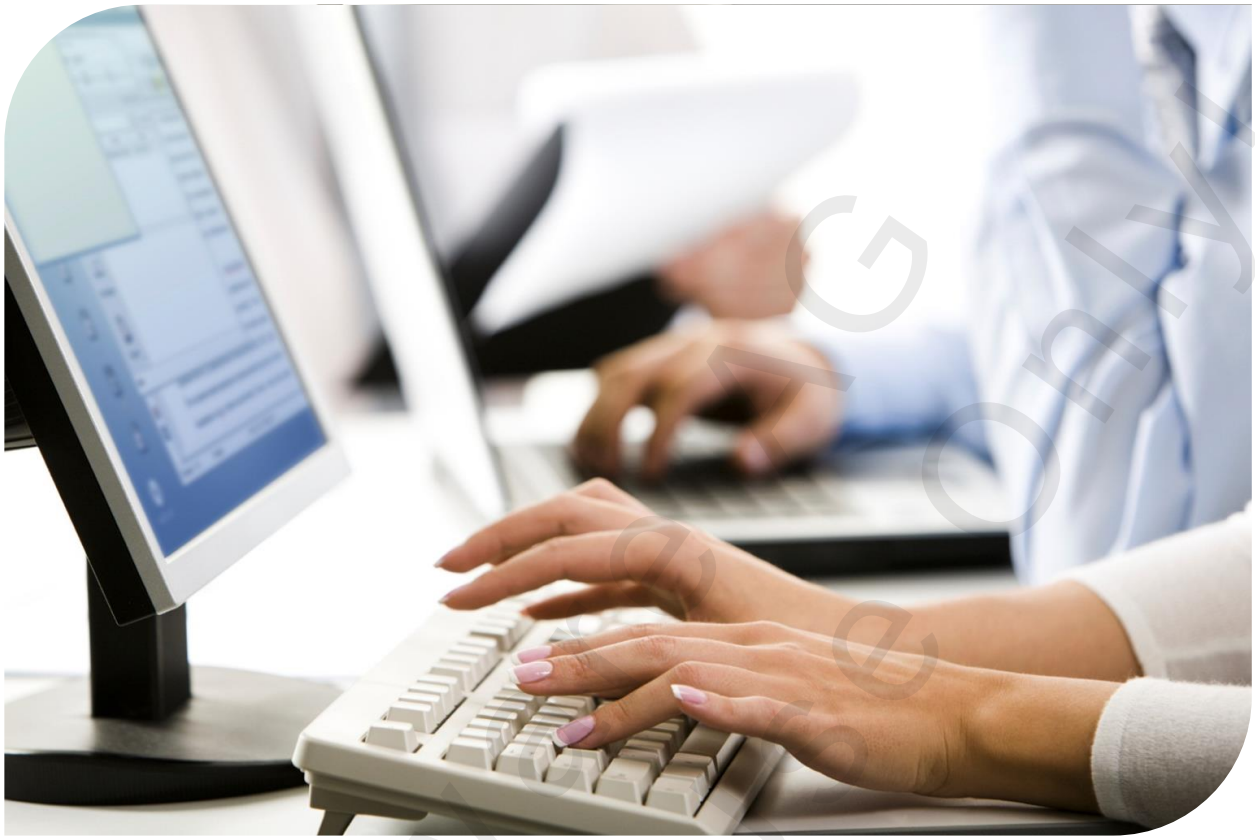


EXERCISE GUIDE

webMethods Business Rules

629-6CE

TABLE OF CONTENTS

This page intentionally left blank

Exercise 01:
Simple Decision Table

Objectives

In this exercise, you will create a Rule project, import Data Models from existing IS documents and configure a first simple Decision Table based on Customer data. The Decision Table determines a discount rate based on various customer parameter elements. You will test your Decision Table in Designer.

In all exercises, replace ***<workshop-dir>*** by **C:\Training\629-6CE\Student**.

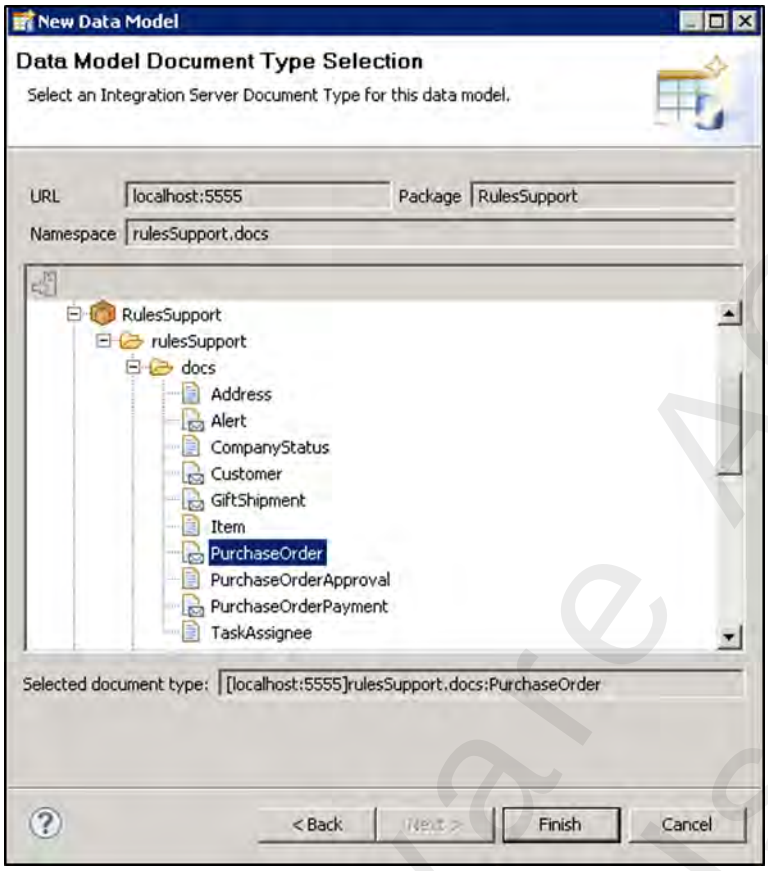
Steps

1. If not already running, start the following Windows service in this sequence.
    - **hMailServer**
    - **Software AG Runtime 9.12**
    - **Software AG Universal Messaging 9.12 (umserver)**
    - **Software AG Integration Server 9.12 (default)**

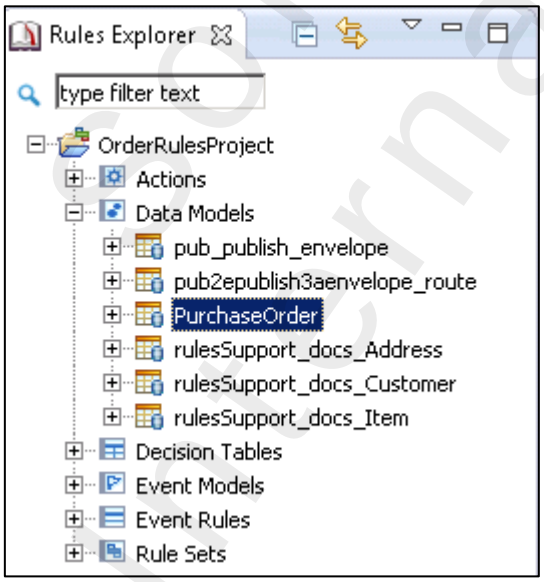
    Important: Leave these services up and running for all exercises until further notice.

2. Copy the provided IS package **<workshop-dir>\Exercise1\Resources\RulesSupport.zip** to the IS folder: **C:\SoftwareAG\IntegrationServer\instances\default\replicate\inbound**
3. Open a browser tab and login to the IS Administration Console (https://localhost:5555) as **Administrator | manage**. Navigate to **Packages** > **Management** > **Install Inbound Releases** to install and activate the package **RulesSupport.zip** into your default Integration Server.
4. Launch **Software AG Designer**. If asked for a secure storage password, provide **manage.** Close the Welcome page, if opened.
5. Switch to **Rules Development** perspective. Use the **Solutions** view to add a new Rules Project named **OrderRulesProject** to your default Designer workspace. Ensure version control is disabled for this project and click **Finish**.

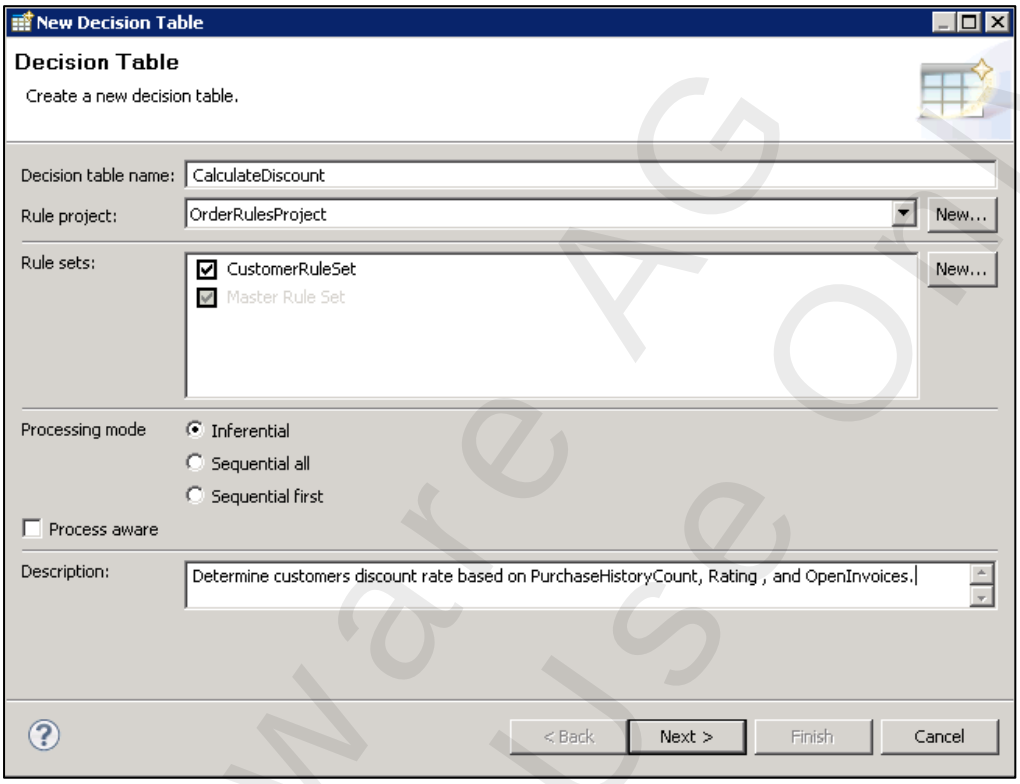
6. Use the **Rules Explorer** view to create a Data Model named **PurchaseOrder** in your rule project **OrderRulesProject**. Select **rulesSupport.docs:PurchaseOrder** as related IS Document Type. Click **Finish**.



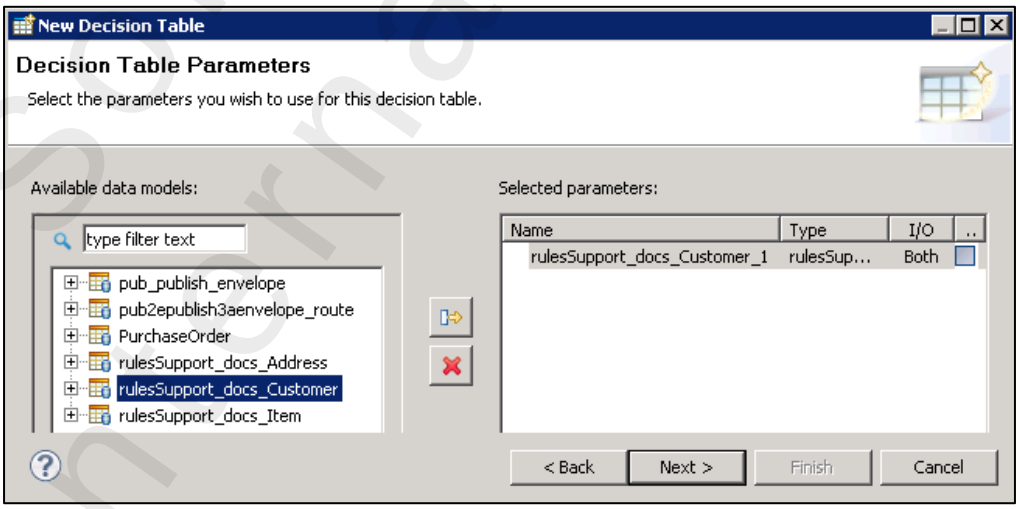
Note: **PurchaseOrder** includes other Document Types, as a result all referenced Document Types have been added as new Data Models with a generated name also.



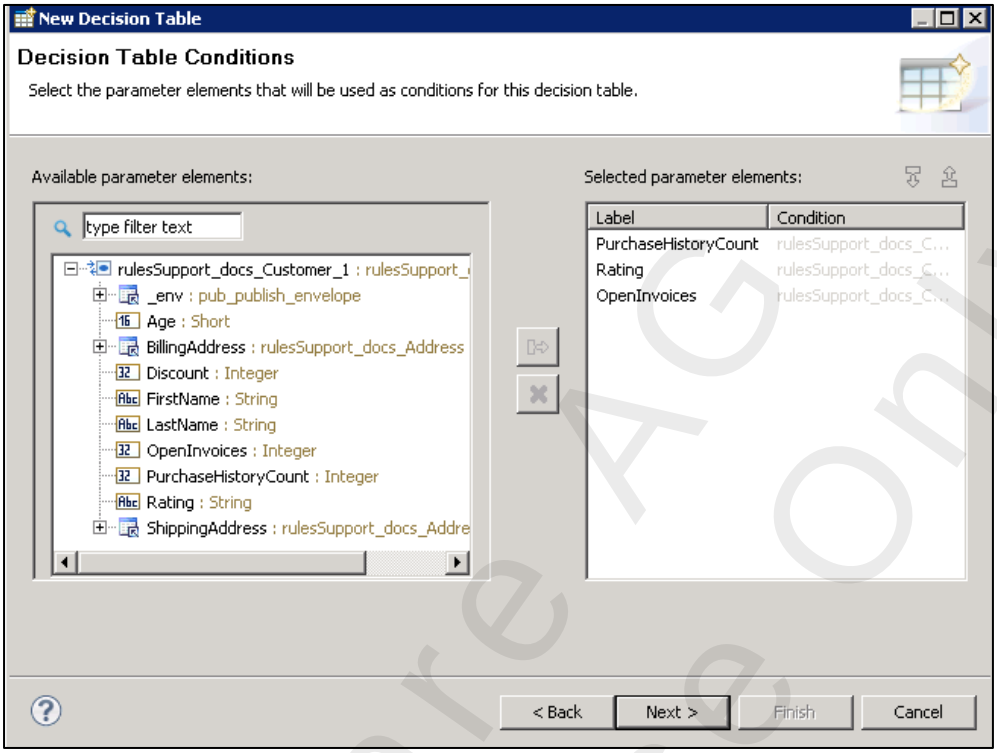
7. Use the Rules Explorer view to create a new Rule Set named **CustomerRuleSet** with **Inferential** Processing mode. This Rule Set will be used to combine assets of the project **OrderRulesProject**.
8. Use the Rules Explorer view to add a Decision Table:
    a. Add a Decision Table named **CalculateDiscount** with **Inferential** Processing mode to your **OrderRulesProject** project. Add this Decision Table to your **CustomerRuleSet** Rule Set and provide a meaningful description text as shown on the screen shot below. Click **Next**.



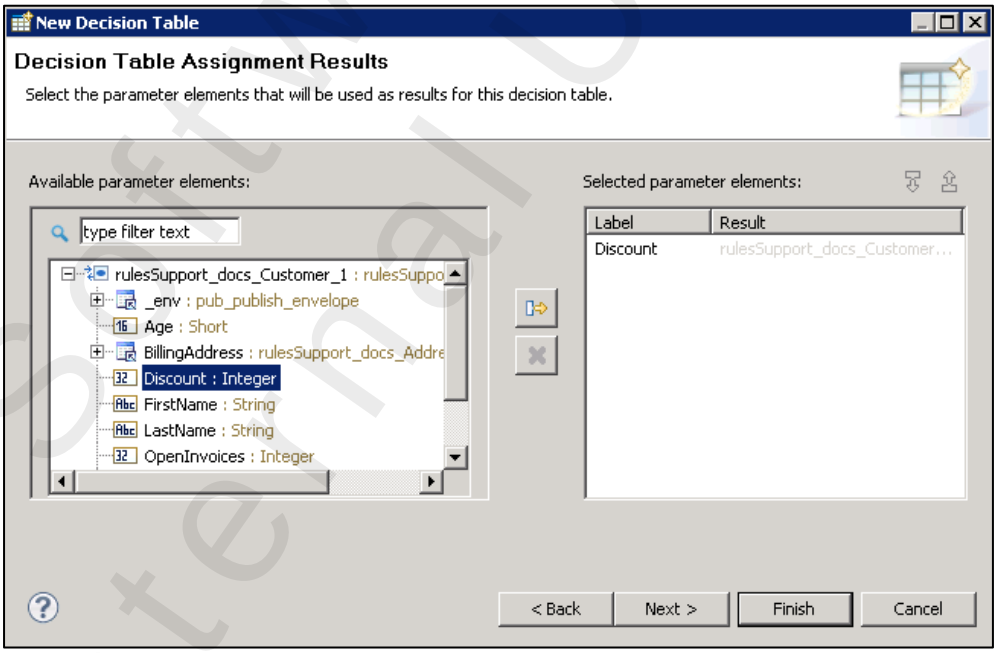
    b. On the subsequent panel, select Data Model **rulesSupport_docs_Customer** as Decision Table parameter with I/O type **Both**. Keep the default parameter name. Click **Next**.



c. On the next panel, select the parameter elements **PurchaseHistoryCount**, **Rating**, and **OpenInvoices**. These will be used later in your rule conditions. Click **Next**.



d. Finally specify the parameter elements used as results for your Decision Table. Select the element **Discount** only:

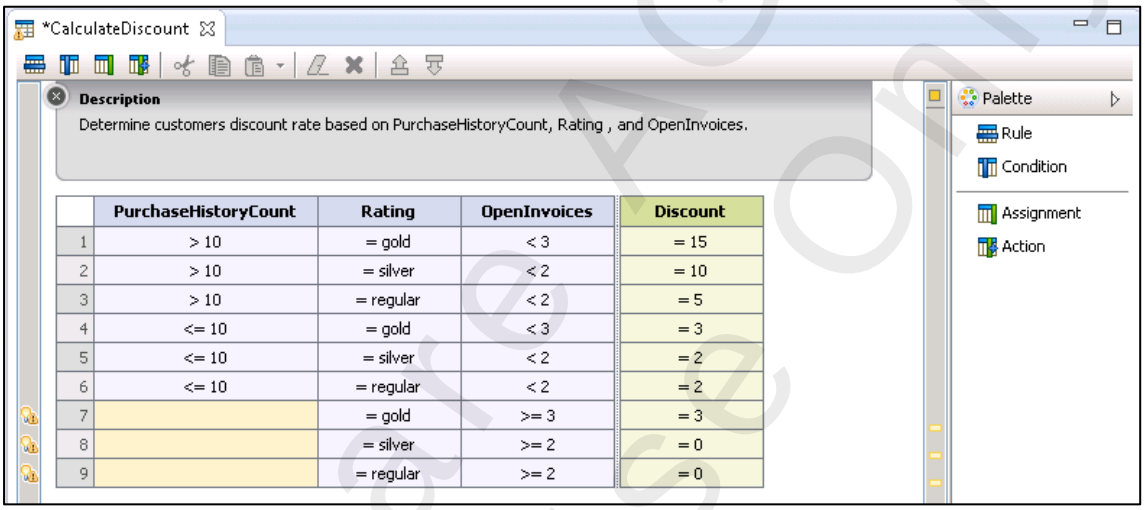


e. Click **Finish** to complete your Decision Table structure.

f. Use the Decision Table Editor for your Decision Table **CalculateDiscount** to insert nine rules.

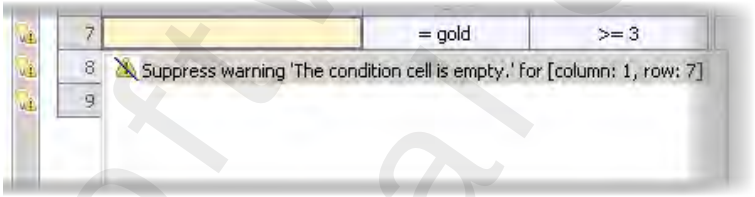
Hints: You can copy one or even multiple rules and paste them as new rules using the appropriate icons from the icon bar of the Decision Table Editor. To create an empty rule, drag from the Palette to the Decision Table or click from the icon bar above the Decision Table.
Comparison operators can by chosen via **Assign Operator** from the context menu or by clicking the pencil icon to open the extended cell editor.

Finally, your Decision Table should look like this:



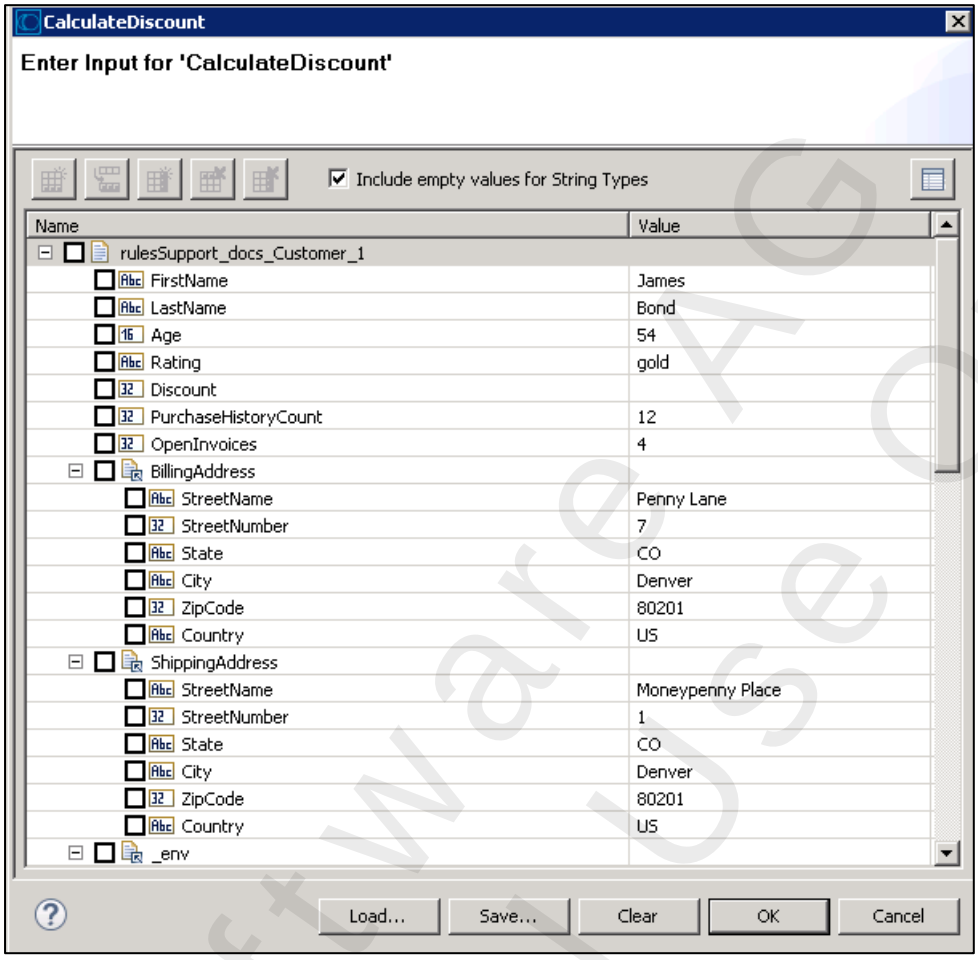
	PurchaseHistoryCount	Rating	OpenInvoices	Discount
1	> 10	= gold	< 3	= 15
2	> 10	= silver	< 2	= 10
3	> 10	= regular	< 2	= 5
4	<= 10	= gold	< 3	= 3
5	<= 10	= silver	< 2	= 2
6	<= 10	= regular	< 2	= 2
7		= gold	>= 3	= 3
8		= silver	>= 2	= 0
9		= regular	>= 2	= 0

g. Note that the last three rules are flagged with warnings because of an empty condition cell. For now, right-click each bulb icon to suppress the warnings for rules 7..9.



h. Save your Decision Table.

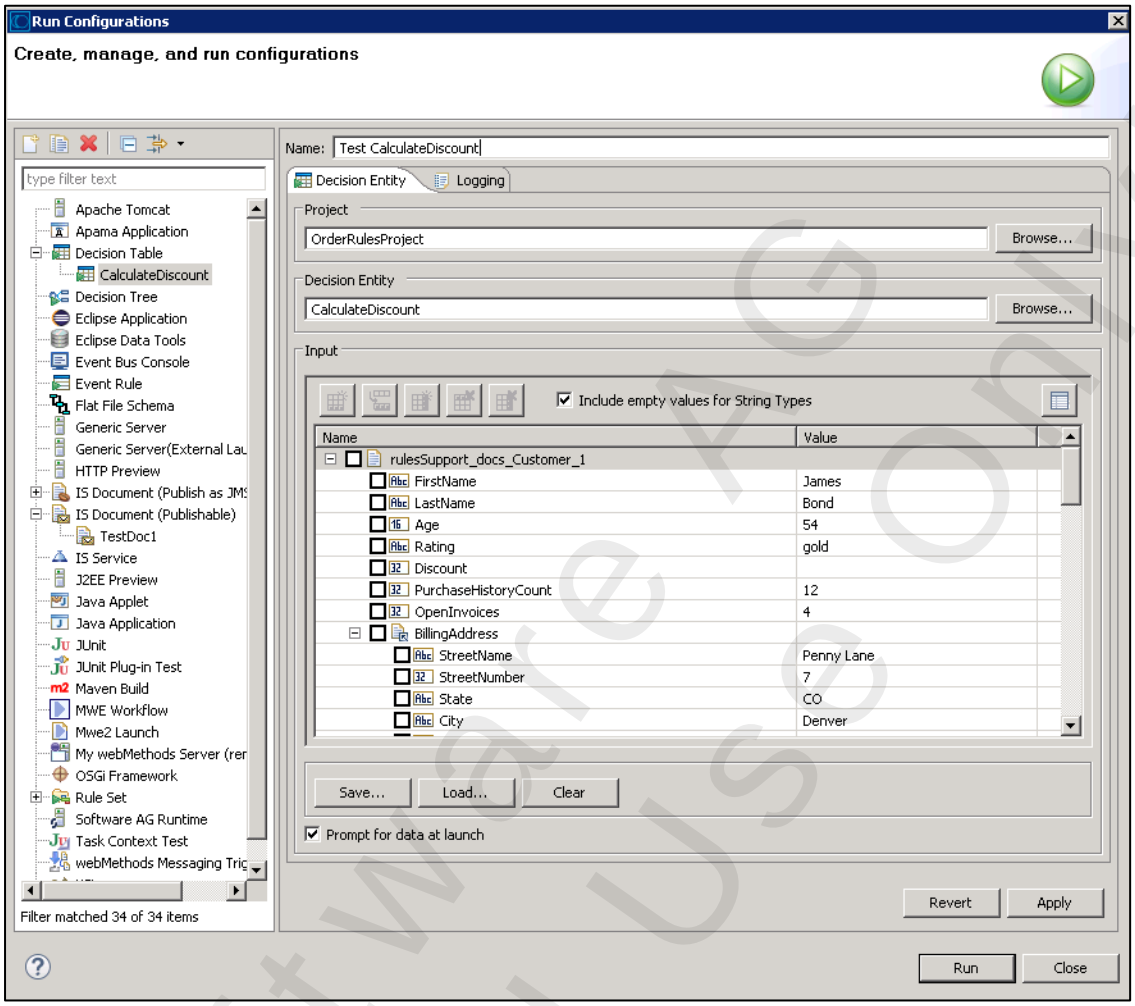
9. Right-click Decision Table **CalculateDiscount** in the Rules Explorer and select **Run As > Run Decision Table**. On the panel popping up, specify at least the necessary input values for **PurchaseHistoryCount**, **Rating**, **OpenInvoices** and check the box beside **Include empty values for String Types**. For testing you can also load the provided sample input **<*workshop-dir*>\Exercise1\Resources\CalculateDiscountDecisionTableInput.xml**.



Click **OK** to run the Decision Table. Inspect the results in the appearing Result view. Repeat testing with different input values to verify that your Decision Table works properly.

10. Right-click the Rule Set **CustomerRuleSet** in the Rules Explorer view and **select Run As > Run Rule Set**. You can reuse the sample input from above.
What is the difference to the behavior of step 9 above?

11. Select **Run > Run Configurations…** from Designer's menu bar. Select the Run Configuration for Decision Table **CalculateDiscount** from here and rename the Run Configuration to **Test Calculate Discount**.



Apply your changes.
Test your Decision Table again by running the preconfigured Run Configuration **Test Calculate Discount**.

This page intentionally left blank

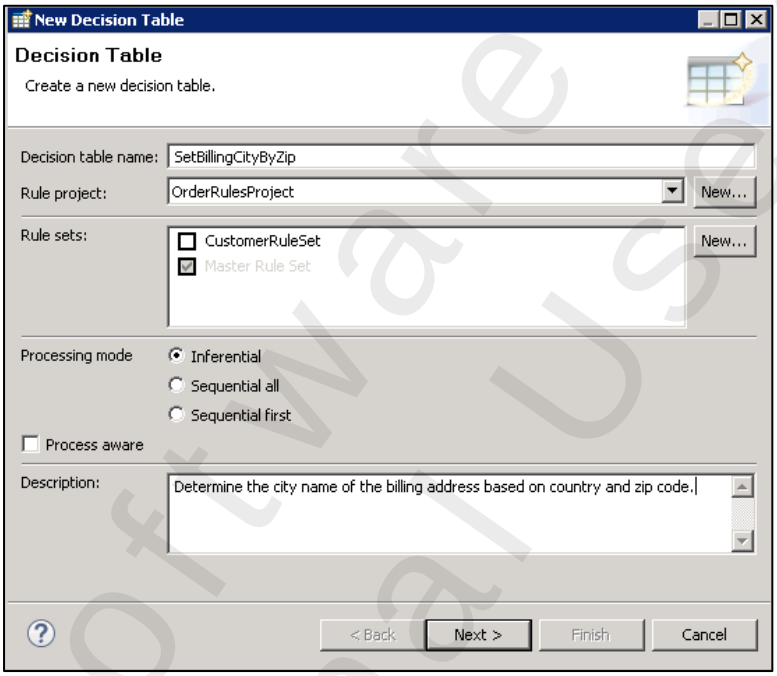
Exercise 02:
Rule Sets

Objectives

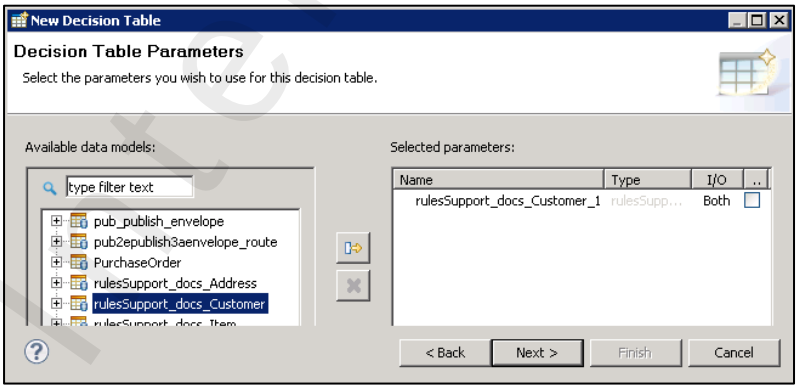
In this exercise, you will add a second Decision Table to your rule project. The Decision Table determines a city by the ZIP code. To combine the execution of the Decision Table with your previous one, both will be contained in one common Rule Set.

