**Cloud Pak for Automation Bootcamp**

**ODM – Hands-on Lab #1: Loyalty program**

Table of Contents

[Introduction 2](#_Toc38992220)

[Useful references 2](#_Toc38992221)

[Airline loyalty scenario 2](#_Toc38992222)

[Scenario realization 3](#_Toc38992223)

[Wait, there’s more! 12](#_Toc38992224)

# Introduction

The goal of this hands-on lab is to familiarize you with the different artifacts that compose an ODM decision project.

You will learn how to create, define and assemble the following essential rule project artifacts in Rule Designer:

* Create a Decision Service project.
* Build a Business Object Model (BOM) from an Execution Object Model (XOM).
* Create the different decision artifacts that are part of a Decision Service project.
* Define the decision operation.

# Useful references

* Documentation for the [decision artifacts](https://www.ibm.com/support/knowledgecenter/SSQP76_8.10.x/com.ibm.odm.dcenter.bu.bconsole/projects/con_bc_project.html) (rule flows, rules, decision tables)
* Documentation for the [BAL rule language constructs](https://www.ibm.com/support/knowledgecenter/SSQP76_8.10.x/com.ibm.odm.dserver.rules.ref.designer/lang_bal_ref_topics/tpc_bal_intro.html)

# Airline loyalty scenario

In this simple scenario, you will implement an operational decision service used by an airline loyalty program to assess the status of a customer, based on the customer’s travel history and other characteristics.

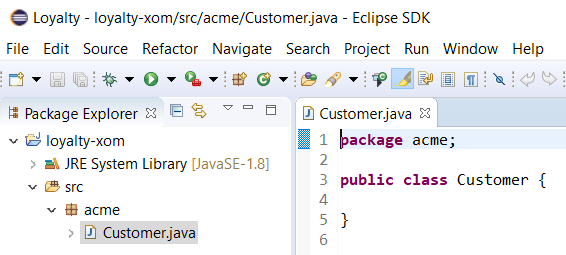
The simple business policy you will implement is the following:

* A customer with who has traveled at least 25K miles and has spent at least $5K worth of airfare has Silver status.
* A customer with who has traveled at least 50K miles and has spent at least $10K worth of airfare has Gold status.
* A customer with who has traveled at least 100K miles and has spent at least $20K worth of airfare has Platinum status.
* Otherwise, the customer has a general Member status.
* Finally, if the customer holds an airline-branded credit card, his status is upgraded by one level, based on the above program rules.

# Scenario realization

Please remember to save your work often as you progress along the following implementation steps. If you repeatedly encounter the same problem with Eclipse, you may want to close then reopen the Eclipse project you are working on, or sometimes use the File > Restart menu item.

1. **Start Rule Designer**
   1. In your chosen bootcamp working directory, create a folder called Loyalty.
   2. Open Rule Designer and open a workspace in the Loyalty folder.
2. **Create the XOM Java project**
   1. Select Windows > Perspective > Open Perspective > Other… and select the Java perspective.
   2. Select File > New > Java Project, and in the dialog, enter loyalty-xom for the project name and click Finish.
3. **Create the Customer Java class**
   1. Select File > New > Class, and in the dialog, enter acme for the package name and Customer for the class name, then click Finish.
   2. You should now see the following:



* 1. Add the following data members to the Customer class:

**public** **int** milesFlown;

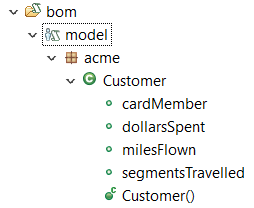
**public** **int** segmentsTravelled;

**public** **int** dollarsSpent;

**public** **boolean** cardMember;

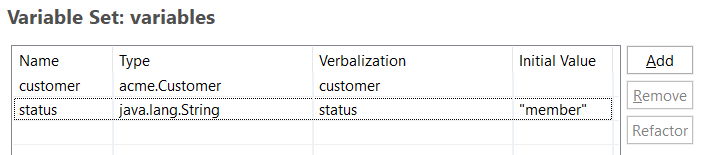
* 1. Save your Java class.

