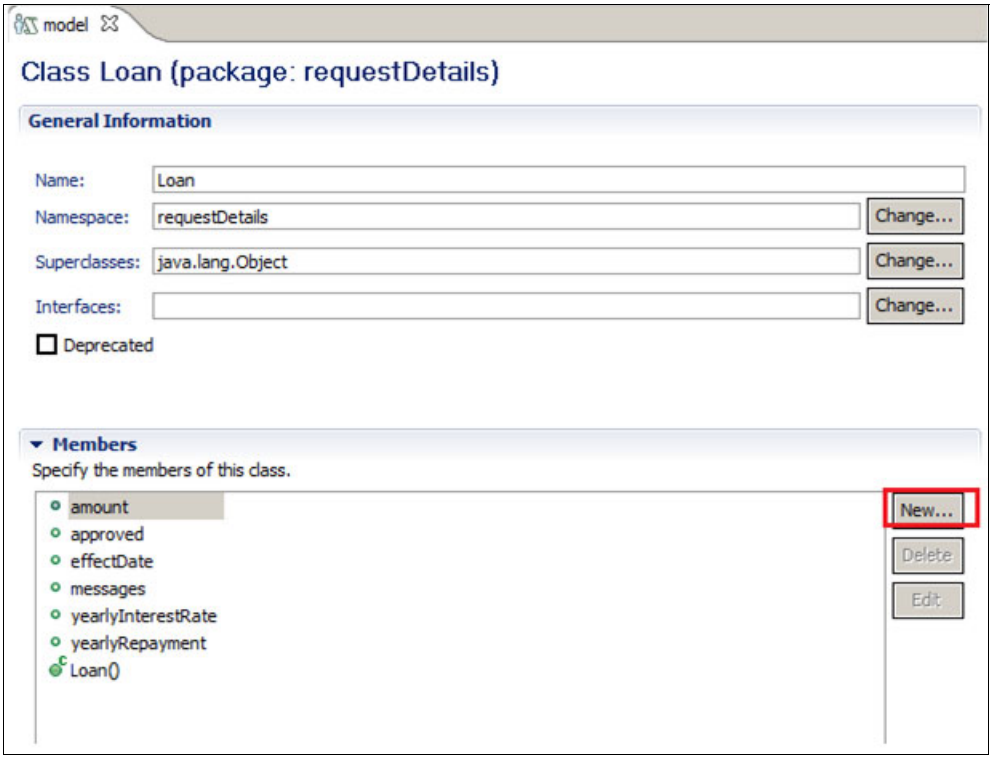


Figure 2-23 Editing the Class Loan

3. In the New Member window, provide the following information:
   - For Type, select **Method**.
   - For Name, type `reject`.
   - For Type, type `void`.

Click **Add** when finished, as shown in Figure 2-24.

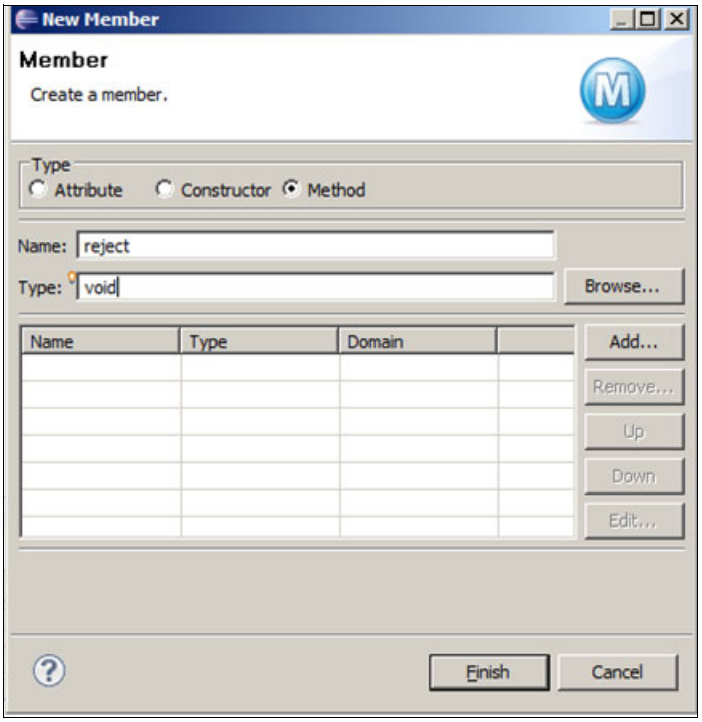


Figure 2-24 New Member

4. In the Method Argument window, enter the following information:
   - For Name, enter `Message`.
   - For Type, enter `java.lang.String`.

   When complete, click **OK**, as shown in Figure 2-25. Back in the New Member window, click **Finish** to create the method.

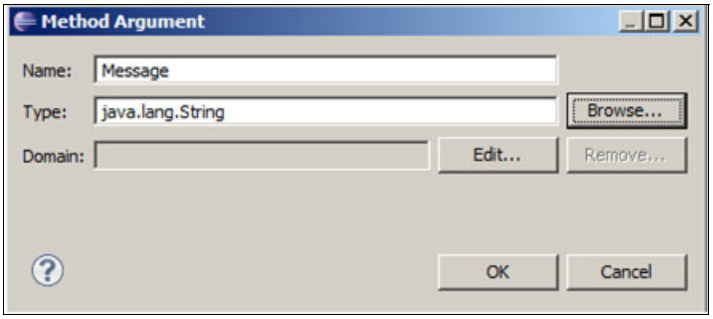


Figure 2-25 Method Argument

5. On the Class page of the BOM editor, the Members list now includes the reject(String) method. Double-click this method, as shown in Figure 2-26.

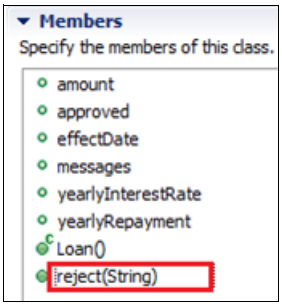


Figure 2-26 Double-click reject(String)

6. In the Members Verbalization section of the BOM editor, click **Create** to view the default verbalization, as shown in Figure 2-27.

Figure 2-27 Member Verbalization

7. The default verbalization of the reject class now displays. Enter the following verbalization into the Template field, as shown in Figure 2-28:

```
reject {this}, reason: ({0})
```

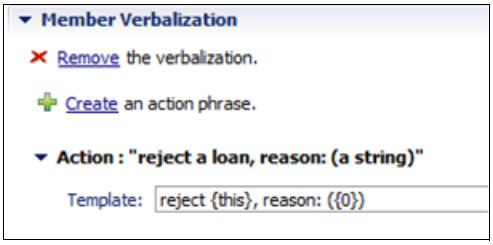


Figure 2-28 Adding an Action Template

8. Scroll down to the BOM to XOM Mapping section of the BOM editor, and then expand BOM to XOM Mapping to activate the BOM to XOM Mapping editor, as shown in Figure 2-29.

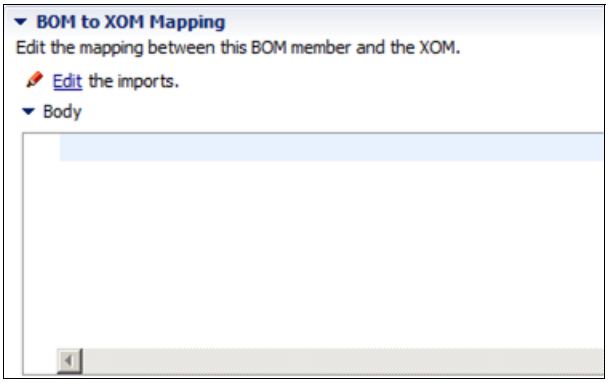


Figure 2-29 BOM to XOM Mapping editor

1. Type the following Java code, shown in Figure 2-30.

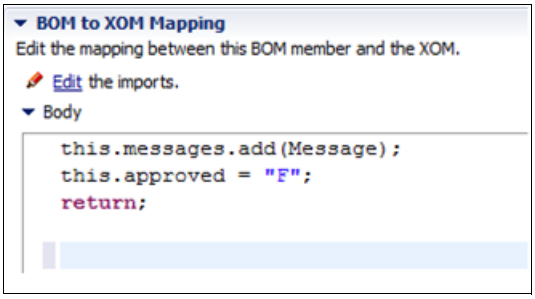


Figure 2-30 Java code entered for the reject method

2. Finalize these steps by saving the method.

2.6 Orchestrating the ruleflow

You need to control the order in which rules are executed by using *ruleflows*. When defining the flow of execution, you organize rules into packages that contain related rules. This section explains how you can create a package that relates to validation rules.

Follow these steps:

1. In the Rule Designer, in the IBM Orchestrate® part of the Rule Project Map, click **Add rule package**, as shown in Figure 2-31.

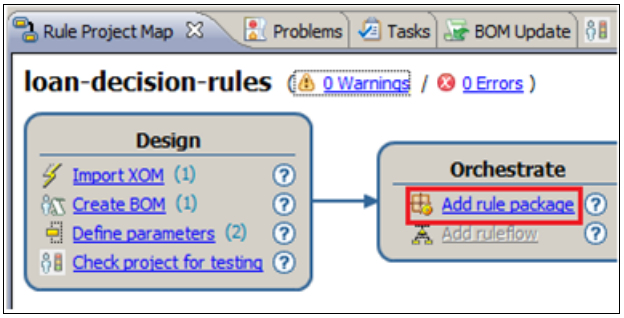


Figure 2-31 Add a rule package

2. In the New Rule Package wizard, type `decideOnLoan` into the Package field, and then click **Finish**, as shown in Figure 2-32.

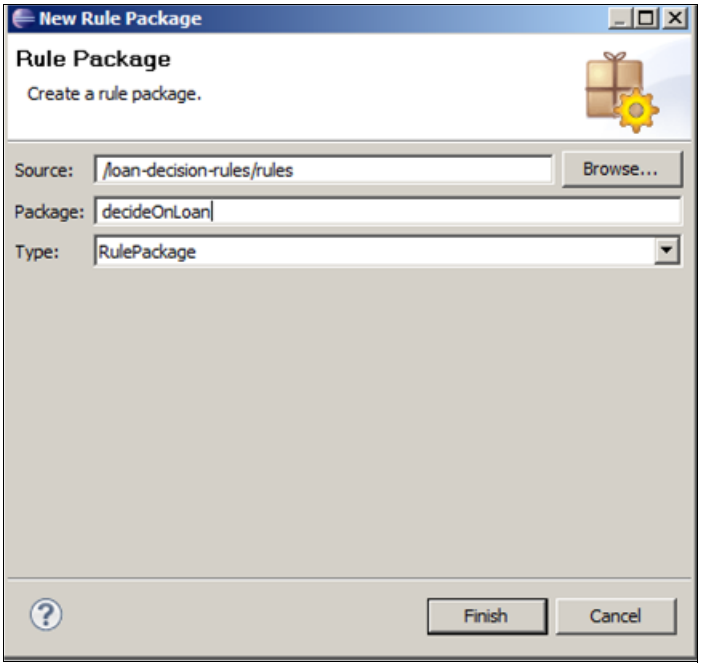


Figure 2-32 Create rule package

3. This creates the rule package, which now displays in the Rule Explorer, as shown in Figure 2-33.

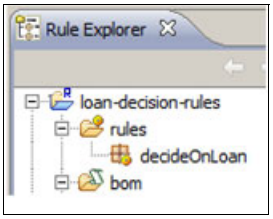


Figure 2-33 New rule package in Rule Explorer

4. To create the ruleflow, in the Orchestrate part of the Rule Project Map, click **Add ruleflow**, as shown in Figure 2-34.

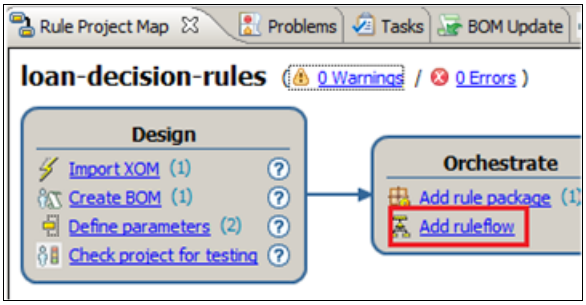


Figure 2-34 Add ruleflow

5. In the New Ruleflow dialog box, type `mainflow` into the Name field, and then click **Finish**, as shown in Figure 2-35.

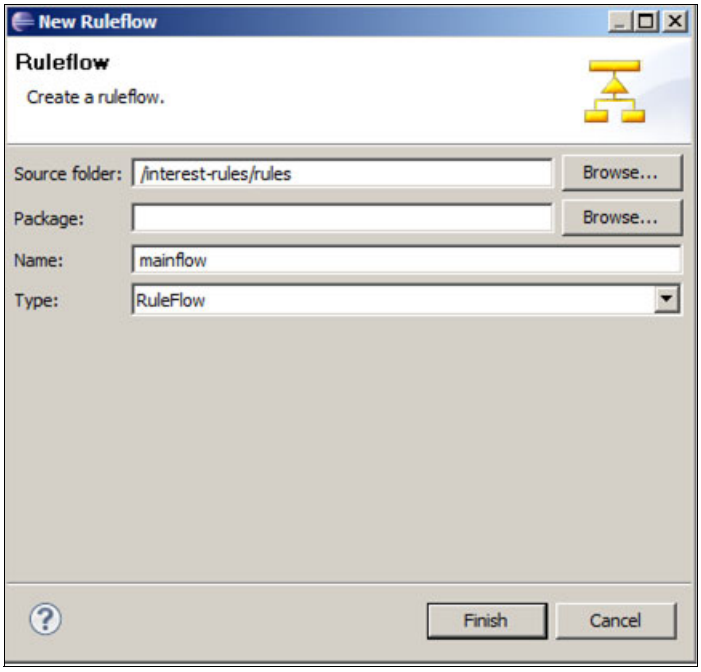


Figure 2-35 Create a ruleflow

6. A blank Ruleflow diagram is created. Follow these steps:

   a. Drag a **Start Node** onto the page.

   b. Drag an **End Node** onto the page.

   c. Drag the task for the **decideOnLoan** rule package onto the page, as shown in Figure 2-36.

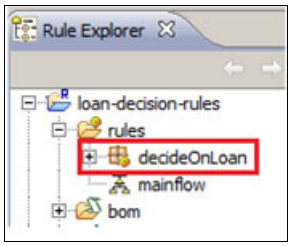


Figure 2-36 Drag the decideOnLoan rule package

   d. Select the **Arrow** icon.

   e. With the arrow icon selected, click your start node to begin the arrow and click the calculation rule package box to complete the arrow. Click the rule package box to start another arrow and click your end node to finish the second arrow.

f. Refine the diagram by clicking the **Refine** icon.

g. The diagram now has the Start Node at the top, an arrow from that to the decideOnLoan rule package, and an arrow from that to the End Node, as shown in Figure 2-37.

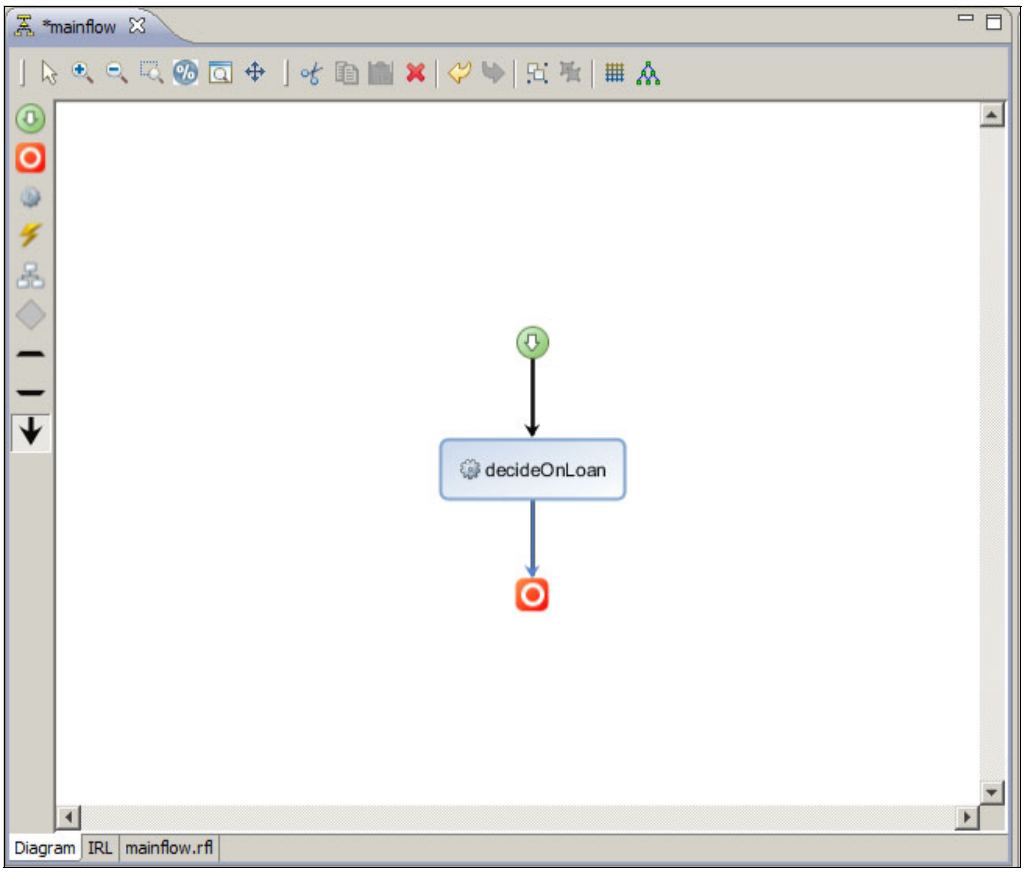


Figure 2-37 Finished ruleflow diagram

h. Save the diagram.

2.7 Authoring the rules

Next, you need to create a number of rules to fit into the calculation area:

- The `belowMaximumAmount` rule ensures that the amount of the loan does not exceed $1,000,000.
- The `belowMaximumAge` rule ensures that the borrower is 65 years of age or younger.
- The `repaymentLessThanMaximum` rule ensures that the loan repayment is less than or equal to 30% of the annual income of the borrower.
- The `aboveMinimumIncome` rule ensures that the yearly income of the borrower is $24,000 or greater.