Steps

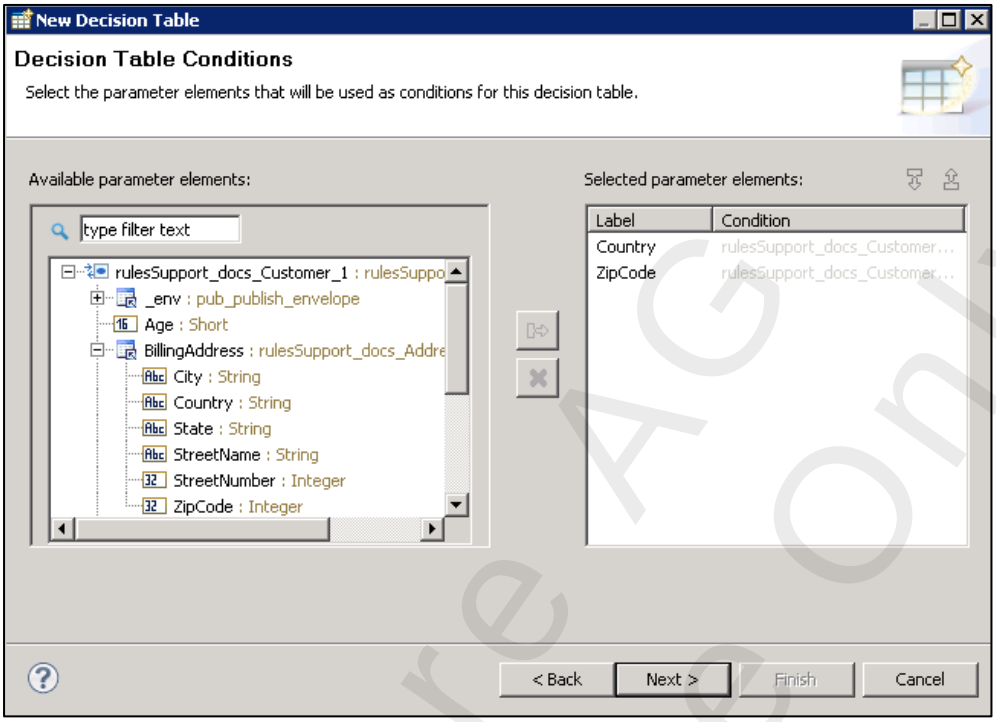
1. Launch **Software AG Designer** and switch to the **Rules Development** perspective.
2. Use the Rules Explorer view to add another Decision Table:
    a. Add a Decision Table named **SetBillingCityByZip** to your **OrderRulesProject** project. Do NOT add this Decision Table to your CustomerRuleSet Rule Set yet. Leave the Processing mode (**Inferential**) unchanged, provide a description text as shown below, and click **Next**.



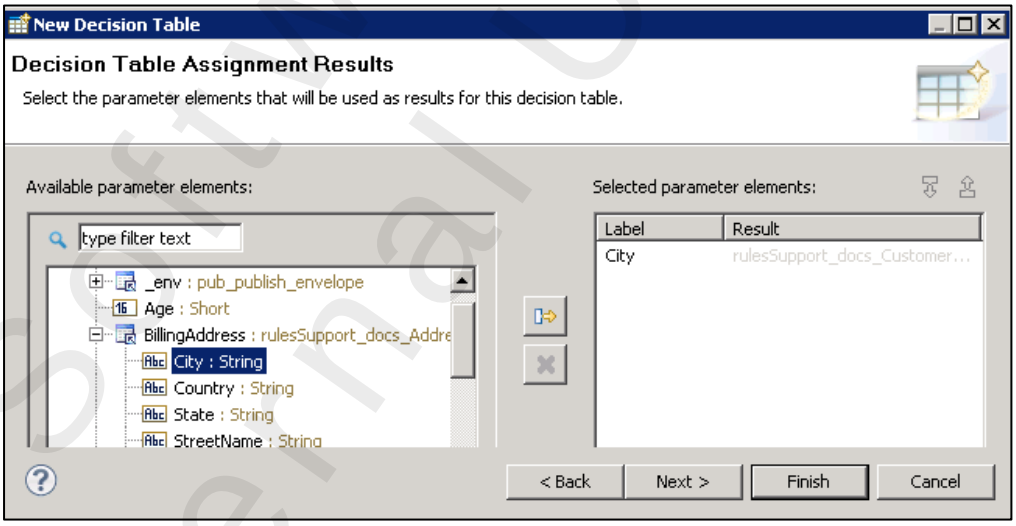
    b. On the subsequent panel select Data Model **rulesSupport_docs_Customer** as Decision Table parameter with I/O type **Both**. Leave the default parameter name unchanged:



c. On the next panel select the parameter elements **Country** and **ZipCode** (of **BillingAddress**) to be used later in your rule conditions. Click **Next**.



d. Finally specify the parameter elements used as assignment results for your Decision Table. Select the element **City** (of **BillingAddress**) only:

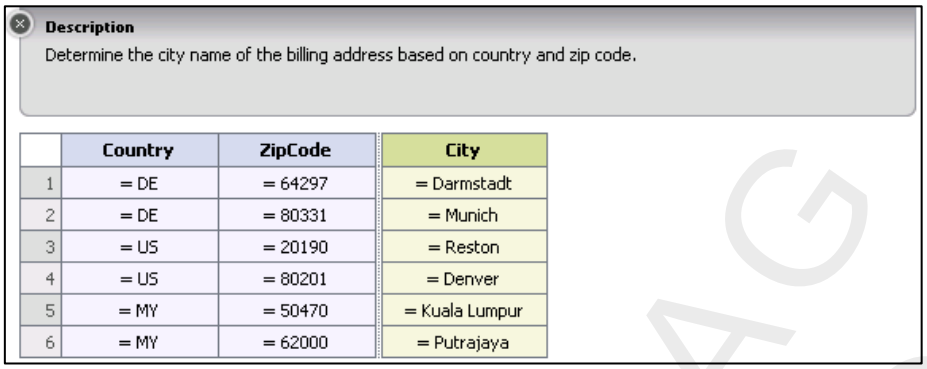


e. Click **Finish** to complete your Decision Table structure.

3. Use the Decision Table Editor for your Decision Table **SetBillingCityByZip** to insert six rules.

   Hint: Create the first rule, then use copy & paste to create the other rules.

   Finally, your Decision Table should look like this:



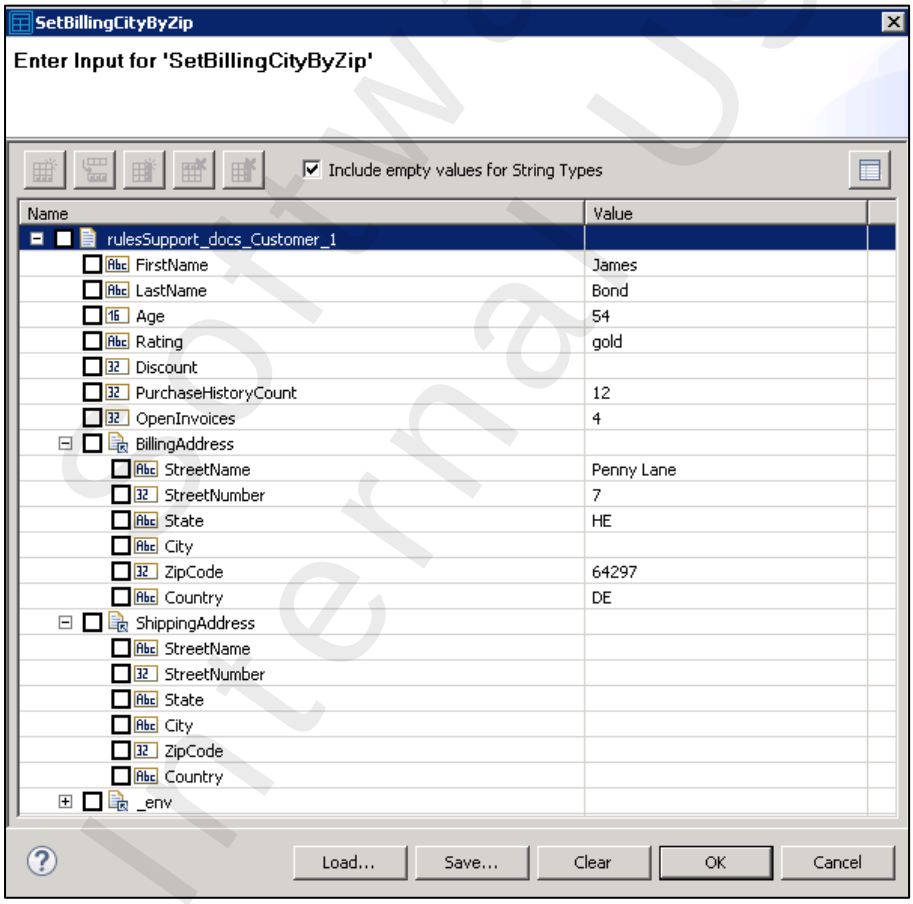
Description

Determine the city name of the billing address based on country and zip code.

	Country	ZipCode	City
1	= DE	= 64297	= Darmstadt
2	= DE	= 80331	= Munich
3	= US	= 20190	= Reston
4	= US	= 80201	= Denver
5	= MY	= 50470	= Kuala Lumpur
6	= MY	= 62000	= Putrajaya

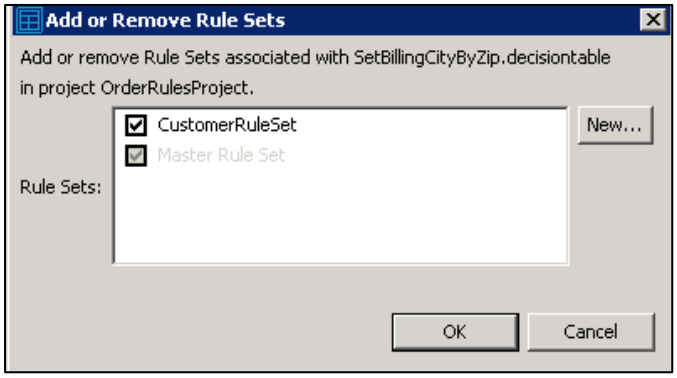
   Save your Decision Table.

4. Right-click Decision Table **SetBillingCityByZip** in the Rules Explorer and select **Run As** > **Run Decision Table** to test it standalone. On the appearing panel, specify at least the necessary input values for **Country** and **ZipCode** within **BillingAddress** and check the box beside **Include empty values for String Types**. For testing you can also load the provided sample input **<*workshop-dir*>\Exercise2\Resources\SetBillingCityByZipDecisionTableInput.xml**.

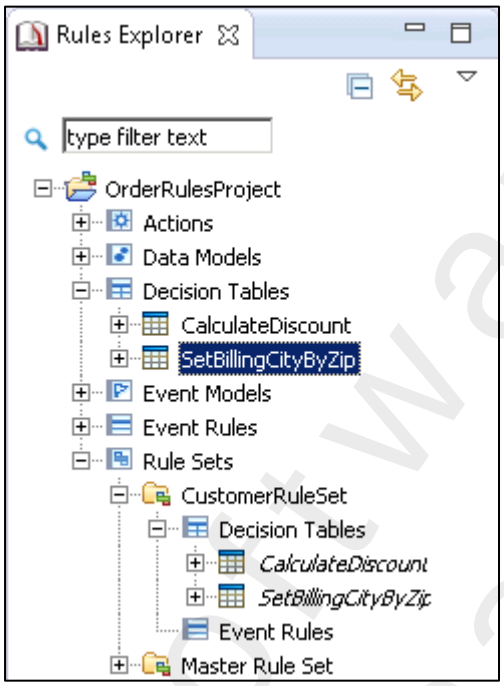


   Inspect the results in the appearing Result view. Repeat testing with different input values to verify that your Decision Table works properly.

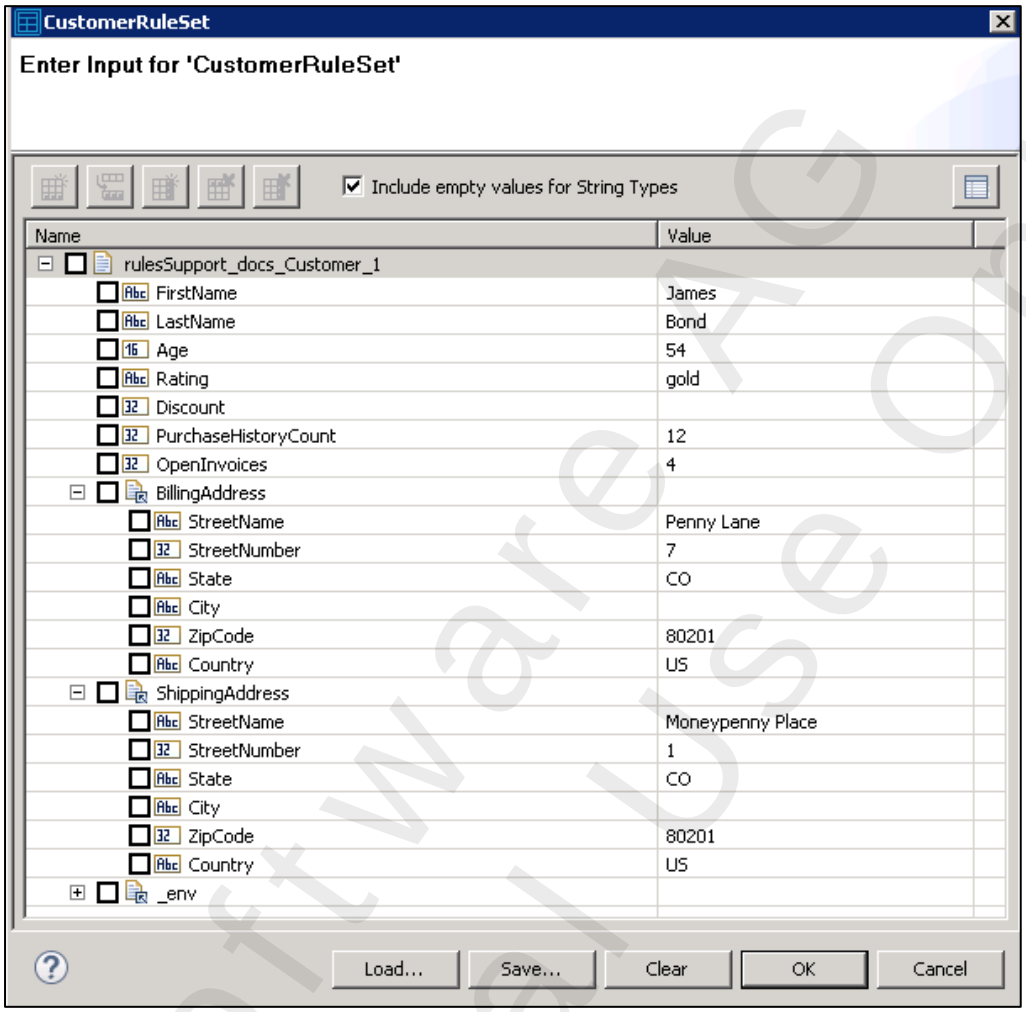
5. Right-click Decision Table **SetBillingCityByZip** in the Rules Explorer and select **RuleSets…** .
On the appearing panel add the Decision Table to your Rule Set **CustomerRuleSet**.



6. Use the Rules Explorer view and expand your Rule Set **CustomerRuleSet**. Double-check, that the Rule Set now contains references to both Decision Tables contained in your Rule project.



7. Right-click the Rule Set **CustomerRuleSet** in the Rules Explorer view and select **Run As** > **Run Rule Set**. On the appearing panel provide at least input values for the parameter elements **PurchaseHistoryCount**, **Rating**, **OpenInvoices**, **Country** (within BillingAddress), and **ZipCode** (within BillingAddress), but leave **Discount** and **City** (within BillingAddress) empty. Also check the box beside **Include empty values for String Types.** For testing you can load the provided sample input **<*workshop-dir*>\Exercise2\Resources\CustomerRuleSetInput.xml**.



Click **OK** to run the Rule Set.
View the results in the **Results** view. Which rules have been executed?
Repeat testing with different input values to verify that your Rule Set works properly.

This page is intentionally left blank.

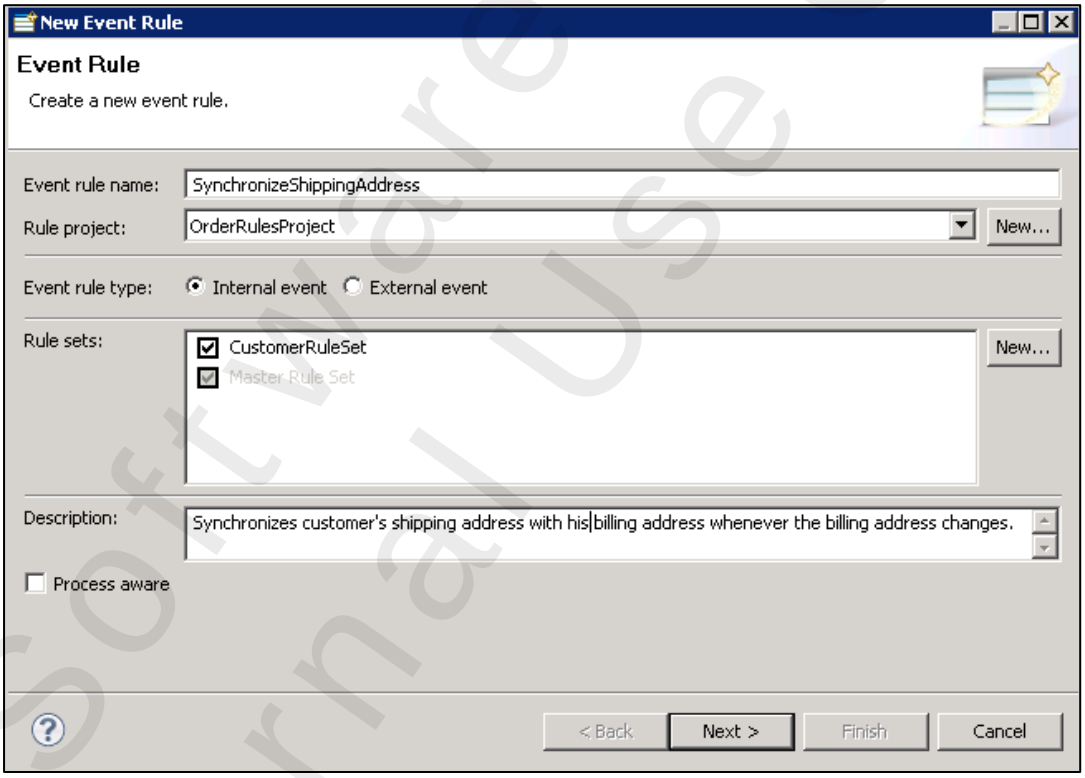
Exercise 03:
Simple Event Rules and Inference

Objectives

In this exercise, you will add an Event Rule to your rule project. The Event Rule will be fired whenever a city of the billing address is changed. In this case the Event Rule copies the changed city value to the shipping address. To enforce inference, the Event Rule belongs to your common customer Rule Set.

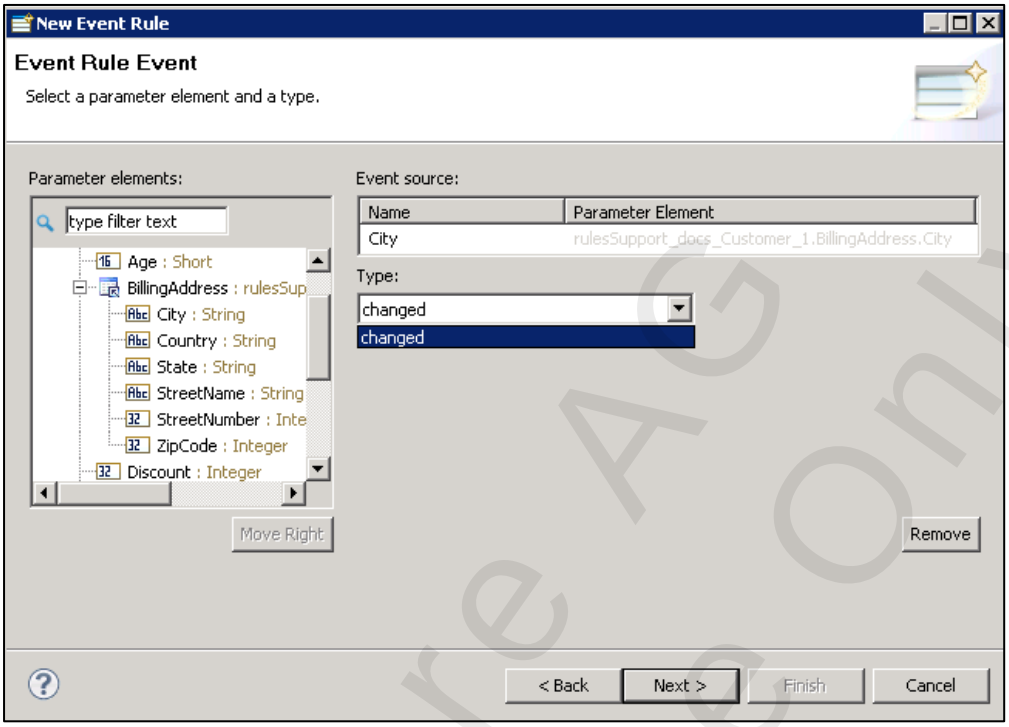
Steps

1. Launch **Software AG Designer** and switch to the **Rules Development** perspective.
2. Use the Rules Explorer view to add an Event Rule:
    a. Add an **internal** Event Rule named **SynchronizeShippingAddress** to your **OrderRulesProject** project. Add this Event Rule to your **CustomerRuleSet** Rule Set and provide a description text as shown below. Click **Next**.



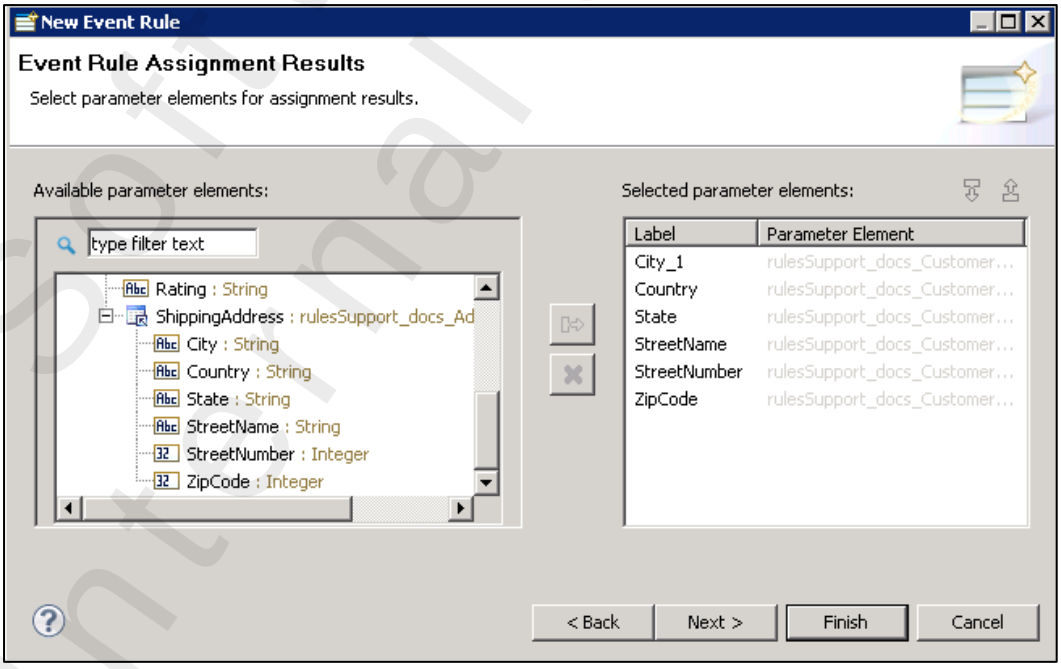
    b. On the subsequent panel select Data Model **rulesSupport_docs_Customer** as Event Rule parameter with I/O type **Both**. Leave the default parameter name unchanged. Click **Next**.

c. On the next panel select the parameter elements **City** (of **BillingAddress**) to be used as triggering element. Select (the one and only) Event type **changed** and click **Next.**



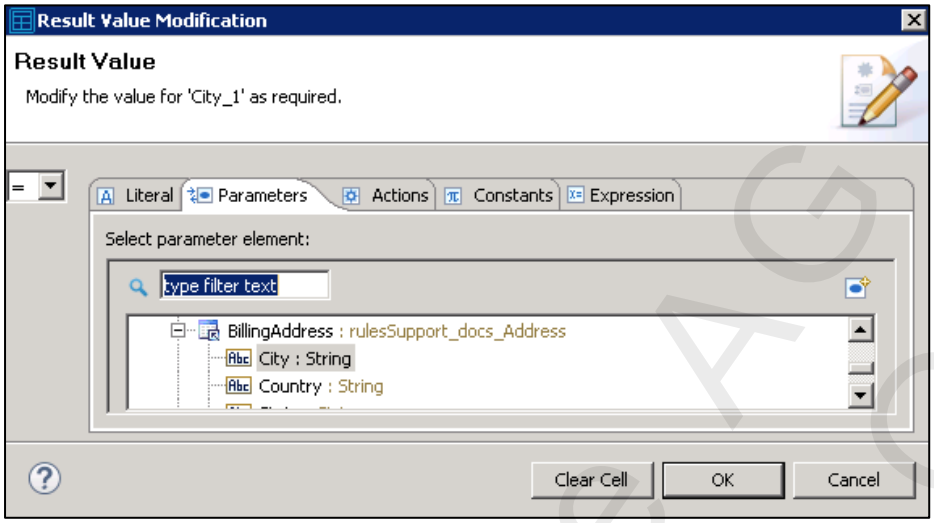
d. Finally specify the parameter elements used as results for your Event Rule. Select all elements (**City, Country, State, StreetName, StreetNumber, ZipCode**) of **ShippingAddress**.

Hint: You can use a multi-select.

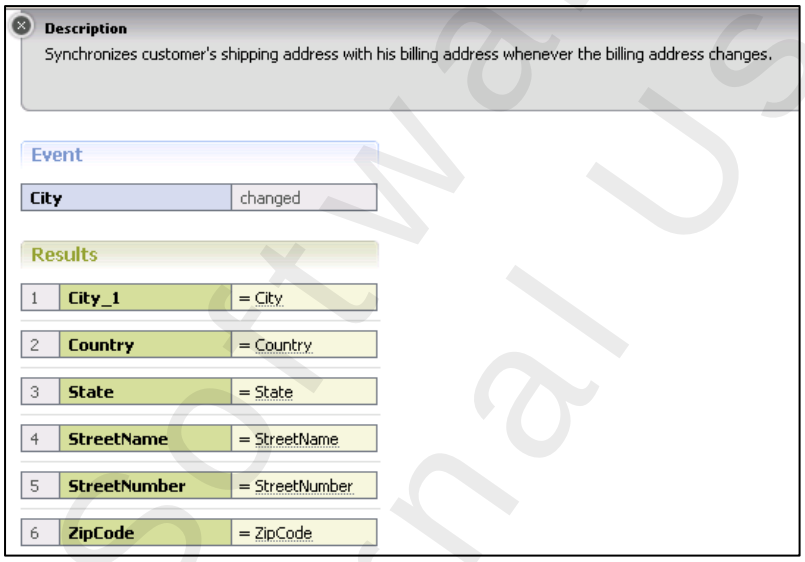


e. As we don't use action results here, click **Finish** to complete your Event Rule structure.

3. Use the Event Editor for your Event Rule **SynchronizeShippingAddress** to add the lacking assignments in the EventRule results section. For each **ShippingAddress** result element (**City**, **Country**, **State**, **StreetName**, **StreetNumber**, **ZipCode**), click the pencil in the Operator cell to open the extended cell editor. In the editor, assign the corresponding parameter element of the **BillingAddress** parameter.



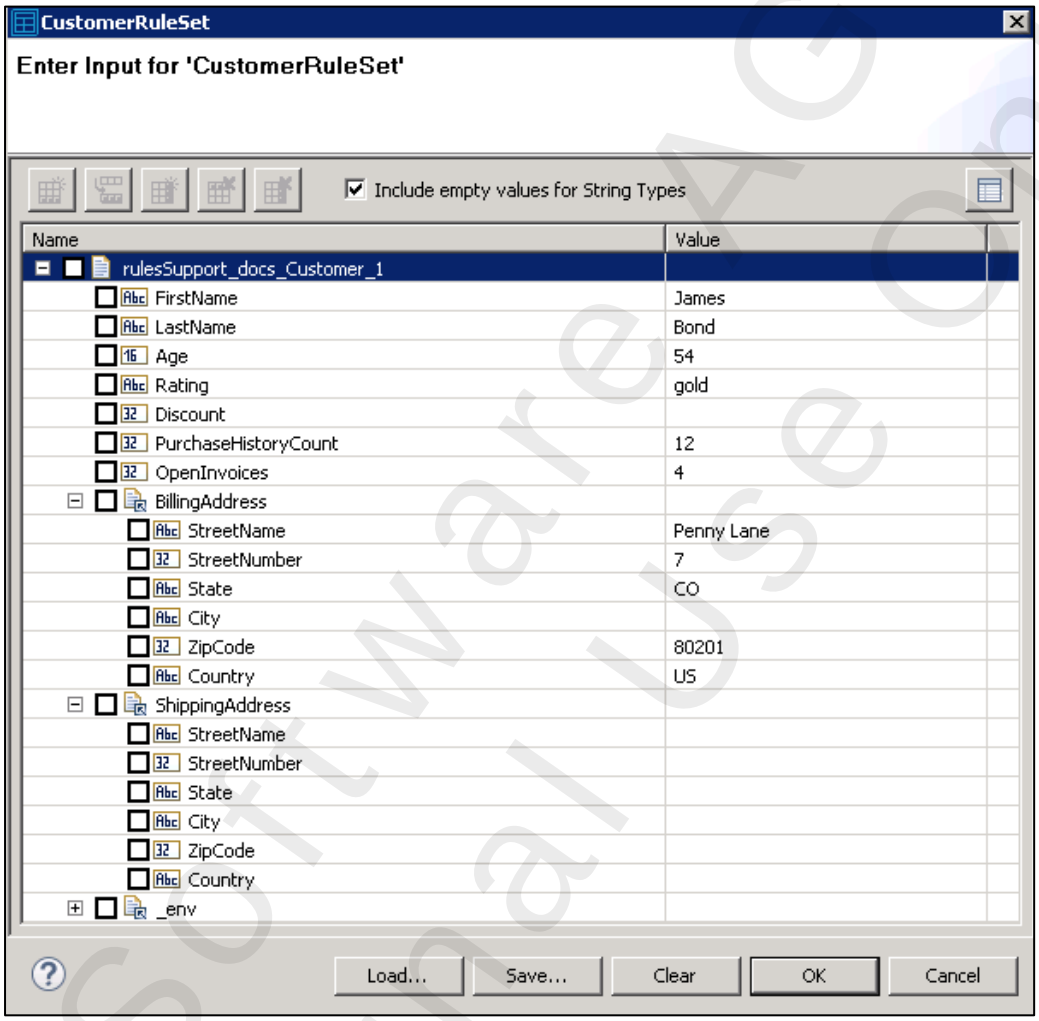
Finally, your Event Rule Table should look like this:



Save your Event Rule.

4. Use the Rules Explorer view and open your Rule Set **CustomerRuleSet**. Double-check, that the Rule Set now contains references to both Decision Tables as well to your Event Rule contained in your Rule project.

5. To test an Event Rule, another asset (e.g. Decision Table) has be executed that changes the appropriate Event Rule parameter element. To do so, right-click the Rule Set **CustomerRuleSet** in the Rules Explorer view and select **Run As** > **Run Rule Set**.
   On the appearing panel provide at least input values for the parameter elements **PurchaseHistoryCount**, **Rating**, **OpenInvoices**, **Country** (within **BillingAddress**), and **ZipCode** (within **BillingAddress**), but leave **Discount**, **City** (within **BillingAddress**), and **ShippingAddress** (all fields) empty. For testing you can also load the provided sample input ***<workshop-dir>*\Exercise3\Resources\CustomerRuleSetInput.xml**.
   Check the box beside **Include empty values for String Types** and click **OK** to run the Rule Set.



   View the results in the **Results** view. Repeat testing with different input values to verify that your Rule Set works properly.