1. **Create the decision Service project**
   1. Select File > New > Project…, and in the dialog, select Rule Project and click Next.
   2. Select Main Rule Project under Decision Service Rule Projects (this should be the highlighted default) and click Next.
   3. Enter loyalty-rules as the project name and click Finish.
   4. Select the Rule perspective when prompted.
2. **Create the Business Object Model**
   1. Right-click on the loyalty-rules project and select Properties in the pop-up menu.
   2. In the dialog, click on Java Execution Object Model, select the loyalty-xom checkbox, then click Apply and Close. This lets the loyalty-rules project know that loyalty-xom is a possible source to define a BOM.
   3. In the loyalty-rules project, right-click on bom and select the New > BOM Entry menu item.
   4. In the dialog, click Next. On the next screen, click Browse XOM, and select loyalty-xom as the XOM element.
   5. The next two screens are asking you to select the classes and members from the XOM that you want to import in the BOM. Select all the checkboxes and click Finish. You should now see the following:

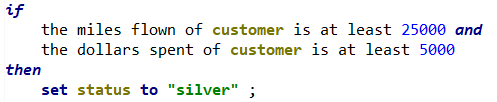


1. **Create the ruleset variables**
   1. Right-click on the rules folder under the loyalty-rules project and select New > Variable Set.
   2. Enter variables as the name of the variable set and click Finish.
   3. In the Variable Set pane, click the Add button, then:
      1. Enter customer for the Name and Verbalization
      2. Select Customer for the Type
   4. Click the Add button again, and this time:
      1. Enter status for the Name and Verbalization
      2. Keep java.lang.String for the Type
      3. Enter “member” for the Initial Value

You should now have the following for variables:

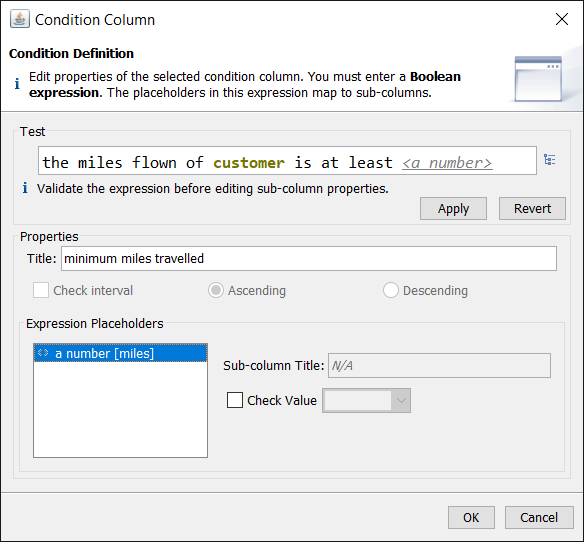


1. **Create an action rule**
   1. Right-click on the rules folder under the loyalty-rules project, select New > Rule Package, enter base status for the name and click Finish.
   2. Right-click this new base status package, select New > Action Rule, enter silver for the name, and click Finish.
   3. Enter the following text in the Action Rule pane:

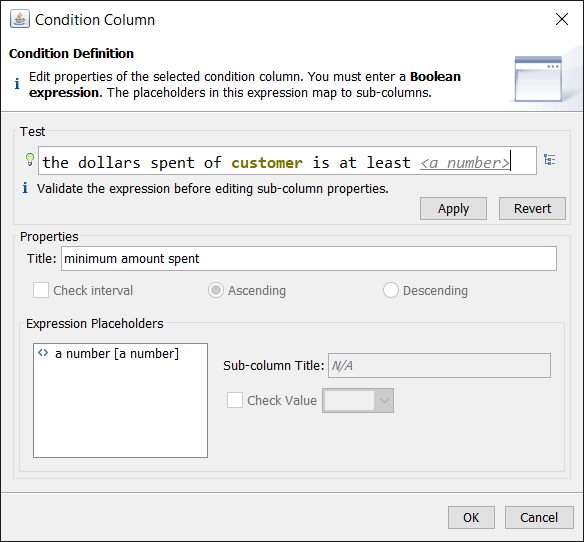


With this, you have just created your first executable ODM rule. For our scenario however, we have 3 rules that are very similar in structure. This is a good indication that we should use a decision table instead of individual action rules. So, let’s delete the rule you just created and start with a decision table instead.