To demonstrate a system like this, it is not necessary to create every rule: only those rules that are created are followed. In fact, the full sample program, on which this example is based,

depends on other rules that are not listed here. More complex rules can also be designed, as you can see in Appendix E, “Further rules definitions” on page 61. The rules specified in the following sections are a set created to demonstrate basic functionality.

2.7.1 Creating the exceedsMaximumAmount rule

To create the exceedsMaximumAmount rule, follow these steps:

1. In the rules project, right-click **decideOnLoan**, and then click **New** → **Action Rule**, as shown in Figure 2-38.

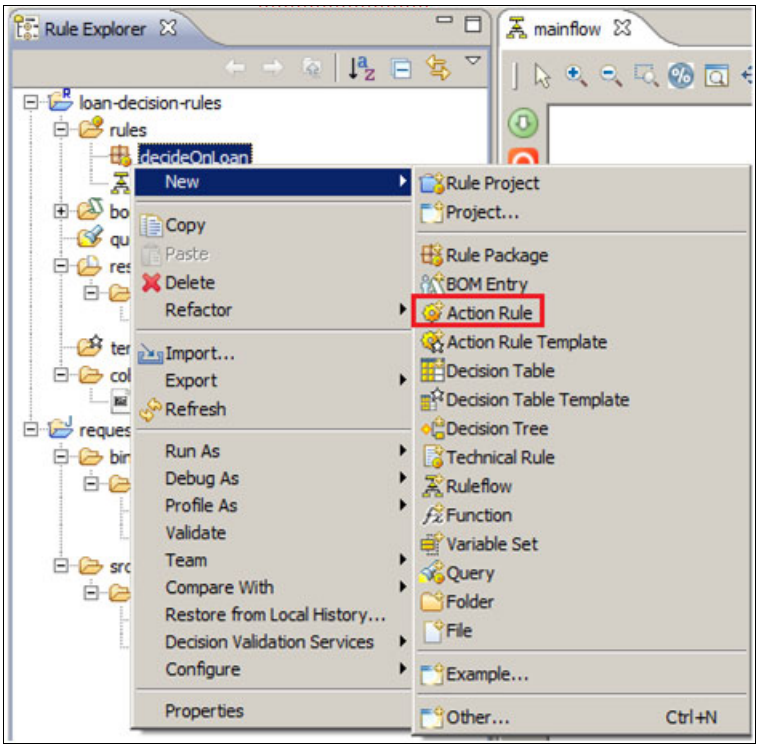


Figure 2-38 Create an Action Rule

2. In the Name field, enter `exceedsMaximumAmount`, as shown in Figure 2-39. Click **Finish** to continue.

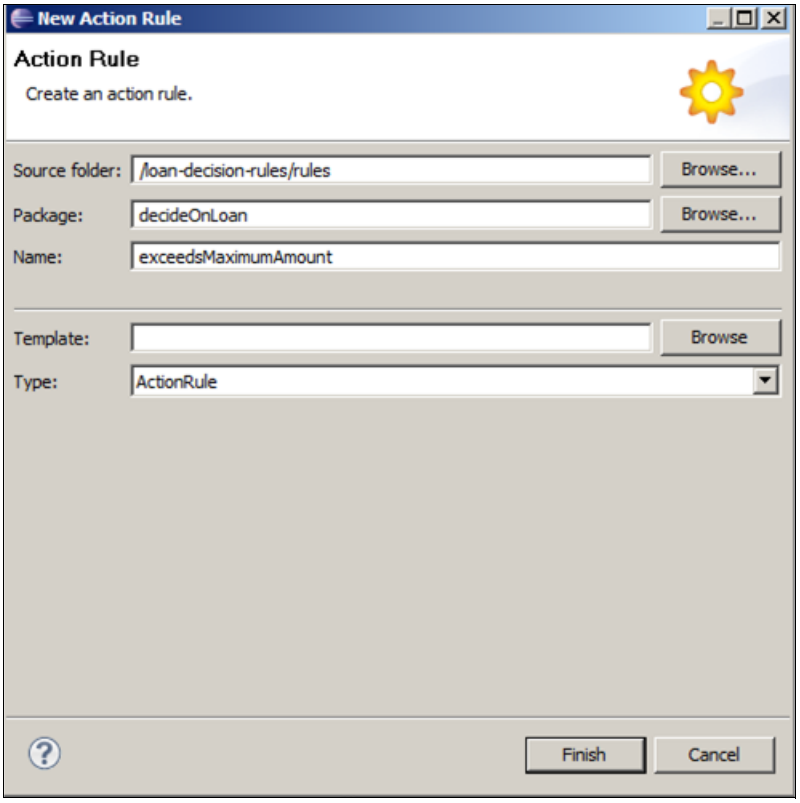


Figure 2-39 New Action Rule dialog

3. The new action rule displays in the Rule Explorer view, and the Intellirule Editor opens.
4. Type the code for the exceedsMaximumAmount rule, as shown in Example 2-1.

Example 2-1 exceedsMaximumAmount rule coding

```
Definitions
   set maximumAmount to 1000000;
if
   the amount of 'the loan' is more than maximumAmount
then
   reject 'the loan' ,
   reason: ("The loan amount is greater than the maximum of "+ maximumAmount );
```

5. Save the code.

2.7.2 Creating the belowMaximumAge rule

To create the belowMaximumAge rule, follow these steps:

1. In the rules project, right-click **decideOnLoan**, and then click **New** → **Action Rule**.
2. In the Name field, enter `belowMaximumAge`. Then, click **Finish**.
3. The new action rule displays in the Rule Explorer view, and the Intellirule Editor opens.

4. Type the code for the belowMaximumAge rule, as shown in Example 2-2.

Example 2-2 belowMaximumAge rule coding

```
Definitions
   set maximumAge to 65;
if
   the age of 'the borrower' is more than maximumAge
then
   reject 'the loan' ,
   reason: ("The borrower is older than the maximum age of "+ maximumAge );
```

5. Save the code.

2.7.3 Creating the repaymentLessThanMaximum rule

To create the repaymentLessThanMaximum rule, follow these steps:

1. In the rules project, right-click **decideOnLoan**, and then click **New** → **Action Rule**.
2. In the Name field, enter `repaymentLessThanMaximum`. Then, click **Finish**.
3. The new action rule displays in the Rule Explorer view, and the Intellirule Editor opens.
4. Type the code for the repaymentLessThanMaximum rule, as shown in Example 2-3.

Example 2-3 repaymentLessThanMaximum rule coding

```
if
   the yearly repayment of 'the loan' is more than (0.3 * the yearly income of 'the borrower')
then
   reject 'the loan' ,
   reason: ("The yearly repayment would be more than 30% of the income of the borrower" );
```

5. Save the code.

2.7.4 Creating the aboveMinimumIncome rule

To create the aboveMinimumIncome rule, follow these steps:

1. In the rules project, right-click **decideOnLoan**, and then click **New** → **Action Rule**.
2. In the Name field, enter `aboveMinimumIncome`. Then, click **Finish**.
3. The new action rule displays in the Rule Explorer view, and the Intellirule Editor opens.
4. Type the code for the aboveMinimumIncome rule, as shown in Example 2-4.

Example 2-4 aboveMinimumIncome rule coding

```
Definitions
   set minimumIncome to 24000;
if
   the yearly income of 'the borrower' is less than minimumIncome
then
   reject 'the loan' ,
   reason: ( "The annual income of the borrower is less than the minumum of "+ minimumIncome );
```

5. Save the code.

2.8 Preparing for rule execution

You now need to deploy the rules to the execution server on z/OS.

2.8.1 Creating a RuleApp project

First, create a RuleApp project to contain the rulesets that you want to execute:

1. In the Rule Designer, in the Rule Project Map, in the Deploy and Integrate section, click **Create RuleApp Project**, as shown in Figure 2-40.

Figure 2-40 Create RuleApp project

2. In the New RuleApp Project wizard, enter `loanRequest` in the Project name field, and ensure that the **Use default location** option is selected, as shown in Figure 2-41. Click **Next**.

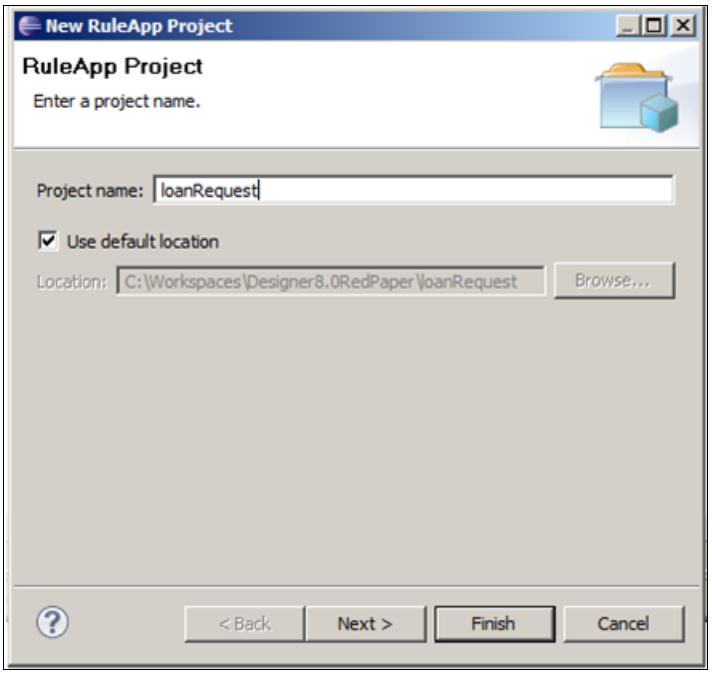


Figure 2-41 Enter a project name

3. The Rule project is listed in the Rule Projects tab, as shown in Figure 2-42.

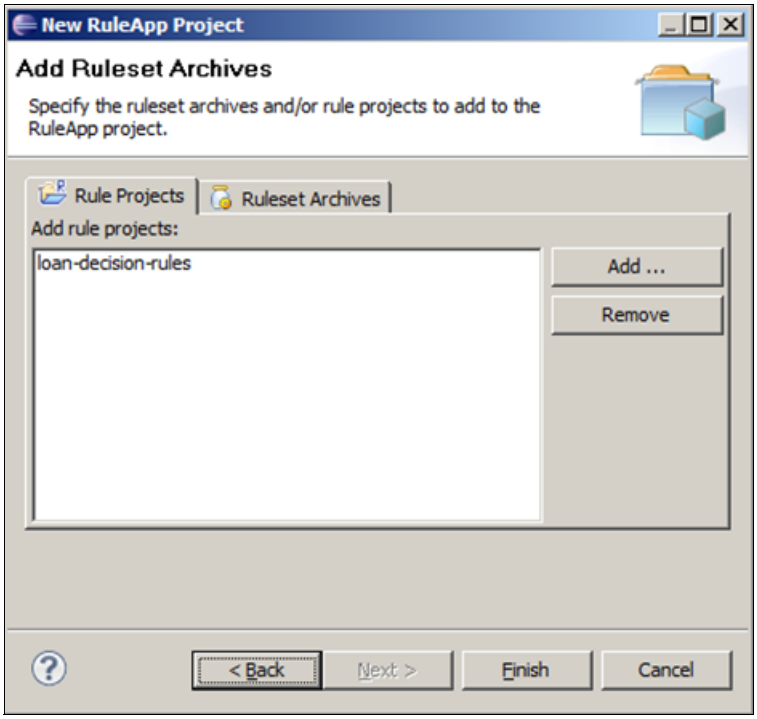


Figure 2-42 Add Ruleset Archives

4. Click **Finish** to create the RuleApp project. It is displayed in the Rule Explorer view, as shown in Figure 2-43.

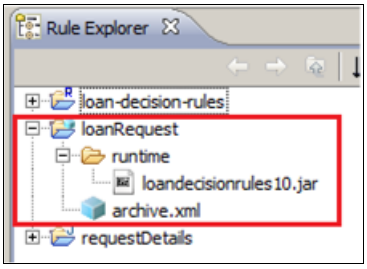


Figure 2-43 Showing the loanRequest RuleApp project

2.8.2 Deploying the RuleApp to the Rule Execution Server for z/OS

In the next step, you have to deploy the Java XOM, the marshaller XOM, and your RuleApp to the Rule Execution Server to be able to execute it:

1. Right-click the RuleApp project **loanRequest** and select **Deploy**, as shown in Figure 2-44.

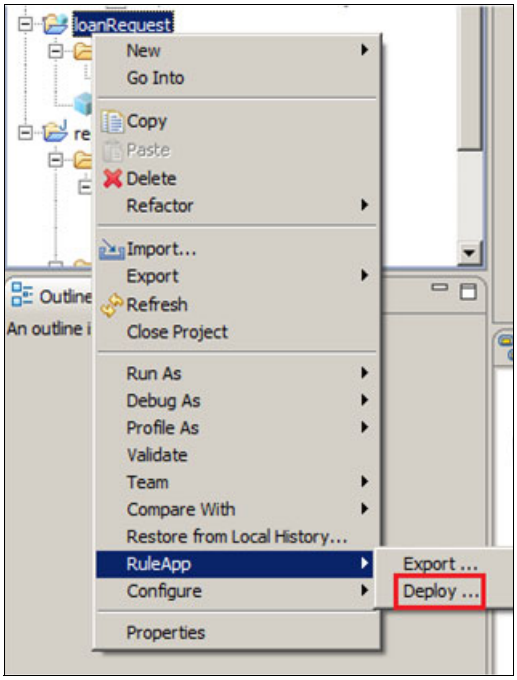


Figure 2-44 Deploy your RuleApp

2. Accept the default option of **Increment RuleApp major version** for the deployment type and click **Next**, as shown in Figure 2-45.

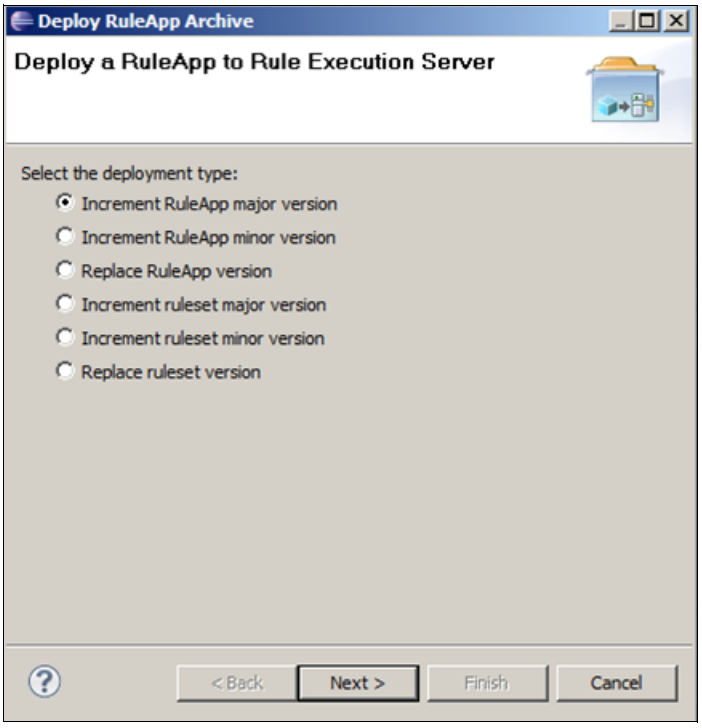


Figure 2-45 Select a deployment type

3. Select **Create a temporary Rule Execution Server configuration** and enter the following details, as shown in Figure 2-46 on page 31:

URL	`http://<yourserveraddress>:<port>/res`
Login	`resAdmin`
Password	`resAdmin`

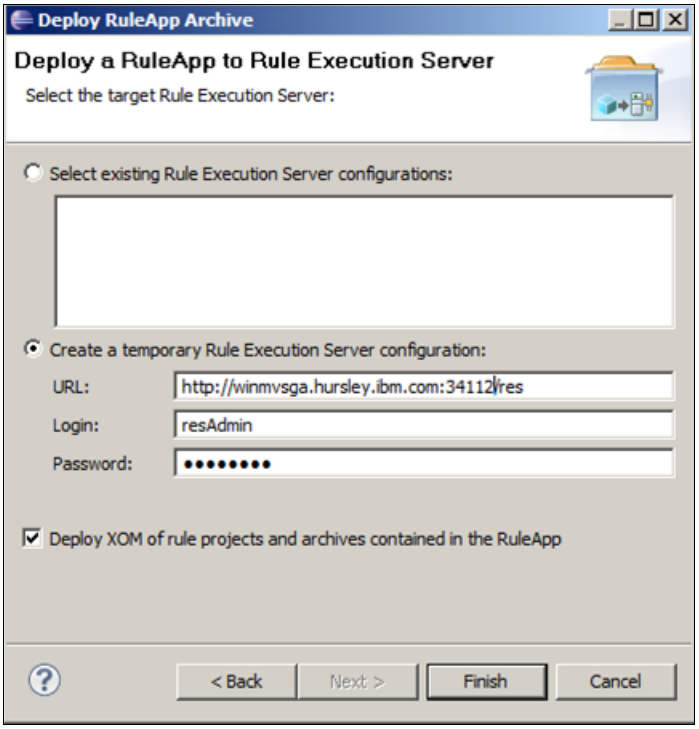


Figure 2-46 Create a temporary Rule Execution Server configuration

4. Click **Finish** to deploy the artifacts to the Rule Execution Server. This deploys the application.

2.8.3 Viewing the deployed rule artifacts in the Rule Execution Server Console

Log in to the Rule Execution Server Console to see the RuleApp and XOM that you have deployed:

1. In a web browser, open the web console for the Rule Execution Server for z/OS by using the URL:

   `http://<yourserveraddress>:<PORT>/res`

2. At the login prompt, enter the User Name `resAdmin` and Password `resAdmin`, as shown in Figure 2-47.

Figure 2-47 Sign in to the Rule Execution Server Console

3. Click **Sign In**, and then select the Explorer tab, as shown in Figure 2-48.

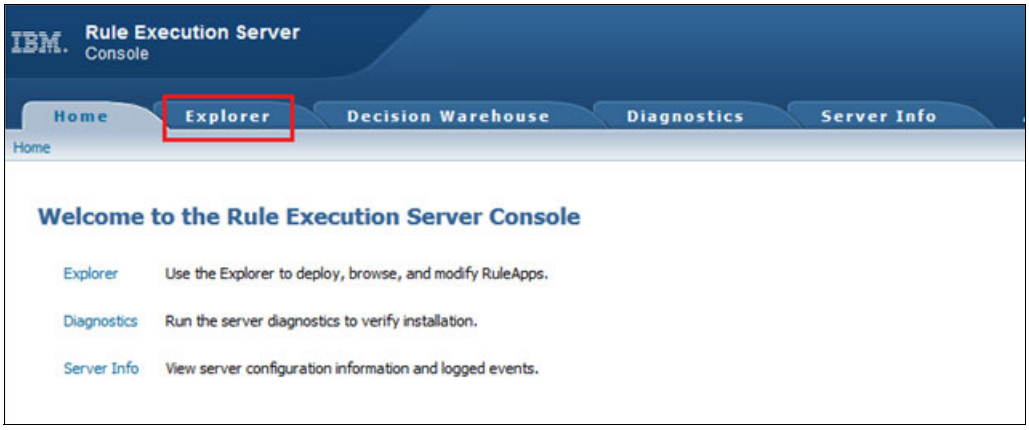


Figure 2-48 Rule Execution Server Console

4. In the Explorer panel, expand **RuleApps** and **Resources** to view the deployed RuleApp and the required resources (Figure 2-49).