Exercise 04:
Decision Trees

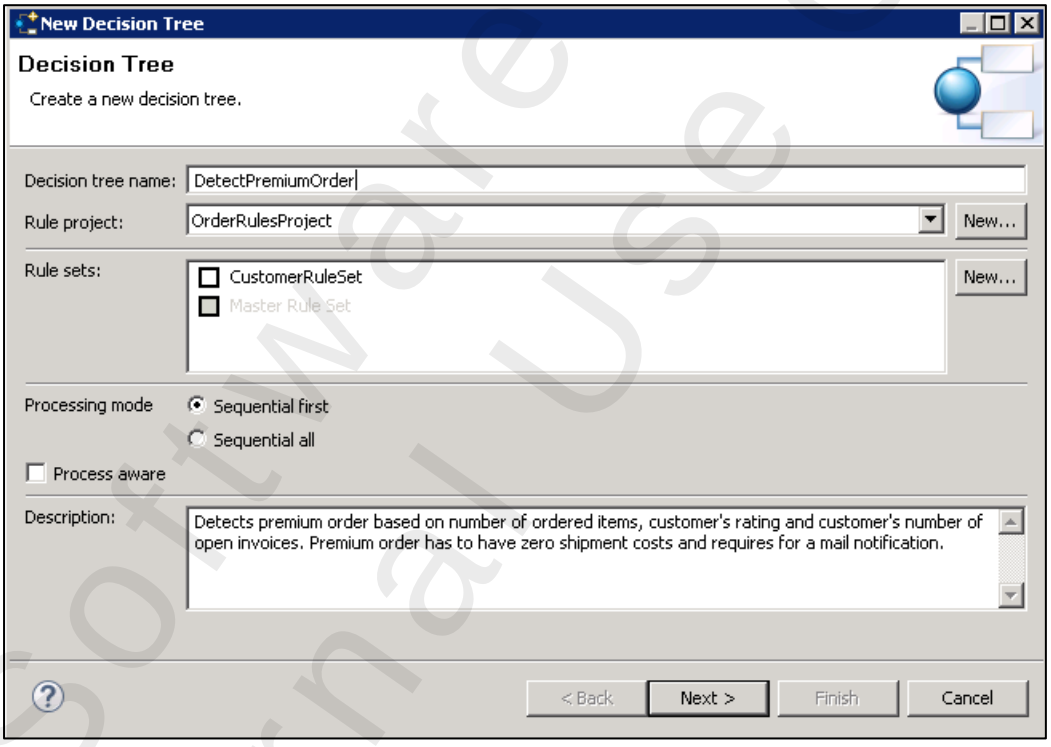
Objectives

In this exercise, you will add a Decision Tree to your rule project. The Decision Tree acts on the PurchaseOrder Data Model and identifies “premium orders” based on customer’s number of open invoices and rating as well as the number of ordered items in an order. “Premium orders” are shipped free of charge, other orders will have assigned ranked shipping costs.

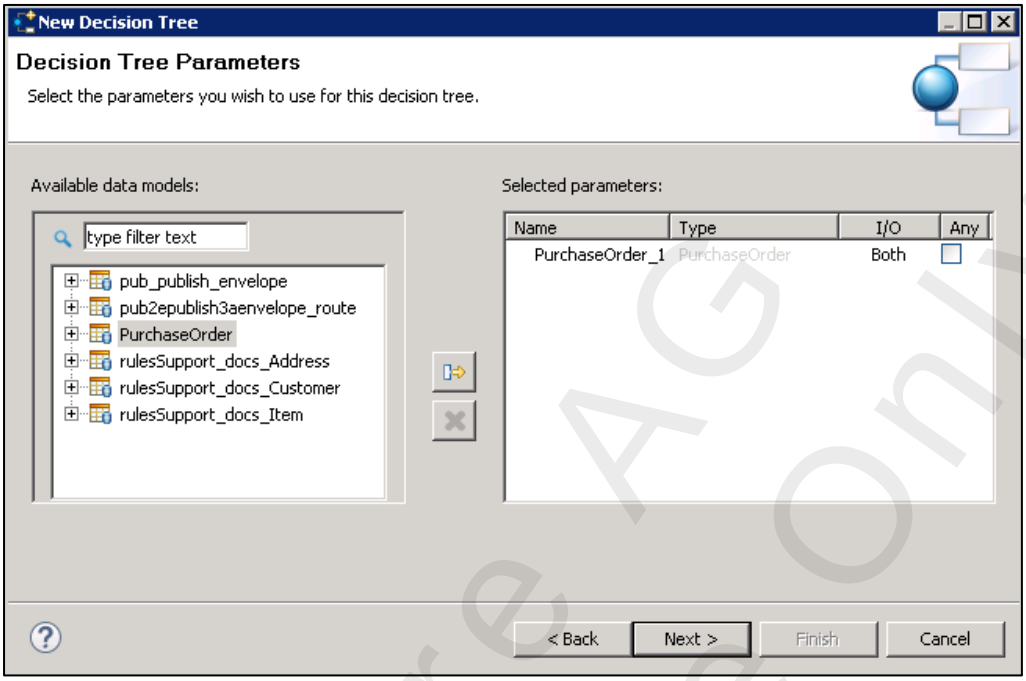
Steps

1. Use the Rules Explorer view to add a Decision Tree:

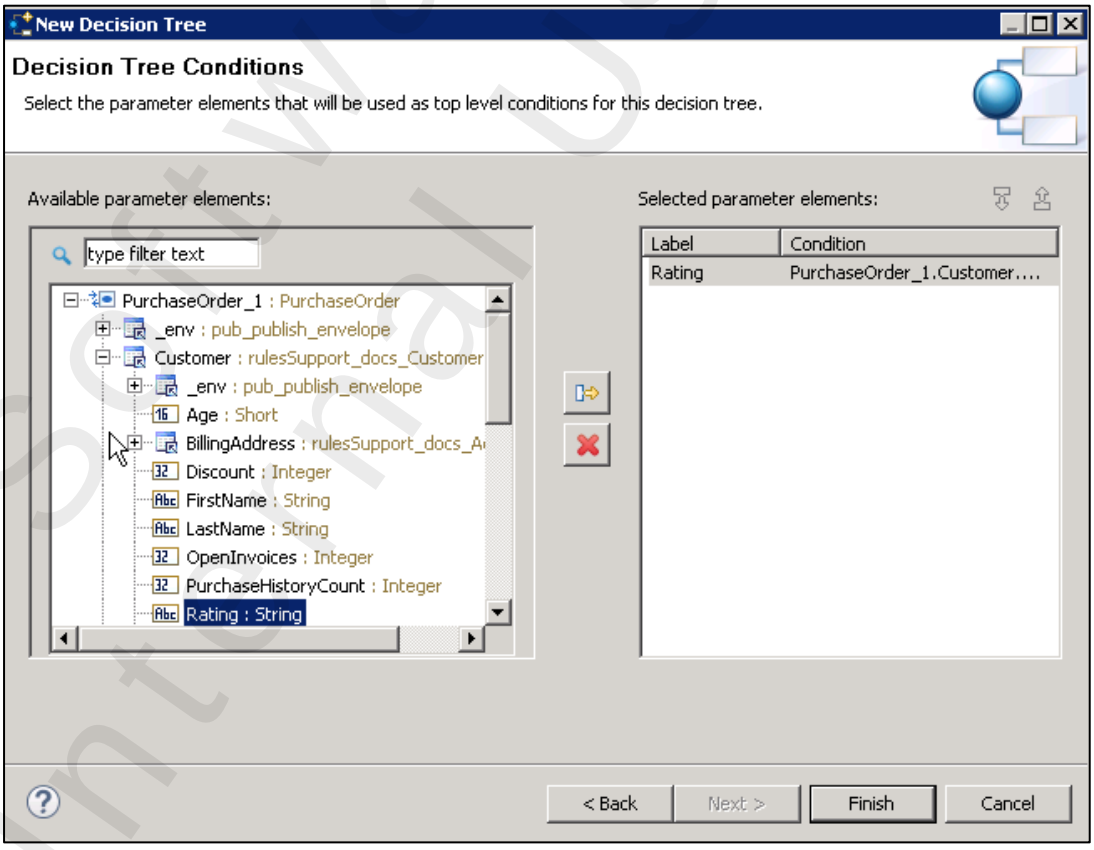
    a. Add a Decision Tree named **DetectPremiumOrder** to your **OrderRulesProject** project. Do NOT add this Decision Tree to your CustomerRuleSet Rule Set. Provide a description text as shown below, choose Processing mode **Sequential first**, and click **Next**.



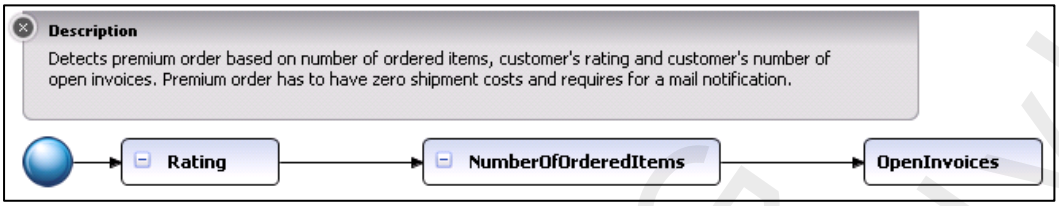
b. On the subsequent panel select Data Model **PurchaseOrder** as Decision Table parameter with I/O type **Both**. Leave the default parameter name unchanged and click **Next**.



c. On the next panel select PurchaseOrder's **Rating** as parameter used in the top level condition for the Decision Tree. Click **Finish**.

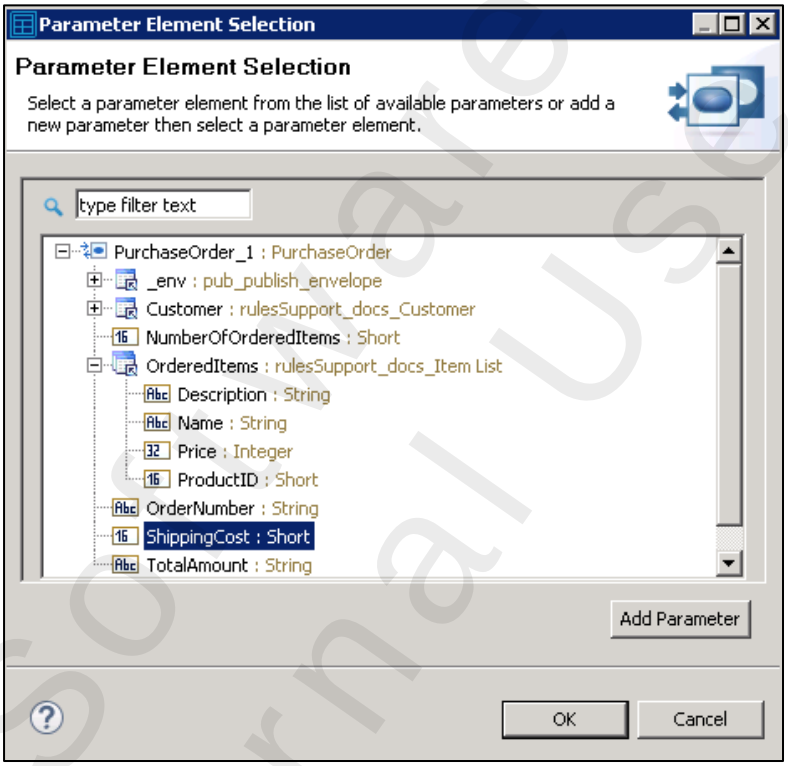


d. The Decision Tree will be opened in the Decision Tree Editor pane. Right-click parameter element **Rating** and select **Add Condition** to add Customer's **NumberOfOrderedItems** as second parameter to be used in a subsequent condition. Right-click parameter element **NumberOfOrderedItems** to add Customer's **OpenInvoices** as third parameter to be used in a subsequent condition.



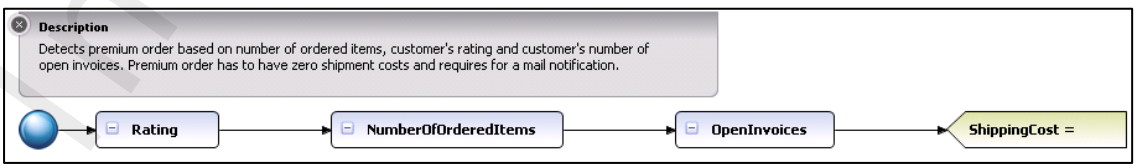
Note: As an alternative, you can also drag conditions from the Palette onto an existing parameter element to form up the rule structure.

e. Right-click parameter element **OpenInvoices** and select **Add Assignment** to add PurchaseOrder's **ShippingCost** to be used in a result assignment.



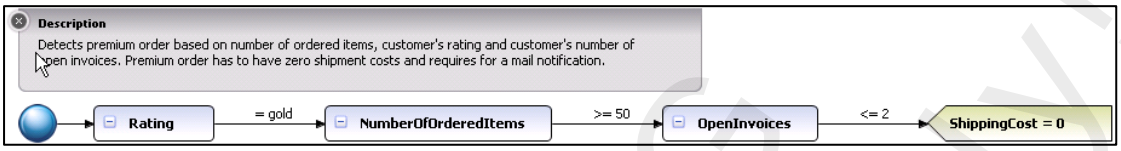
Note: As an alternative, you can also drag assignments from the Palette onto an existing parameter element to form up the rule structure.

Your Decision Tree structure should now look like this:

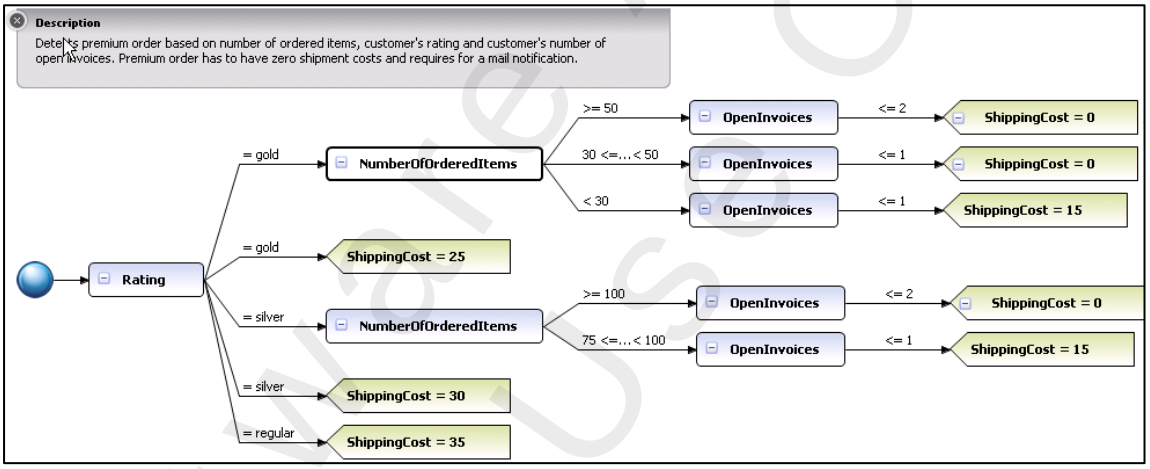


2. Use the Decision Tree Editor for your Decision Tree **DetectPremiumOrder** to implement eight rules:

   a. To build up the first rule, configure the transition from **Rating** to **NumberOfOrderedItems** to express the condition **Rating = gold.** Next modify the transition from **NumberOfOrderedItems** to **OpenInvoices** to express the condition **NumberOfOrderedItems >= 50**. Transition from **OpenInvoices** to **ShippingCost** should express the condition **OpenInvoices <= 2**. Finally assign value **0** to **ShippingCost**.



   b. To create the other rules, copy and paste nodes including their descendant nodes to add seven additional rules as tree branches. Adjust their condition and assignment values to reflect the screen shot below:



   c. Save your Decision Tree.

3. Right-click Decision Tree **DetectPremiumOrder** in the Rules Explorer and select **Run As** > **Run Decision Tree**. On the appearing panel, specify at least the necessary input values for **OrderNumber**, **NumberOfOrderedItems**, **Rating**, and **OpenInvoices**. For testing you can also load the provided sample input ***<workshop-dir>*\Exercise4\Resources\DetectPremiumOrderDecisionTreeInput.xml**. Check the box beside **Include empty values for String Types** and click **OK**.
   View the results in the **Results** view. Depending on your input for **NumberOfOrderedItems**, **Rating**, and **OpenInvoices**, different values for **ShippingCost** should be returned. Repeat testing with different input to verify that your Rule Set works properly.

Name	Value
PurchaseOrder_1	
NumberOfOrderedItems	120
ShippingCost	0
OrderNumber	4711
TotalAmount	

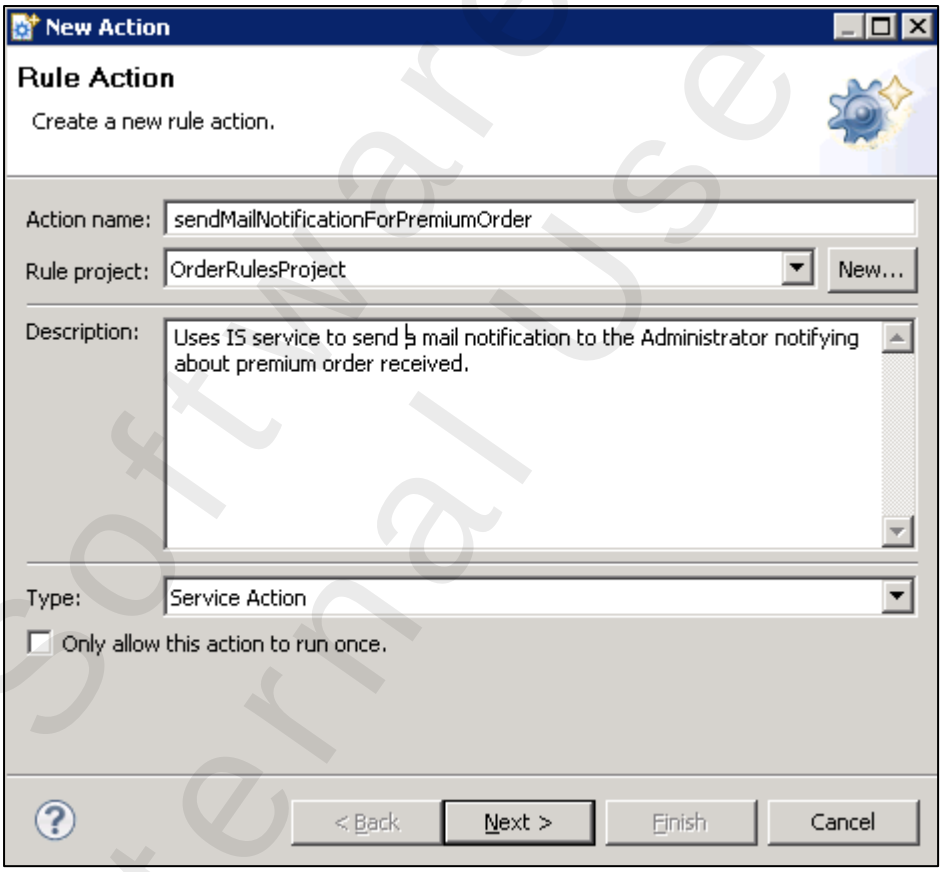
Exercise 05:
Service Action as Result

Objectives

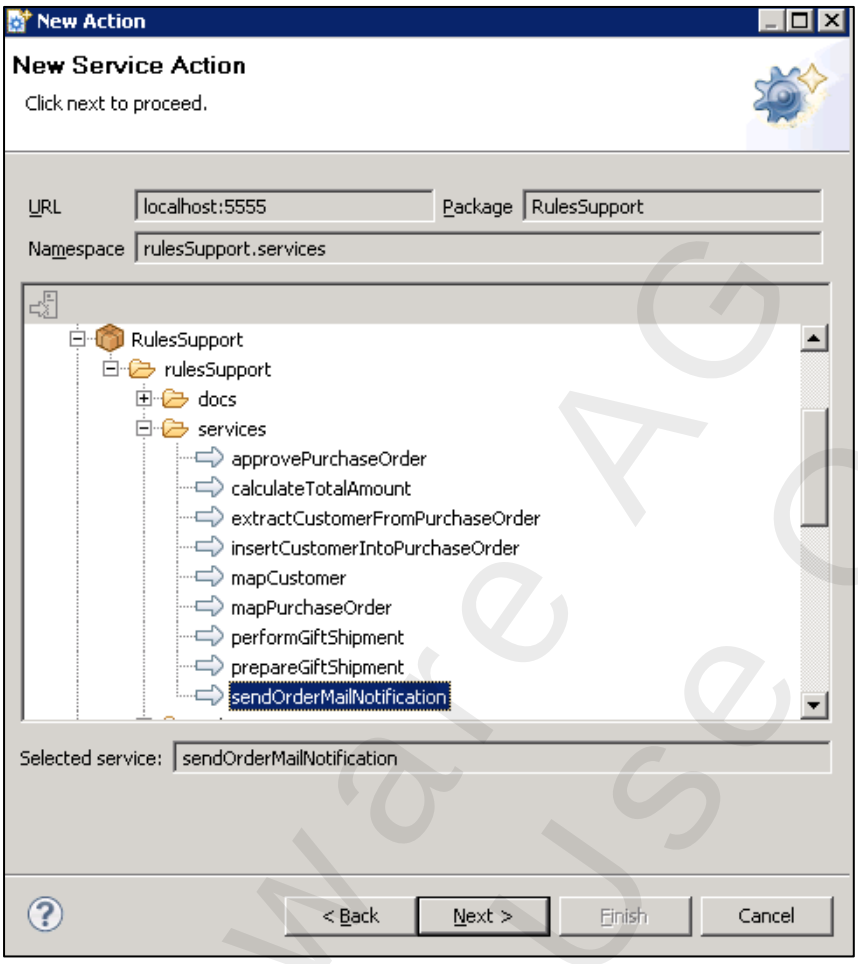
In this exercise, you will enhance the Decision Tree created in the previous exercise by the invocation of a Service Action. The invoked Service Action also acts on the PurchaseOrder Data Model and uses an IS service to send an email notification to the process administrator. It will be invoked by the Decision Tree as a result whenever a “premium order” has been detected.

Steps

1. Launch **Software AG Designer** and switch to the **Rules Development** perspective.
2. Use the Rules Explorer view to add an Action:
    a. Add an Action named **sendMailNotificationForPremiumOrder** to your **OrderRulesProject** project. Select action type **Service Action** and provide a description text as shown below. Click **Next**.



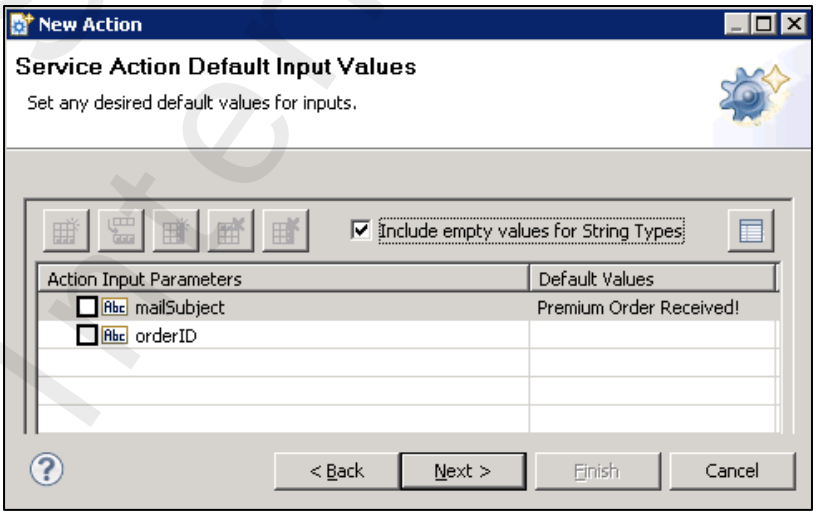
b. On the subsequent panel select the existing IS service **rulesSupport.services:sendOrderMailNotification** as service to be invoked. Click **Next**.



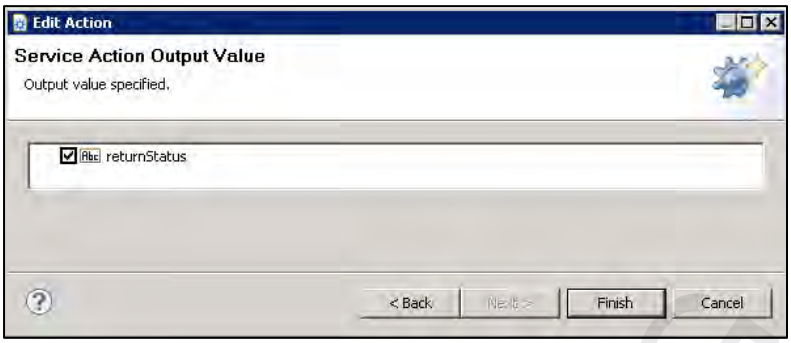
c. On the next panel confirm the action input parameters **mailSubject** and **orderID**. Specify the text **Premium Order Received!** as default value for the parameter **mailSubject**.

Hint: Mail receiver (Administrator@company.com), mail sender (IsProcessEngine@company.com), and mail body are set by the invoked IS service.

Check the box beside **Include empty values for String Types** and click **Next**.



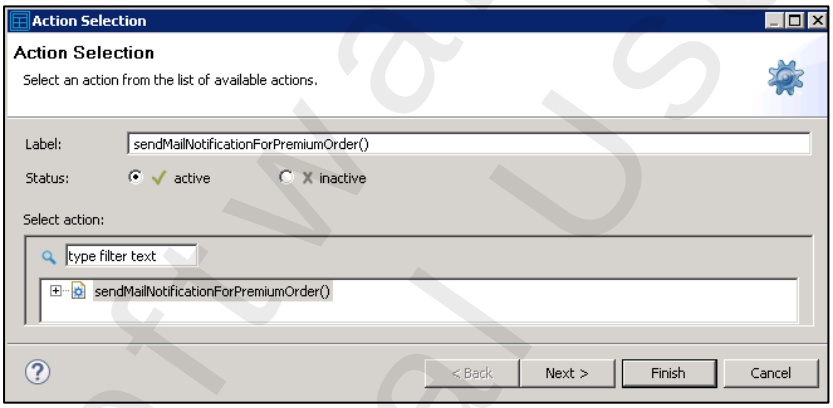
d. Finally select the IS service return parameter **returnStatus** as action service output value:



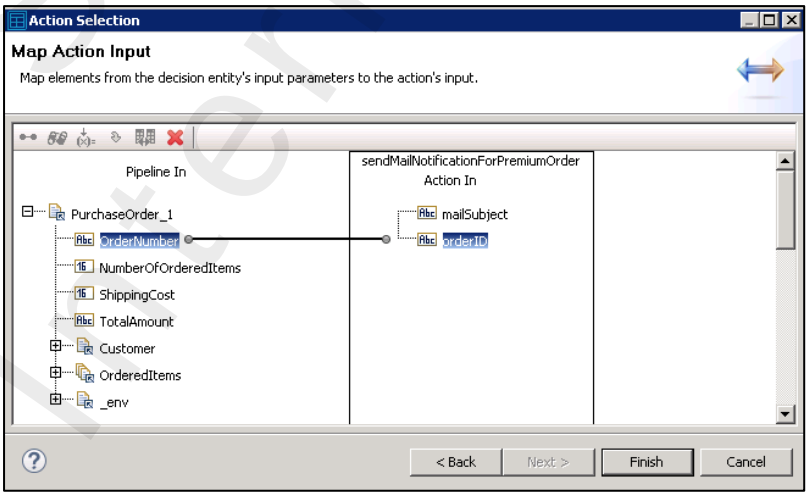
3. Click **Finish** to complete your Service Action.
4. Add your Service Action to be invoked as a result to your Decision Tree DetectPremiumOrder:
   a. If not opened yet, open Decision Tree **DetectPremiumOrder** from the Rules Explorer view.
   b. Drag the Action icon from the Palette onto assignment node ShippingCost of the first rule.

   Note: You can also right-click node ShippingCost and select **Add Action**.

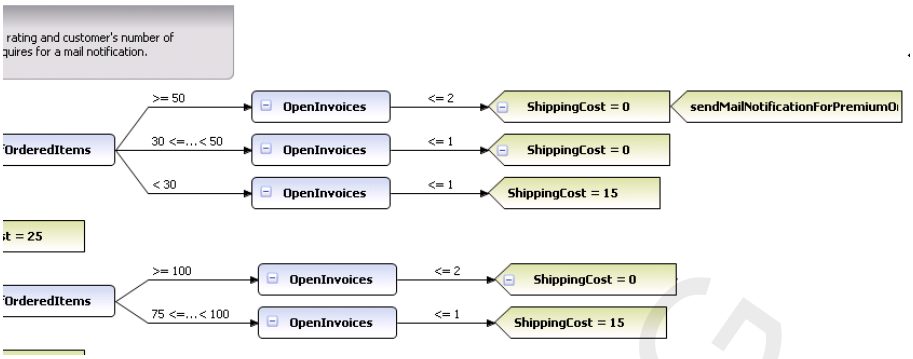
   c. Choose Service Action **sendMailNotificationForPremiumOrder** to be invoked. Select the action to be **active** for this rule. Click **Next**.



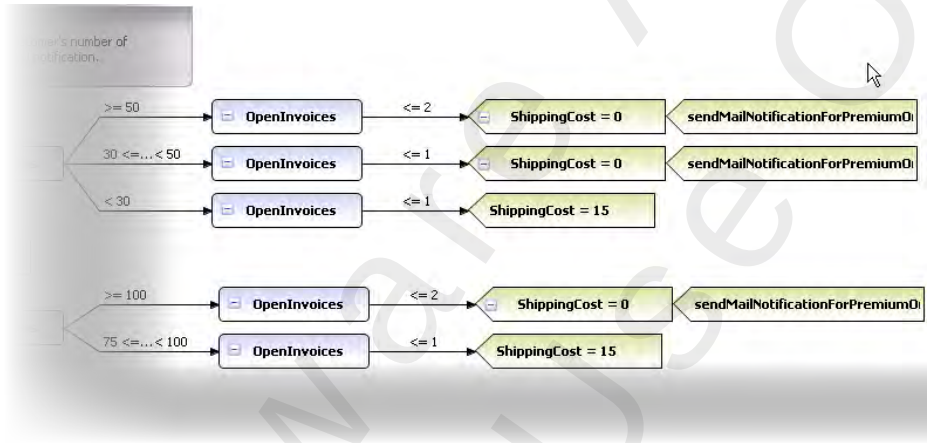
   d. On the next panel map the Decision Tree parameter element **OrderNumber** of parameter **PurchaseOrder_1** to the input parameter **orderID** of your Service Action.



e. Click **Finish**.



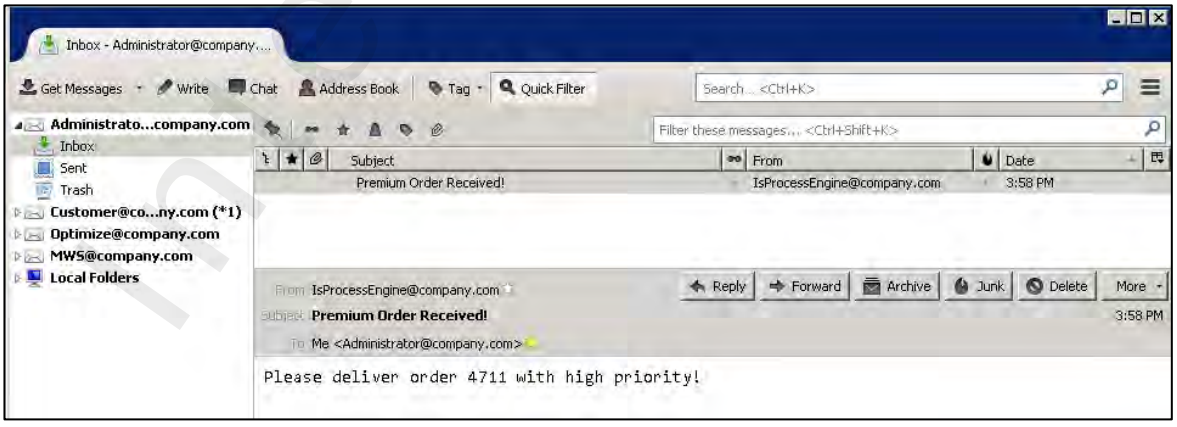
f. Copy action node **sendMailNotificationForPremiumOrder()** and paste it to the **ShipmentCost** node of each other rule that returns zero shipping costs.



Save your Decision Tree.

5. Right-click Decision Tree **DetectPremiumOrder** in the Rules Explorer and select **Run As** > **Run Decision Tree**. On the appearing panel, specify at least the necessary input values for **OrderNumber**, **NumberOfOrderedItems**, **Rating**, and **OpenInvoices**. For testing you can also load the provided sample input ***<workshop-dir>*\Exercise5\Resources\DetectPremiumOrderDecisionTreeInput.xml**. Check the box beside **Include empty values for String Types** and click **OK**.

   Depending on your input for **NumberOfOrderedItems**, **Rating**, and **OpenInvoices**, different values for **ShippingCost** should be returned. For a “premium order” (ShippingCost = 0) you should receive an additional mail notification mail as a result of the invoked Service Action. To check your received emails, start the preconfigured **Mozilla Thunderbird** mail client on your VM. Use the mail account **Administrator@company.com**. If prompted, provide **manage** as password.