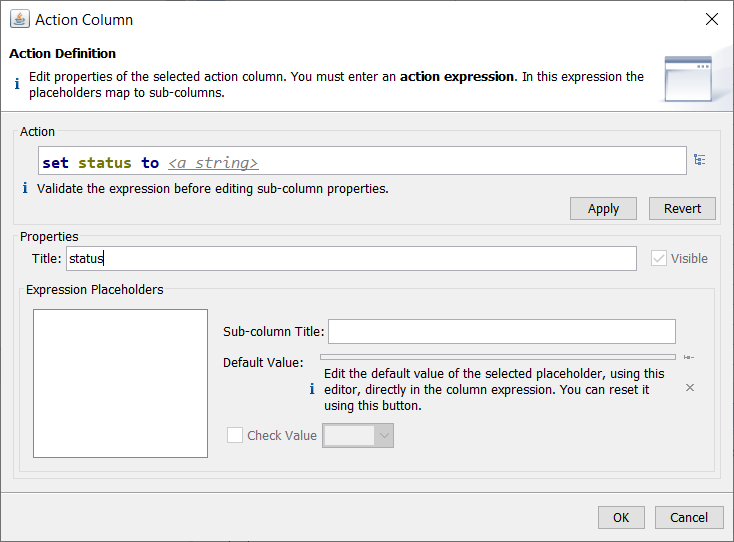
1. **Create the structure for a decision table**
   1. Right-click on the rules folder under the loyalty-rules project, select New > Decision Table, enter set base status for the name and click Finish. A default skeleton for a decision table is created, with three condition columns (A, B and C) and one action column (D). We are now going to configure these columns.
   2. Double-click on the header of column A, enter the Test definition and the column Title as shown below, then click OK:



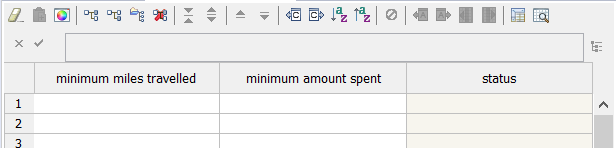
* 1. Double-click on the header of column B, enter the Test definition and the column Title as shown below, then click OK:



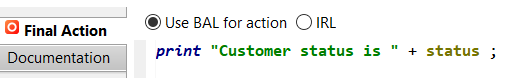
* 1. Right-click on the header of column C and select Remove Condition Column since we only have 2 condition in our decision table.
  2. The last column, column D, is an action column for the decision table. Double-click on the header of column D, enter the Test definition and the column Title as shown below, then click OK:



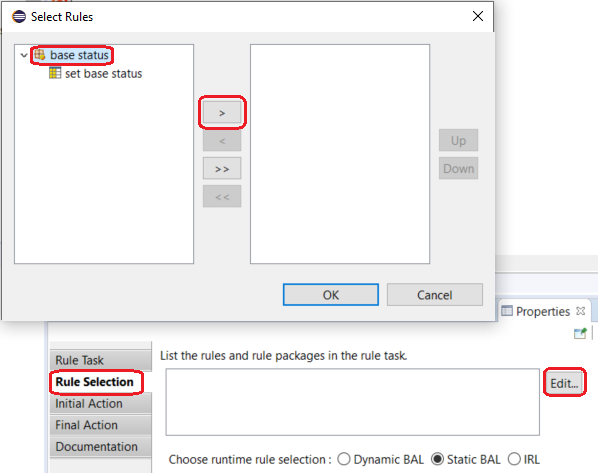
Your final decision table structure is now ready and should look like this:



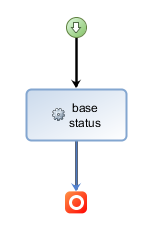
1. **Populate the decision table**
   1. Enter the values provided by the scenario for each status in the decision table. You will see a warning about overlapping cells; you can ignore it for now.
2. **Create the rule flow**
   1. Right-click on the rules folder, select New > Ruleflow, enter customer status for the name and click Finish. The ruleflow canvas pane opens, with a palette of different ruleflow elements on the left.
   2. Click on the Create a start node element, then click on the ruleflow canvas to create the start node.
   3. Click on Create an end node, then click on the ruleflow canvas to create the end node. Double-click on the end node you just created and in the property pane under the ruleflow canvas, enter the print action shown below. This will enable us to see the resulting status when testing the ruleset execution from Eclipse.