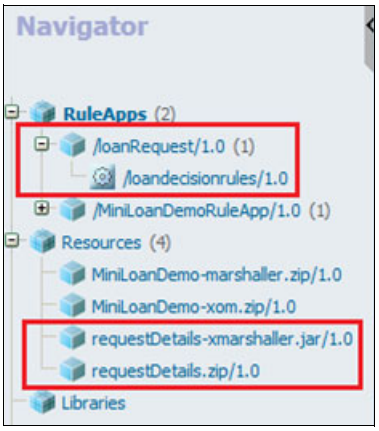


Figure 2-49 Locate your RuleApps and Resources

The rules are now ready to be used.

3

Coding IMS programs to use IBM Operational Decision Manager for rules management

To enable IMS programs to use the Rule Execution Server for z/OS (Rule Execution Server) for rules management, you first need to establish a connection between the program and the server. After that, the program can invoke the Rule Execution Server for rules checking as many times as it needs to. Finally, before the program ends, you need to disconnect it from the server.

This chapter contains details about each of these steps (connection, invocation, and disconnection). First, however, it discusses the COBOL copybooks that are used by the applications program.

3.1 Including copybooks

During each of the steps (connection, invocation, and disconnection), parameters are passed between the program and the Rule Execution Server. This parameter area is defined in Example 3-1.

Example 3-1 HBRA-CONN-AREA definition

```
01 HBRA-CONN-AREA.
    10 HBRA-CONN-EYE                PIC X(4)  VALUE 'HBRC'.
    10 HBRA-CONN-LENGTH             PIC S9(8) COMP  VALUE +3536.
    10 HBRA-CONN-LENTH              REDEFINES HBRA-CONN-LENGTH
                                    PIC S9(8) COMP.
    10 HBRA-CONN-VERSION            PIC S9(8) COMP  VALUE +2.
    10 HBRA-CONN-RETURN-CODES.
       15 HBRA-CONN-COMPLETION-CODE PIC S9(8) COMP  VALUE -1.
       15 HBRA-CONN-REASON-CODE     PIC S9(8) COMP  VALUE -1.
    10 HBRA-CONN-FLAGS              PIC S9(8) COMP  VALUE +1.
    10 HBRA-CONN-INSTANCE.
       15 HBRA-CONN-PRODCODE        PIC X(4) VALUE SPACES.
       15 HBRA-CONN-INSTCODE        PIC X(12) VALUE SPACES.
       15 HBRA-CONN-SSID            PIC X(4) VALUE SPACES.
       15 HBRA-CONN-RESERVED        PIC X(4) VALUE SPACES.
    10 HBRA-RESERVED01              PIC S9(8) COMP  VALUE 0.
    10 HBRA-RESERVED02              PIC S9(8) COMP  VALUE 0.
    10 HBRA-RESERVED03              PIC S9(8) COMP  VALUE 0.
    10 HBRA-CONN-RULE-CCSID         PIC S9(8) COMP  VALUE 0.
    10 HBRA-CONN-RULEAPP-PATH       PIC X(256) VALUE SPACES.
    10 HBRA-RESPONSE-AREA           VALUE SPACES.
       15 HBRA-RESPONSE-MESSAGE     PIC X(1024).
    10 HBRA-RA-INIT                 VALUE LOW-VALUES.
       15 HBRA-RESERVED04           PIC X(1792).
    10 HBRA-RA-PARMETERS            REDEFINES HBRA-RA-INIT.
       15 HBRA-RA-PARMS             OCCURS 32.
          20 HBRA-RA-PARAMETER-NAME PIC X(48).
          20 HBRA-RA-DATA-ADDRESS   USAGE POINTER.
          20 HBRA-RA-DATA-LENGTH    PIC 9(8) BINARY.
    10 HBRA-RESERVED.
      15 HBRA-RESERVED05            PIC X(12).
      15 HBRA-RESERVED06            PIC X(64).
      15 HBRA-RESERVED07            PIC X(64).
      15 HBRA-RESERVED08            PIC X(128).
      15 HBRA-RESERVED09            PIC X(132).
```

This HBRA-CONN-AREA is provided as a copybook in <HBRHLQ>.SHBRCOBS(HBRWS)

After each call (connection, invocation, and disconnection), the success or failure of the call is returned in HBRA-CONN-COMPLETION-CODE.

These are the possible values for HBRA-CONN-COMPLETION-CODE:

```
10 HBR-CC-OK      PIC S9(9) BINARY VALUE 0.
10 HBR-CC-WARNING PIC S9(9) BINARY VALUE 4.
10 HBR-CC-ERROR   PIC S9(9) BINARY VALUE 8.
10 HBR-CC-SEVERE  PIC S9(9) BINARY VALUE 12.
```

If HBRA-CONN-COMPLETION-CODE = HBR-CC-OK, the call was successful.

Otherwise, HBRA-CONN-REASON-CODE and HBRA-RESPONSE-MESSAGE can provide additional information as to why the call was not successful.

Both the completion codes and reason codes are provided as a copybook in <HBRHLQ>.SHBRCOBS(HBRC).

Both the HBRWS and HBRC copybooks have been provided in Appendix A, “COBOL copybooks” on page 45.

3.2 Connecting to the Rule Execution Server

This is the API to connect to the Rule Execution Server:

```
call 'HBRCONN' using HBRA-CONN-AREA
```

It is not necessary to define any additional information in the HBRA-CONN-AREA before issuing the HBRCONN call.

You can configure Rule Execution Server instances as server groups to enable rule execution to be transferred to another server in case a server fails, or if there is a planned outage.

A server group can include 1 - 32 server instances. The list of Rule Execution Servers (HBRSSIDLIST) is specified in a data set pointed to by the HBRENVPR DD statement in your JCL. Appendix B, “HBRENVPR DD statement” on page 51 has more information about the HBRENVPR DD statement. When you issue the HBRCONN call, it establishes the connection with the first available server in the list.

If the HBRCONN was successful, HBR-CONN-INSTANCE contains the details of the Rule Execution Server instance that was selected from the server group defined by the HBRENVPR DD statement.

3.3 Invoking rules in the Rule Execution Server

This is the API to invoke the Rule Execution Server for rules checking:

```
call 'HBRRULE' using HBRA-CONN-AREA
```

These are the steps to invoke rules in the Rule Execution Server:

1. Before issuing the HBRRULE call, you need to specify in the program which rules (or RuleApps) to check. The location of the rules is specified in the following path:

   ```
   HBRA-CONN-RULEAPP-PATH
   ```

2. Specify the input and output parameters associated with the rules in HBRA-RA-PARMS of the HBR-CONN-AREA, as shown in Example 3-2.

Example 3-2 HBRA-RA-PARMS of the HBR-CONN-AREA

```
15 HBRA-RA-PARMS             OCCURS 32.
   20 HBRA-RA-PARAMETER-NAME PIC X(48).
   20 HBRA-RA-DATA-ADDRESS   USAGE POINTER.
   20 HBRA-RA-DATA-LENGTH    PIC 9(8) BINARY
```

You can use up to 32 parameters for input or output. Define these parameters using the structure shown previously in Example 3-2 on page 37, giving the parameter name, its location in storage, and the length of the data that storage contains.

3. You can use the Rule Application to pass back error text or informational messages. This is done in `HBRA-RESPONSE-MESSAGE`.

3.4 Disconnecting from the Rule Execution Server

This is the API to disconnect from the Rule Execution server:

```
call 'HBRDISC' using HBRA-CONN-AREA
```

It is not necessary to define any additional information in the `HBRA-CONN-AREA` before issuing the `HBRDISC` call.

Miniloan application program example

Chapter 2, “Setting up rules in ODM” on page 3, described an example of authoring and deploying rules where a loan company is trying to establish whether a loan request is acceptable according to their business policy.

In this section, you walk through the steps involved in coding a COBOL program that invokes the Rules Execution Server to apply the previous rules.

Specifically, the COBOL program reads the following information pertaining to the borrower and their loan from a file:

- Borrower:
  - Name
  - Credit Score
  - Yearly Income
  - Age
- Loan:
  - Amount
  - Yearly Interest Rate
  - Yearly Repayment
  - Effective Date

The COBOL program then invokes the Rules Execution Server to determine whether the loan is approved.

Follow these steps to disconnect from the Rules Execution Server:

1. When you code COBOL programs that use the Rules Execution Server for rules validation, make sure to include the copybooks:

   ```
   01 WS-REASON-CODES
         COPY HBRC.
         COPY HBRWS.
   ```

2. Include the file record layout copybook, because the program will read (from a file) information about the borrower and their loan:

   ```
   Copy HBRLDAT1.
   ```

3. Establish a connection to the Rules Execution Server using the `HBRCONN` call, and then check the completion and reason codes to confirm that the operation completed successfully. Example 3-3 on page 39 shows that code.

Example 3-3 Check for successful completion and reason codes

```
DISPLAY WS-PROGRAM  '--Connecting to zRule Execution Server'
           call 'HBRCONN' using HBRA-CONN-AREA
           IF HBRA-CONN-COMPLETION-CODE = HBR-CC-ERROR OR
              HBRA-CONN-COMPLETION-CODE = HBR-CC-SEVERE THEN
                 move 'F' to ws-demo-outcome
                 DISPLAY WS-PROGRAM  ' --'
                     'HBRCONN FAILED'
                     '-CC->' HBRA-CONN-COMPLETION-CODE
                     '-RC->' HBRA-CONN-REASON-CODE
                     '-MSG->' HBRA-RESPONSE-MESSAGE
           ELSE
                IF HBRA-CONN-COMPLETION-CODE IS EQUAL TO HBR-CC-WARNING THEN
                  DISPLAY WS-PROGRAM  ' --'
                     'HBRCONN WARNING'
                     '-CC->' HBRA-CONN-COMPLETION-CODE
                     '-RC->' HBRA-CONN-REASON-CODE
                     '-MSG->' HBRA-RESPONSE-MESSAGE
                END-IF
```

4. The program reads the borrower and loan information from the input file, as shown in Example 3-4.

Example 3-4 Reading the input file

```
perform until WS-EOF IS EQUAL TO 'Y'
                  READ SCENARIO-FILE AT END
                    MOVE 'Y' TO WS-EOF
                  END-READ
                  IF NOT-AT-EOF THEN
                    add 1 to ws-customerNumber
                    perform PROCESS-DATA
                  END-IF
                end-perform
```

5. For each input (borrower and their loan), you have to invoke the Rules Execution Server for rules checking. To do that, specify the rules location:

```
MOVE "/MiniLoanDemoRuleApp/MiniLoanDemo" TO HBRA-CONN-RULEAPP-PATH
```

6. Initialize the parameters that are needed for the rules checking, as shown in Example 3-5.

Example 3-5 Initialize the parameters

```
MOVE ALL SPACES TO Borrower Loan
MOVE ALL LOW-VALUES TO HBRA-RA-PARMETERS
move LENGTH OF Borrower to HBRA-RA-DATA-LENGTH(1)
move "borrower"         to  HBRA-RA-PARAMETER-NAME(1)
set HBRA-RA-DATA-ADDRESS(1) to address of Borrower
move LENGTH OF Loan         to HBRA-RA-DATA-LENGTH(2)
multiply length of messages by 99 giving WS-maxMessageLen
add WS-maxMessageLen to HBRA-RA-DATA-LENGTH(2)
move "loan"                 to HBRA-RA-PARAMETER-NAME(2)
set HBRA-RA-DATA-ADDRESS(2) to address of Loan
```

7. Populate the parameters with the borrower and loan information from the input record that was just read, as shown in Example 3-6 on page 40.

Example 3-6 Populate the parameters with the borrower and loan information

```
MOVE ALL LOW-VALUES TO WS-IN
UNSTRING SCENARIO-DATA DELIMITED BY ','
                  INTO
              WS-IN-data(1) WS-IN-data(2) WS-IN-data(3)
              WS-IN-data(4) WS-IN-data(5) WS-IN-data(6)
              WS-IN-data(7) WS-IN-data(8)
 MOVE  WS-IN-data(1)     TO name
 Compute creditScore   = Function numval(WS-IN-data(2))
 Compute yearlyIncome  = Function numval(WS-IN-data(3))
 Compute age           = Function numval(WS-IN-data(4))
 Compute amount        = Function numval(WS-IN-data(5))
 Compute yearlyInterestRate = Function numval(WS-IN-data(6))
 Compute yearlyRepayment = Function numval(WS-IN-data(7))
 MOVE WS-IN-data(8)      TO effectDate
 MOVE 'T'                TO approved
 MOVE 0                  TO messageCount
```

Note: The `approved` parameter has been initialized to **T** and the messageCount to **0**. These parameters will be updated by the `reject` method if it is called; otherwise, the parameters will not be updated and you can assume that the request is successful.

8. Invoke the Rules Execution Server by using the `HBRRULE` call, and then check the completion and reason codes, as shown in Example 3-7.

Example 3-7 Check the completion and reason codes

```
      *
      * Invoke the rule
      *
           DISPLAY WS-PROGRAM
              '--Invoking rules in zRule Execution Server'
           call 'HBRRULE' using HBRA-CONN-AREA
      *
      * Display rule responses, or error code, as appropriate
      *
           IF HBRA-CONN-COMPLETION-CODE = HBR-CC-OK THEN
             DISPLAY WS-PROGRAM  ' --'
                     ' Rule executed in->' HBRA-CONN-SSID
             DISPLAY WS-PROGRAM  '-'
                     '-name->' name
                     ' -loan amount->' amount
                     ' -approved->' approved
             move 1 to ws-msgcount
             display WS-PROGRAM '-'
                     '-messages->'
             perform until ws-msgcount >  messageCount
               display WS-PROGRAM  '->' messages(ws-msgcount)
               add 1 to ws-msgcount
             end-perform
           ELSE
             move 'F' to ws-demo-outcome
             DISPLAY WS-PROGRAM  '-'
             '-CC->' HBRA-CONN-COMPLETION-CODE
             '-RC->' HBRA-CONN-REASON-CODE
```

```
            '-MSG->' HBRA-RESPONSE-MESSAGE
          END-IF.
```

You can find the complete sample miniloan COBOL application program in Appendix C, “Sample miniloan application program” on page 55.

3.5 Connecting to Rule Execution Server with WebSphere z/OS Optimized Local Adapters and IMS

You can also connect to Rule Execution Server for WebSphere Application Server for z/OS by using WebSphere z/OS Optimized Local Adapters (WOLA). This requires that you set up both IMS and the Rule Execution Server to establish a connection with the correct WebSphere Application Server.

3.5.1 Setting up IMS

In order to connect IMS to WOLA, you need to follow these steps:

1. Create an external subsystem IMS PROCLIB member, or update your existing member to include the following entry:

   `WOLA,BBOA,BBOAIEMT`

2. Pass the SSM parameter into your IMS startup data.
3. Include the WOLA load library, created during the WOLA setup, in your IMS Control region startup in both the `STEPLIB` and the `DFSESL DDs`.
4. Restart IMS to pick up the changes.

3.5.2 Setting up Rule Execution Server for WebSphere Application Server for z/OS

Currently, this is only available for message processing programs (MPP) and batch message processing programs (BMP), and not for DL/I programs.

When connecting via WOLA, you need to specify to which WebSphere Application Server system the WOLA is used to connect. You define all of the necessary parameters in a member specified by the `HBRENVPR DD` statement. These consist of the following parameters:

- HBRWOLALOADLIB

  The load library that is created as part of the setting up of WOLA with IMS (see 3.5.1, “Setting up IMS” on page 41): for example, `<HLQ>.WAS.OLA.LOADLIB`

- HBRTARGETRES

  This indicates that the connection is to WOLA rather than the Rule Execution Server for z/OS: enter the value `WOLA`

- HBRWOLACELL

  The cell name of the WebSphere Application Server for the connection: for example, `CSO3A1`

- HBRWOLANODE

  The node name of the WebSphere Application Server for the connection: for example, `NSO3A1`

- HBRWOLASERVER

  The server name of the WebSphere Application Server for the connection: for example, `WSSO3A1`

A typical sample of such a setup can be seen in `<HBRHLQ>.SHBRPARM(HBRWOLA)`.

3.6 Using IMS APIs

The use of the Rule Execution Server for rules management does *not* limit the IMS program in any way. In addition to the API to communicate with the Rule Execution Server, the program can continue to use other APIs, for example, DL/I calls, SQL calls, message queue (MQ) calls, and so on.

3.7 Preparing the program

All IMS application programs need to be link-edited with the DL/I Language interface module (DFSLI000). To do so, follow these steps:

1. Ensure that the link-edit step has the following include:

   ```
   INCLUDE RESLIB (DFSLI000)
   ```

   This is standard procedure for IMS.