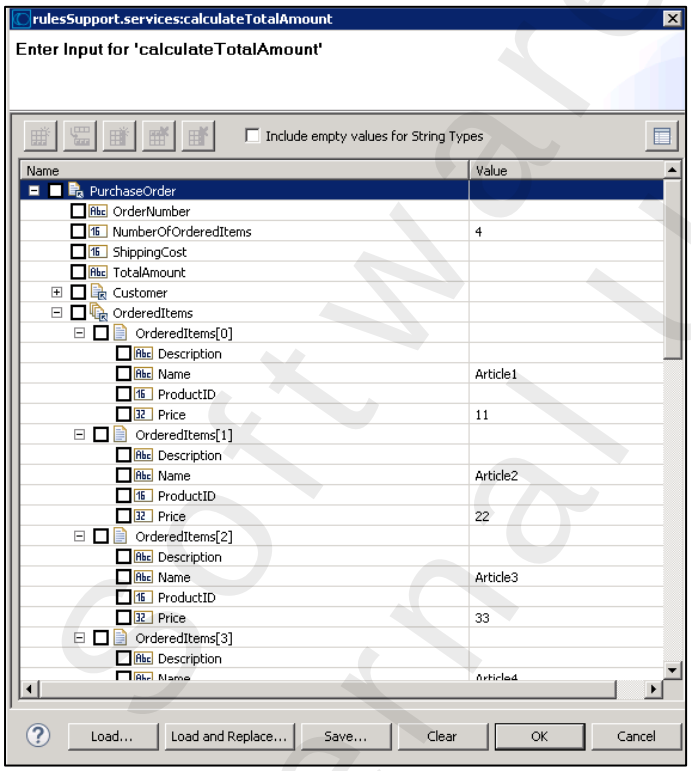
Exercise 06:
Service Action in an Assignment

Objectives

In this exercise, you will add a Service Action that calculates the total amount to be invoiced for all items in an order. This Service Action will be used in a new Decision Table within an assignment. The execution of the Service Action depends on the customer's rating.

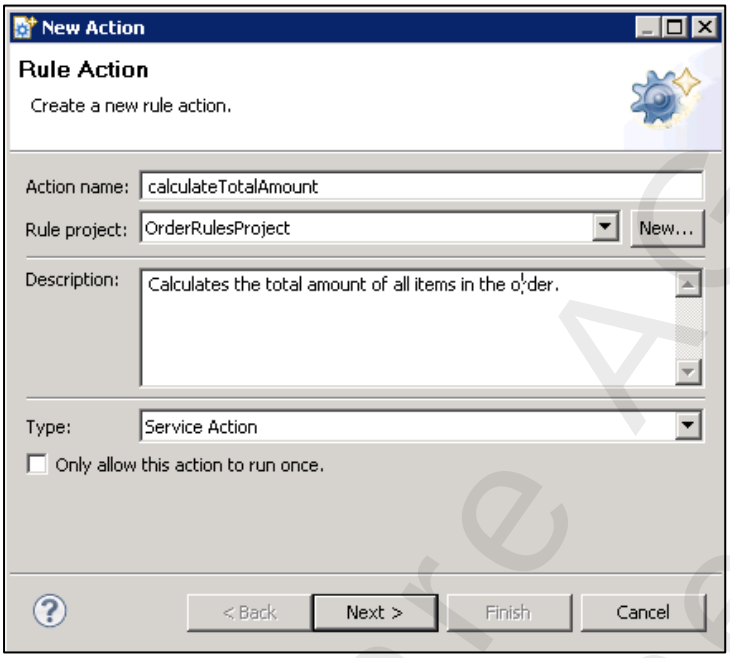
Steps

1. Launch **Software AG Designer** and switch to the **Rules Development** perspective.
2. Use the Package Navigator view and navigate to the service **rulesSupport.services:calculateTotalAmount** within IS package **RulesSupport**. To test the service, right-click the service and select **Run As** > **Run Flow Service**.
   For testing you can load the provided sample input **<workshop-dir>\Exercise6\Resources\CalculateTotalAmountServiceInput.xml**.
   Check the box beside **Include empty values for String Types** and click **OK**.

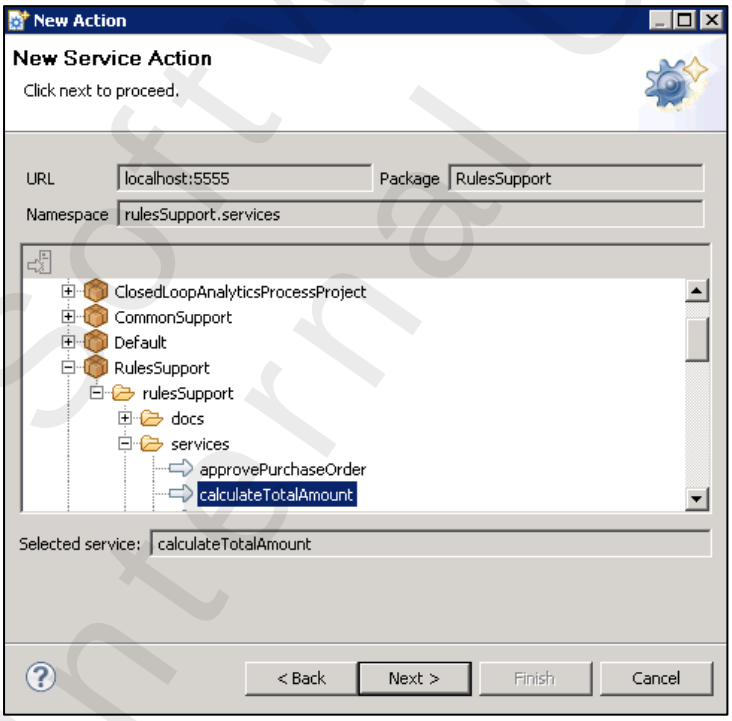


3. Use the Rules Explorer view to add a another Action:

   a. Add an Action named **calculateTotalAmount** to your **OrderRulesProject** project. Select action type **Service Action**, provide a description text as shown on the screen shot below, and click **Next**.



   b. On the subsequent panel select the existing IS service **rulesSupport.services:calculateTotalAmount** from IS package **RulesSupport** as service to be invoked. Hit **Next**.

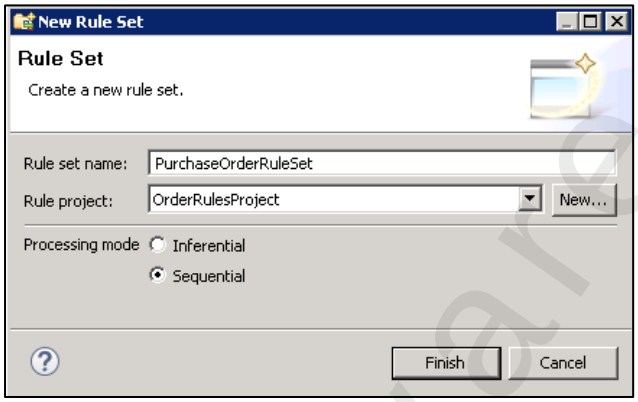


   c. On the next panel confirm the action input parameters **PurchaseOrder**. Don't specify a default value for any parameter element. Just click **Next**.

d. Finally select the IS service return parameter **totalAmount** as action service output value:

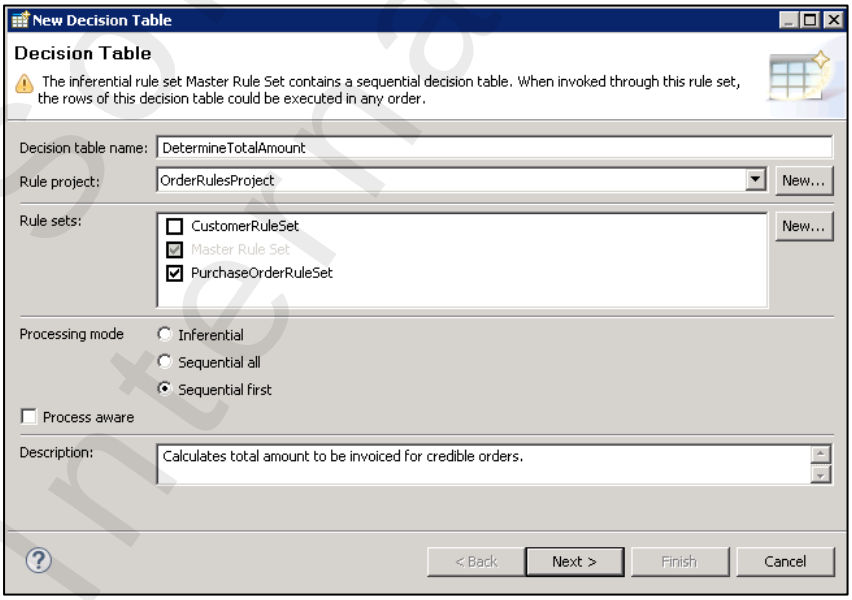
e. Click **Finish** to complete your Service Action.

4. Use the Rules Explorer view to create a new Rule Set named **PurchaseOrderRuleSet** in your OrderRulesProject. Choose Processing mode **Sequential** and click **Finish**.



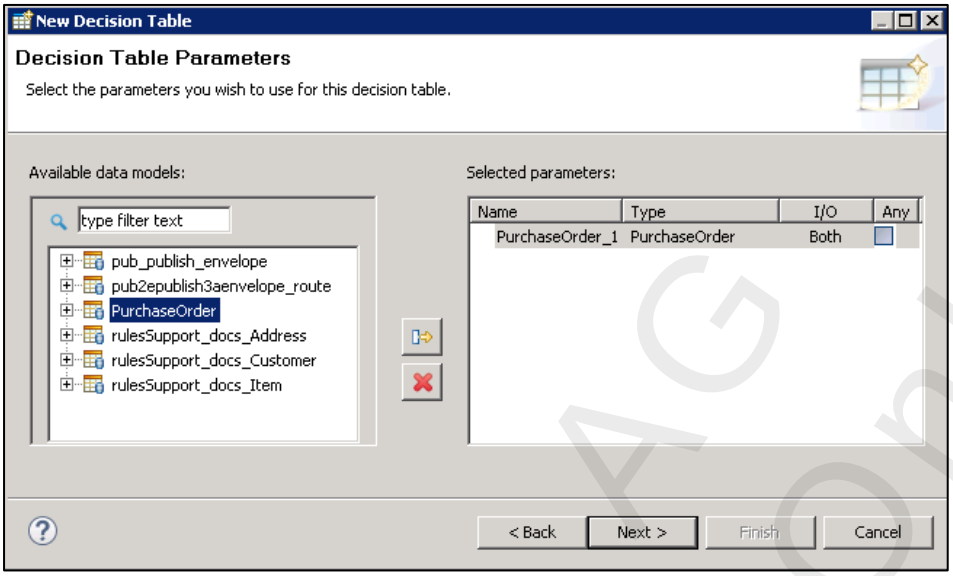
5. Use the Rules Explorer to view to add another Decision Table:

a. Add a Decision Table named **DetermineTotalAmount** to your **OrderRulesProject** project. Add this Decision Table to your **PurchaseOrderRuleSet** Rule Set, choose Processing mode **Sequential first** and provide a description text as shown on the screen shot below. Click **Next**.

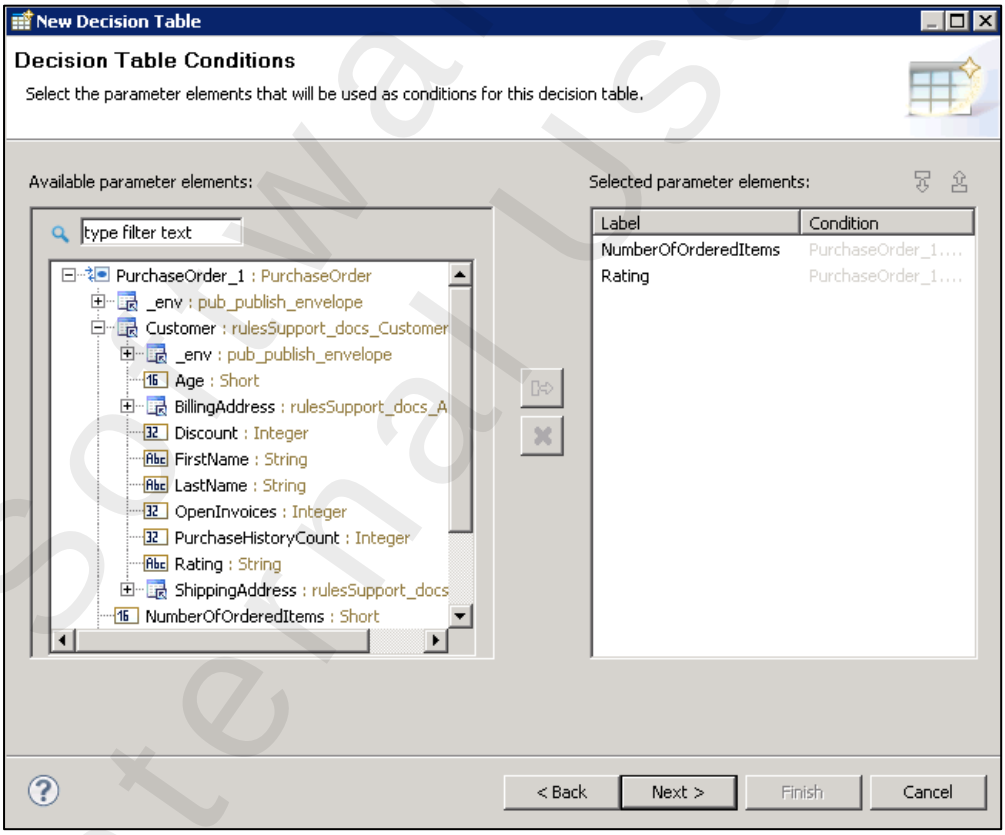


Note: Accept the warning about the inferential Processing mode of the Master Rule Set.

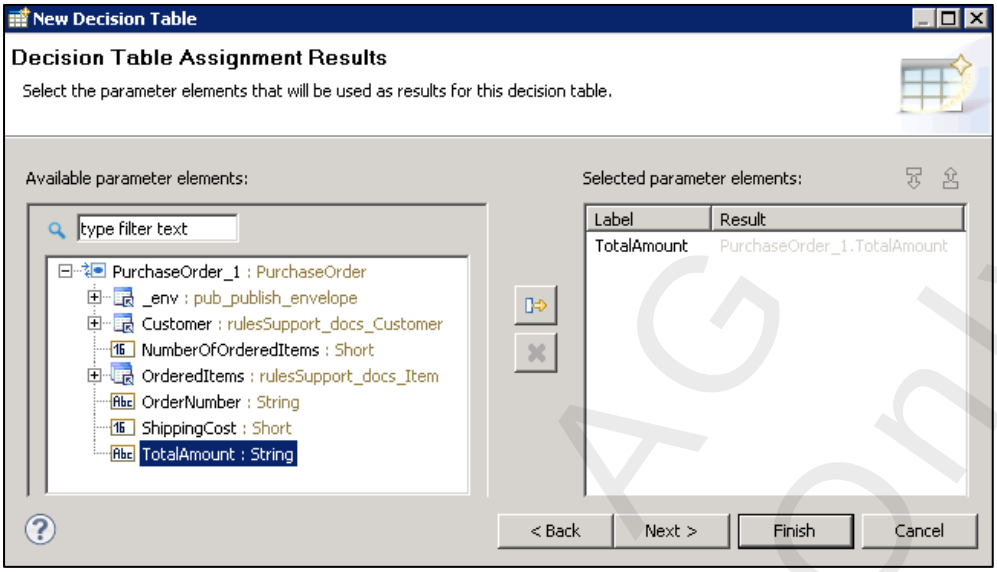
b. On the subsequent panel select Data Model **PurchaseOrder** as Decision Table parameter with I/O type **Both**. Leave the default parameter name unchanged and click **Next**.



c. On the next panel select the parameter elements **NumberOfOrderedItems** and **Rating** (contained in **Customer**) to be used later in your rule conditions. Click **Next**.



d. Finally specify the parameter element used as results for your Decision Table. Select the element **TotalAmount** only:

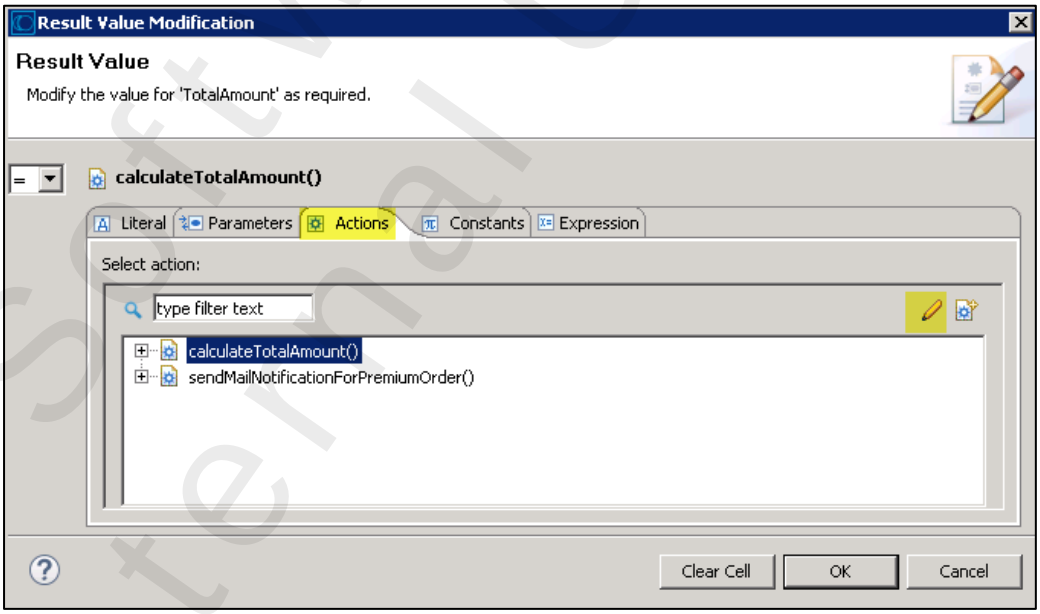


e. Click **Finish** to complete your Decision Table structure.

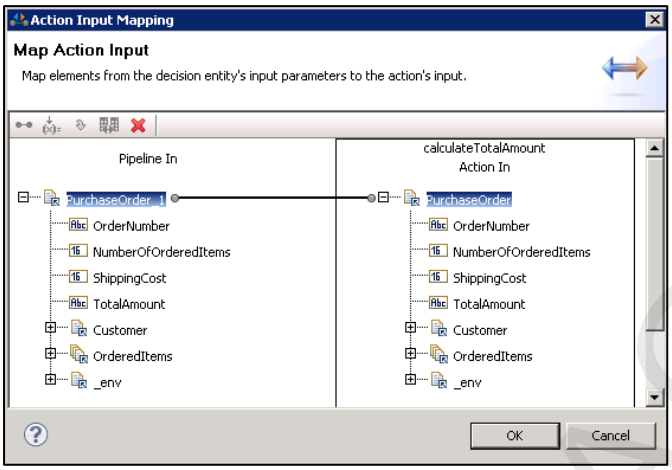
6. Use the Decision Table Editor for your Decision Table **DetermineTotalAmount** to configure the first (empty) rule:

   a. Click the pencil in the assignment cell **TotalAmount** to configure the assignment.

   b. On the extended cell editor panel, open the **Actions** tab. Highlight Service Action **calculateTotalAmount** as action to return the value to be assigned. Do NOT click OK.



c. Still on the Actions tab, click on the pencil icon to perform the following action input mapping:



d. Hit **OK** twice**.** Your Decision Table should now look like this:

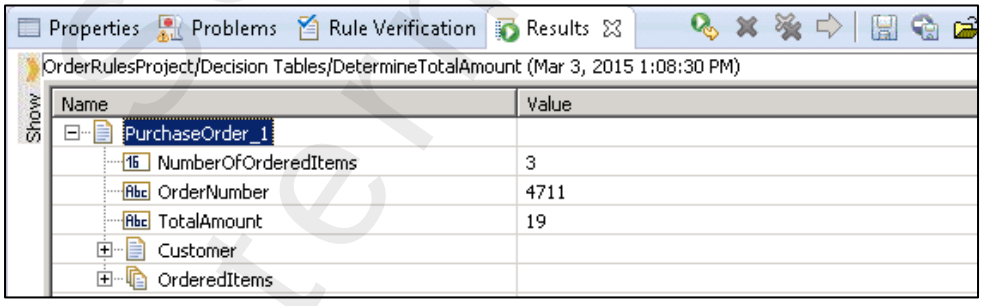
	NumberOfOrderedItems	Rating	TotalAmount
1			= calculateTotalAmount()

7. Copy rule one and paste it three times. Adjust the copied rules to correspond to the following screen shot:

	NumberOfOrderedItems	Rating	TotalAmount
1	> 0	= gold	= calculateTotalAmount()
2	> 0	= silver	= calculateTotalAmount()
3	> 0	= regular	= calculateTotalAmount()
4		= fraud	= calculateTotalAmount()

Suppress warning for rule four and save your Decision Table.

8. Right-click Decision Table **DetermineTotalAmount** in the Rules Explorer and select **Run As > Run Decision Table**. On the appearing panel, specify at least the necessary input values for **NumberOfOrderedItems**, **Rating**, and add some **OrderedItems**, each having an assigned **Price**. For your convenience you can also load the provided sample input ***<workshop-dir>*\Exercise6\Resources\DetermineTotalAmountDecisionTableInput.xml**. Check the box beside **Include empty values for String Types** and click **OK**.



9. *For extra credit*: Run your Rule Set **PurchaseOrderRuleSet**. For testing you can use the same sample input from above.

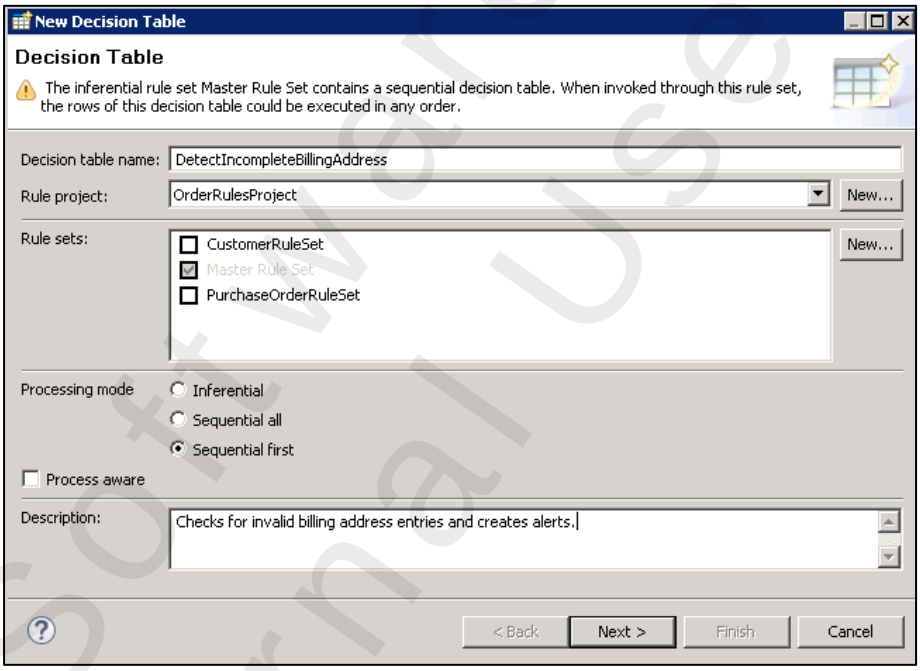
Exercise 07:
Empty Values and New Data Actions

Objectives

In this exercise, you will add a Decision Table and a New Data Action to your rule project. The New Data Action returns an Alert parameter with some preconfigured parameter elements. This Action will be executed by the Decision Table when an incomplete billing address has been detected. Depending on the number of empty parameter elements, different alerts will be thrown.

Steps

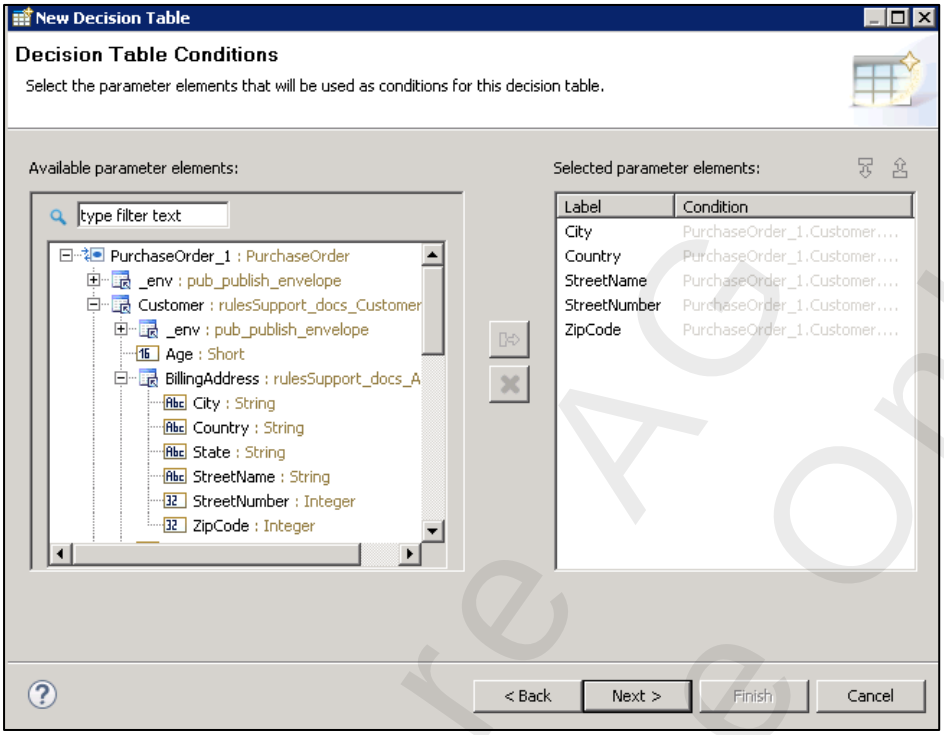
1. Launch **Software AG Designer** and switch to the **Rules Development** perspective.
2. Use the Rules Explorer view to add a Decision Table:
    a. Create a new Decision Table named **DetectIncompleteBillingAddress**. Do NOT add this Decision Table to any additional Rule Set (except for the default Master Rule Set) yet. Select Processing mode **Sequential first**, provide a description text as shown on the screen shot below, and click **Next**.



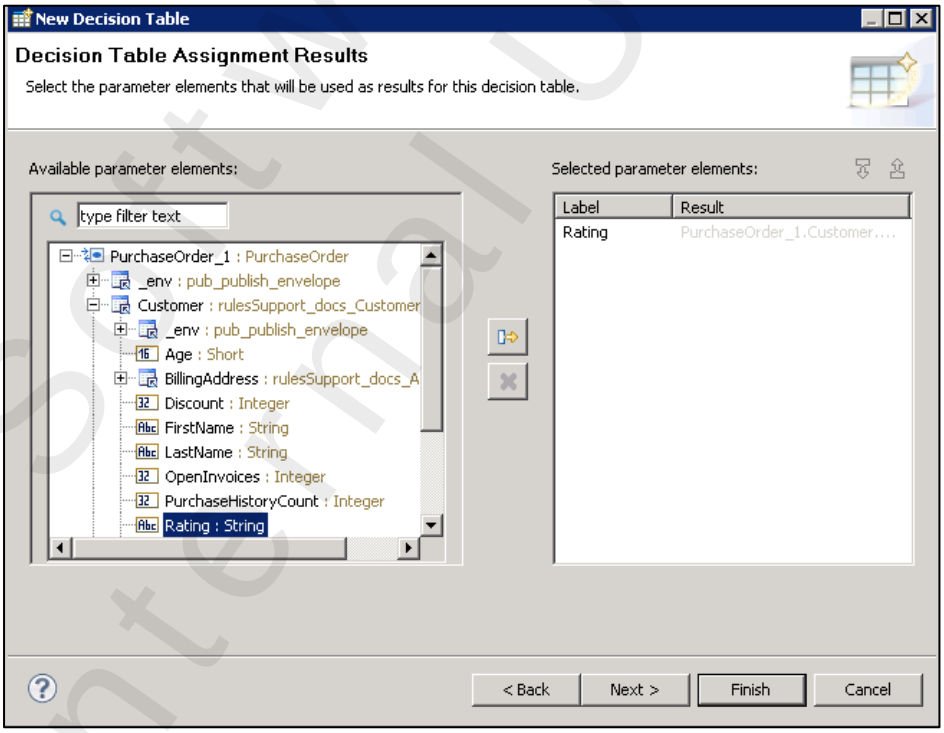
Note: Accept the warning about the inferential Processing mode of the Master Rule Set.

    b. On the subsequent panel select Data Model **PurchaseOrder** as Decision Table parameter with I/O type **Both**. Leave the default parameter name unchanged and click **Next**.

c. On the next panel select the parameter elements **City**, **Country**, **StreetName**, **StreetNumber**, **ZipCode** (all of Customer's **BillingAddress**) to be used later in your rule conditions.

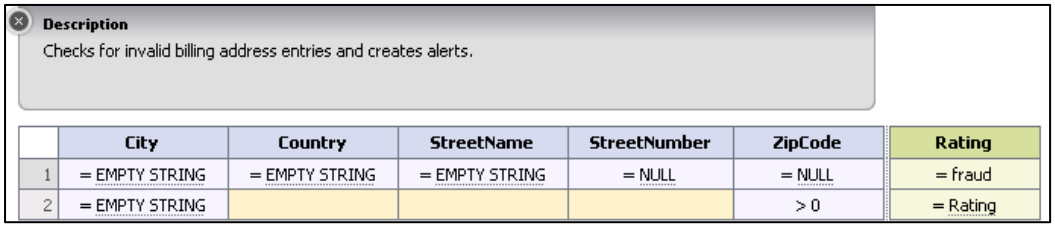


d. Finally specify the parameter elements used as results for your Decision Table. Select the element **Rating (**within **Customer**) only:

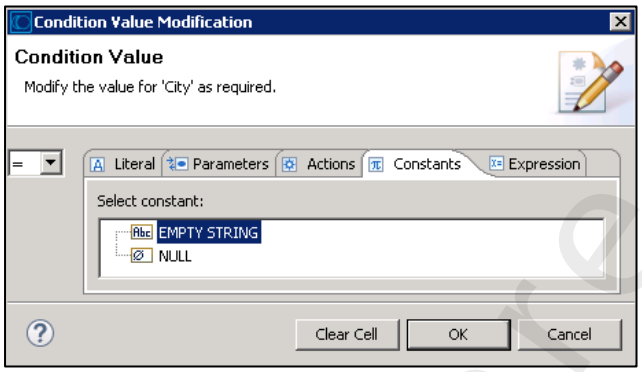


e. Click **Finish** to complete your Decision Table structure.

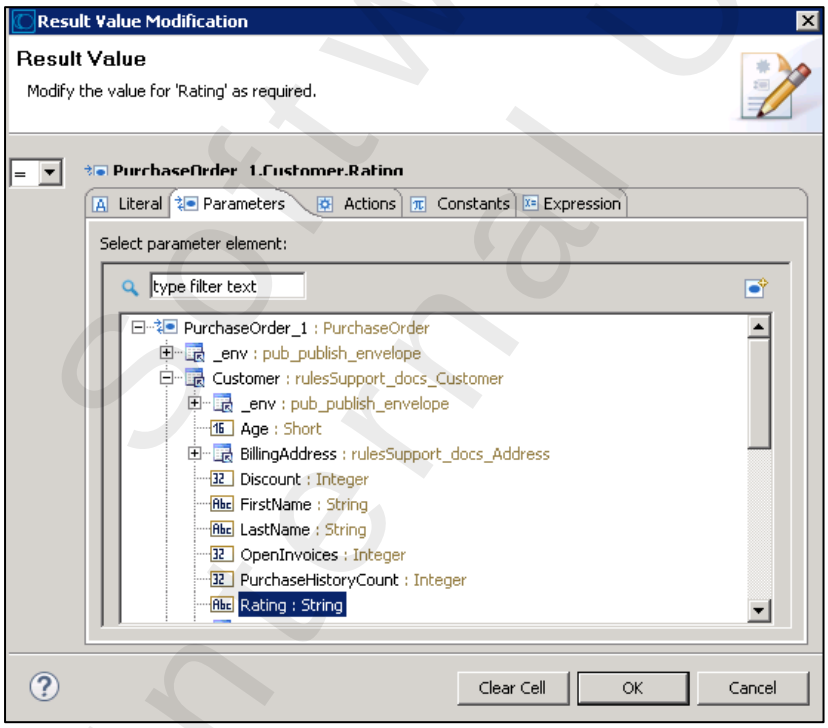
3. Use the Decision Table Editor for your Decision Table **DetectIncompleteBillingAddress** to insert two rules. The final Decision Table should look like this:



Hint: Both rules check for empty values contained in string or numeric parameter elements. Appropriate constants to be used could be found in the extended cell editor on tab **Constants**.