* 1. Click on Create a rule task, then click on the ruleflow canvas to create a rule task.
  2. Double-click on the task, and in the properties panel below the ruleflow canvas, enter base status in the ID field.
  3. Click the Rule Selection tab, click Edit, and in the Select Rules panel, select the base status rule package and click the right arrow to select it, as shown below:

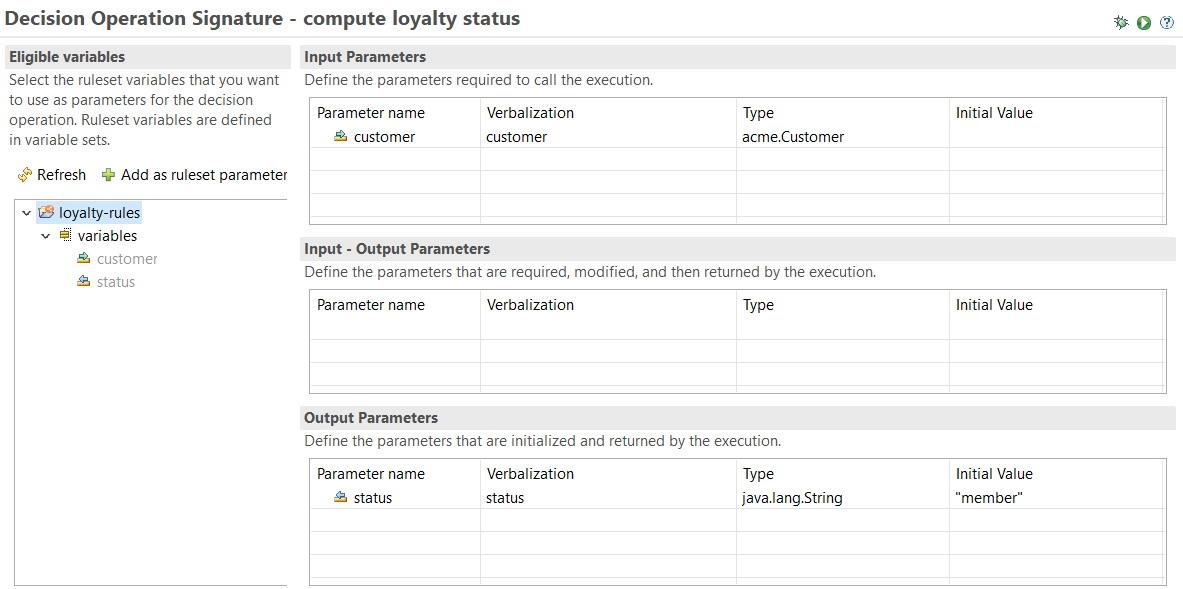


* 1. Click on Create a transition in the ruleflow palette, then link the start node to the task and the task to the end node. To link two nodes, start by clicking on the origin node, then click on the desired destination node. Your final ruleflow should look like this:



1. **Create the decision operation**
   1. Right-click on the deployment folder, select New > Decision Operation, enter compute loyalty status for the name and click Next.
   2. In the next screen, select loyalty-rules for the Source Rule Project, and click Next.
   3. In the next screen, expand loyalty-rules until you see the customer status ruleflow, select it and click Finish.
   4. In the Decision Operation pane that opens, select the Signature tab.
   5. Expand the Eligible variables, then:
      1. Select customer and click Add as ruleset parameter, then Use as input.
      2. Select status and click Add as ruleset parameter, then Use as output.

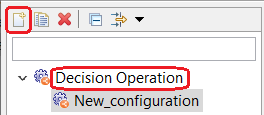
You should now have the following setup:



1. **Execute the decision operation**

You can execute the decision operation in Eclipse as you would for a Java program.

1. Select Run > Run Configurations.
2. In the dialog, select Decision Operation and create a new run configuration.



1. Select compute loyalty status as the decision operation.
2. Click on the Parameters & Arguments tab, select customer and click Edit value…
3. Select Function body and enter the following code:

acme.Customer result = **new** acme.Customer();

result.milesFlown = 53000;

result.dollarsSpent = 11200;

**return** result;

1. Click OK, then Apply and Run on the parent window.
2. You should see the following result of the execution in the Console pane of Eclipse:

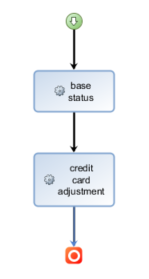


# Wait, there’s more!

We did not take care of the bit about the airline-branded credit card:

“if the customer holds an airline-branded credit card, his status is upgraded by one level”

1. **Add the missing rule**
   1. Create a new rule package called credit card adjustment.
   2. In this new rule package, create a decision table to implement the missing policy statement.
   3. Add the new rule package to the ruleflow, which should then look like this:

****

1. **Re-execute the decision operation**
   1. Execute the run configuration again, this time with the following input data:

acme.Customer result = **new** acme.Customer();

result.milesFlown = 53000;

result.dollarsSpent = 11200;

result.cardMember = true;

**return** result;

The output status should now be platinum.