   In addition, to resolve the API calls (`HBRCONN, HBRRULE & HBRDISC`), the IMS program also needs to be link-edited with the `HBRISTUB` module.

2. Ensure that the link-edit step also has an `INCLUDE` for `HBRISTUB`:

   ```
   INCLUDE HBRLIB (HBRISTUB)
   ```

Figure 3-1 depicts how the call from the Cobol application goes to the stub module, and the stub module issues a PC call into the Rule Execution Server Address space.

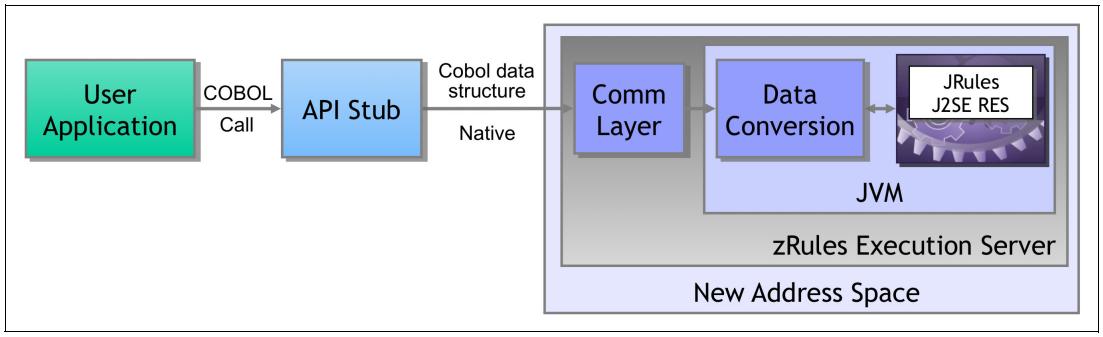


Figure 3-1 Call flow from a COBOL program to the Rules Execution Server

3.8 Changing the execution Job Control Language

You need to make some changes to the execution Job Control Language (JCL). Follow these steps:

1. In the `STEPLIB` of the execution JCL, concatenate the `SHBRLOAD` library. The `SHBRLOAD` library contains the IBM Operational Decision Manager (ODM) Load modules.
2. Ensure that the JCL has an `HBRENVPR DD` statement that points to a data set that specifies which Rules Execution Server to use.

You can run ODM servers within WebSphere Application Server for z/OS, or you can run them stand-alone on z/OS (referred to as *ODM on z/OS)*. These are some of the advantages and disadvantages for these setups:

- ODM for z/OS advantages:
  - Lightweight address space
  - Highest performing option
  - Less real storage consumption
- WebSphere Application Server advantage:

  Ability to run rules engine side-by-side with Java applications
- WebSphere Application Server disadvantages:
  - Not always a preferred solution on z/OS
  - Slower performance than stand-alone ODM on z/OS
  - Heavy architecture (real storage usage, admin costs) to support a rules engine

ODM servers running within WebSphere Application Server for z/OS communicate using WOLA, so the data set that the `HBRENVPR DD` statement points to must contain information about which WebSphere Application Server to connect to. These parameters for WOLA are described in 3.5.2, “Setting up Rule Execution Server for WebSphere Application Server for z/OS” on page 41, and also in “Rules Execution Server on WebSphere Application Server for z/OS” on page 53.

If you are using a stand-alone ODM on z/OS, the data set that the `HBRENVPR DD` statement points to can contain either one or a list of several ODM on z/OS servers.

The reason why you might want to specify a list is because ODM on z/OS Server for instances are configured as server groups to enable rule execution to be transferred to another server in case a server fails, or if there is a planned outage.

Appendix B, “HBRENVPR DD statement” on page 51 provides more information about specifying this list of ODM on z/OS Servers.

Aside from the `SHBRLOAD` library and the `HBRENVPR DD` statement, there are no other required JCL changes.

3.9 Setting up IMS definitions

In this chapter, you have learned about these changes:

- The link-edit step changes (3.7, “Preparing the program” on page 42)
- The execution JCL changes (3.8, “Changing the execution Job Control Language” on page 43)

Aside from these, you do not need to perform any other setup in IMS before you can use the Rules Execution Server from an IMS application program. There are no special definitions in the IMS System definition, or in the IMS startup parameters.

Everything else in IMS is standard. For example, if the COBOL program that uses the Rules Execution Server is a BMP, the definition in the IMS system definition for the BMP does not change. In addition, there are no changes in the program specification block (PSB) associated with the program.

3.10 Conclusion

As demonstrated in this Redpaper, you can use the functionality provided by IBM Operational Decision Manager from IMS COBOL MPP, BMP, and DLIBATCH applications. IBM ODM can provide such applications with a powerful decision engine, which you can use to change rules conveniently to support a quickly-moving business. The steps described in this Redpaper configure IMS to exploit this functionality.

A

COBOL copybooks

This appendix contains the following COBOL copybooks:

- HBRWS copybook
- HBRC copybook

HBRWS copybook

```
01 HBRA-CONN-AREA.
   10 HBRA-CONN-EYE                PIC X(4)  VALUE 'HBRC'.
   10 HBRA-CONN-LENGTH             PIC S9(8) COMP  VALUE +3536.
   10 HBRA-CONN-LENTH              REDEFINES HBRA-CONN-LENGTH
                                   PIC S9(8) COMP.
   10 HBRA-CONN-VERSION            PIC S9(8) COMP  VALUE +2.
   10 HBRA-CONN-RETURN-CODES.
      15 HBRA-CONN-COMPLETION-CODE PIC S9(8) COMP  VALUE -1.
      15 HBRA-CONN-REASON-CODE     PIC S9(8) COMP  VALUE -1.
   10 HBRA-CONN-FLAGS              PIC S9(8) COMP  VALUE +1.
   10 HBRA-CONN-INSTANCE.
      15 HBRA-CONN-PRODCODE        PIC X(4) VALUE SPACES.
      15 HBRA-CONN-INSTCODE        PIC X(12) VALUE SPACES.
      15 HBRA-CONN-SSID            PIC X(4) VALUE SPACES.
      15 HBRA-CONN-RESERVED        PIC X(4) VALUE SPACES.
   10 HBRA-RESERVED01              PIC S9(8) COMP  VALUE 0.
   10 HBRA-RESERVED02              PIC S9(8) COMP  VALUE 0.
   10 HBRA-RESERVED03              PIC S9(8) COMP  VALUE 0.
   10 HBRA-CONN-RULE-CCSID         PIC S9(8) COMP  VALUE 0.
   10 HBRA-CONN-RULEAPP-PATH       PIC X(256) VALUE SPACES.
   10 HBRA-RESPONSE-AREA           VALUE SPACES.
      15 HBRA-RESPONSE-MESSAGE     PIC X(1024).
   10 HBRA-RA-INIT                 VALUE LOW-VALUES.
      15 HBRA-RESERVED04           PIC X(1792).
   10 HBRA-RA-PARMETERS            REDEFINES HBRA-RA-INIT.
      15 HBRA-RA-PARMS             OCCURS 32.
         20 HBRA-RA-PARAMETER-NAME PIC X(48).
         20 HBRA-RA-DATA-ADDRESS   USAGE POINTER.
         20 HBRA-RA-DATA-LENGTH    PIC 9(8) BINARY.
   10 HBRA-RESERVED.
      15 HBRA-RESERVED05            PIC X(12).
      15 HBRA-RESERVED06            PIC X(64).
      15 HBRA-RESERVED07            PIC X(64).
      15 HBRA-RESERVED08            PIC X(128).
      15 HBRA-RESERVED09            PIC X(132).
```

HBRC copybook

```
** Completion codes
      10 HBR-CC-OK      PIC S9(9) BINARY VALUE 0.
      10 HBR-CC-WARNING PIC S9(9) BINARY VALUE 4.
      10 HBR-CC-ERROR   PIC S9(9) BINARY VALUE 8.
      10 HBR-CC-SEVERE  PIC S9(9) BINARY VALUE 12.
   ** Reason codes
   ** The call completed normally.
        10 HBR-RC-NONE PIC S9(9) BINARY VALUE  0.
   ** An unexpected error occurred.
        10 HBR-RC-UNEXPECTED PIC S9(9) BINARY VALUE  2195.
   ** Unable to load Decision Server load modules.
        10 HBR-RC-ERROR-HBRBCON PIC S9(9) BINARY VALUE  3001.
   ** Unable to load Decision Server load modules.
```

```
     10 HBR-RC-ERROR-HBRCCON PIC S9(9) BINARY VALUE  3002.
** Unable to load Decision Server load modules.
     10 HBR-RC-ERROR-HBRBDSC PIC S9(9) BINARY VALUE  3003.
** Unable to load Decision Server load modules.
     10 HBR-RC-ERROR-HBRCDSC PIC S9(9) BINARY VALUE  3004.
** The subsystem specified by the HBRSSID variable in
**  the HBRENVPR data set is not defined to z/OS.
     10 HBR-RC-SERVER-NOT-DEFINED PIC S9(9) BINARY VALUE  3005.
** The subsystem specified by the HBRSSID variable in
**  the HBRENVPR data set is not active.
     10 HBR-RC-SERVER-NOT-ACTIVE PIC S9(9) BINARY VALUE  3006.
** A parameter is incorrectly specified in the HBRA-R
** A-PARMS structure in the HBRA-CONN-AREA data area.
** The number of the invalid parameter is returned
** in the HBRA-RESPONSE-MESSAGE field of the HBRA-CON
** N-AREA data area.
     10 HBR-RC-INVALID-NUMBER-PARMS PIC S9(9) BINARY VALUE  3007.
** The subsystem specified by the HBRSSID variable in
**  the HBRENVPR data set is already in use.
     10 HBR-RC-SERVER-ID-INVALID PIC S9(9) BINARY VALUE  3008.
** The server is unable to accept work because it is
** paused.
     10 HBR-RC-NOT-ACCEPTING-WORK PIC S9(9) BINARY VALUE  3009.
** The CICS version used to connect to the server is
** unsupported. zRule Execution Server for z/OS suppo
** rts only CICS version 3.2, 4.1, and 4.2.
     10 HBR-RC-ERROR-INVALID-CICS PIC S9(9) BINARY VALUE  3010.
** The version of CICS used with HBRUSEJVMS=YES is un
** supported. The HBRUSEJVMS=YES parameter is support
** ed only on CICS TS version 4.1 or later.
     10 HBR-RC-INVALID-JVMS-RELEASE PIC S9(9) BINARY VALUE  3011.
** An attempt to obtain storage has failed. There is
** not enough storage available to complete the reque
** st.
     10 HBR-RC-ERROR-STORAGE-FAIL PIC S9(9) BINARY VALUE  3012.
** The connection to JVM Server HBRJVM has failed as
** the JVM Server is not installed.
     10 HBR-RC-CICS-NO-JVMSERVER PIC S9(9) BINARY VALUE  3013.
** The connection to JVM Server HBRJVM has failed as
** the JVM Server is not enabled.
     10 HBR-RC-CICS-NOTENABLED-JVMS PIC S9(9) BINARY VALUE  3014.
** Unable to load Decision Server load modules.
     10 HBR-RC-ERROR-HBRICON PIC S9(9) BINARY VALUE  3015.
** Unable to load Decision Server load modules.
     10 HBR-RC-ERROR-HBRIDSC PIC S9(9) BINARY VALUE  3016.
** The code page specified in HBRA_CONN_RULE_CCSID is
**  invalid.
     10 HBR-RC-ERROR-INVALID-CCSID PIC S9(9) BINARY VALUE  3017.
** A HBRSSIDLIST value is missing in a data set specif
** ied by the HBRENVPR DD card.
     10 HBR-RC-MISSING-HBRSSIDLIST PIC S9(9) BINARY VALUE  3018.
** The HBRSSIDLIST value is incorrect. Please provide
**  a comma separated list of up to 32 SSIDs of 4 cha
** racters  e.g. HBR1,HBR2,HBR3
     10 HBR-RC-INVALID-HBRSSIDLIST PIC S9(9) BINARY VALUE  3019.
```

```
** No valid zRule Execution Server was found in the H
** BRSSIDLIST property.
     10 HBR-RC-NO-VALID-SERVER-CONN PIC S9(9) BINARY VALUE  3020.
** A zRule Execution Server in the HBRSSIDLIST proper
** ty could not be connected to.
     10 HBR-RC-WARNING-SERVER-LIST PIC S9(9) BINARY VALUE  3021.
** No valid zRule Execution Server was found to execu
** te the rule request.
     10 HBR-RC-NO-VALID-SERVER-RULE PIC S9(9) BINARY VALUE  3022.
** Could not open HBRENVPR which is defined in the HB
** RENVPR DD statement in the CICS JCL.
     10 HBR-RC-INVALID-CICS-ENV-DD PIC S9(9) BINARY VALUE  3023.
** The user ID of the application issuing the HBRCONN
**  API call is not authorized to connect to the serv
** er.
     10 HBR-RC-CONN-NOT-AUTH PIC S9(9) BINARY VALUE  4035.
** Missing Ruleapp name.
      10 HBR-RC-MISSING-RA-NAME PIC S9(9) BINARY VALUE  4084.
** Invalid Ruleapp name.
   10 HBR-RC-INVALID-RA-NAME PIC S9(9) BINARY VALUE  4085.
** CICS is not connected to zRule Execution Server fo
** r z/OS.
     10 HBR-RC-ADAPTER-NOT-AVAILABLE PIC S9(9) BINARY VALUE  4086.
** The HBRC structure in the HBRA-CONN-AREA data area
**  passed either to the HBRCONN API or the HBRRULE A
** PI is invalid or contains invalid data.
     10 HBR-RC-INVALID-HBRC PIC S9(9) BINARY VALUE  4087.
** The user ID is already connected to the server. Th
** e user ID is passed back in the HBRA-RESPONSE-MESS
** AGE field of the HBRA-CONN-AREA data area.
     10 HBR-RC-ALREADY-CONNECTED PIC S9(9) BINARY VALUE  4088.
** A HBRSSID value is missing in a data set specified
** by the HBRENVPR DD card.
     10 HBR-RC-MISSING-HBRSSID PIC S9(9) BINARY VALUE  4089.
** The application is not connected to the server.
     10 HBR-RC-NOT-CONNECTED PIC S9(9) BINARY VALUE  4090.
** An unexpected exception occurred in the JRules eng
** ine.
     10 HBR-RC-JRULES-UNEXPECTED PIC S9(9) BINARY VALUE  5000.
** An exception occurred in the JRules engine.
     10 HBR-RC-JRULES-EXCEPTION PIC S9(9) BINARY VALUE  5001.
** An exception occurred converting the parameter dat
** a.
   10 HBR-RC-RAW-DATA-EXCEPTION PIC S9(9) BINARY VALUE  5002.
** An exception occurred parsing the rule application
**  path.
     10 HBR-RC-ILR-FORMAT-EXCEPTION PIC S9(9) BINARY VALUE  5003.
** Unable to load WOLA load modules.
     10 HBR-RC-WOLA-LOAD PIC S9(9) BINARY VALUE  6000.
** The WAS server specified by HBRWOLASERVER cannot b
** e located.
     10 HBR-RC-WOLA-BAD-DAEMON-GROUP PIC S9(9) BINARY VALUE  6001.
** The node name or server name is not found.
     10 HBR-RC-WOLA-BAD-CELL-OR-NODE PIC S9(9) BINARY VALUE  6002.
** An error occurred while connecting to WAS.
```

```
     10 HBR-RC-WOLA-WAS-ERROR PIC S9(9) BINARY VALUE  6003.
** The WOLA EJB could not be found.
     10 HBR-RC-WOLA-EJB-NOT-FOUND PIC S9(9) BINARY VALUE  6004.
```

B

HBRENVPR DD statement

This appendix discusses the HBRENVPR DD statement for the following two cases:

- Rules Execution Server on z/OS
- Rules Execution Server on WebSphere Application Server for z/OS

Rules Execution Server on z/OS

Rule Execution Server for z/OS instances are configured as server groups to enable rule execution to be transferred to another server in case a server fails, or if there is a planned outage.

A server group can include from 1 - 32 server instances. You specify the list of Rule Execution servers in a data set pointed to by the HBRENVPR DD statement in your job control language (JCL). There are several ways to code this list in the HBRENVPR DD data set.

You can code:

```
HBRSSIDLIST=++HBRSSIDLIST++
```

This means that the list comes from the server group membership list variable, `++HBRSSIDLIST++`, in the `SHBRPARM(HBRINST)` data set member.

Alternatively, you can specify your own list, for example:

```
HBRSSIDLIST=HBR1,HBR2,HBR3
```

Here the server group consists of three servers whose subsystem IDs are HBR1, HBR2, and HBR3.

The execution JCL for an IMS application program that uses the Rule Execution Server for Rules checking must include a data set pointed to by the HBRENVPR DD statement.

Rule execution begins on the first available server in the list. Other servers execute rulesets only if rule execution is transferred to them. To route rule execution to a particular server, specify its ID first.

If a server crashes, hangs, abnormally ends, or shuts down, rule execution transfers automatically to the next active server in the list. The inactive server remains in the server group, and when it restarts, rule execution automatically transfers back to the original server. No action is required by the COBOL application to accomplish these transfers.

Rules Execution Server on WebSphere Application Server for z/OS

If you are using WebSphere z/OS Optimized Local Adapters (WOLA), a different set of parameters is required. You can see an example of this in the `SHBRPARM(HBRWOLA)` data set member. These values are required to identify the specific WOLA in use to the COBOL application.

This data set consists of the following parameters:

- `HBRWOLALOADLIBRARY`, the WOLA load library, for example:

  ```
  HBRWOLALOADLIBRARY=USER.V80.WOLA.LOADLIBRARY
  ```

- `HBRTARGETRES`, an indication that you are connecting to WebSphere Application Server via WOLA, for example:

  ```
  HBRTARGETRES=WOLA
  ```

- `HBRWOLACELL`, the short name of the WebSphere Application Server cell in use, for example:

  ```
  HBRWOLACELL=CIL1
  ```

- `HBRWOLANODE`, the short name of the WebSphere Application Server node in use, for example:

  ```
  HBRWOLANODE=NIL1
  ```

- `HBRWOLASERVER`, the WebSphere Application Server server name, for example:

  ```
  HBRWOLASERVER=server1
  ```

C

Sample miniloan application program

This appendix contains the source code for a sample miniloan application program, as shown in Example C-1.