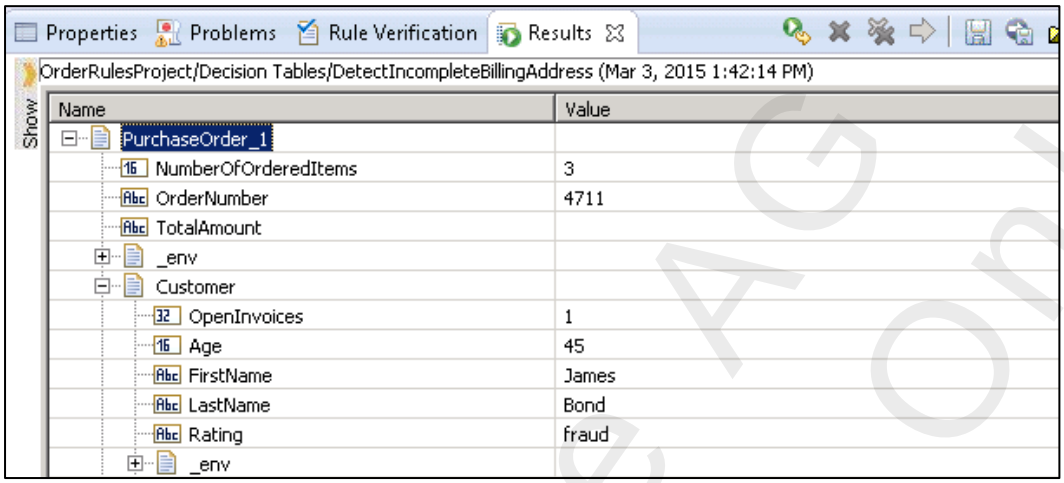


Hint: In the second rule, Rating should be assigned to its original value as contained in the parameter element.



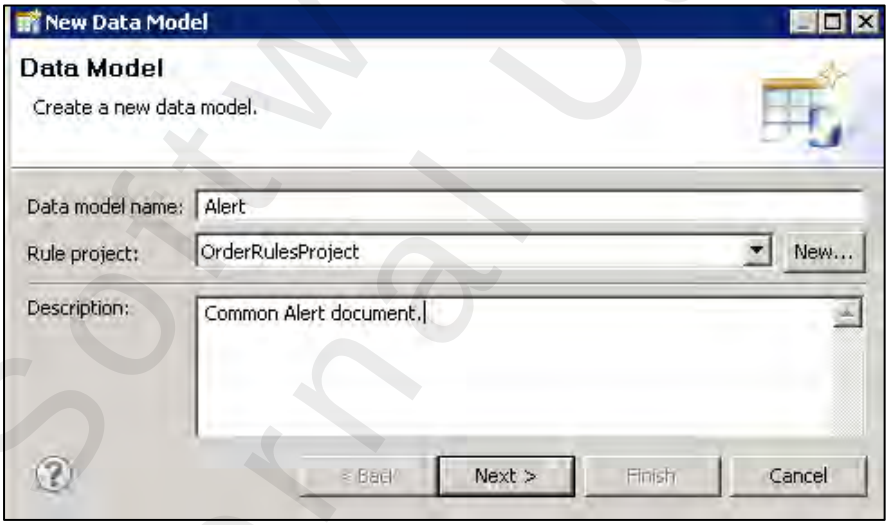
Suppress all warnings of the second rule and save your Decision Table in your Rules Project.

4. Right-click Decision Table **DetectIncompleteBillingAddress** in the Rules Explorer and select **Run As > Run Decision Table**. On the appearing test panel, check the box beside **Include empty values for String Types**. Provide input to ensure both rules are working properly. For testing you can load provided sample input from file:
<*workshop-dir*>\Exercise7\Resources\DetectIncompleteBillingAddressDecisionTableInput.xml
Click **OK** to run the Decision Table in Designer.

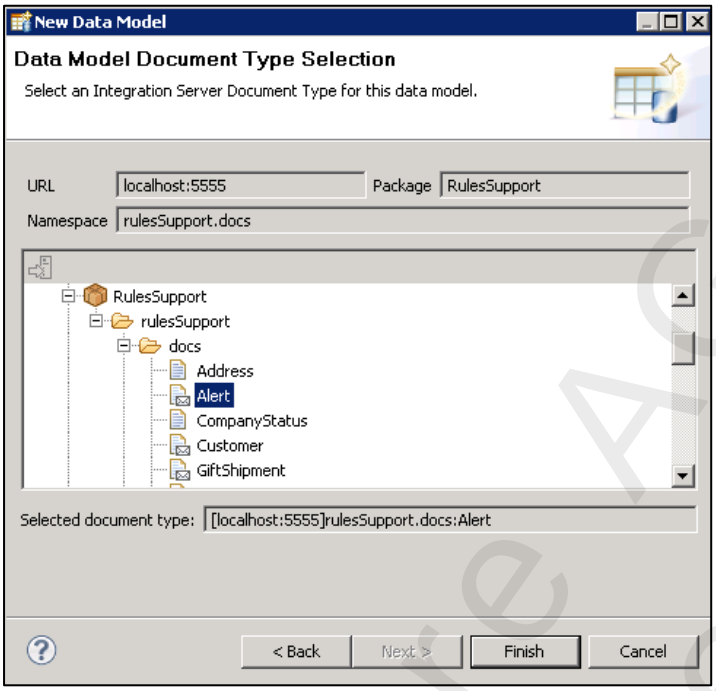


5. Add a Data Model:

   a. Create a Data Model named **Alert** to your rule project. Provide a description text as shown on the screen shot below and click **Next**:



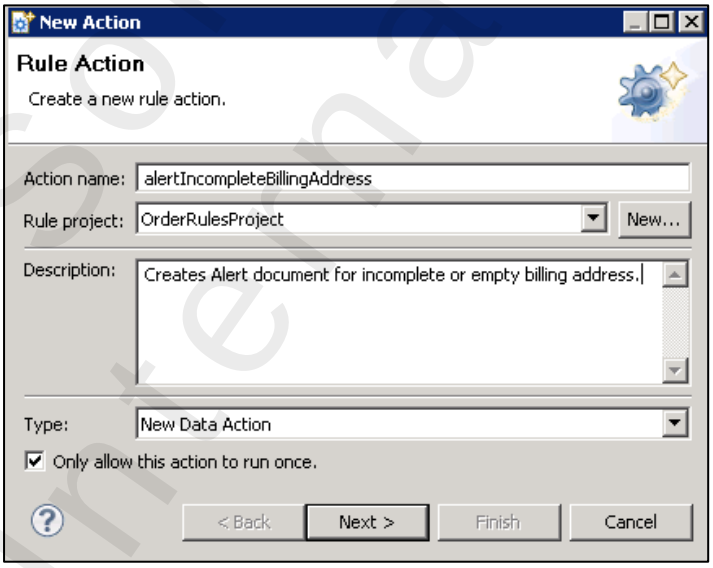
b. On the next panel select the existing IS document **rulesSupport.docs:Alert** of IS package **RulesSupport** as related document type for your Data Model and click **Finish** to complete your Data Model.



c. Because the Alert document also contains e.g. **pub.publish_envelope.datamodel**, click **Select All** on the appearing pop up to allow overwriting of existing Data Models in your project. Click **OK** to overwrite.

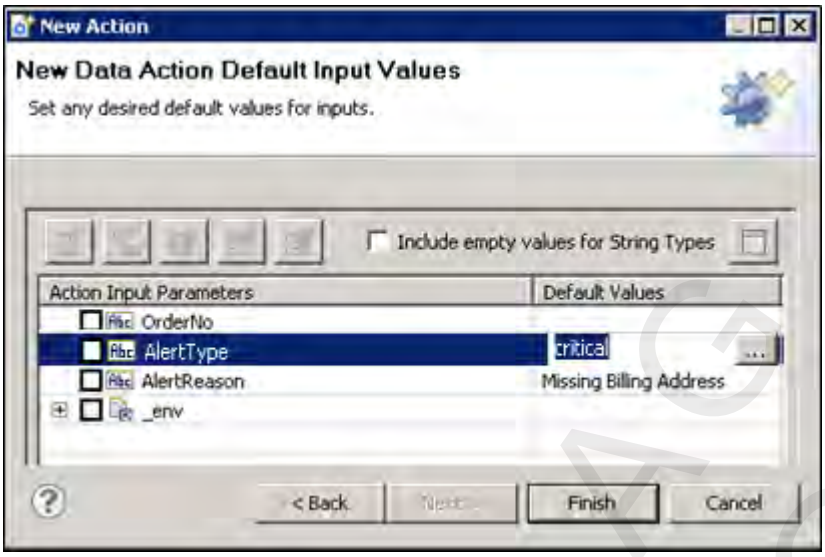
6. Add an Action to your rule project:

a. Add an **Action** of type **New Data Action** named **alertIncompleteBillingAddress** to your **OrderRulesProject** and provide a description text as shown on the screen shot below. Check your action to **run once** only and click **Next**.



b. On the next panel, select Data Model **Alert** as data model for which an instance should be created as action result. Click **Next**.

c. Specify the following default values for your alert data:

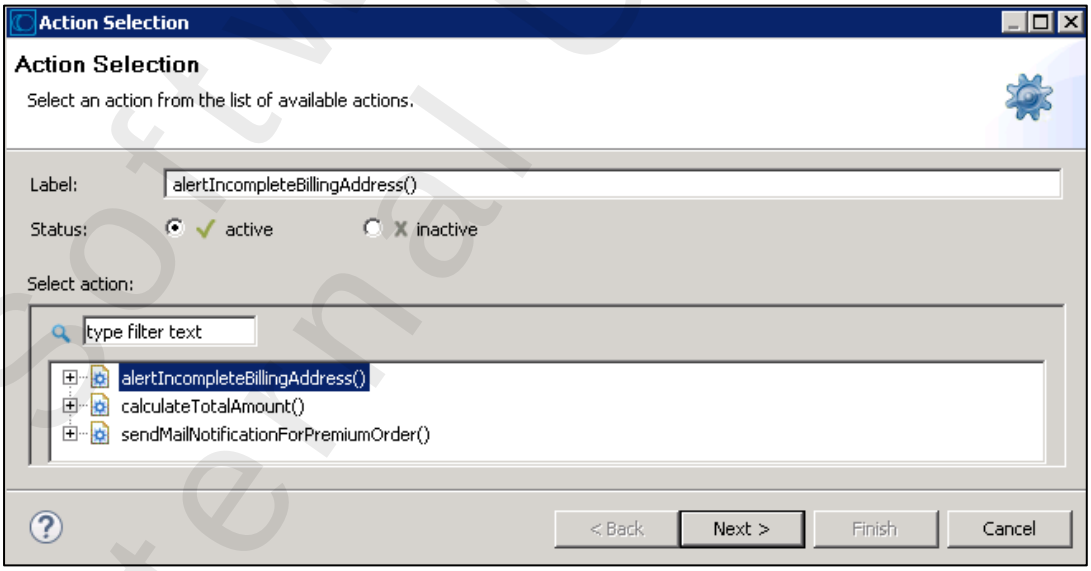


d. Click **Finish** to complete your New Data Action.

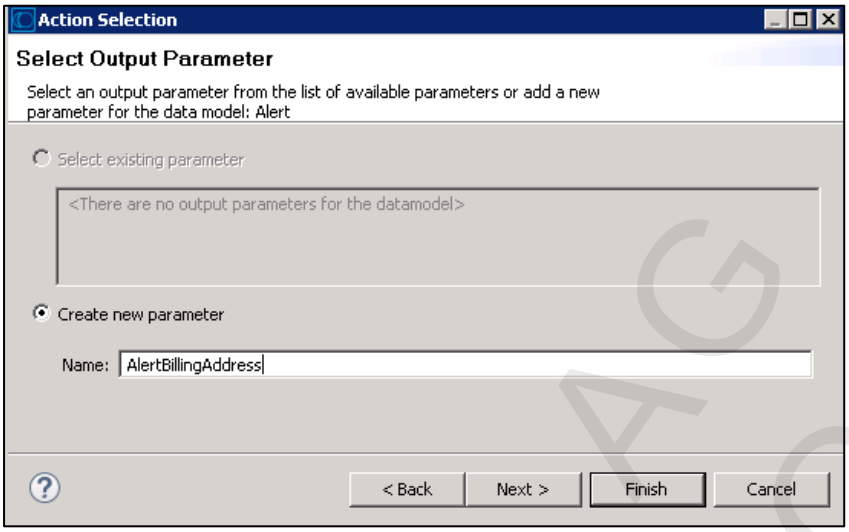
7. Extend your existing Decision Table **DetectIncompleteBillingAddress** opened in the Decision Table Editor pane:

   a. To add another result column of type Action, drag the New Data Action **alertIncompleteBillingAddress** from the Rules Explorer view behind the last column named **Rating** of Decision Table **DetectIncompleteBillingAddress**.

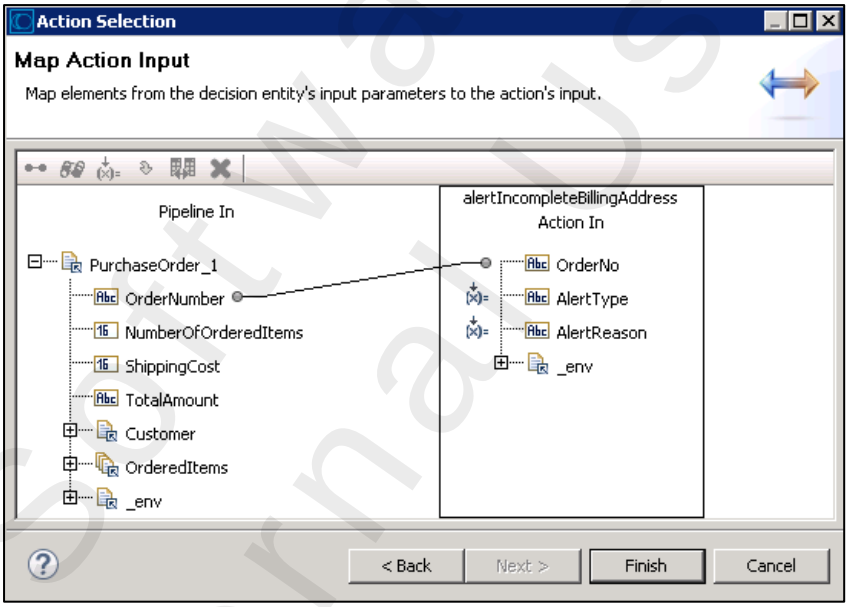
   b. Confirm Action **alertIncompleteBillingAddress** and default status active. Click **Next**.



c. Because your Decision Table does not contain any Alert as (output) parameter yet, you get prompted to add a new parameter of Data Model type **Alert** now. Enter **AlertBillingAddress** as parameter name and click **Next**.

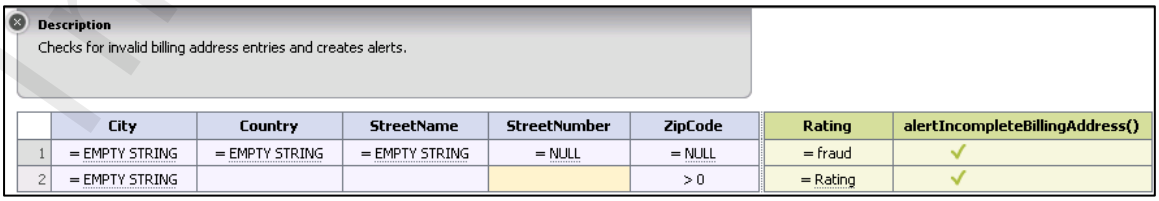


d. Map **OrderNumber** of parameter **PurchaseOrder_1** to the **OrderNo** input element of the **alertIncompleteBillingAddress** action. Highlight action input parameter AlertType and click to set its value to **critical**. In the same way assign the text string **Empty billing address** as value to action input parameter **AlertReason**. In both cases allow to overwrite pipeline values.



Hint: These mapping and assignments will be used as defaults for all rows of your Decision Table.

e. Click **Finish**. Your Decision Table should now look like this:



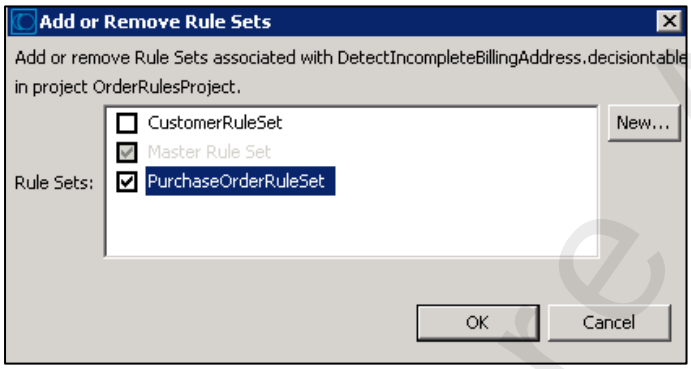
	City	Country	StreetName	StreetNumber	ZipCode	Rating	alertIncompleteBillingAddress()
1	= EMPTY STRING	= EMPTY STRING	= EMPTY STRING	= NULL	= NULL	= fraud	✓
2	= EMPTY STRING				> 0	= Rating	✓

f. For the second rule, click into action result cell of column **alertIncompleteBillingAddress()** and click the green pencil icon to modify the input value assignments of your action. On the Action Input Mapping panel, modify the assigned input value of **AlertType** to **medium** and set assigned value of **AlertReason** to the text string **Missing city in billing address**. Hit **OK**.

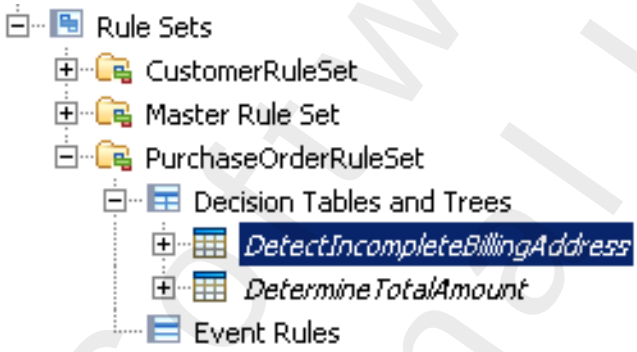
g. Suppress all warnings of the second rule and ensure the result action **alertIncompleteBillingAddres()** is still enabled for both rules.

h. Save Decision Table **DetectIncompleteBillingAddress**.

8. In the Rules Explorer view, right-click Decision Table **DetectIncompleteBillingAddress,** select **Rule Sets…** to add the Decision Table to the existing Rule Set **PurchaseOrderRuleSet**.

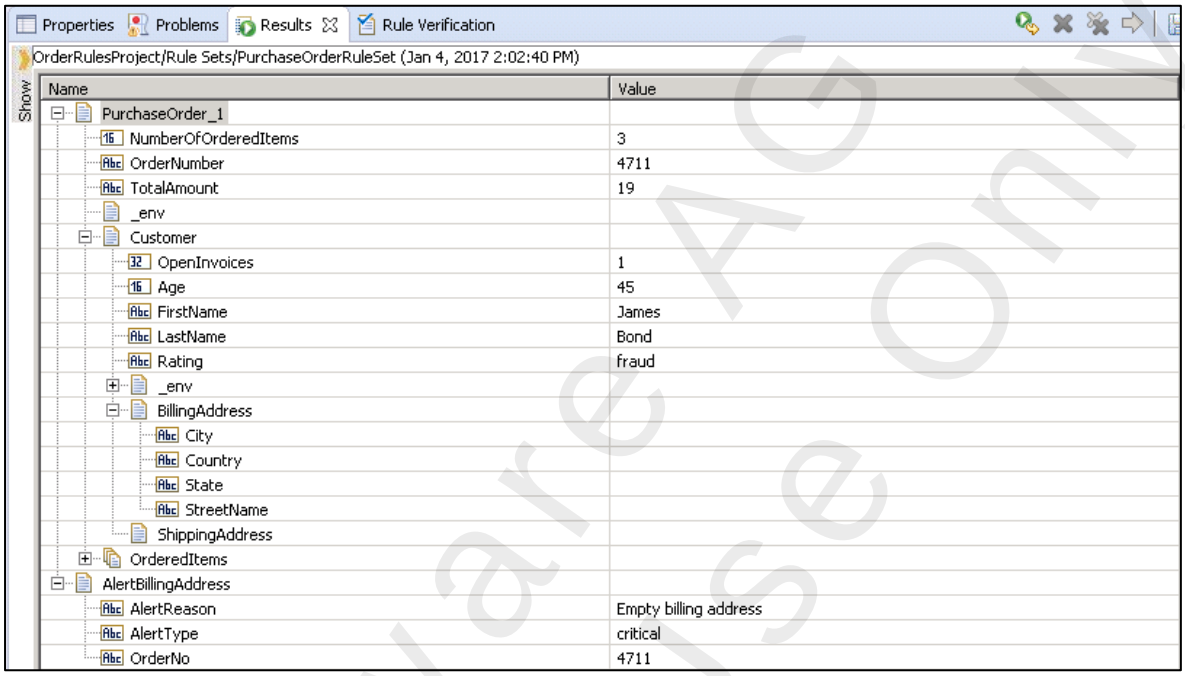


9. Because the Processing mode of Rule Set **PurchaseOrderRuleSet** is sequential, ensure that Decision Table **DetectIncompleteBillingAddress** will be executed first.



Hint: To adjust the sequence, you can right-click a Decision Entity under a Rule Set and use the **Move Up/Down** commands.

10. Run the Rule Set **PurchaseOrderRuleSet**. For testing you can load the provided sample input **<*workshop-dir*>\Exercise7\Resources\PurchaseOrderRuleSetInput.xml**.
    Important: Don't forget to check the box beside **Include empty values for String Types**.
    Depending on the input values of **City**, **Country**, **StreetName**, **StreetNumber**, **ZipCode** (all of Customer's **BillingAddress**), you should see a new result parameter named **AlertBillingAddress** with filled **AlertReason**, **AlertType** and **OrderNo** values.
    Moreover, **Rating** could have changed to **fraud**.
    Also ensure, **TotalAmount** has been calculated by the second Decision Table in the Rule Set.



Note: We will use the "thrown" **AlertBillingAddress** parameter of type **Alert** later in another Decision Entity to make some "intelligent" decisions.

This page is intentionally left blank.

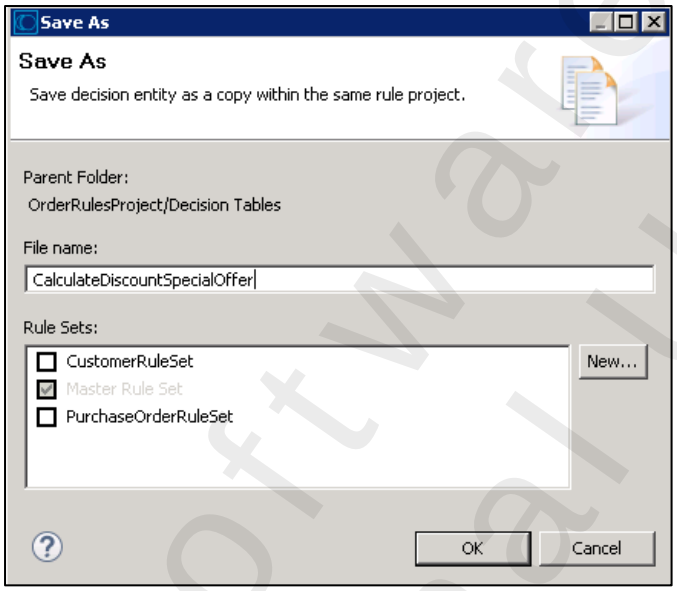
Exercise 08:
Functions and Expressions

Objectives

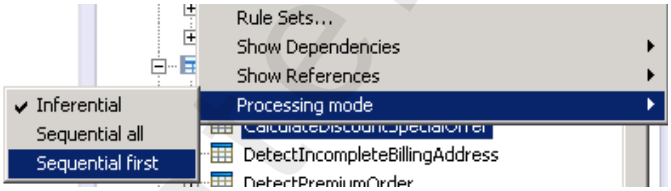
Sales department likes to offer a special discount rate for customers of a particular region in Europe regardless of their status and rating. In this exercise you will created an enhanced Decision Table based on the existing CalculateDiscount Decision Table that returns the special discount rate for a selected customer subset.

Steps

1. Launch **Software AG Designer** and switch to the **Rules Development** perspective.
2. Use the Rules Explorer view to open the existing Decision Table **CalculateDiscount**.
3. Save Decision Table **CalculateDiscount** as a new Decision Table **CalculateDiscountSpecialOffer** within the same Rules project **OrderRulesProject**. Do NOT add this Decision Table to any custom Rule Set.



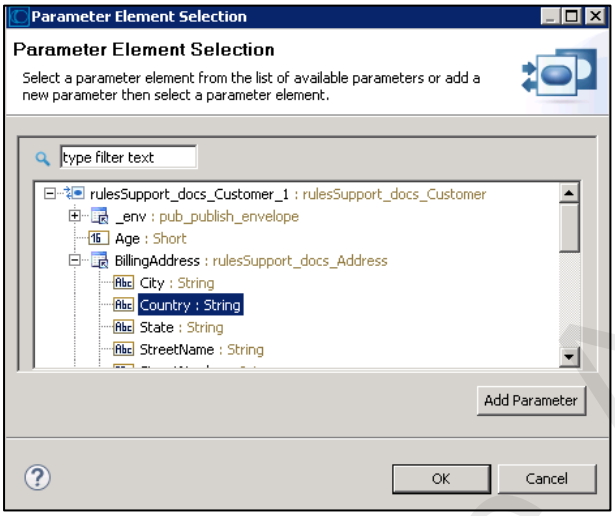
4. Right-click Decision Table **CalculateDiscountSpecialOffer** in the Rules Explorer view and change its Processing mode to **Sequential first.**



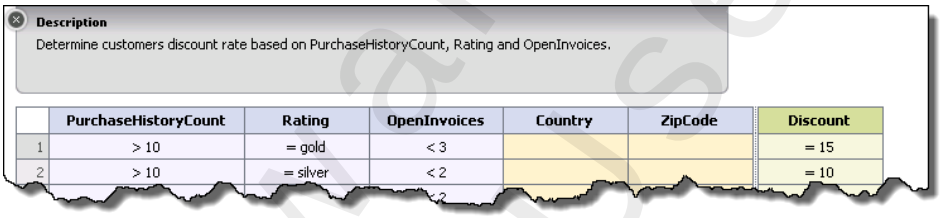
Hint: You can also change the Processing mode in the Properties view of a Decision Entity.

5. Modify the Decision Table like this:

   a. In the Decision Table Editor pane, use the Palette icon to insert a condition column behind column **OpenInvoices**. Select parameter **Country** of **BillingAddress** to be added.



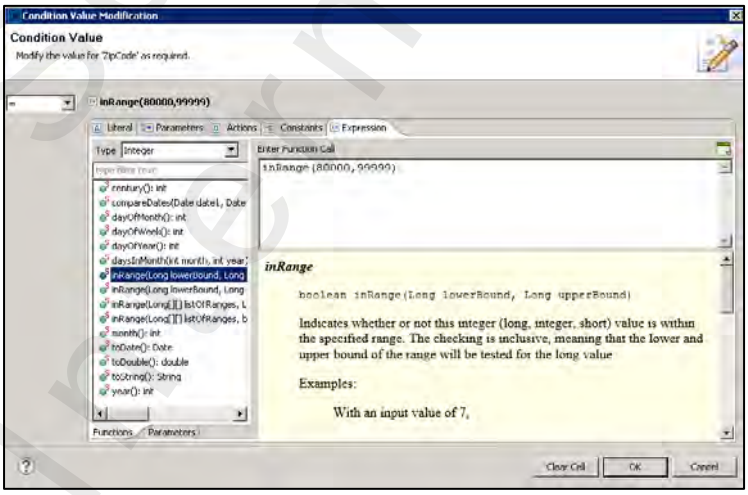
   b. In the same way, insert another condition column behind column **Country**. Select parameter **ZipCode** of **BillingAddress** to be added.



	PurchaseHistoryCount	Rating	OpenInvoices	Country	ZipCode	Discount
1	> 10	= gold	< 3			= 15
2	> 10	= silver	< 2			= 10

   c. In the Decision Table Editor, use the Palette icon to insert two empty rules at the very beginning of the entire Decision Table.

   d. Configure the first new rule to return a discount rate of **20** for customers with a country code of **DE** and a ZIP code in the range of **80000**…**99999**. To do so, use the extended cell editor, open tab **Expressions** and use the Integer Function **inRange** for that purpose.



	PurchaseHistoryCount	Rating	OpenInvoices	Country	ZipCode	Discount
1				= DE	= inRange(80000,99999)	= 20

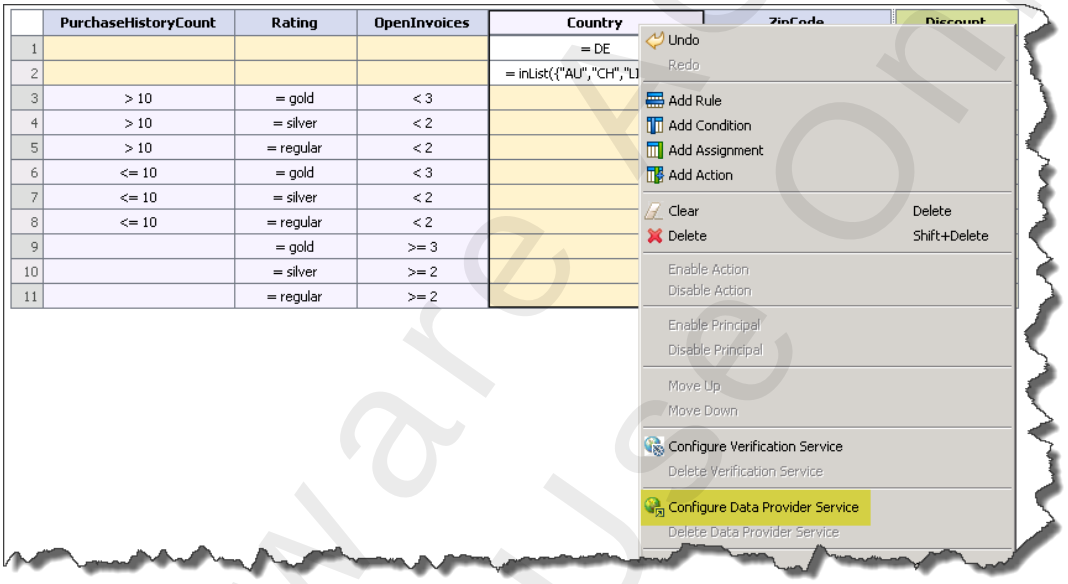
e. Configure the second new rule to return a discount rate of **23** for customers with a 5-digit ZIP code greater than **10000** and country code of **AU**, **CH**, **LI**, or **LU**. In the extended cell editor, use Integer Function **inRange** and String Function **inList** for that purpose.

	PurchaseHistoryCount	Rating	OpenInvoices	Country	ZipCode	Discount
1				= DE	= inRange(80000,99999)	= 20
2				= inList({"AU","CH","LI","LU"})	= inRange(10000,99999)	= 23

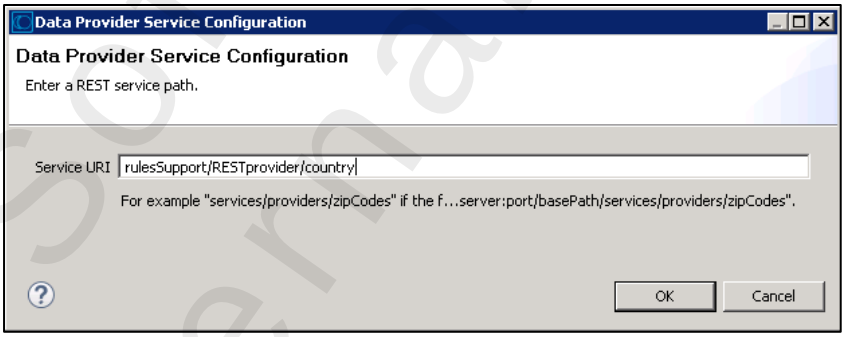
f. Save your Decision Table.

6. To offer better comfort to Business Users working on the Decision Table in the web-based RMC, configure a Data Provider Service that will return valid country codes for a selected country:

a. In Designer, right-click the column header of condition **Country** and select **Configure Data Provider Service**.



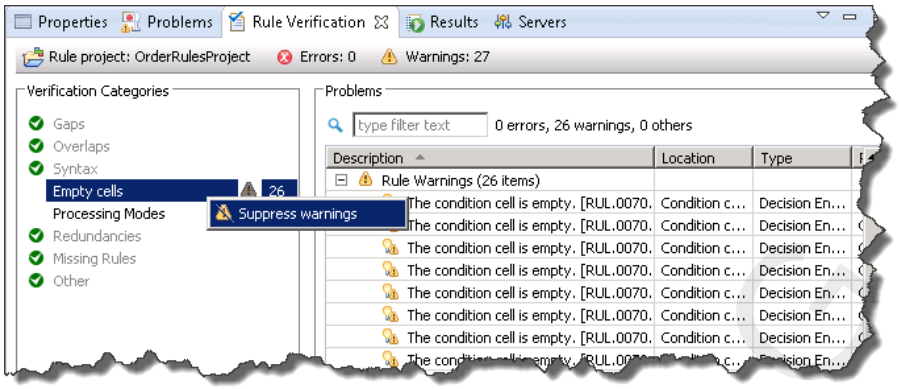
b. Your IS package RulesSupport already contains a ready-made REST resource that returns valid country codes by a _get method. Specify **rulesSupport/RESTprovider/country** as the trailing REST service path and click **OK**.