Example C-1 Sample miniloan application program

```
* @START_COPYRIGHT_NONOCO@
* Licensed Materials - Property of IBM
*
* 5655-Y07, 5655-ILG
* (c) Copyright IBM Corp. 2011  All Rights Reserved.
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
* @END_COPYRIGHT_NONOCO@

 IDENTIFICATION DIVISION.
PROGRAM-ID. HBRMINI.

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT SCENARIO-FILE
ASSIGN TO S-SCENARIO
ORGANIZATION IS SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD  SCENARIO-FILE
RECORDING MODE IS F
LABEL RECORD STANDARD.
01 SCENARIO-AREA.
05 SCENARIO-DATA            PIC X(496).

WORKING-STORAGE SECTION.
```

```
01 WS-EOF          PIC X    VALUE "N".
88 AT-EOF                VALUE "Y".
88 NOT-AT-EOF            VALUE "N".
01  WS-IN.
05  WS-IN-data    PIC  X(50) occurs 30 times.
01 MY-LOCAL-STORAGE.
02 WS-PROGRAM     PIC X(8) VALUE "MINILOAN".
02 FILLER         PIC X(4) VALUE "-WS>".
COPY HBRLDAT1.
01 WS-maxMessageLen   PIC 9(10).
01 ws-customerNumber  PIC 9(4) VALUE ZERO.
01 ws-msgcount        PIC 9(10).
01 ws-demo-outcome    PIC X value "T".
01 WS-REASON-CODES.
COPY HBRC.
COPY HBRWS.
PROCEDURE DIVISION.
Display WS-PROGRAM  ' -Miniloan Demo on zOS Batch '
*
* Open input file
*
OPEN INPUT SCENARIO-FILE
*
* Connect
*
DISPLAY WS-PROGRAM  '--Connecting to zRule Execution Server'
call 'HBRCONN' using HBRA-CONN-AREA
IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK THEN
move 'F' to ws-demo-outcome
DISPLAY WS-PROGRAM  '-'
   'HBRCONN FAILED'
   '-CC->' HBRA-CONN-COMPLETION-CODE
   '-RC->' HBRA-CONN-REASON-CODE
ELSE
perform until WS-EOF IS EQUAL TO 'Y'
     READ SCENARIO-FILE AT END
    MOVE 'Y' TO WS-EOF
        END-READ
        IF NOT-AT-EOF THEN
     add 1 to ws-customerNumber
     perform PROCESS-DATA
     END-IF
end-perform
END-IF
*
* Disconnect
*
DISPLAY WS-PROGRAM
' --disconnecting from zRule Execution Server'
call 'HBRDISC' using HBRA-CONN-AREA
IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK THEN
move 'F' to ws-demo-outcome
    DISPLAY WS-PROGRAM  '-'
   'HBRDISC FAILED'
   '-CC->' HBRA-CONN-COMPLETION-CODE
```

```
   '-RC->' HBRA-CONN-REASON-CODE
END-IF
*
* Close input file
*
CLOSE SCENARIO-FILE
if ws-demo-outcome is equal to 'T'
display WS-PROGRAM ' --SUCCESSFUL COMPLETION of demo'
MOVE ZERO to RETURN-CODE
else
display WS-PROGRAM ' --demo completed with ERRORS'
display WS-PROGRAM ' --please review logs and rerun '
display WS-PROGRAM ' --with trace as needed.'
MOVE 8 to RETURN-CODE
end-if
MOVE ZERO to RETURN-CODE
GOBACK.
*
* For each scenario....
*
PROCESS-DATA.
*
* Initialize call parameters
*
MOVE ALL SPACES TO Borrower Loan
MOVE ALL LOW-VALUES TO HBRA-RA-PARMETERS
MOVE "/MiniLoanDemoRuleApp/MiniLoanDemo" TO
HBRA-CONN-RULEAPP-PATH
move LENGTH OF Borrower to HBRA-RA-DATA-LENGTH(1)
move "borrower"         to  HBRA-RA-PARAMETER-NAME(1)
set HBRA-RA-DATA-ADDRESS(1) to address of Borrower
move LENGTH OF Loan         to HBRA-RA-DATA-LENGTH(2)
multiply length of messages by 99 giving WS-maxMessageLen
add WS-maxMessageLen to HBRA-RA-DATA-LENGTH(2)
move "loan"                 to HBRA-RA-PARAMETER-NAME(2)
set HBRA-RA-DATA-ADDRESS(2) to address of Loan
*
* Read scenario data
*
MOVE ALL LOW-VALUES TO WS-IN
UNSTRING SCENARIO-DATA DELIMITED BY ','
       INTO
WS-IN-data(1) WS-IN-data(2) WS-IN-data(3)
WS-IN-data(4) WS-IN-data(5) WS-IN-data(6)
WS-IN-data(7) WS-IN-data(8)
MOVE  WS-IN-data(1)     TO name
Compute creditScore   = Function numval(WS-IN-data(2))
Compute yearlyIncome  = Function numval(WS-IN-data(3))
Compute age           = Function numval(WS-IN-data(4))
Compute amount        = Function numval(WS-IN-data(5))
Compute yearlyInterestRate = Function numval(WS-IN-data(6))
Compute yearlyRepayment = Function numval(WS-IN-data(7))
MOVE WS-IN-data(8)      TO effectDate
MOVE 'T'                TO approved
MOVE 0                  TO messageCount
```

```
display WS-PROGRAM  ' --Loan customer ' ws-customerNumber
*
* Invoke the rule
*
DISPLAY WS-PROGRAM
'--Invoking rules in zRule Execution Server'
call 'HBRRULE' using HBRA-CONN-AREA
*
* Display rule responses, or error code, as appropriate
*
IF HBRA-CONN-COMPLETION-CODE = HBR-CC-OK THEN
   DISPLAY WS-PROGRAM  '-'
   '-name->' name
   ' -loan amount->' amount
   ' -approved->' approved
move 1 to ws-msgcount
display WS-PROGRAM '-'
      '- messages->'
perform until ws-msgcount >  messageCount
display WS-PROGRAM  '->' messages(ws-msgcount)
add 1 to ws-msgcount
end-perform
ELSE
move 'F' to ws-demo-outcome
DISPLAY WS-PROGRAM  '-'
'-CC->' HBRA-CONN-COMPLETION-CODE
'-RC->' HBRA-CONN-REASON-CODE
'-MSG->' HBRA-RESPONSE-MESSAGE
END-IF
```

D

JCL to run an IMS DLIBatch program or BMP that uses Rule Execution Server for z/OS

This appendix contains Job Control Language (JCL) code to run an IMS DLIBatch or batch message processing program (BMP) that uses Rule Execution Server for z/OS, which is shown in Example D-1.

Example D-1 JCL to run an IMS DLIBatch or BMP that uses Rule Execution Server for z/OS

```
//HBRMINI JOB NOTIFY=&SYSUID,MSGCLASS=A
//*********************************************************************
//* @START_COPYRIGHT_NONOCO@                                          *
//* Licensed Materials - Property of IBM                              *
//*                                                                   *
//* 5655-Y17, 5655-Y31                                                *
//* (c) Copyright IBM Corp. 2012  All Rights Reserved.                *
//* US Government Users Restricted Rights - Use, duplication or       *
//* disclosure restricted by GSA ADP Schedule Contract with           *
//* IBM Corp.                                                         *
//* @END_COPYRIGHT_NONOCO@                                            *
//*********************************************************************
//JCLLIB JCLLIB ORDER=(++IMSREGHLQ++.PROCLIB)
//*
//*Select whether BMP or DLI is required.
//*For DLI, replace <IMSPLEX> with the name of your IMSPLEX and
//*uncomment the additional STEPLIB line below.
//HBRSAMP2 EXEC PROC=IMSBATCH,
//*HBRSAMP2 EXEC PROC=DLIBATCH,IMSPLEX=<IMSPLEX>,
//                     MBR=HBRMINI,PSB=HBRMINI,
//                     IMSID=++IMSREGID++,SOUT='*',TIME=(60)
//         SET HBRHLQ=++HBRHLQ++
//         SET HBRWDS=++HBRWORKDS++
//         SET IMSHLQ=++IMSHLQ++
```

```
//         SET IMSREG=++IMSREGHLQ++
//STEPLIB  DD DISP=SHR,DSN=&HBRHLQ..SHBRLOAD
//         DD DSN=&IMSHLQ..&SYS2.SDFSRESL,DISP=SHR
//         DD DSN=&IMSREG..PGMLIB,DISP=SHR
//*Uncomment the following line if running DLI.
//*           DD DSN=&IMSREG..MDALOCAL,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//* ADDITIONAL DD STATEMENTS
//*
//IMS      DD DISP=SHR,DSN=&IMSREG..PSBLIB
//         DD DISP=SHR,DSN=&IMSREG..DBDLIB
//DFSSTAT  DD SYSOUT=*
//*
//*INPUT FOR MINILOAN PROGRAM.
//SCENARIO DD DISP=SHR,DSN=&HBRWDS..SHBRPARM(HBRSCEN)
//*IDENTIFY THE ZRULES SERVER.
//HBRENVPR DD DISP=SHR,DSN=&HBRWDS..SHBRPARM(HBRBATCH)
//*
//*********************************************************************
//* See miniloan-test.xls
//* name,creditscore,yearlyIncome,age,amount,
//* yearlyinterestRate,yearlyRepayment, effectiveDate
//*
```

E

Further rules definitions

You can write rules in a variety of ways, not all of which are described in the main part of this paper. This appendix describes a few ways to write rules.

Decision tables

A *decision table* is concerned with decisions that are more than 2-way, and allow multiple responses to the rules to be considered. The best way to illustrate the decision table is with an example similar to the one used in Chapter 2, “Setting up rules in ODM” on page 3.

A loan request is made, and whether it is approved or rejected is based on two factors, the *amount of the loan* and the *age of the customer*, based on the following table delete, as shown in Table E-1.

Table E-1 Decision factors for loan

Amount of loan	Age	Decision
Less than the yearly income of borrower	<=21	Approved
	21< age <=60	Approved
	>60	Approved
More than the yearly income but less than 3x yearly income	<=21	Rejected
	21< age <=60	Approved
	>60	Approved
Greater than the yearly income	<=21	Rejected
	21< age <=60	Approved
	>60	Rejected

This information can be expressed by using a decision table, which makes it easier for you to view. Follow these steps:

1. In the Rules Designer, click **New** → **Create Decision Table**, as shown in Figure E-1**.**

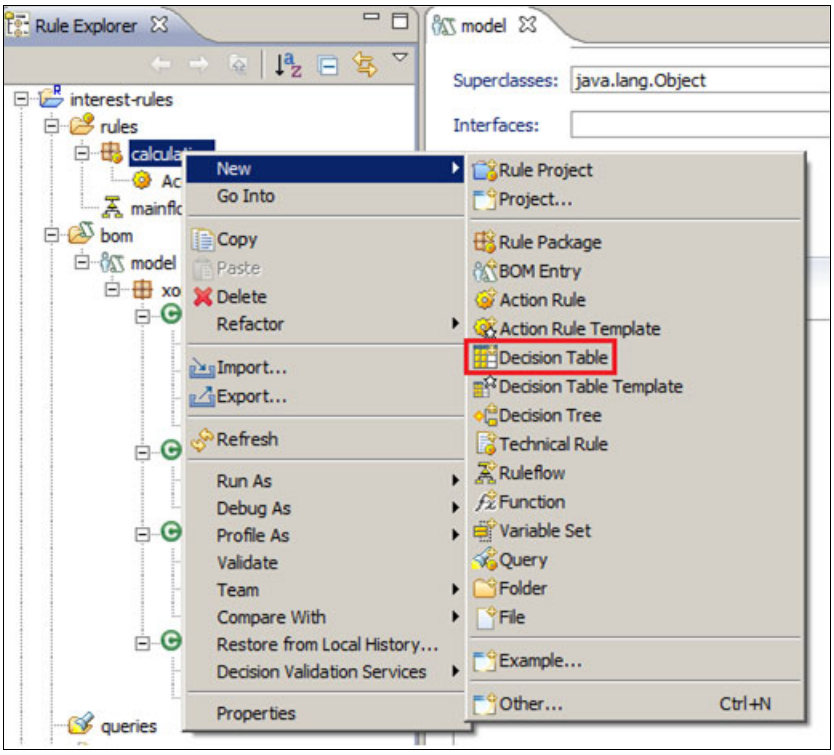


Figure E-1 Create a decision table

2. Select a source folder, enter the name `DecideOnLoan`, and click **Finish**, as shown in Figure E-2.

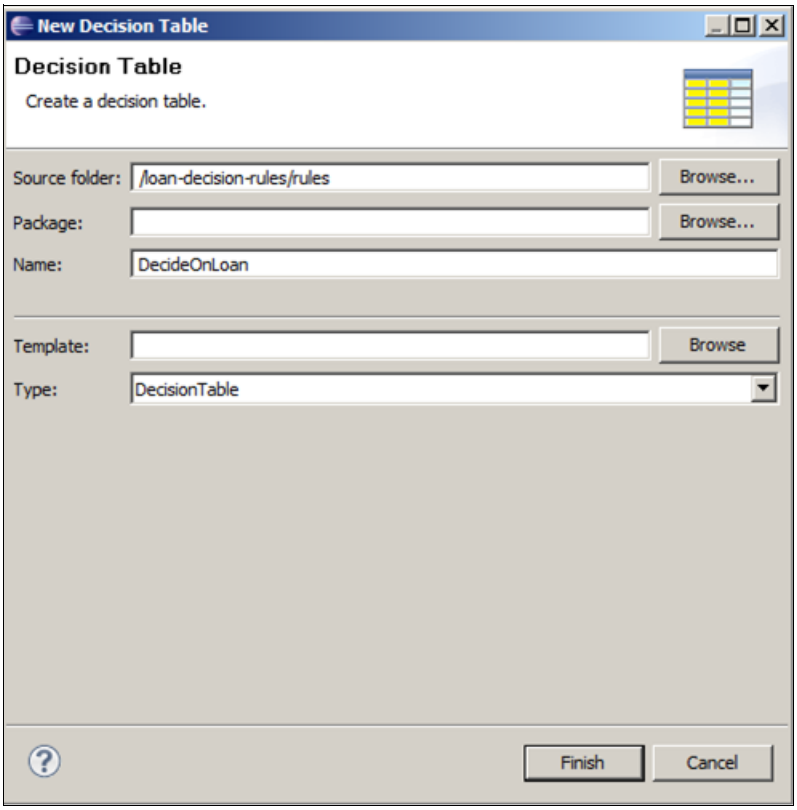


Figure E-2 New decision table details

3. There are two columns in your decision table, and the default created is three, so remove one of the condition columns. Right-click column C and select **Remove Condition Column**, as shown in Figure E-3.

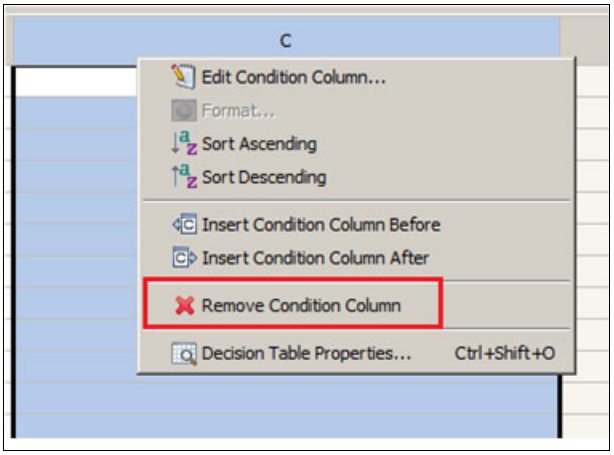


Figure E-3 Remove a condition column

4. To display the Condition Definition panel, right-click the heading of column A and select **Edit Condition Column**, as shown in Figure E-4.

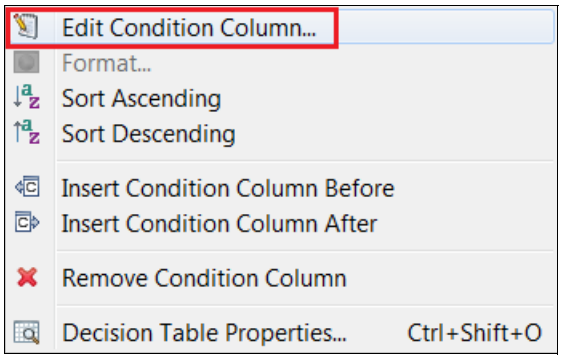


Figure E-4 Edit a condition column

5. Enter these values, as shown in Figure E-5. You can use the editor to help enter this information:
   a. In the Test field, enter `the amount of 'the loan' is more than <min> and at most <max>`.
   b. In the Title field, enter `Amount`.
   c. In the Expression Placeholders field, enter `<> a boolean [a boolean]`.