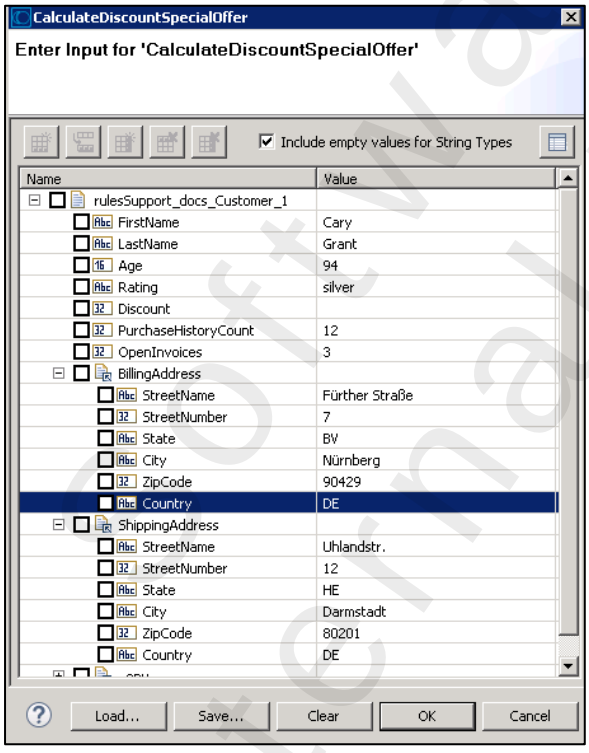
Note: The leading portion of the Data Provider service URL has to be configured on the My webMethods Business Rules Settings page (see next exercise).
Service URI must be formed up using “/”. Do NOT use “\”.

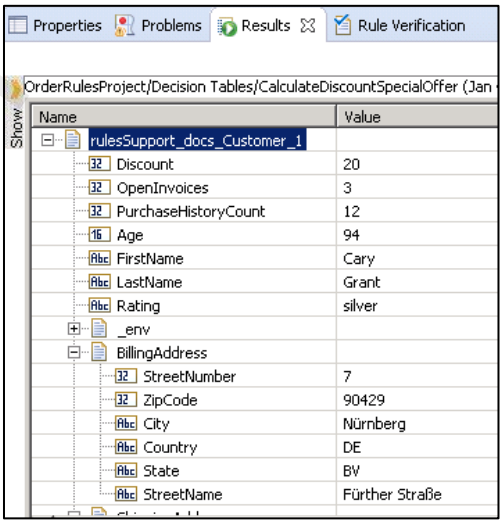
c. Save your Decision Table.

7. To get rid of all warning for Decision Table CalculateDiscountSpecialOffer, suppress warnings for **Empty cells** and **Syntax** in the Rule Verification view.



8. Finally right-click Decision Table **CalculateDiscountSpecialOffer** in the Rules Explorer and select **Run As > Run Decision Table** to test it. On the appearing panel, specify at least the necessary input values for **PurchaseHistoryCount**, **Rating**, **OpenInvoices**, **Country** (of **Billing Address**), **ZipCode** (of **BillingAddress**). Instead of typing you can load input data from the provided sample input file ***<workshop-dir>*\Exercise8\Resources\CalculateDiscountSpecialOfferDecisionTableInput.xml**. In any case check the box beside **Include empty values for String Types**. Click **OK** to run the Decision Table. Inspect the results in the appearing **Result** view. Repeat testing with different input values to verify that your Decision Table works as expected.





9. *For extra credit:* What would happen if you change the Decision Table's Processing mode to **Sequential all** or **Inferential**?

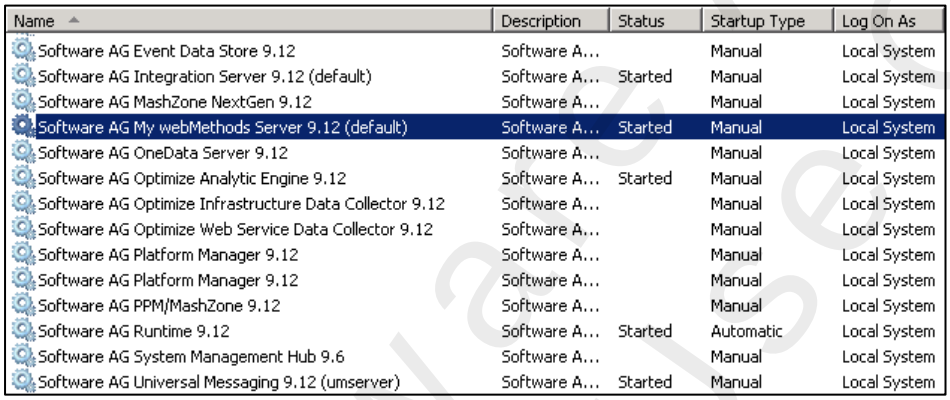
Exercise 09:
Deployment to IS and MWS

Objectives

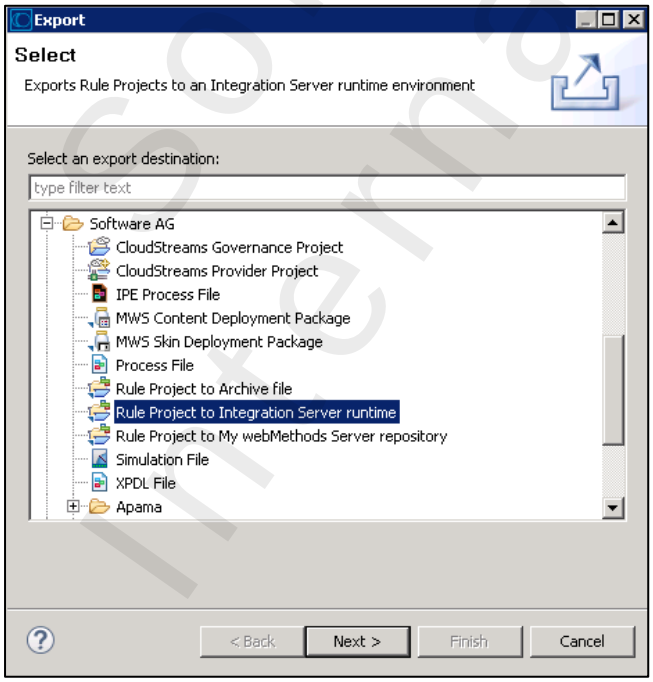
In this exercise, you will deploy your rules project to your Integration Server first.
Then, by using a configured MWS Repository connection, you will deploy your rule project to the Rules Management Console (RMC) in My webMethods. This requires the configuration of several MWS security settings in advance.

Steps

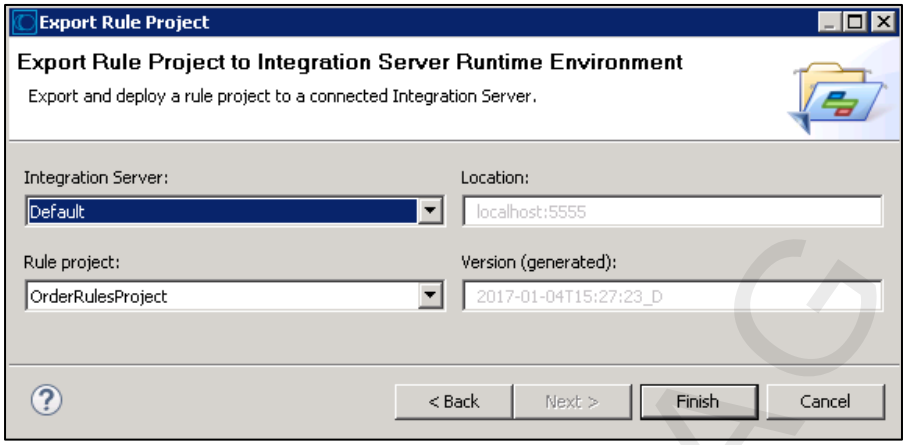
1. Start your **Universal Messaging** realm as a Windows service, if not active.
2. Start your **Integration Server** as a Windows service, if not active.
3. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.



4. Launch **Software AG Designer** and switch to the **Rules Development** perspective.
5. To deploy your entire Rules Project to your IS as runtime environment, right-click **OrderRulesProject** in the Solutions view and choose **Export…** . On the appearing panel select export type **Software AG > Rule Project to Integration Server runtime**.



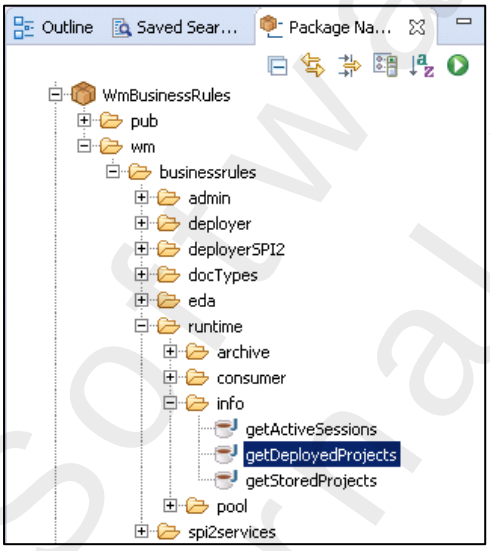
On the next panel confirm the rule project **OrderRulesProject** as source and your local Integration Server named **Default** as target.



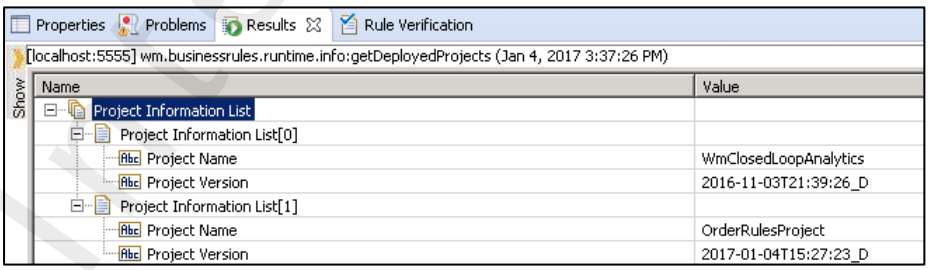
Click **Finish** to perform the IS deployment.

6. *Optionally*: Double-check the successful deployment to your Integration Server:

    a. In Designer, use the Package Navigator view and execute (**Run As > Run Service**) the (non-public) IS service **wm.businessrules.runtime.info:getDeployedProjects** contained in the IS package **WmBusinessRules**. The service does not require any inputs.

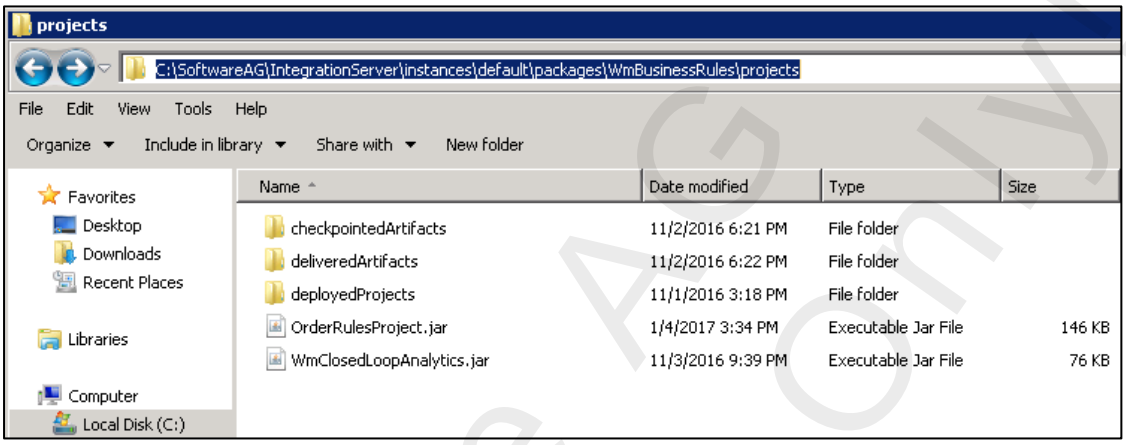


View the results in the **Results** view:



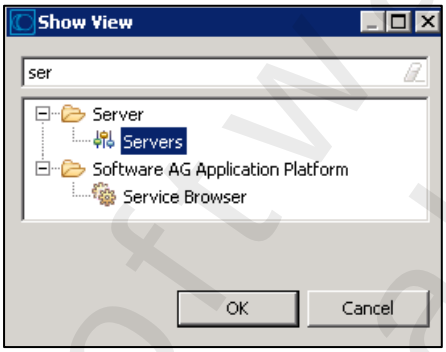
Hint: Rule project WmClosedLoopAnalytics is a deployed system Business Rules project that comes with the installed BPM Closed Loop Analytic feature.

b. As an alternative, open the Windows Explorer and navigate to:
C:\SoftwareAG\IntegrationServer\instances\default\packages\WmBusinessRules\projects
Your rule project should exist here as a .jar file:

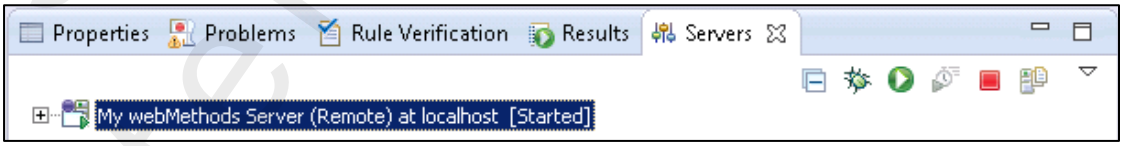


7. Prepare your local MWS for later User Task deployment:

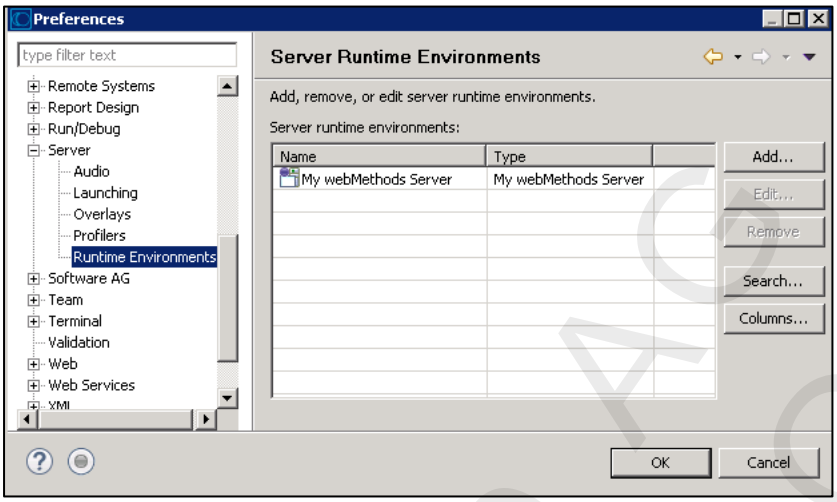
a. In Designer, open the **Servers** view.
If the Servers view is not displayed in the Rules Development perspective, add this view.
To do so, select **Window** > **Show View** > **Other...** . Filter for **ser** and select the **Servers** view to be added.



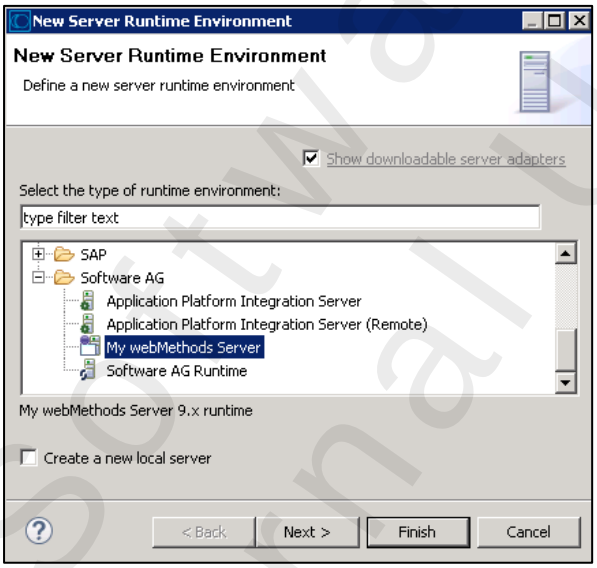
b. Move the **Servers** view to the lower area within Designer next to the **Results** view. In the Servers view, check the availability of your local MWS displayed as **My webMethods Server (Remote) at localhost**.



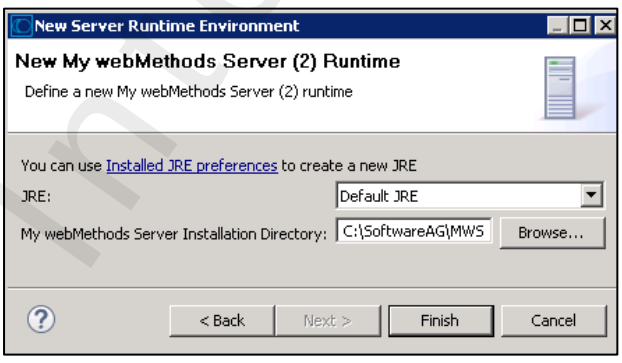
c. Only do this step if the MWS server is not available in the Servers view. Otherwise continue with the next step.
To add the MWS to the Servers view , select **Window** > **Preferences**, then select **Server** > **Runtime Environments**.



If the MWS is missing here too, click on **Add** to add it. Otherwise click **Cancel** and continue with the next step.
In case of **Add**, select **Software AG > My webMethods Server** and click **Next**.

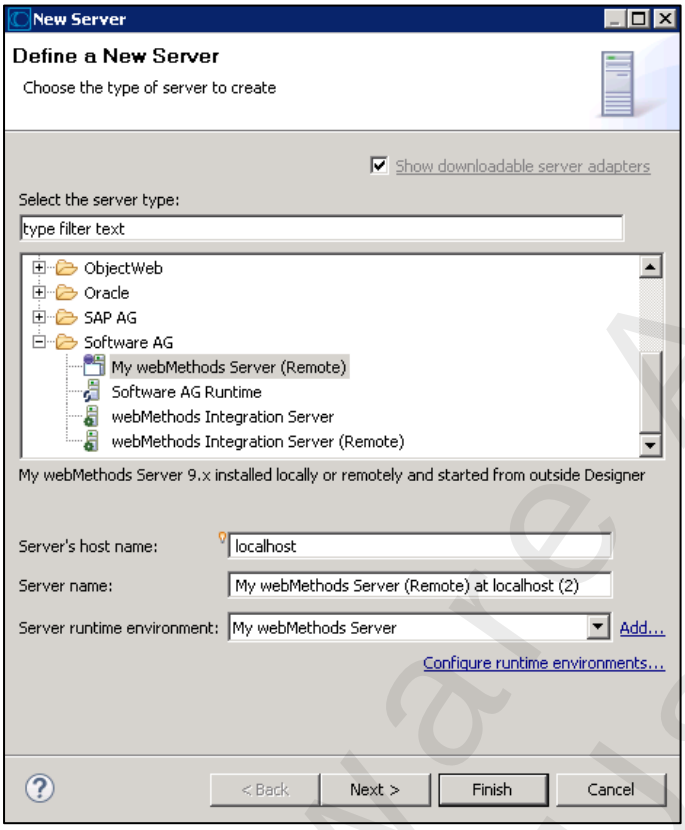


On the next panel browse for the MWS root folder and click on **Finish**.

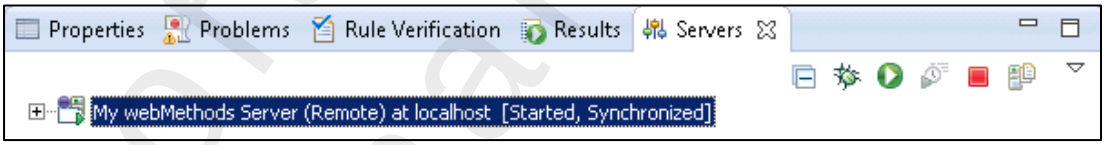


Restart Designer.

Now go back to the **Servers** view. If your MWS is not shown up here, right-click in the whitespace. Select **New** > **Server**. On the appearing panel select **My webMethods Server (Remote)** and click **Next**.

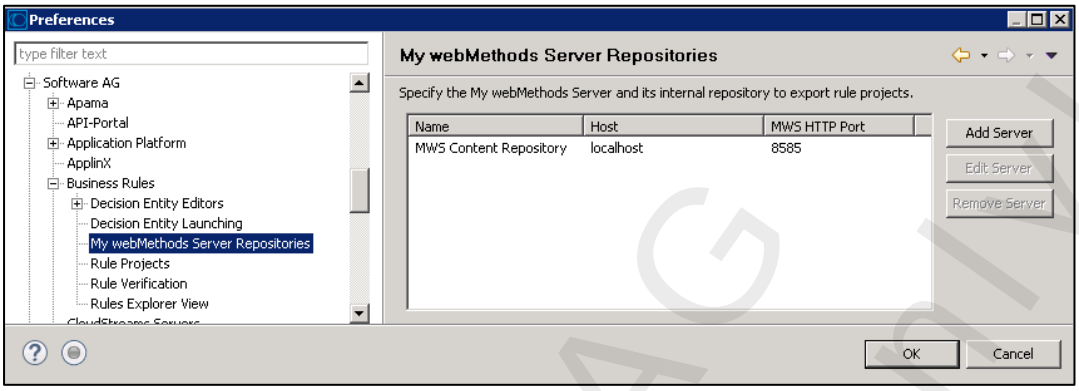


On the last panel confirm all connection parameters and click **Finish**.
The Servers view should now show your MWS. Click on to get connected.

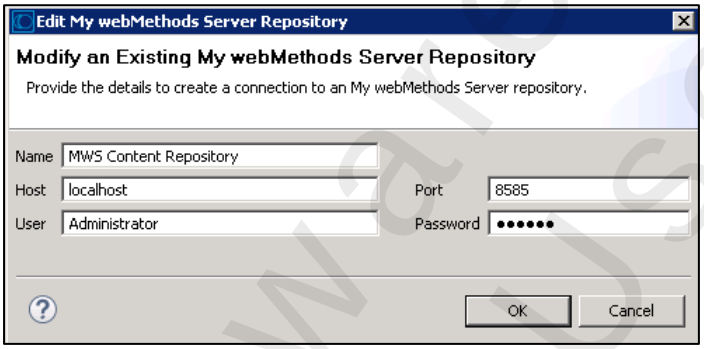


8. Check/configure a My webMethods Server repository connection in Designer:

   a. Click **Window** > **Preferences** > **Software AG** > **Business Rules** > **My webMethods Server Repositories** in the menu bar.



   b. In case a Server Repository connection is missing here, click **Add Server**, otherwise select the existing one and click **Edit Server**. On the appearing panel provide the following settings or double-check the existing ones (password is **manage**). Finally click **OK.**



Note: The default repository port is **10999**. The default port can be changed on the My webMethods Server by a system administrator.

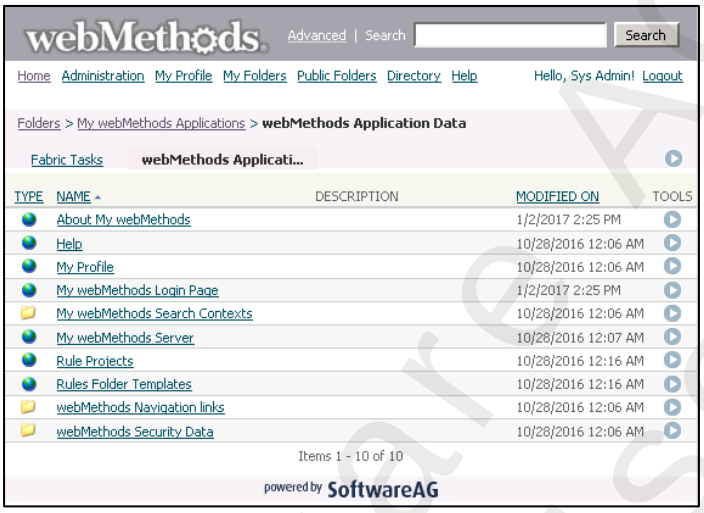
9. Before you can deploy a rule project into the RMC in My webMethods, a **Rule Projects** folder must be created in the repository and you must have full permissions (write/delete) for this Rule Projects folder in MWS.

> *Note*: For your convenience, the **Rule Projects** folder has already been created in MWS on your VM.

The following steps grant full write/delete permissions for the Rule Projects folder to the My webMethods Administrator role in MWS:

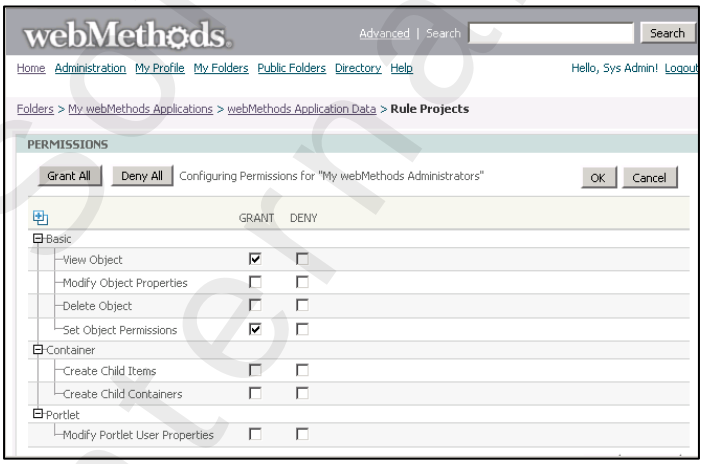
a. Open a browser tab and connect to My webMethods using the URL http://localhost:8585. Login as **Sysadmin | manage**.

b. Navigate to **Folders** > **My webMethods Applications** > **webMethods Application Data**.



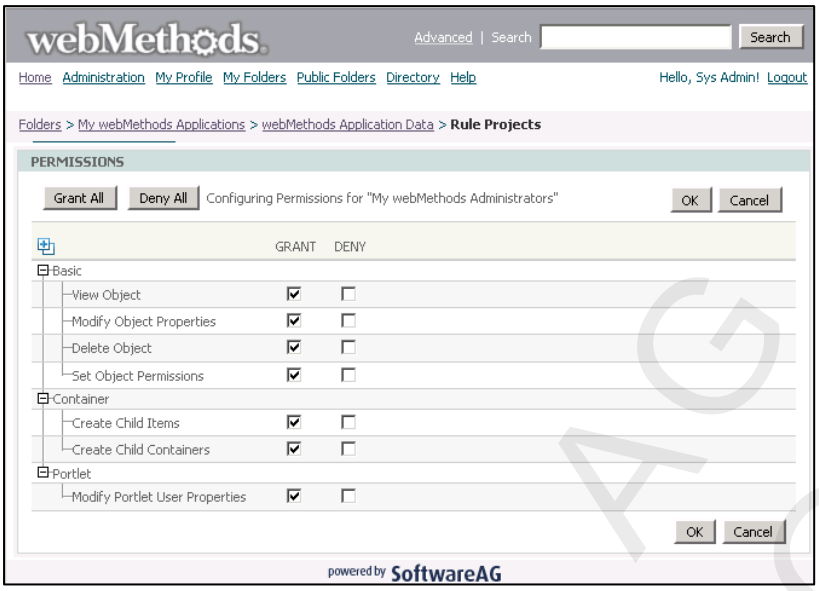
c. In the **TOOLS** column, click for folder **Rule Projects** and select **Permissions**. On the appearing view, ensure permissions are displayed as **Granted All** for Role (Principal) **My webMethods Administrators**.

d. Perform this step only in the case the permissions are displayed as Custom. Otherwise continue with the next step.
Click the **Custom** link. On the appearing Permissions view, click **Grant All**:



Confirm your changes by clicking **OK**.

The Permissions should now look like this:



Click **OK** again.
On the Edit Permissions view, role **My webMethods Administrators** should now have permissions titled **Granted All**. Click **Apply**.

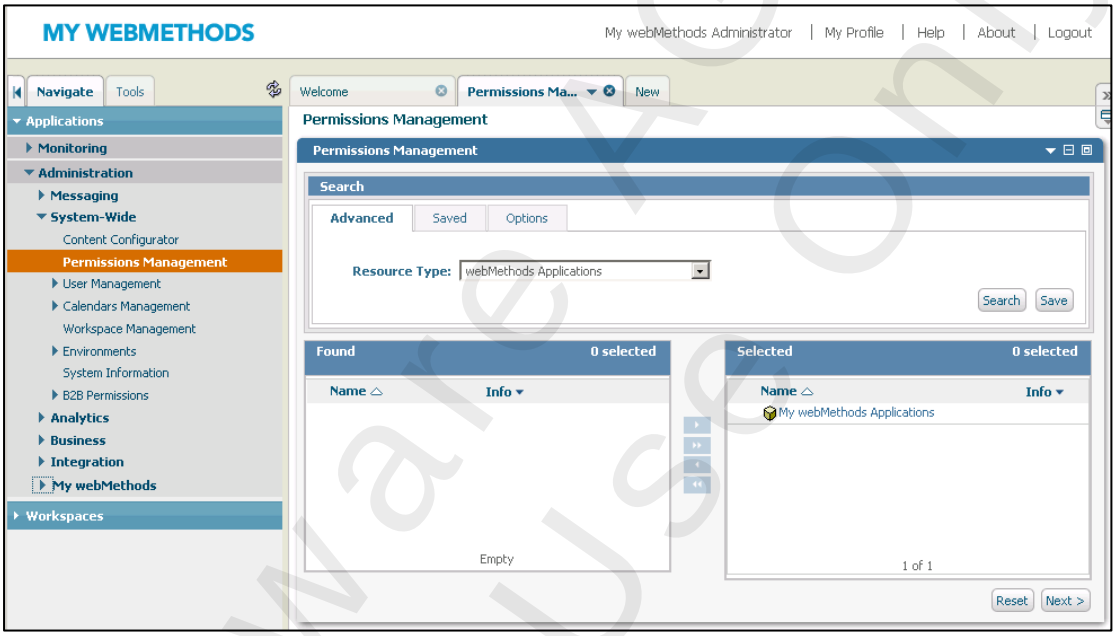
e. Logout from the MWS Administration Console.

Note: Usually the MWS administrator (Sysadmin) must provide/create a different role (not My webMethods Administrators) for Business Rules users and assign all access rights to this role for the Rule Projects folder. Then the MWS Server administrator (Sysadmin) must add all MWS user who will be accessing the rule projects to this role.

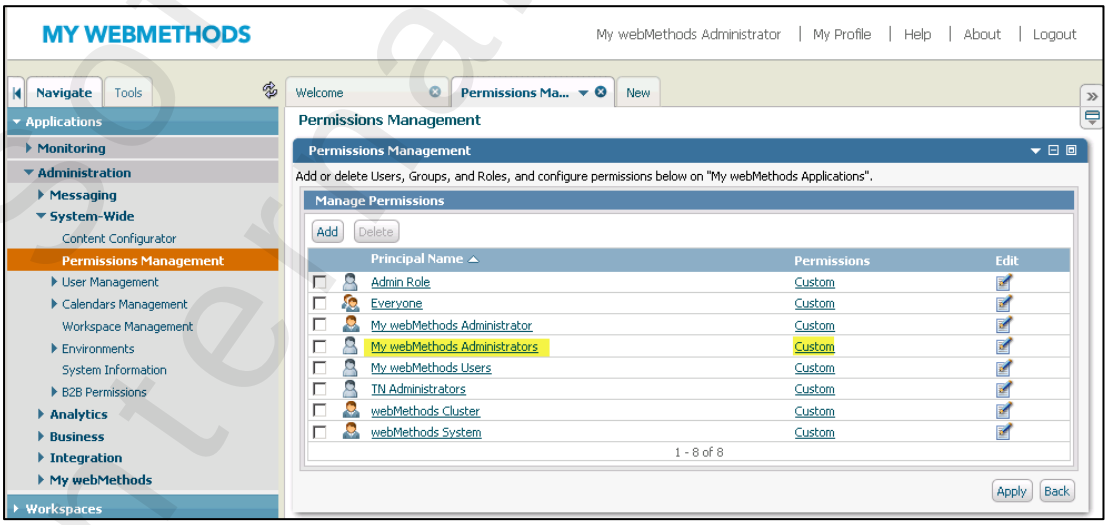
10. Additionally you have to ensure that a Business Rule user will see the newly exported rule project in My webMethods. To do this, you have to provide access permissions in My webMethods to see the appropriate Business Rule portlets and pages. In the exercises, we will act as MWS user Administrator (who belongs to the role My webMethods Administrators) who has full access permissions in My webMethods. To double-check this, perform the following steps:

    a. Use a browser tab to connect to My webMethods using the http:/localhost:8585. Login as **Administrator | manage**.

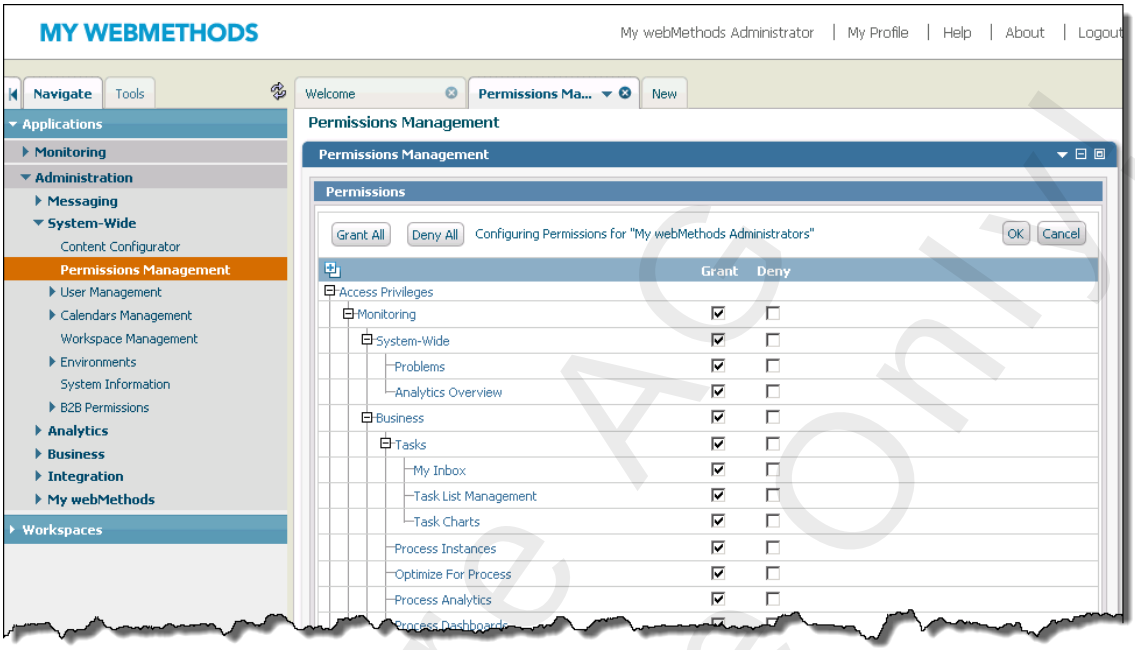
    b. Navigate to **Applications** > **Administration** > **System-Wide** > **Permissions Management**. In the Permissions Management view, select Resource Type **webMethods Applications**, and click **Next**:



    c. On the next panel, click on **Granted All** or **Custom** for Principal Name **My webMethods Administrators**:



d. On the Permissions panel ensure that My webMethods Administrators (already) have full access permissions for all webMethods Business Rule assets in My webMethods. If not, click **Grant All** and hit **OK**.

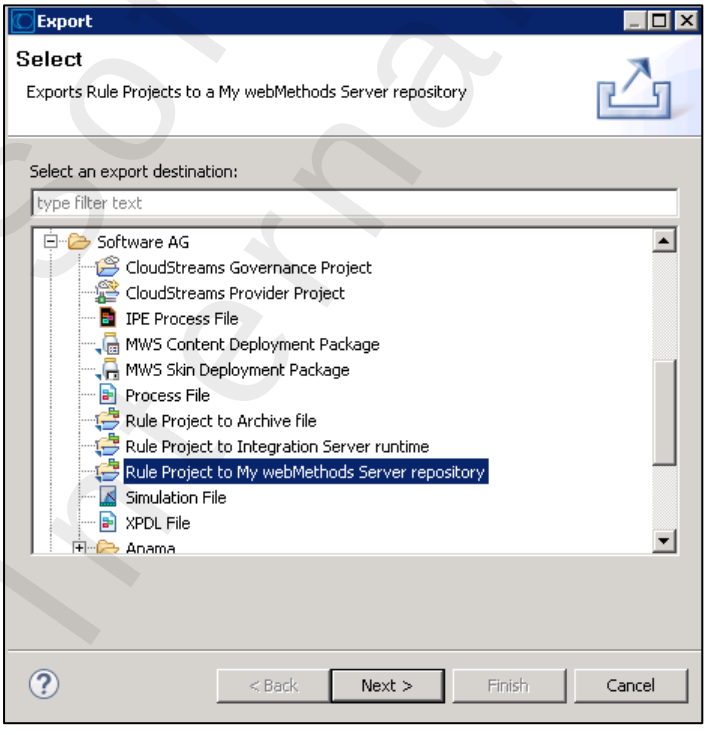


e. Click **OK,** then click **Apply**.

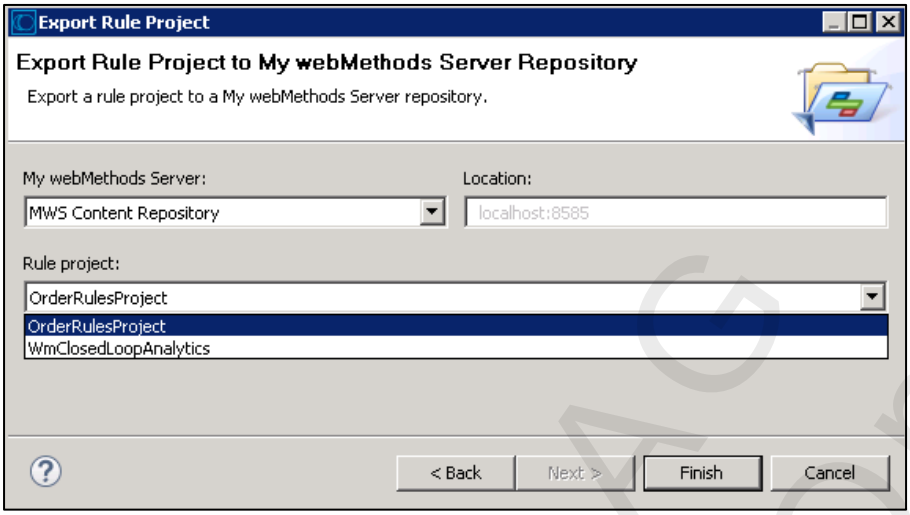
f. Logout from My webMethods.

11. Switch back to Designer and deploy your rule project to MWS repository:

a. To deploy your Rules Project into the RMC in your MWS, right-click **OrderRulesProject** in the Solutions view and choose **Export…** . In the appearing wizard select export type **Software AG > Rule Project to My webMethods Server repository**. Click **Next**.



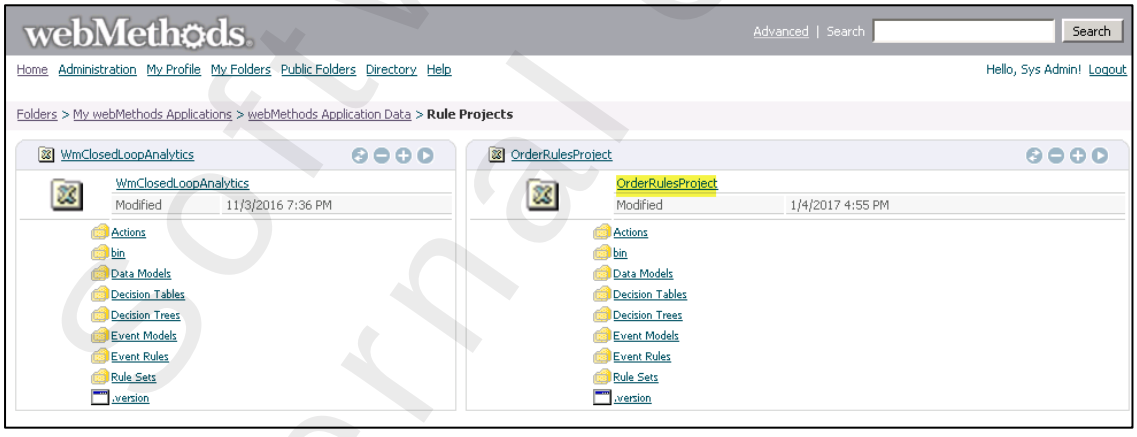
b. On the next panel confirm the rule project **OrderRulesProject** as source and select **MWS Server Repository** as target.



c. Click **Finish** to start the deployment.

Hint: We are using **Administrator | manage** (as defined in the MWS Server Repository connection) to authenticate against MWS.

12. *Optionally*: To double check the MWS deployment, use a browser tab to login to MWS (http://localhost:8585) as user **Sysadmin | manage**. Navigate to **Folders** > **My webMethods Applications** > **webMethods Application Data** > **Rule Projects**. Your **OrderRulesProject** should be stored here:



13. Logout from MWS.

14. Check whether you can see your deployed rule project **OrderRulesProject** in the Rules Management Console (RMC):

    a. Use a browser tab and connect to My webMethods using the URL http://localhost:8585. Login as **Administrator | manage**.

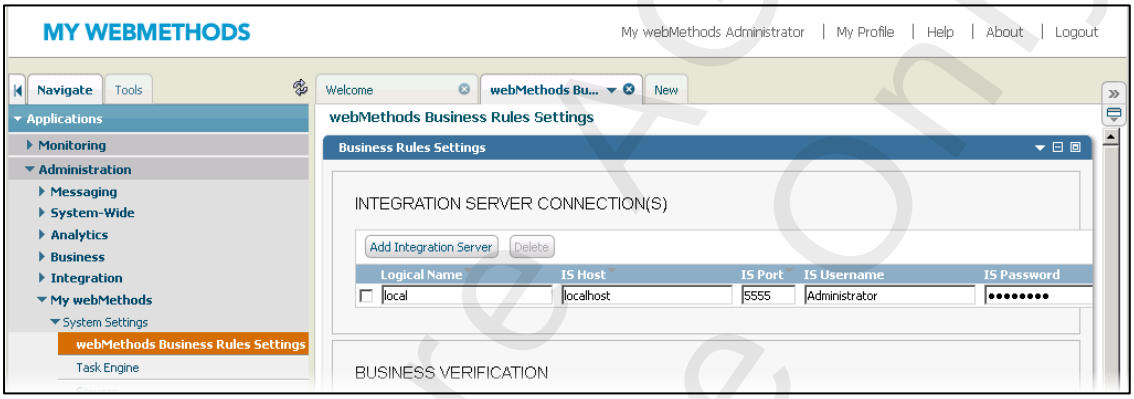
    b. Navigate to **Administration** > **Business** > **webMethods Business Rules.**
    You should see your Business Rules project listed in the Navigation tree.

Hint: In case, the project isn't listed yet, click the Refresh icon to reload the Navigation tree.

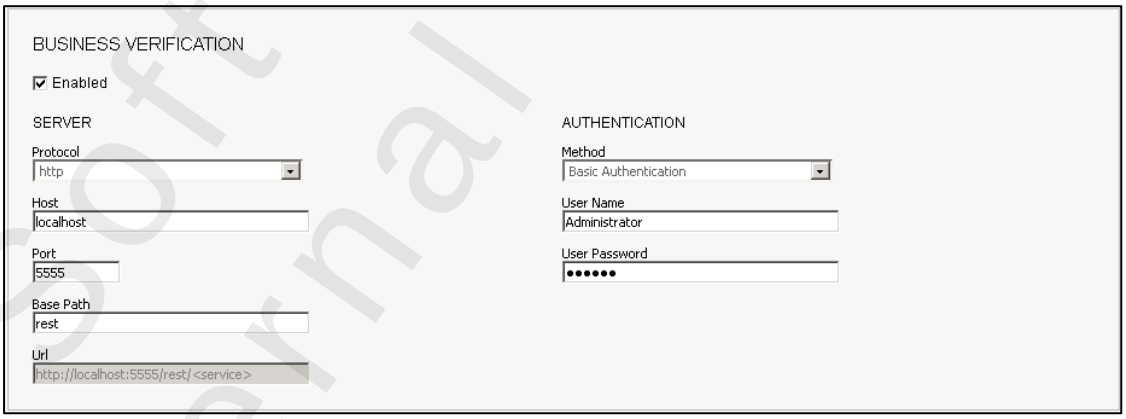
15. A Rules Management Console (RMC) allows a Business User to view, modify and delete assets within a deployed rule project from a web UI (My webMethods). Moreover, a rule project can be hot-deployed from the RMC into one or even multiple IS runtime environments. To allow this, several connection settings need to be pre-configured in My webMethods for webMethods Business Rules:

    a. Navigate to **Administration** > **My webMethods** > **System Settings** > **webMethods Business Rule Settings**.

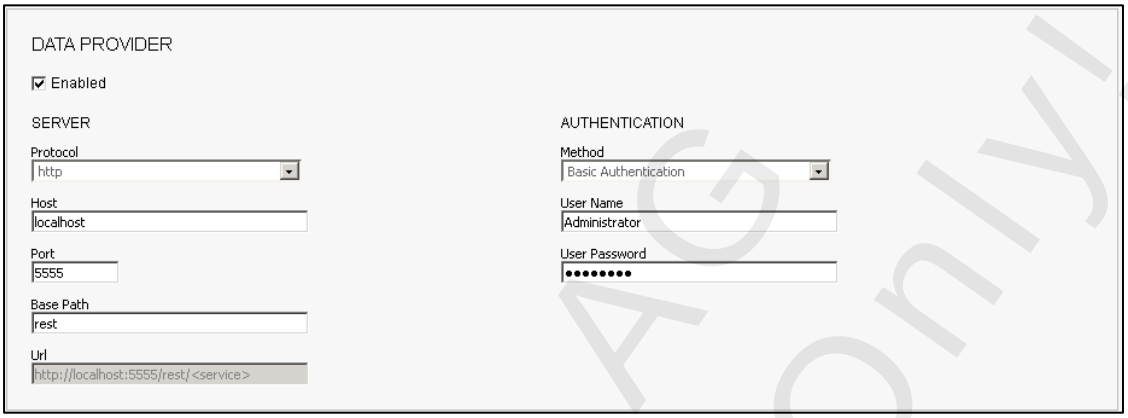
    b. Adjust (only if necessary) the **INTEGRATION SERVER CONNECTION(S)** on the webMethods Business Rules Settings page to allow a later hot-deploy of (modified) rule projects from the RMC to your local Integration Server (**localhost:5555**) with a Logical Name of **local**. IS Username and IS Password are **Administrator** | **manage**.



    c. To allow the RMC to perform Business Verifications assigned in Designer for values in specific Decision Table columns, the base path portion of the implementing REST service URL needs to be configured and enabled. In the **BUSINESS VERIFICATION** section of the webMethods Business Rules Settings page specify the values as shown on the screen shot below:



d. To allow the RMC to provide the data and descriptions in drop-downs for cells in specific Decision Table columns that have a configured Data Provider Service in Designer, the base path portion of the implementing REST service URL needs to be configured and enabled, too. In the **DATA PROVIDER** section of the webMethods Business Rules Settings page specify the values as shown on the screen shot below:



e. Leave the **PRINCIPAL TYPES** section unchanged (all principal types are allowed/ checked).

f. Click **Save** to persist your entries.

This page is intentionally left blank.

Exercise 10:
Using the RMC and Hot Deploy

Objectives

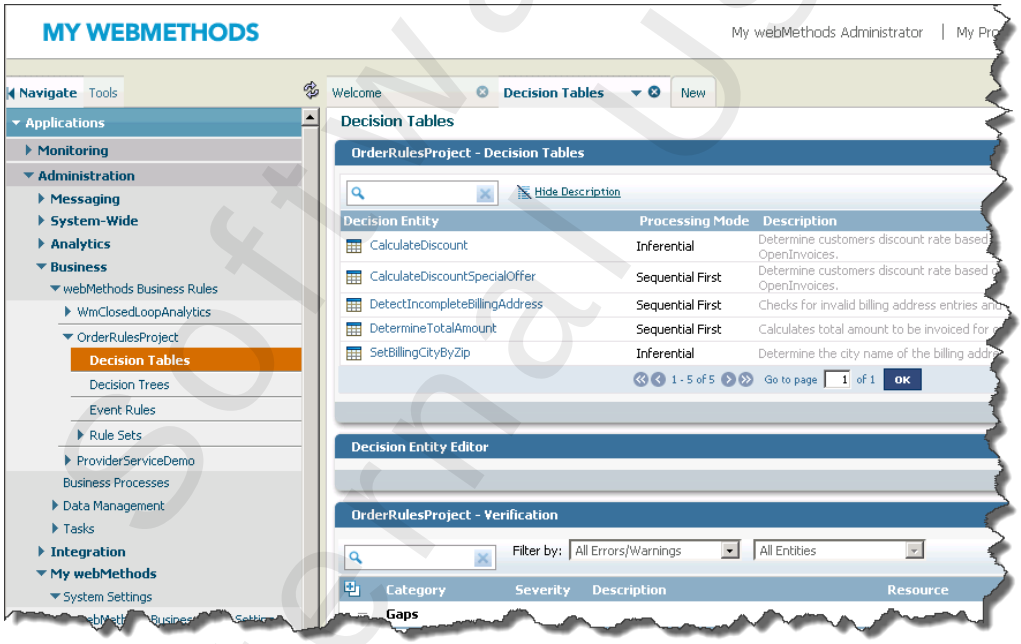
In this exercise, you will use the Rules Management Console (RMC) in My webMethods to modify your deployed rule project as a Business User. You will change a Decision Table and a Decision Tree and see the advantage of a Data Provider Service. Finally you will hot deploy the entire rule project to your Integration Server runtime, and synchronize your rule project with Designer.

Steps

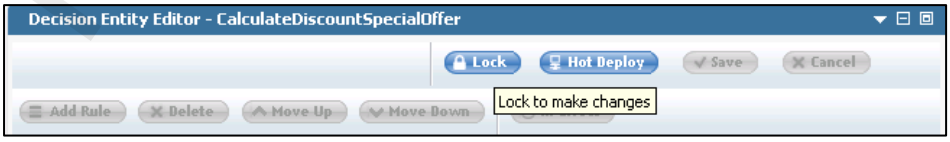
1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. To use the Rules management Console open a browser tab and connect to My webMethods using the URL http://localhost:8585. Login as **Administrator | manage**.

   Note: In a more realistic scenario, a Business User without global administrative permissions would login to My webMethods to work on rules using the RMC.

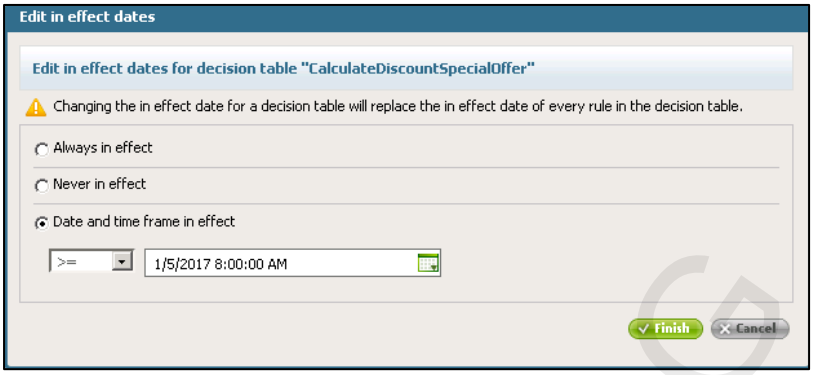
4. In My webMethods, navigate to **Administration** > **Business** > **webMethods Business Rules** > **OrderRulesProject** > **Decision Tables**.



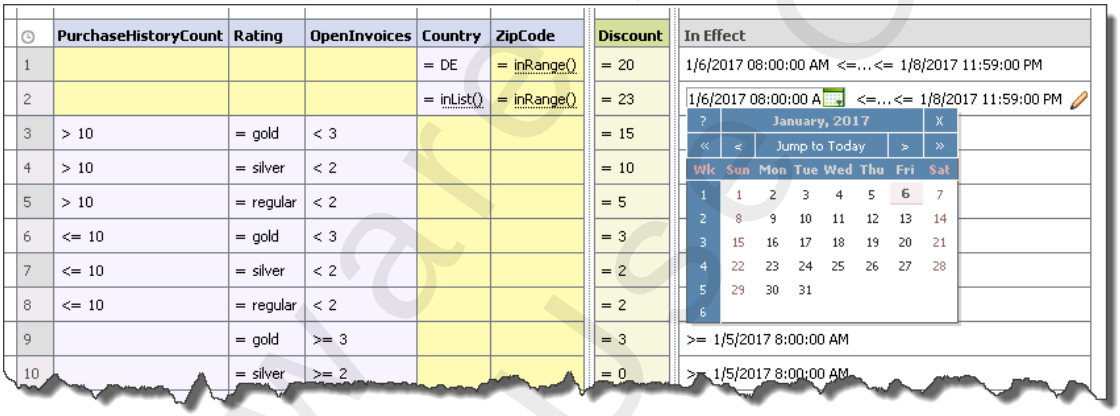
5. Select Decision Table **CalculateDiscountSpecialOffer** to open it in the RMC.
   a. **Lock** the Decision Table to perform changes.



b. Click the button **In Effect**. In the pop-up panel customize all rules to become effective for <***today***> **08:00 AM** and later (**>=**). Click **Finish**.

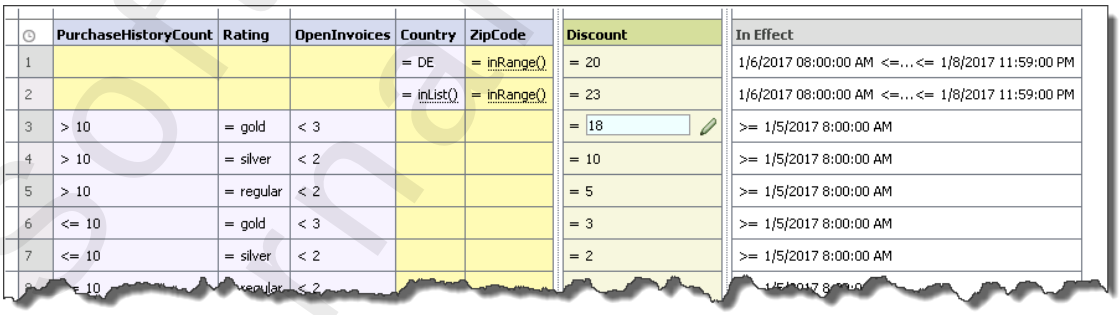


c. Rule one and two implement a “special offer” discount rate calculation. They should be only effective starting from **<*tomorrow*> 08:00 am** for **three days** ending at **11:59 PM**. Adjust the **In Effect** column comparison operator and values for both rules accordingly.



	PurchaseHistoryCount	Rating	OpenInvoices	Country	ZipCode	Discount	In Effect
1				= DE	= inRange()	= 20	1/6/2017 08:00:00 AM <=...<= 1/8/2017 11:59:00 PM
2				= inList()	= inRange()	= 23	1/6/2017 08:00:00 A <=...<= 1/8/2017 11:59:00 PM
3	> 10	= gold	< 3			= 15	
4	> 10	= silver	< 2			= 10	
5	> 10	= regular	< 2			= 5	
6	<= 10	= gold	< 3			= 3	
7	<= 10	= silver	< 2			= 2	
8	<= 10	= regular	< 2			= 2	
9		= gold	>= 3			= 3	>= 1/5/2017 8:00:00 AM
10		= silver	>= 2			= 0	>= 1/5/2017 8:00:00 AM

d. Change the result (**Discount**) of the third rule from 15 to **18**:

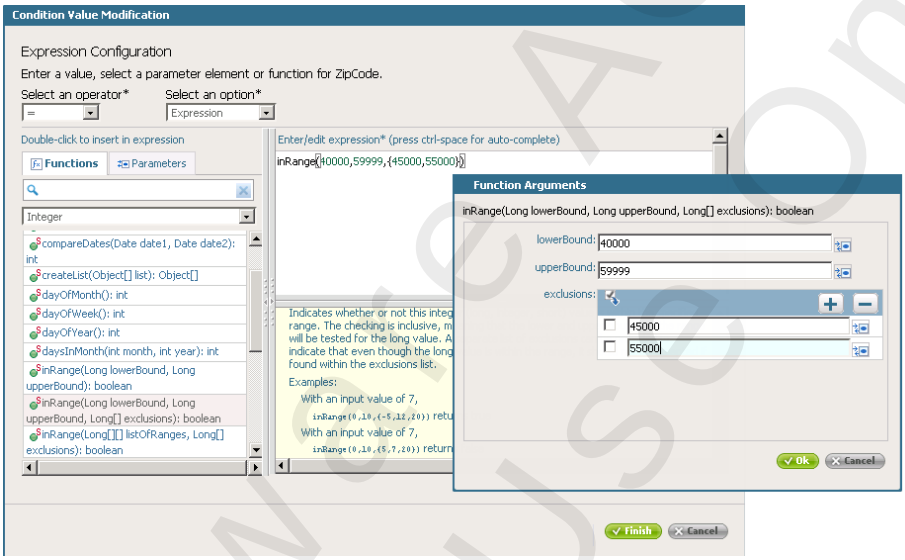


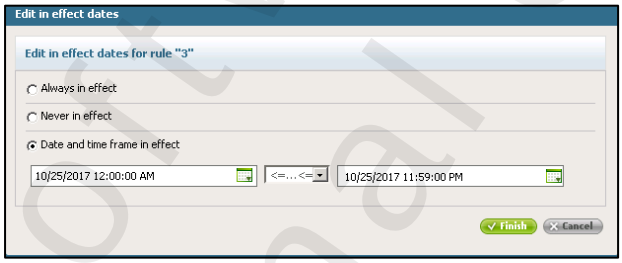
	PurchaseHistoryCount	Rating	OpenInvoices	Country	ZipCode	Discount	In Effect
1				= DE	= inRange()	= 20	1/6/2017 08:00:00 AM <=...<= 1/8/2017 11:59:00 PM
2				= inList()	= inRange()	= 23	1/6/2017 08:00:00 AM <=...<= 1/8/2017 11:59:00 PM
3	> 10	= gold	< 3			= 18	>= 1/5/2017 8:00:00 AM
4	> 10	= silver	< 2			= 10	>= 1/5/2017 8:00:00 AM
5	> 10	= regular	< 2			= 5	>= 1/5/2017 8:00:00 AM
6	<= 10	= gold	< 3			= 3	>= 1/5/2017 8:00:00 AM
7	<= 10	= silver	< 2			= 2	>= 1/5/2017 8:00:00 AM
8	<= 10	= regular	< 2				

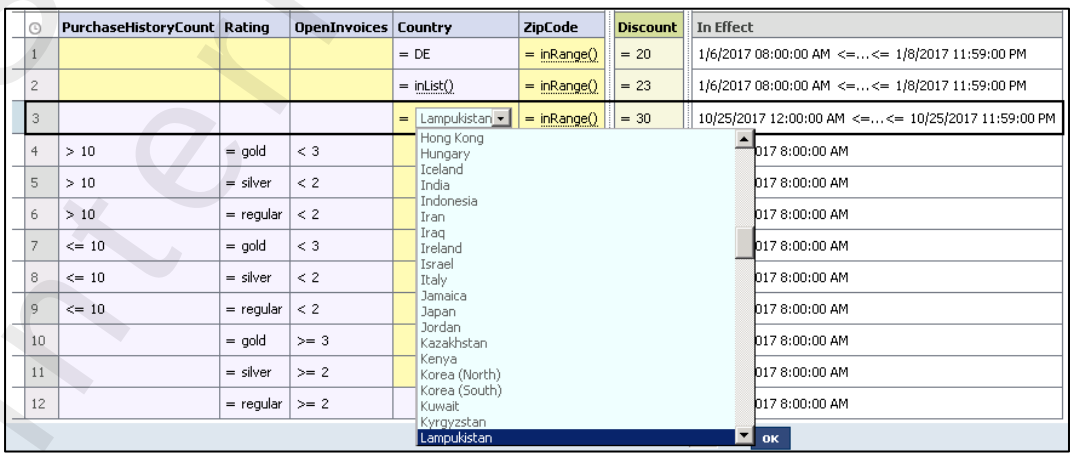
e. Click **Add Rule** to append an empty rule at the end of your Decision Table. Use button **Move Up** to move the empty rule up to row **3**.

f. The new (third) rule should fire for orders with a billing address **country** of **Lampukistan (LP)** and with billing address zip codes in the range between **40000** and **59999**. Zip codes **45000** and **55000** should not be valid. The resulting **Discount** value should be **30**.
This rule (offering) should be valid only at <***this year***>**/Oct/25** (birthday of head of state Joltawan Barodscheff).

Hint: You can select the country code value for the Country condition from the drop-down filled by the Data Provider Service. To build the ZipCode condition, open the extended cell editor and create an Expression using the inRange with exclusions function.







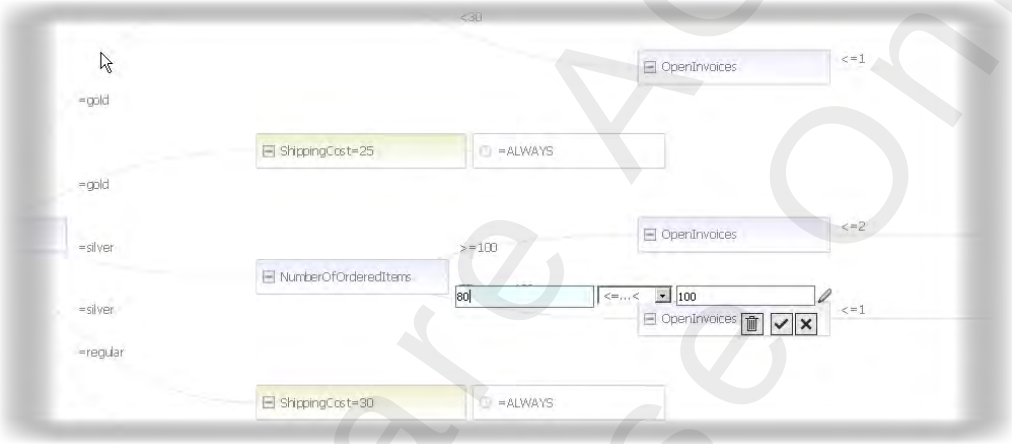
g. Click **Save** to persist your changes in the MWS repository.

h. **Unlock** your Decision Table in the RMC.

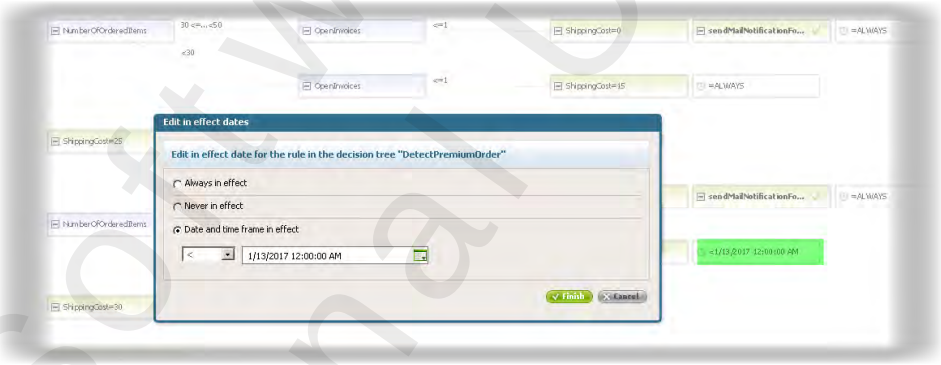
6. Select Decision Tree **DetectPremiumOrder** to open it in the RMC.

   a. **Lock** the Decision Tree to perform changes.

   b. Modify the condition for **NumberOfOrdereditems** of rule six to be become **80 <= NumberOfOrdereditems < 100.** Click ☑.



   c. Modify the In Effect date for rule six in a way that this rule is in effect for the next seven days. Click ☑.

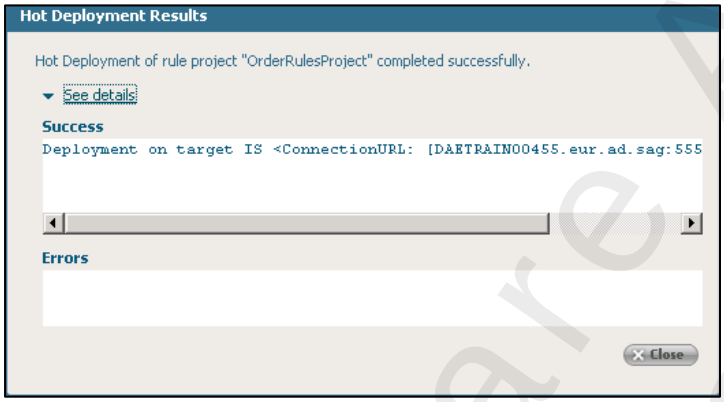


   d. Decrease the returned **ShippingCost** value for rule seven from 30 to **25**. Click ☑.

   e. **Unlock** your Decision Tree in the RMC.