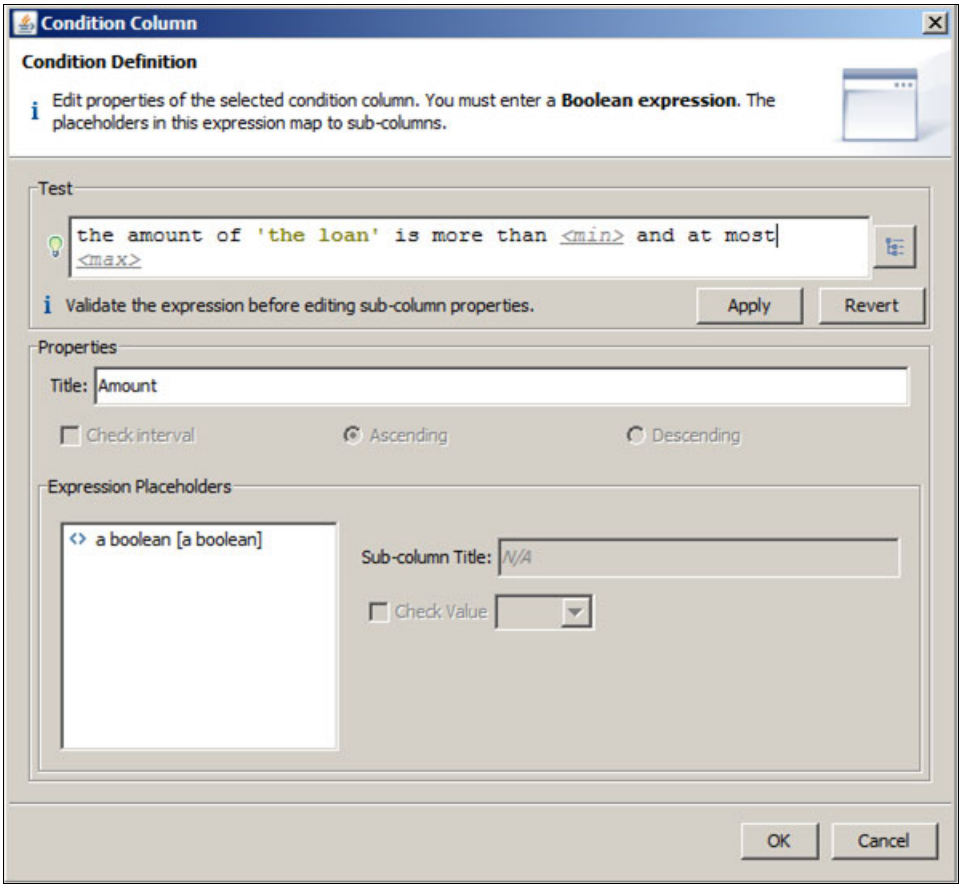


Figure E-5 Values for the Amount condition column

6. Click **OK** to populate the column.
7. In row 1, column A, right-click and select **Operator** and then select the less than or equal to symbol (**<=**), as shown in Figure E-6.

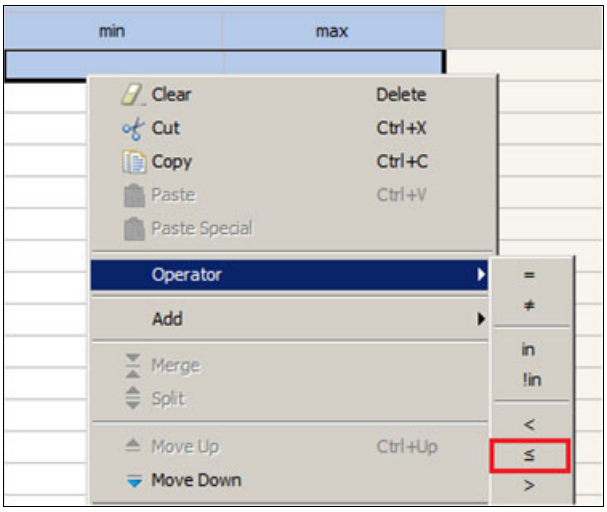


Figure E-6 Select an operator

8. Repeat the process for row 3, except this time, select the greater than symbol (**>**).
9. Populate the values in this column. Click row 1, column Amount to display the rule, as shown in Figure E-7.

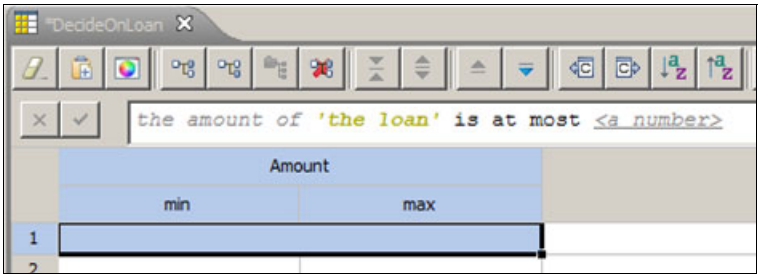


Figure E-7 Populating values

10. Click ***<a number>*** to display the help. Select **the yearly income of <a borrower>**, as shown in Figure E-8.

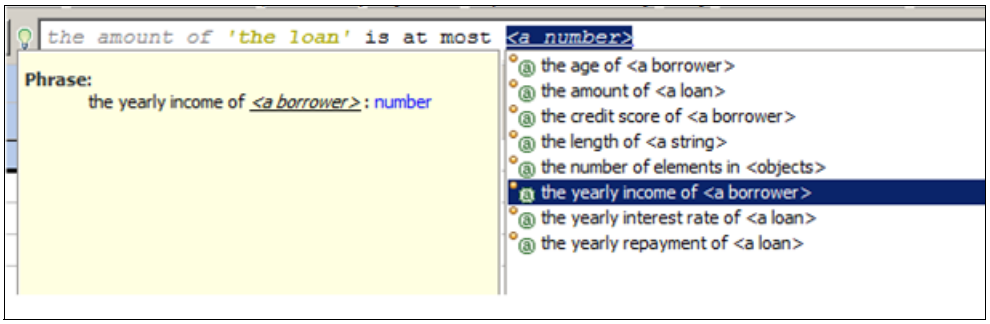


Figure E-8 Selecting a value

11. Enter `the borrower`, so that the decision is `the amount of 'the loan' is at most the yearly income of 'the borrower'`, as shown in Figure E-9.

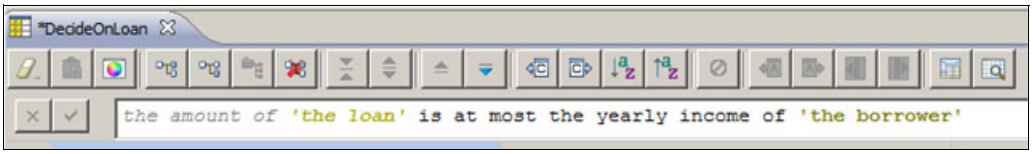


Figure E-9 Entering a value

12. Repeat this process to populate the cells, as shown in Figure E-10.

Amount	
min	max
≤ the yearly income of 'the borrower'	
the yearly income of 'the borrower'	*the yearly income of 'the borrower' * 3*
*> the yearly income of 'the borrower' * 3*	

Figure E-10 Completed values

13. Select column B, and populate the Condition Column box using the following information, as shown in Figure E-11 on page 68:
    a. In the Test field, enter `the age of 'the borrower' is more than <min> and at most <max>`.
    b. In the Title field, enter `Age`.

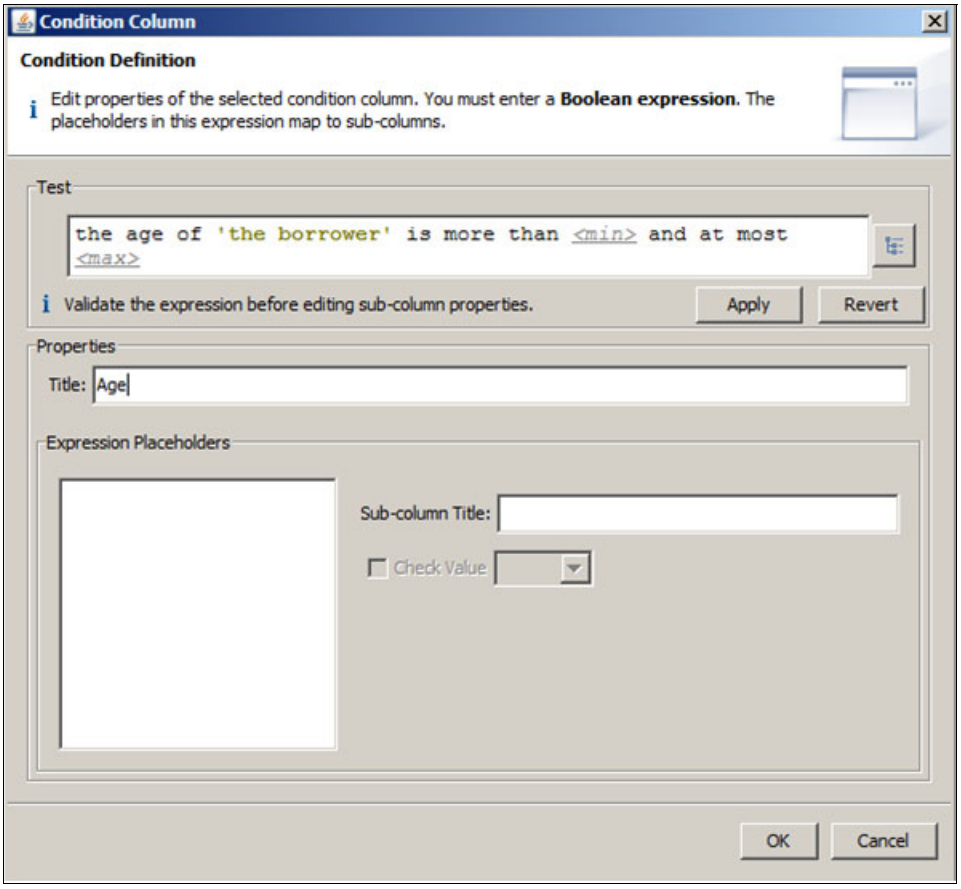


Figure E-11 Condition Column values

14. In row 1, column Age, right-click and select **Add** → **Insert New Row After**, as shown in Figure E-12.

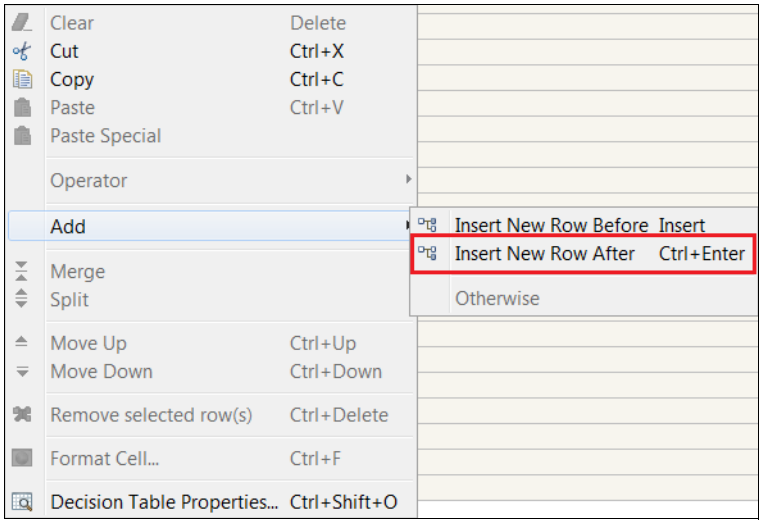


Figure E-12 Insert new row to decision table

15. Repeat this step to add an extra row. Add more rows so that there are three rows opposite each of the three possible entries in the Amount row. The table now shows columns for Amount (with min and max) and Age (with min and max), as shown in Figure E-13. There are three possibilities (rows) for Age next to each possible Amount (income).

Amount		Age	
min	max	min	max
≤ the yearly income of 'the borrower'			
the yearly income of 'the borrower'	*the yearly income of 'the borrower' * 3*		
*> the yearly income of 'the borrower' * 3*			

Figure E-13 Adding more rows

16. Add the less than or equal to (**<=**) operator to row 1 for Age, and the greater than (**>**) operator to row 3. Enter 21 for the minimum and 60 for the maximum, as shown in Figure E-14.

Age	
min	max
≤ 21	
21	60
> 60	

Figure E-14 Age values and operators

17. Use copy and paste operations to copy the contents of these cells to the corresponding cells beneath them, so that each set of rows has the same values, as shown in Figure E-15.

Amount		Age	
min	max	min	max
≤ the yearly income of 'the borrower'		≤ 21	
		21	60
		> 60	
the yearly income of 'the borrower'	*the yearly income of 'the borrower' * 3*	≤ 21	
		21	60
		> 60	
*> the yearly income of 'the borrower' * 3*		≤ 21	
		21	60
		> 60	

Figure E-15 Age values and operators filled across decision table

18. You are now ready to populate the actions. First, create a new action column. Right-click the header of column C, and select **Insert Action Column After**, as shown in Figure E-16.

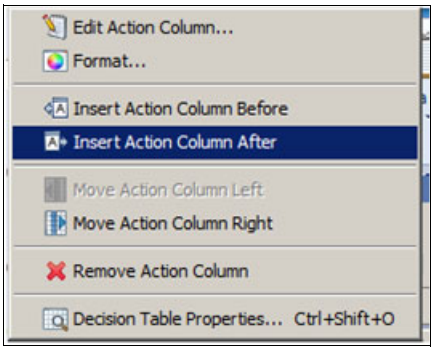


Figure E-16 Insert an action column

19. Double-click the header of column C and enter the action, as shown in Figure E-17, using this information:

    a. In the Action field, enter `reject 'the loan' , reason: (<a string>)`.
    b. In the Title field, enter `RejectLoan`.
    c. Select **Visible**.
    d. In the Expression Placeholders field, enter `<> a string [a string]`.

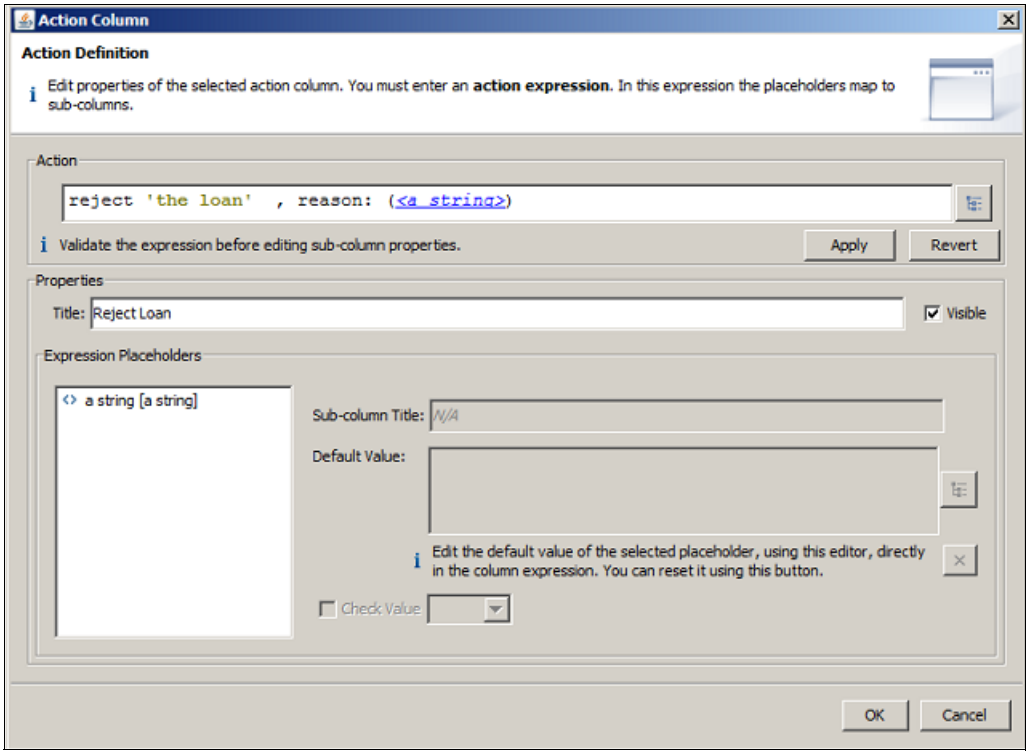


Figure E-17 Action Definition - RejectLoan

20. Select **OK**. Enter the messages for each of the three cells, as shown in Figure E-18 on page 71:

    a. For the first cell, enter `Borrower is below the age limit for this amount.`
    b. For the second cell, enter `Borrower is below the age limit for this amount.`

c. For the third cell, enter `Borrower is above the age limit for this amount.`

Amount		Age		Reject Loan
min	max	min	max	
≤ the yearly income of 'the borrower'		≤ 21		
		21	60	
		> 60		
the yearly income of 'the borrow...	the yearly income of 'the borrow...	≤ 21		Borrower is below the age limit for this amount
		21	60	
		> 60		
> the yearly income of 'the borrower' * 3		≤ 21		Borrower is below the age limit for this amount
		21	60	
		> 60		Borrower is above the age limit for this amount

Figure E-18 Entering messages for RejectLoan

21. In each of the remaining cells, right-click and select **Enable / Disable Action** to disable this action for each of these cells, as shown in Figure E-19.

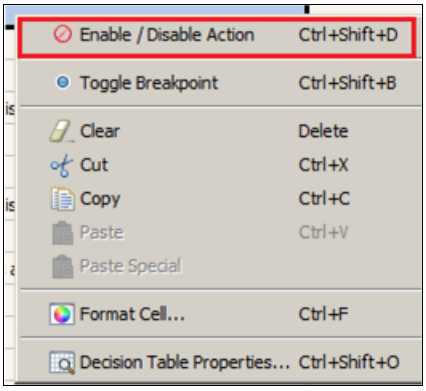


Figure E-19 Enable or disable the action

22. Double-click column D and enter information in the Action Column panel, as shown in Figure E-20 on page 72:
    a. In the Action field, enter `set the approved of 'the loan' to <a string>`.
    b. In the Properties field, enter `Decision`.
    c. In the Expression Placeholders field, enter `<> a string [a string]`.

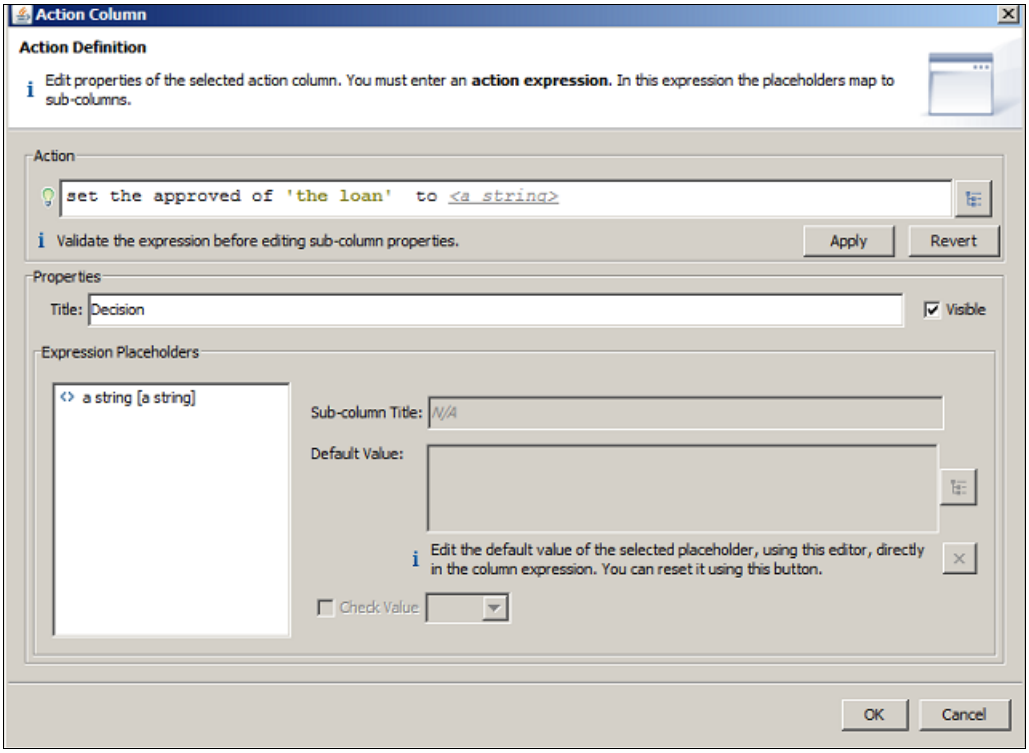


Figure E-20 Add an Action and a Title

23. Click **OK** and populate the strings in the column, disabling the action in the appropriate cells, as shown in Figure E-21. The cells in the Reject Loan column without a message are disabled, and their corresponding cells in the Decision column have a value of T.

Amount		Age		Reject Loan	Decision
min	max	min	max		
≤ the yearly income of 'the borrower'		≤ 21		⊘	T
		21	60	⊘	T
		> 60		⊘	T
the yearly income of...	the yearly income of ...	≤ 21		Borrower is below the age limit for this amount	
		21	60	⊘	T
		> 60		⊘	T
> the yearly income of 'the borrower' * 3		≤ 21		Borrower is below the age limit for this amount	
		21	60	⊘	T
		> 60		Borrower is above the age limit for this amount	

Figure E-21 Configure Actions for the Decision column

24. Save the decision table.

You have created a 9-way decision matrix based on two parameters, *age* and *amount of loan*. Clearly, it is possible to extend this decision table much further to make more fine-grained decisions. In some circumstances, it is easier to view the rules in this way rather than using a rule flow. Not only does this method produce the decision, it also sends a message explaining any rejections.

Decision trees

A *decision tree* describes rules in a tree-like structure, providing an alternative way of viewing and managing sets of business rules. Conditions are declared in a diamond-shaped node. The branches of the tree represent the possible conditions, and the actions are declared at the end of those branches.

To create a decision tree, use the identical rules that you used to create the decision table in Table E-1 on page 62.

When you create your own rules, you can use whichever design is most appropriate for your configuration. Follow these steps:

1. In the Rules Designer, right-click the **loan-decision-rules** project and select **New** → **Decision Tree**, as shown in Figure E-22.

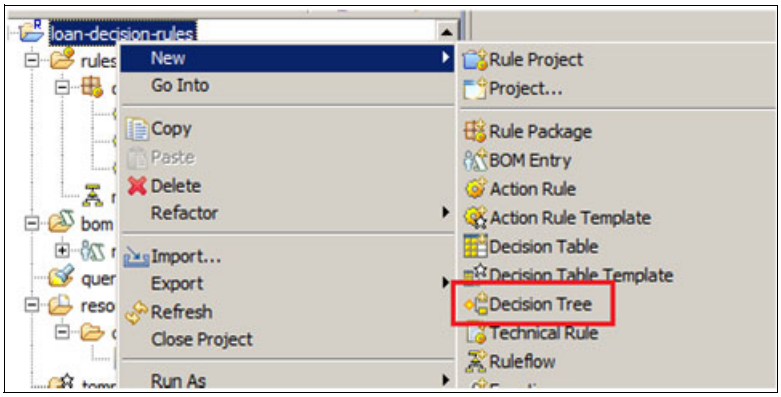


Figure E-22 Create a decision tree

2. Enter the name `LoanDecision`, as shown in Figure E-23.

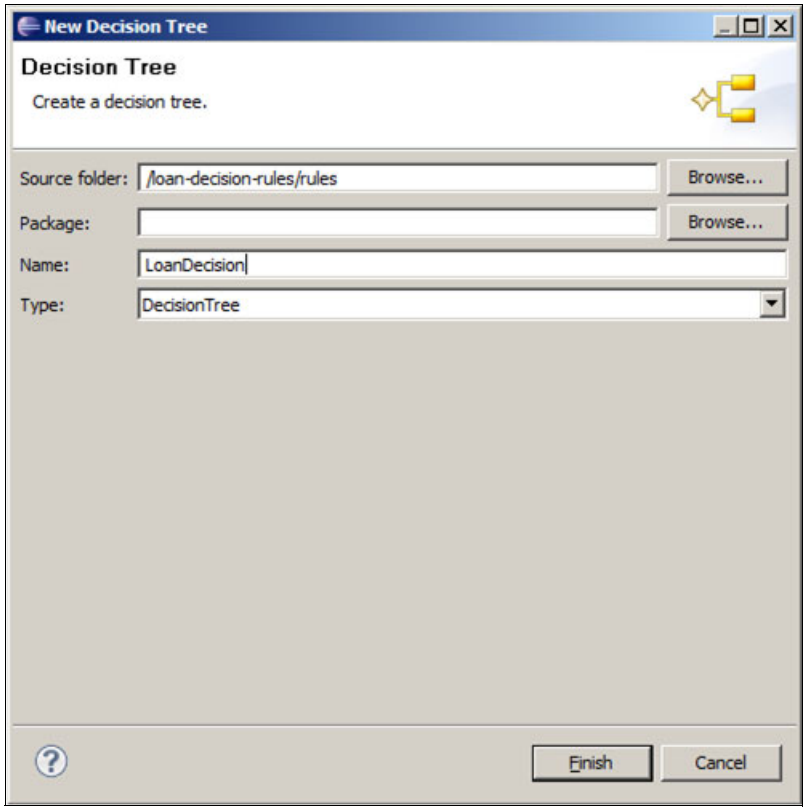


Figure E-23 New decision tree details

3. Click **Finish** to display the Rules decision tree, as shown in Figure E-24.

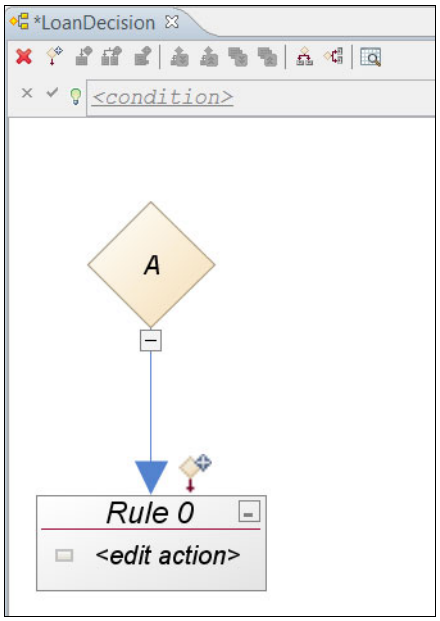


Figure E-24 Rules decision tree

4. Construct the following tree:
   a. Click Node **A** to select it.
   b. Use the **Add Branch** icon not button at the top to add two new branches from node A. You must reselect node **A** in between creating the first and second branches.
   c. Now**,** select the branches to Rule 1 and Rule 2 in turn and insert an extra condition node, using the **Insert a Condition Node** icon button. Do *not* insert an extra condition node for Rule 0, as shown in Figure E-25.

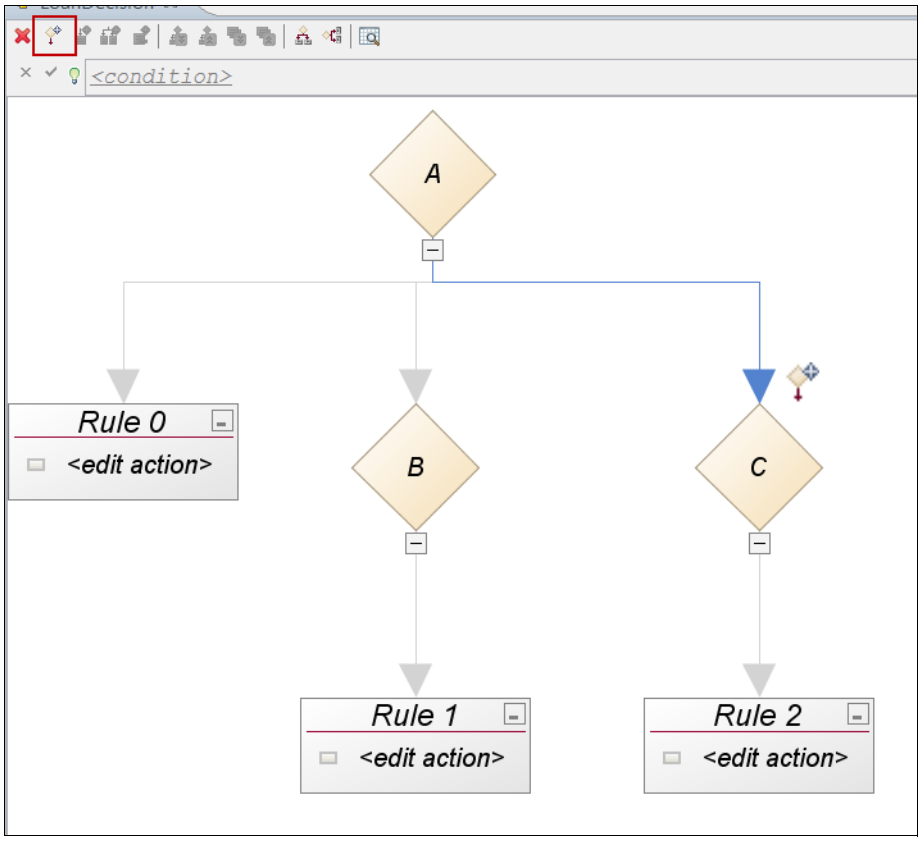


Figure E-25 Add condition nodes

   d. Create an extra branch from node B to Rule 1 and Rule 2, and extra branches from node C to Rule 3, Rule 4, and Rule 5, so that the diagram looks like the diagram in Figure E-26 on page 76.

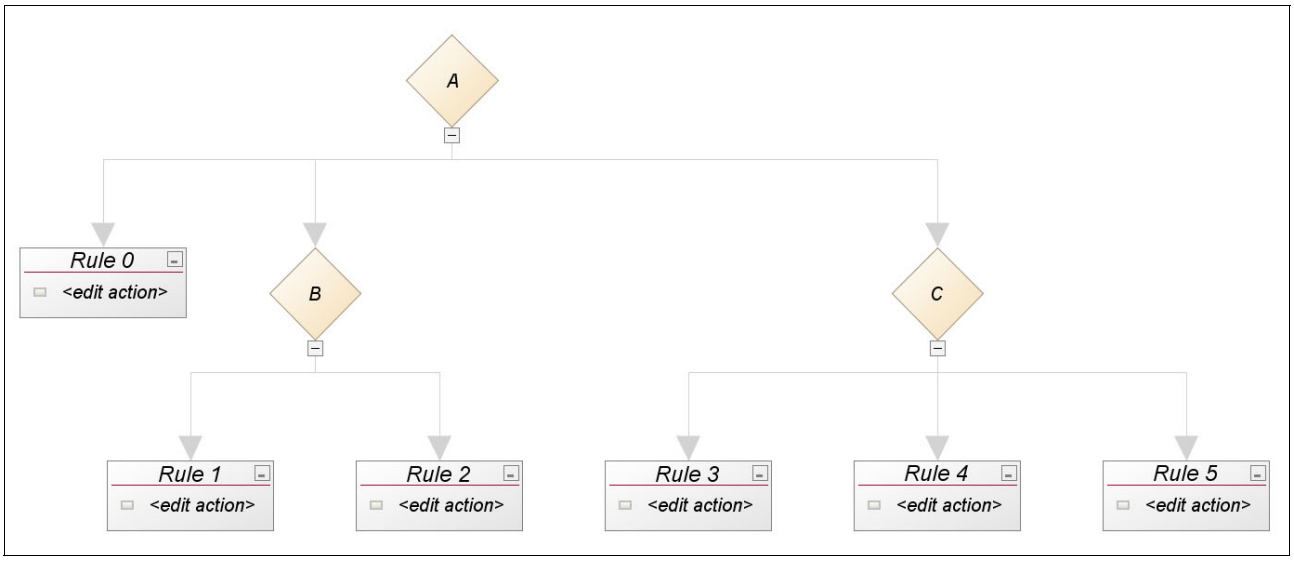


Figure E-26 Finished decision tree layout

5. Label the nodes:

   a. Node A represents the decision made on the amount of the loan. Double-click node **A** to open the Node Editor, and enter `Amount of Loan`. Click **OK**, as shown in Figure E-27.

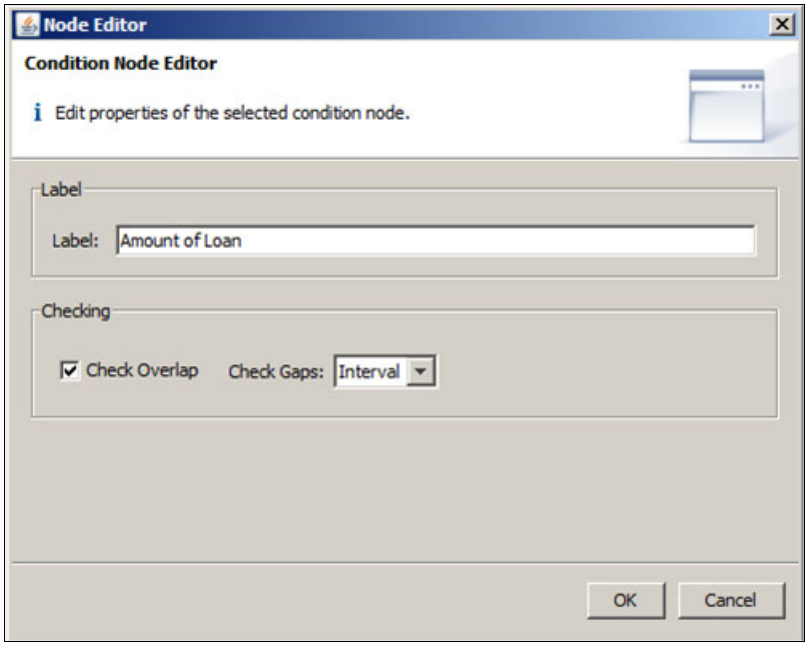


Figure E-27 Labeling the nodes

   b. In the same way, label node B `Age more than 65`, and label node C `Age between 21 and 65`. Both nodes are making a decision based on age, although the decision is different for each node.

6. Populate the decision-making in the Amount of Loan node:

   a. Click the **Amount of Loan is this correct?** condition node to open the condition editor.

b. Enter `the amount of 'the loan' is more than <a number> and at most <a number>` as shown in Figure E-28 on page 77. Click the green check mark on the left to save your changes.

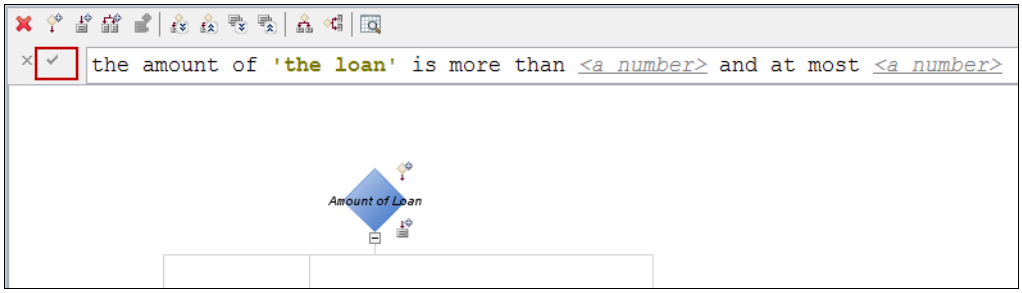


Figure E-28 Adding conditions

c. Click the branch to Rule 0. Modify the rule so that it reads `the amount of 'the loan' is at most the yearly income of 'the borrower'`. Click the green check mark to save. When you click the diagram, the branch will be labeled accordingly, as shown in Figure E-29.

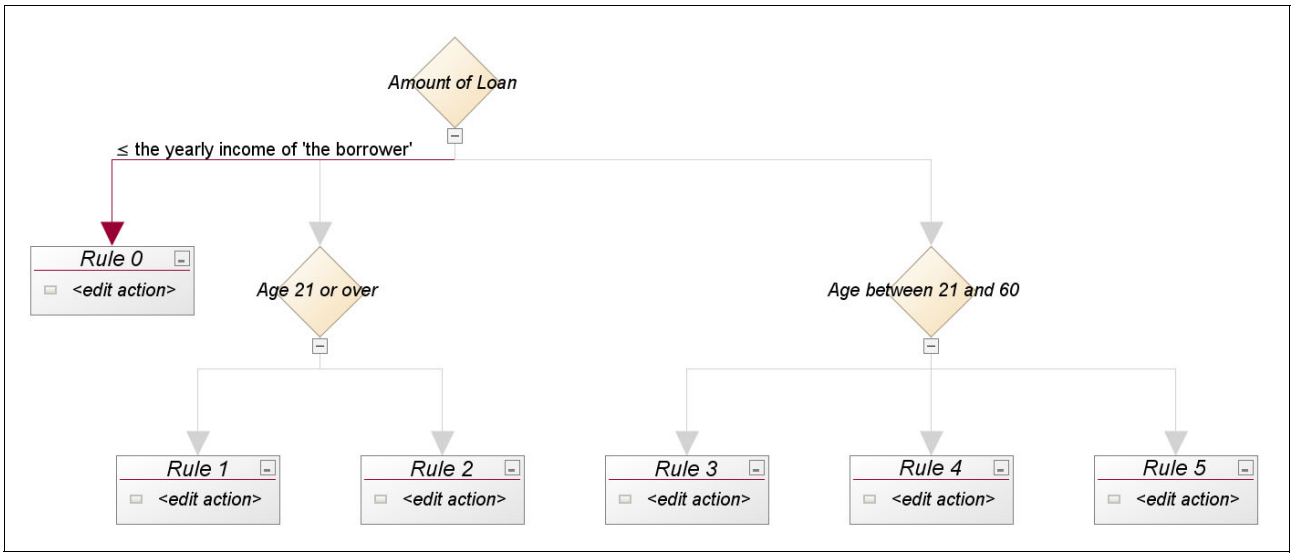


Figure E-29 Decision tree with some labels

d. Repeat for the other two branches. Remember to click the green check mark to save.

   i. The text for the middle branch is `the amount of 'the loan' is more than the yearly income of 'the borrower' and at most 3*the yearly income of 'the borrower'`. Click the green check mark to save.

   ii. The text for the right branch is `the amount of 'the loan' is more than 3 * the yearly income of 'the borrower'`. Click the green check mark to save.

e. The tree displays these changes, as shown in Figure E-30 on page 78.

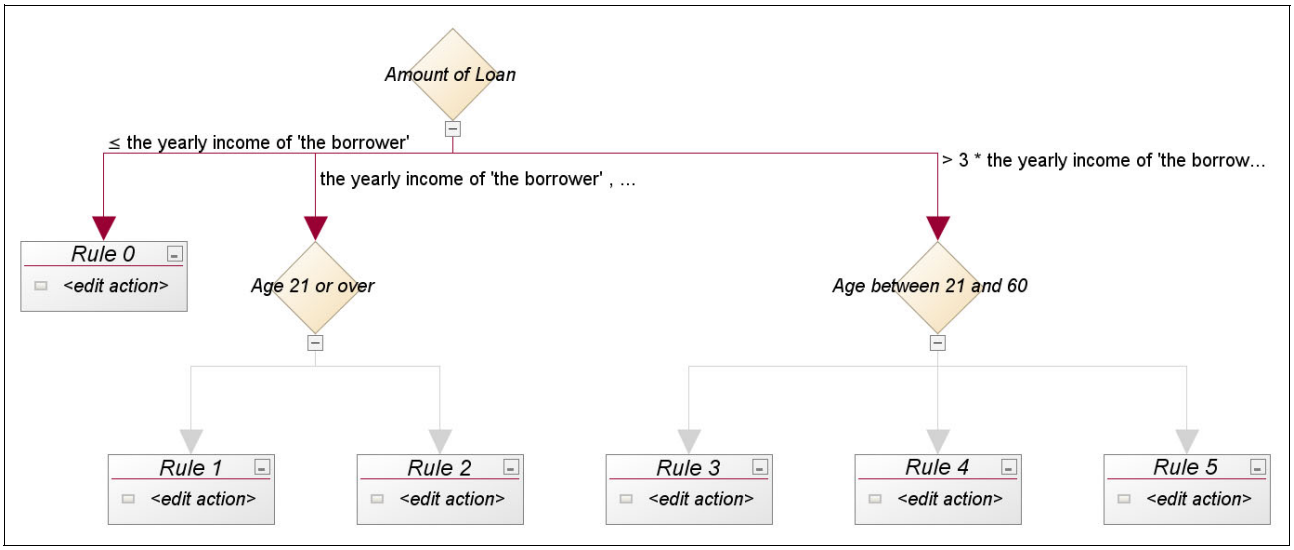


Figure E-30 Decision tree with all labels

7. Populate the Age 21 or over node:

   a. If you consider this example, loans between 1 and 3 times the borrower's salary are only available to borrowers over the age of 21. Therefore, we have two possible outcomes, depending on whether the borrower is over 21, which is why there are two branches.

   b. Click the **Age 21 or over** node and populate the rule with `the age of 'the borrower' is at least 21 is` *`<a boolean>`*. Click the green check mark to save.

   c. Populate the branch to Rule 1 with `the age of 'the borrower' is at least 21 is true`. Click the green check mark to save.

   d. Right-click the branch to Rule 2 and select **Set/Unset as Otherwise**, as shown in Figure E-31.

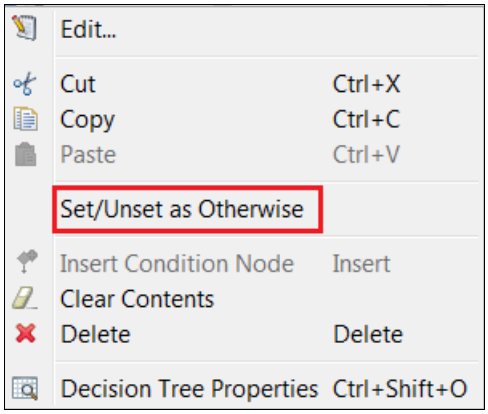


Figure E-31 Set/Unset as Otherwise

   e. The diagram displays these changes, as shown in Figure E-32 on page 79.

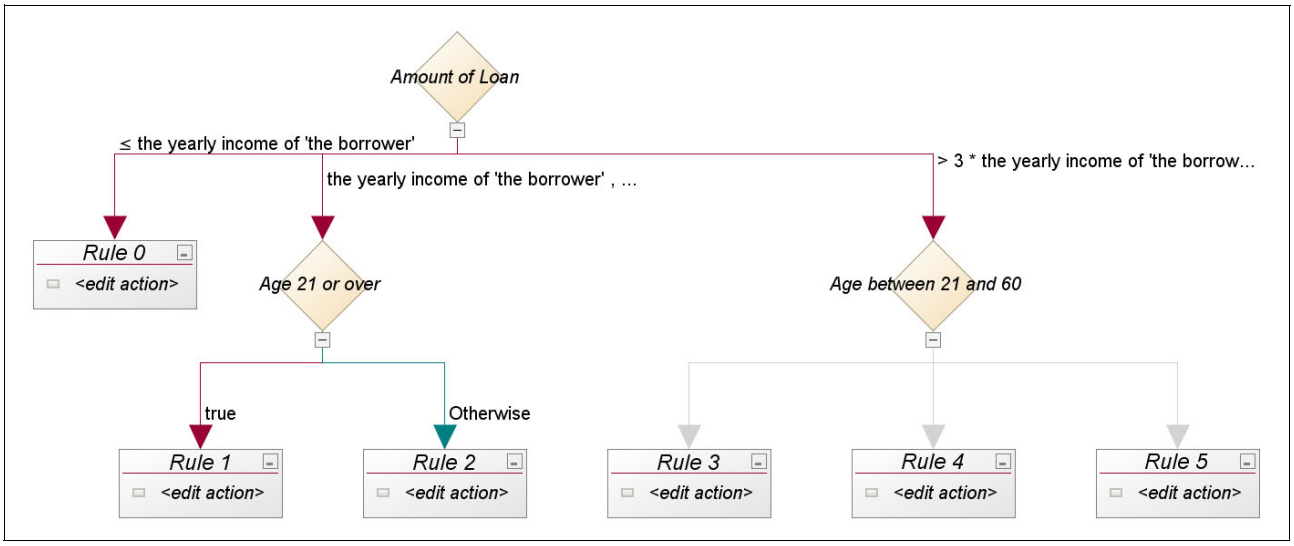


Figure E-32 Decision tree with more details

8. Now, select and populate the **Age between 21 and 60 node**:

   a. Enter the rule as `the age of 'the borrower' is more than <a number> and at most <a number>`. Click the green check mark to save.

   b. On the first branch, modify the rule so that it reads `the age of 'the borrower' is more than 21 and at most 60`. Click the green check mark to save.

   c. On the middle branch, modify the rule so that it reads `the age of 'the borrower' is at most 21`. Click the green check mark to save.

   d. On the right branch, modify the rule so that it reads `the age of 'the borrower' is more than 60`. Click the green check mark to save.

   e. The diagram now displays these changes, as shown in Figure E-33.

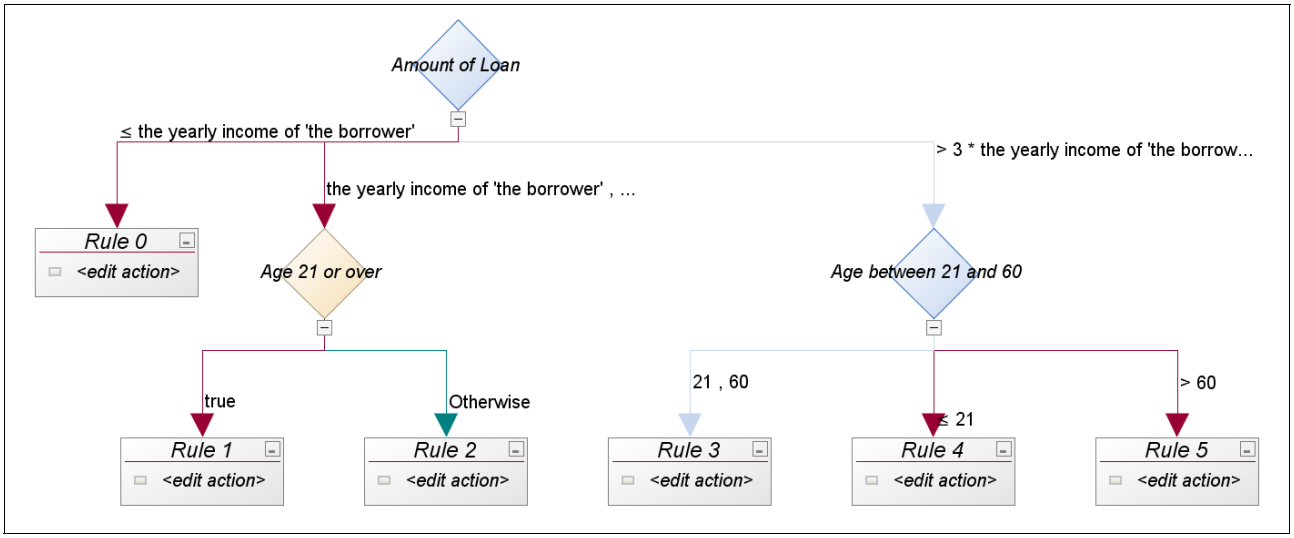


Figure E-33 Decision tree with more details

9. Now, populate the actions. Rules 0, 1, and 3 need to indicate that the loan has been accepted.

   a. Right-click Rule 0, and select **Edit**. Enter `Approved` as the label for this rule. Click **OK** to save, as shown in Figure E-34.

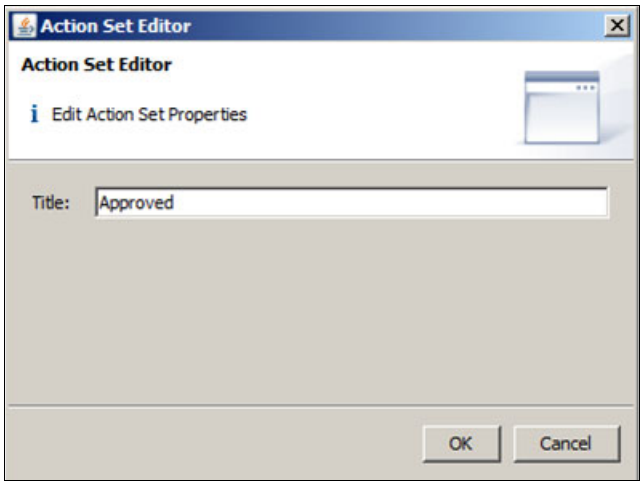


Figure E-34 Rule 0 is titled Approved

   b. Click **<edit action>**, as shown in Figure E-35.

Figure E-35 Edit Action

   c. Replace the action with `set the approved of 'the loan' to "T"`. Click the green check mark to save the action, as shown in Figure E-36.

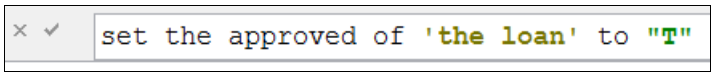


Figure E-36 Edit Action

   d. Repeat the process to populate Rules 1 and 3, as shown in Figure E-37 on page 81.

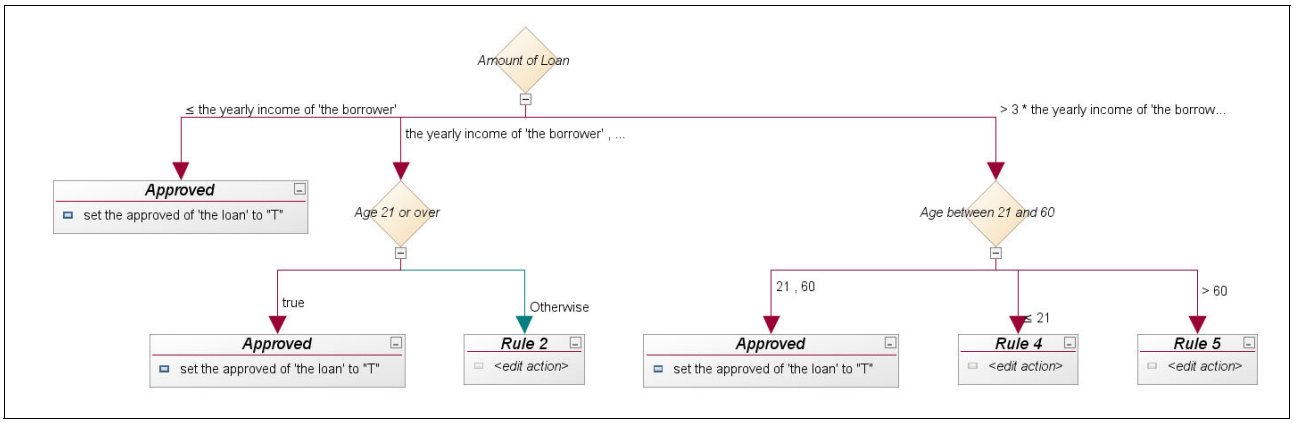


Figure E-37 Decision tree with final details

10. Rules 2 and 4 need to indicate that the borrower age is too young for this loan:
    a. Label these two Rules `Rejected, age too low`.
    b. Populate the action with `reject 'the loan', reason: ("Borrower below the age limit for this amount")`. Click the green check mark to save the action.
11. Rule 5 needs to indicate that the borrower's age is too old for this loan:
    a. Label this Rule `Rejected, age too high`.
    b. Populate the action with `reject 'the loan', reason: (“Borrower above the age limit for this amount”)`. Click the green check mark to save the action.
12. The decision tree is now complete. The decisions are identical to the ones that you defined in the table, but are being viewed in a different way. It might be impossible to see the entire decision tree at one time, necessitating scrolling. If this is the case, you can change the view to horizontal layout by clicking **Switch to horizontal layout** for a different view, as shown in Figure E-38.

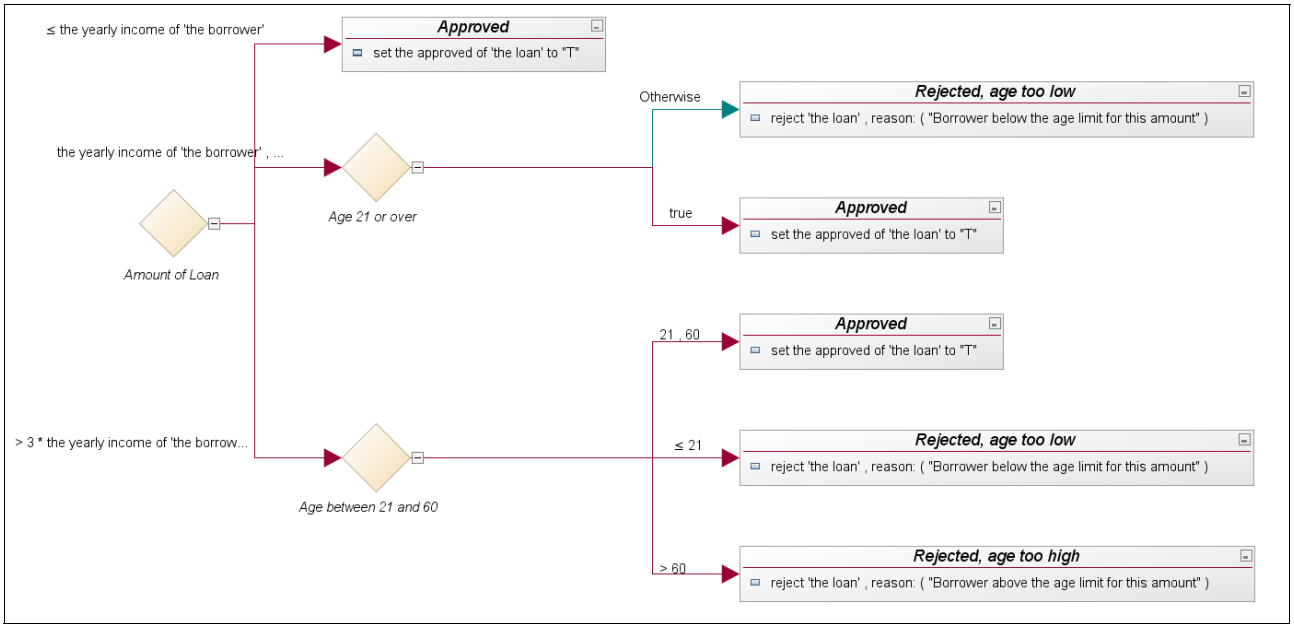


Figure E-38 Decision tree with final details, horizontal layout

Using IBM Operational Decision Manager

IMS COBOL BMP, COBOL DLIBATCH, and COBOL MPP

Redpaper™

Provides a step-by-step guide for calling ODM from IMS MPP, BMP, and DL/I applications

Discusses IMS applications' usage of ODM as a Rules System

Considers the configuration changes required in IMS

IBM Operational Decision Manager (ODM) is an implementation of a Business Rule Management System (BRMS). It enables you to create, manage, test, and govern business rules and events. You can store these in a central repository where multiple individuals and software products can access them.

IBM ODM Version 8.0 provides support for IBM IMS COBOL programs. This IBM Redpaper publication walks you through a step-by-step approach for using IBM ODM for rules management from an IMS COBOL MPP, BMP, or DL/IBATCH program.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

REDP-4997-00