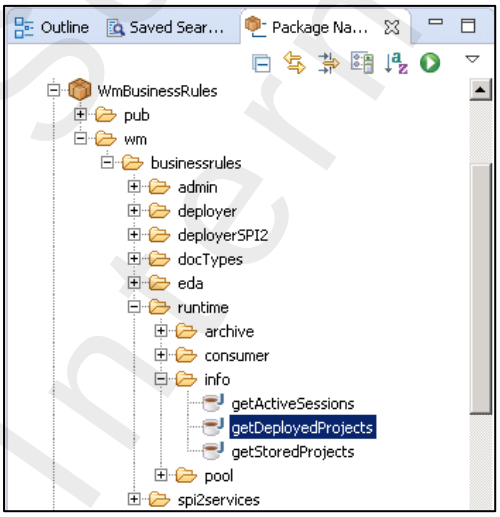
7. Click **Hot Deploy** to deploy (and overwrite) your entire rule project OrderRulesProject in the IS runtime (as configured in the previous exercise). Confirm the hot deploy with **OK**.

Double-check the reported results:

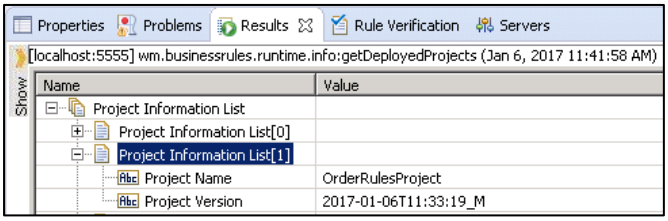


Note: A Hot Deploy from the RMC uses all Integration Server as targets with a configured **Integration Server Connection** on the webMethods Business Rules Settings page.

8. *Optionally*: Check the successful Hot Deploy to your Integration Server runtime:
   a. Switch back to Designer and use the **Package Navigator** view. Execute **(Run As > Run Service**) service **wm.businessrules.runtime.info:getDeployedProjects** contained in the IS package **WmBusinessRules**. The service does not require any inputs.



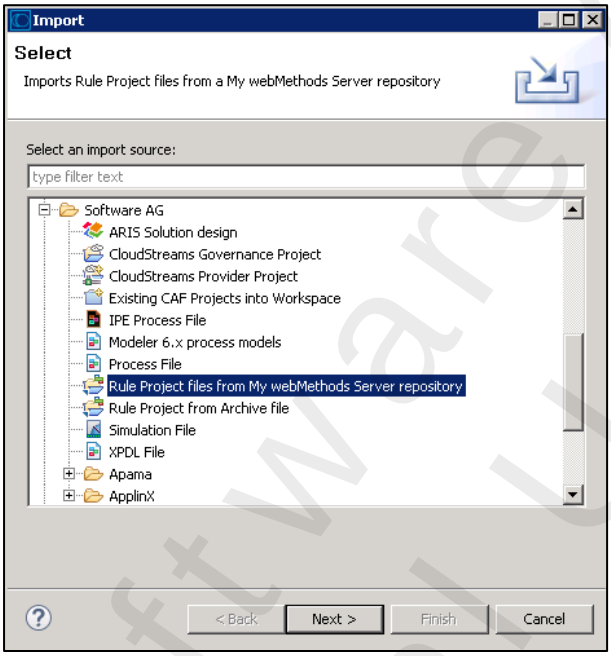
b. View the results in the **Results** view. The version number should be replaced and should be suffixed with a **M** (like **M**y webMethods) instead of **D** (like **D**esigner):



9. Use Designer to reload (synchronize) your modified rule project from the MWS repository:

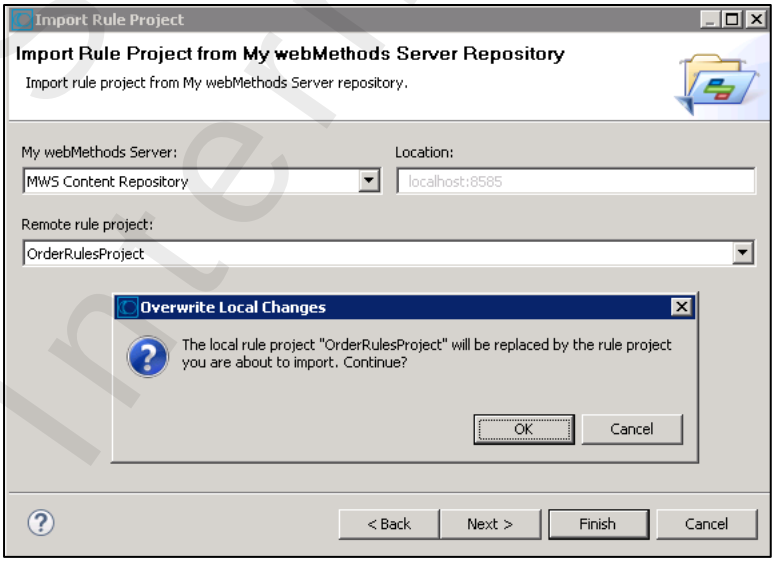
   a. Close all opened rule assets in Designer.

   b. Select **File - Import…** from Designer's menu bar. In the appearing wizard select import type **Software AG - Rule Project files from My webMethods Server repository**. Click **Next**.



   c. On the next panel select **MWS Content Repository** as source and the rule project **OrderRulesProject** as remote rule project.

   d. Click **Finish** to start the import. Allow overwriting:



10. Re-open your Decision Table **CalculateDiscountSpecialOffer** from the Rules Explorer of Designer.

	PurchaseHistoryCount	Rating	OpenInvoices	Country	ZipCode	Discount
1				= DE	= inRange(80000,99999)	= 20
2				= inList({"AU","CH","LI","LU"})	= inRange(10000,99999)	= 23
	Date and time in effect Jan 6, 2017 8:00:00 AM <=...<= Jan 8, 2017 11:59:00 PM			= LP	= inRange(40000,59999,{45000,50000})	= 30
4	> 10	= gold	< 3			= 18
5	> 10	= silver	< 2			= 10
6	> 10	= regular	< 2			= 5
7	<= 10	= gold	< 3			= 3
8	<= 10	= silver	< 2			= 2
9	<= 10	= regular	< 2			= 2
10		= gold	>= 3			= 3
11		= silver	>= 2			= 0
12		= regular	>= 2			= 0

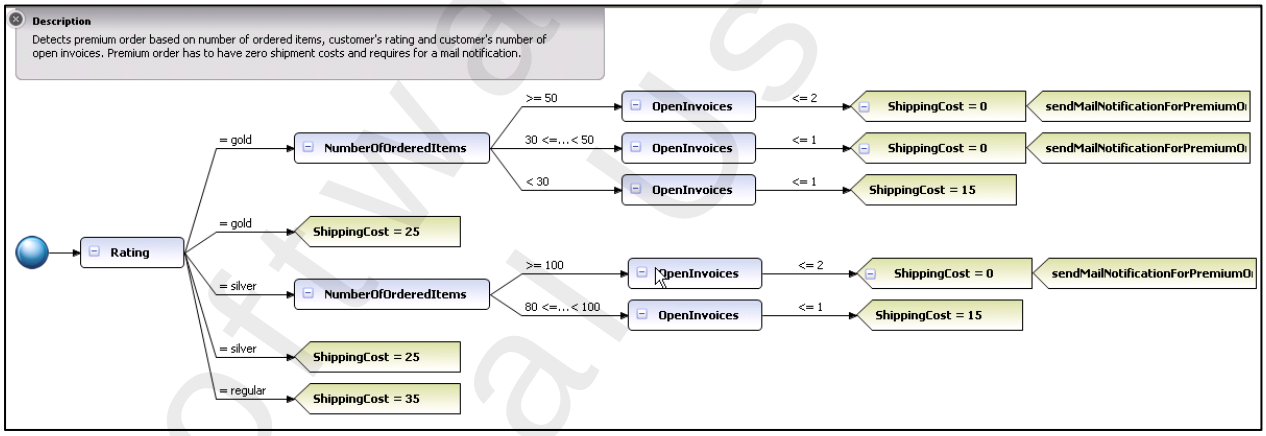
Check for the synchronized modifications:

a. Move your mouse over first cell for rule one to see its assigned In Effect time frame.

b. Check for the modified Discount value of rule four.

c. Verify a last third rule has been added to the Decision Table.

11. Re-open your Decision Tree **DetectPremiumOrder** from the Rules Explorer of Designer.
Check for the synchronized modifications in rules six and seven.



Exercise 11:
Process Invokes Decision Tree

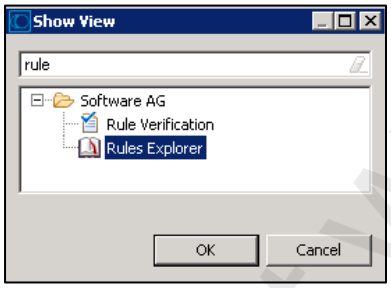
Objectives

In this exercise, you will enhance a preconfigured process model by adding a Rule Task Activity. The Rule Task Activity has to be configured to invoke your Decision Tree DetectPremiumOrder. You will debug the process to see the returned results of your invoked Decision Tree.

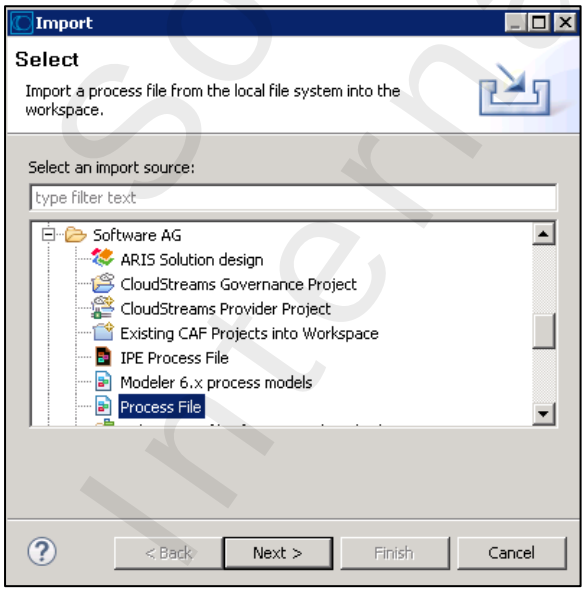
Steps

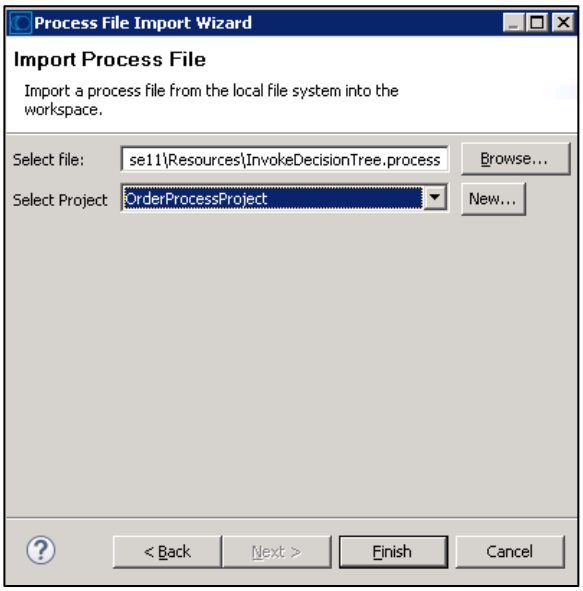
1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. Start **Designer** and switch to the **Process Development** perspective.
4. Use the Solutions view and add a new Process project named **OrderProcessProject**.
5. If the Rules Explorer view is not visible, choose **Window** > **Show View** > **Other…** and select the Rules Explorer view to be displayed in the Process Development perspective.

   Hint: Use the search filter in the Show View panel to find the Rules Explorer view.

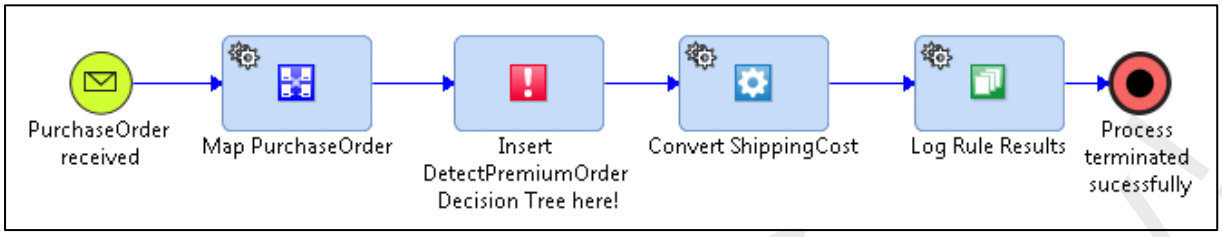


6. Select **File** > **Import** to import the Process File **<*workshop-dir*>\Exercise11\Resources\InvokeDecisionTree.process** into your process project **OrderProcessProject**.



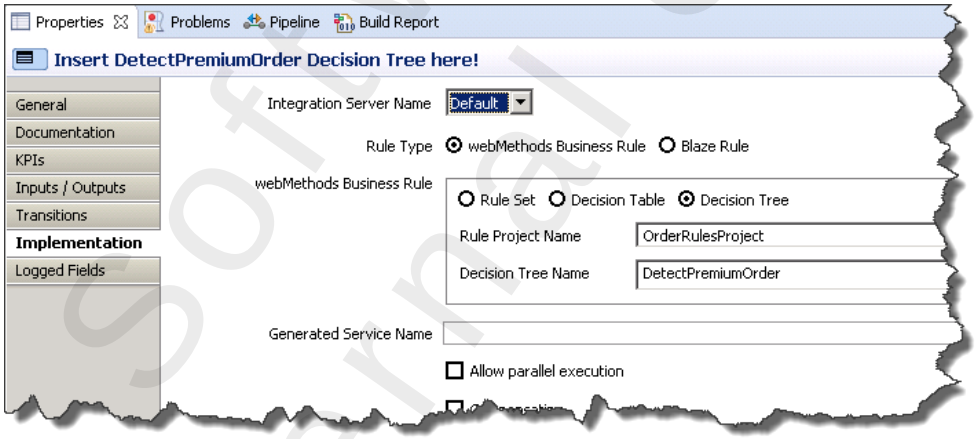


7. If not opened automatically, open the process model InvokeDecisionTree in the process editor. Ensure you are working in the **Process Developer** mode  . The opened process should look like this:

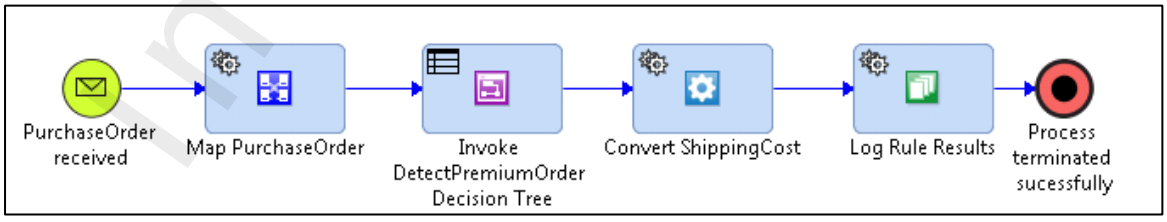


Remarks: The prepared process model receives **PurchaseOrder** documents by its Start Message Event named **PurchaseOrder received**. In the third Task Activity, your **DetectPremiumOrder** Decision Tree should be invoked. This is not done yet, instead an Abstract Task Activity named **Insert DetectPremiumOrder Decision Tree here!** acts as a placeholder. Because your Decision Tree expects an input parameter of type **PurchaseOrder** named **PurchaseOrder_1**, a preceding mapping step named **MapPurchaseOrder** returns this named parameter as a result in the process pipeline. Task Activities **ConvertShippingCost** and **LogRuleResults** map the shipping costs (returned by the Decision Tree) into a string value and create an appropriate message in the IS Server log.

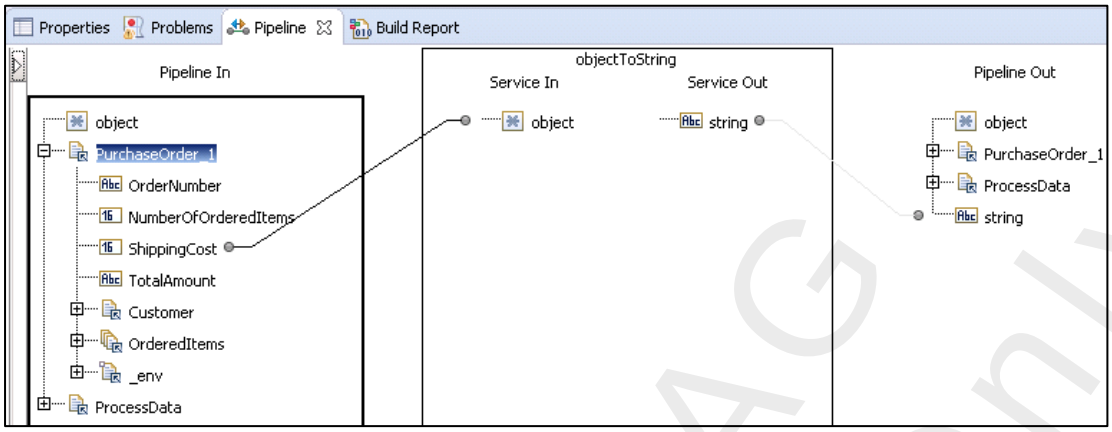
8. Use the Rules Explorer view and drag the Decision Tree **DetectPremiumOrder** contained in your rule project **OrderRuleProject** onto the Activity **Insert DetectPremiumOrder Decision Tree here!** in the process editor.
Select the Task Activity and inspect its **Properties** view. Dragging should have adjusted the Task Activity type to **Rule** and the rule type to **webMethods Business Rule**, sub type **Decision Tree**:



9. Right-click the Rule Task in the editor to rename it to **Invoke DetectPremiumOrder Decision Tree** and to adjust its image to fit to the following screen shot:



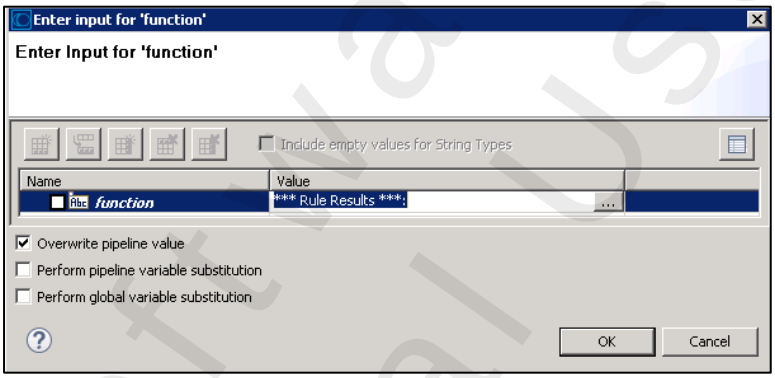
10. Save your process model. Right-click Task Activity **Convert Shipping Cost** in the process editor and choose **Edit Data Mapping**. Select the invocation of the IS service **pub.string.objectToString** in the Tree view and adjust the corresponding mapping in the Pipeline view to fit to the following screen shot:



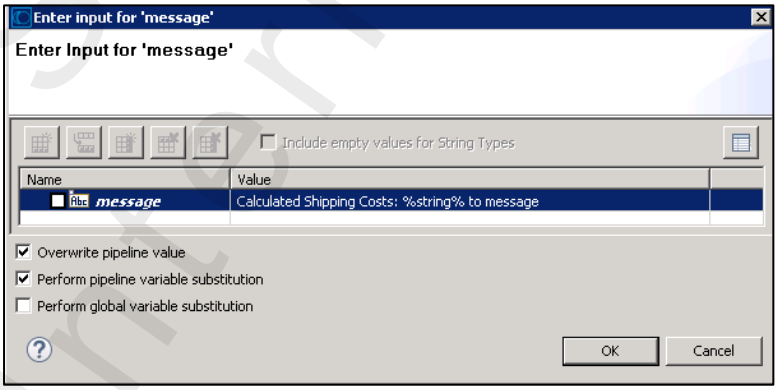
Save your changes.

11. Switch back to the process model **InvokeDecisionTree**. Right-click Task Activity **Log Rule Results** in the process editor and choose **Edit Data Mapping**.

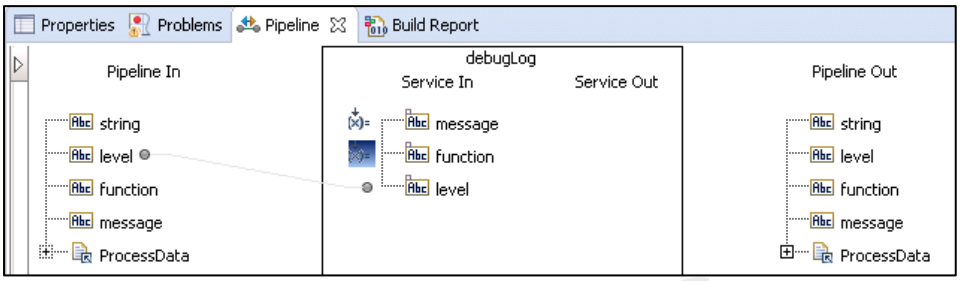
    a. Select the invocation of the IS service **pub.flow.debugLog** in the Tree view.
    In the Pipeline view, set the value *** Rule Results ***: to function under Service In:



    b. Set value **Calculated Shipping Costs: %string%** to **message** (under Service In) and check **Perform pipeline variable substitution**.



c. Adjust the mapping in the Pipeline view to fit to the following screen shot:



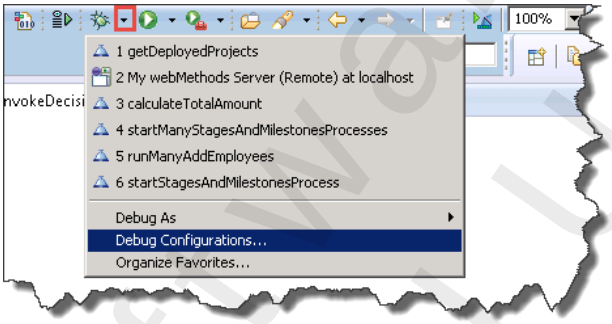
d. Save all your changes.

12. Save, Build and Upload your process model **InvokeDecisionTree**. If asked to automatically enable your process model for execution, click **Yes**.

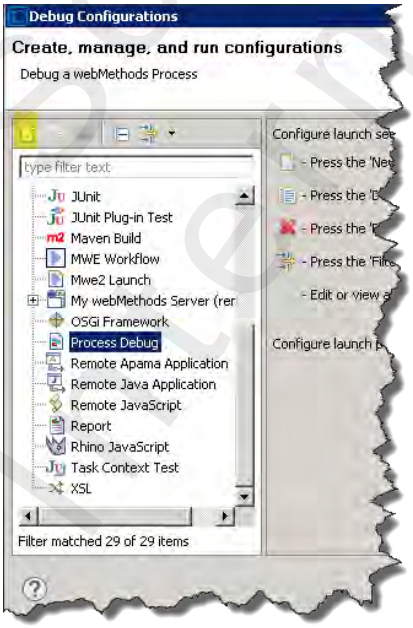
13. In the Package Navigator view, right-click document **rulesSupport.docs:PurchaseOrder** within package **RulesSupport** and select **Sync Document Type** to push the Document Type to the Messaging Provider.

14. Now test your process model **InvokeDecisionTree** using the Process Debugger:

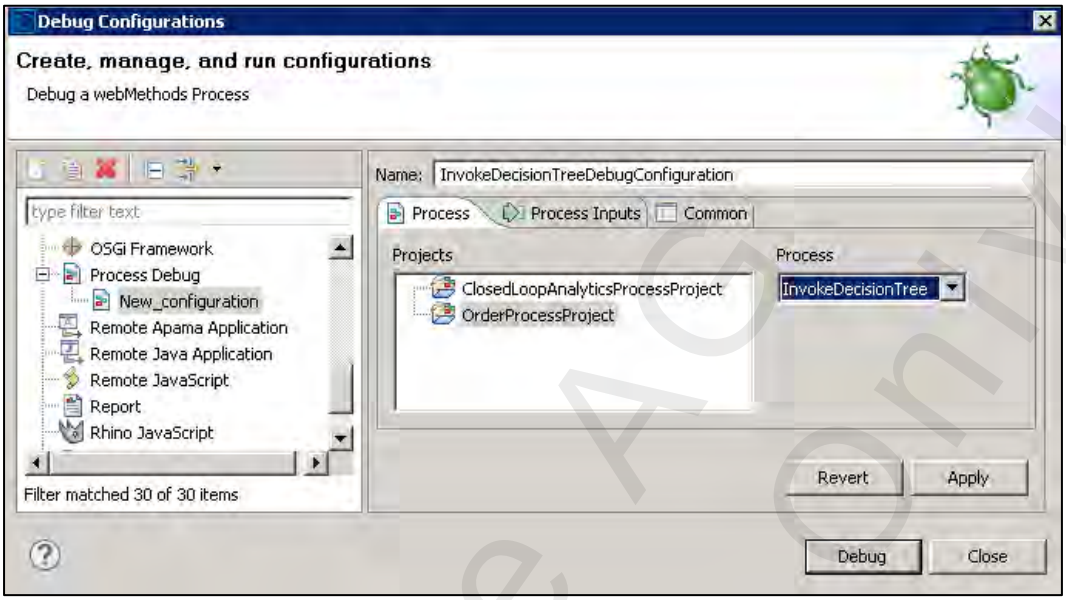
a. Before you can debug a process model for the very first time, you have to create a Debug (Launch) Configuration in your Designer workspace. To do so, click the triangle next to the Debug icon having the process model loaded in the process editor. Select **Debug Configurations…** .



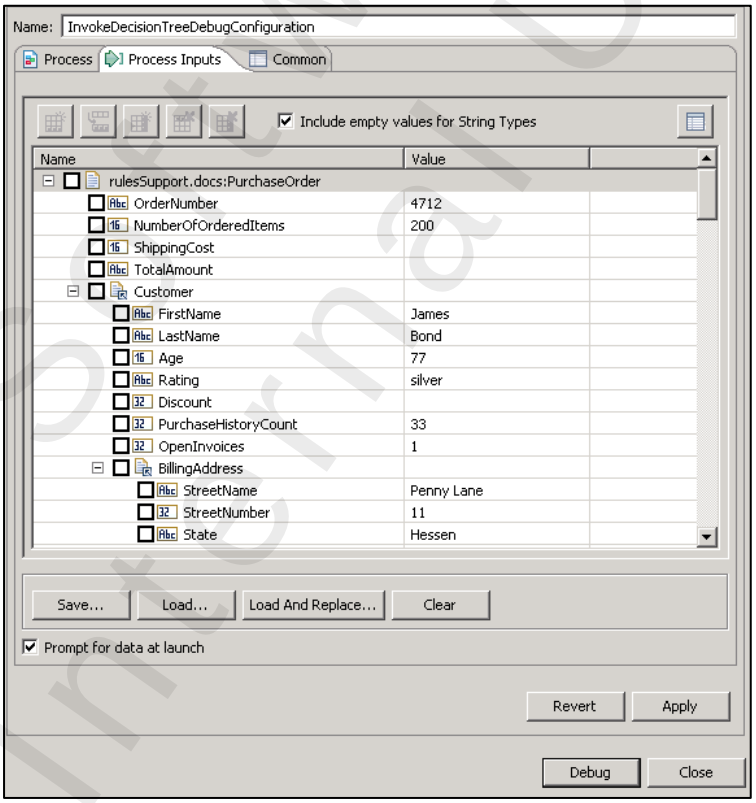
b. Select asset type **Process Debug** and hit the **New launch configuration** button.



c. On the **Process** tab of your new Debug Configuration, specify **InvokeDecisionTreeDebugConfiguration** as Run Configuration name and select the process model **InvokeDecisionTree** within the process project **OrderProcessProject**.



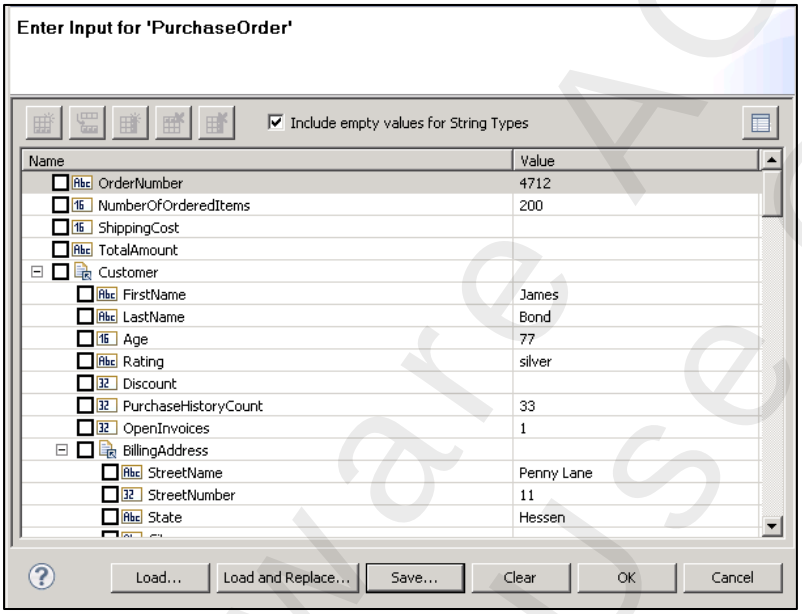
d. On the **Process Inputs** tab, provide input data for the Start Message Event and check to get prompted to type/alter those data when debugging starts.
Instead of typing you can load input data from **<*workshop-dir*>\Exercise11\Resources\InvokeDecisionTreeDebugConfigInput.xml**.
Check the box beside **Include empty values for String Types**. Click **Apply** to save the Debug Configuration. Then click **Debug** to start process debugging.



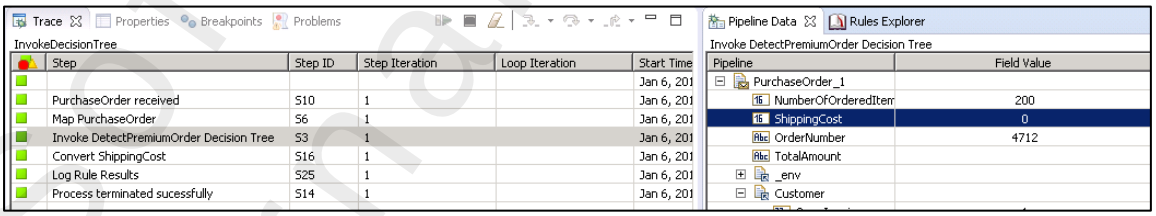
e. When get prompted, allow switching to the **Process Debug** perspective.

f. Because of the settings you made in the Debug Configuration you get prompted for the input data having your data specified as in the Debug Configuration preloaded. (Same) sample input can optionally be loaded from **<*workshop-dir*>\Exercise11\Resources\InvokeDecisionTreeDebugInput.xml**.
Again, check the box beside **Include empty values for String Types** and click **OK**.

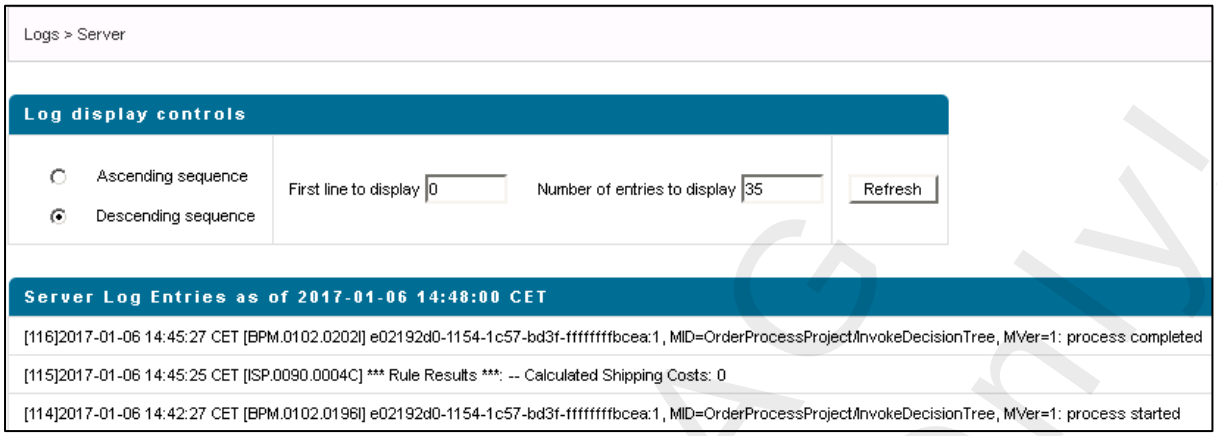
Note: When you load the test data for debugging, feel free to update the data, including the date fields with current dates/times. It's not mandatory but your test data will appear more current. If you change any data, remember the changes you made so that you can find it in the server log when testing.



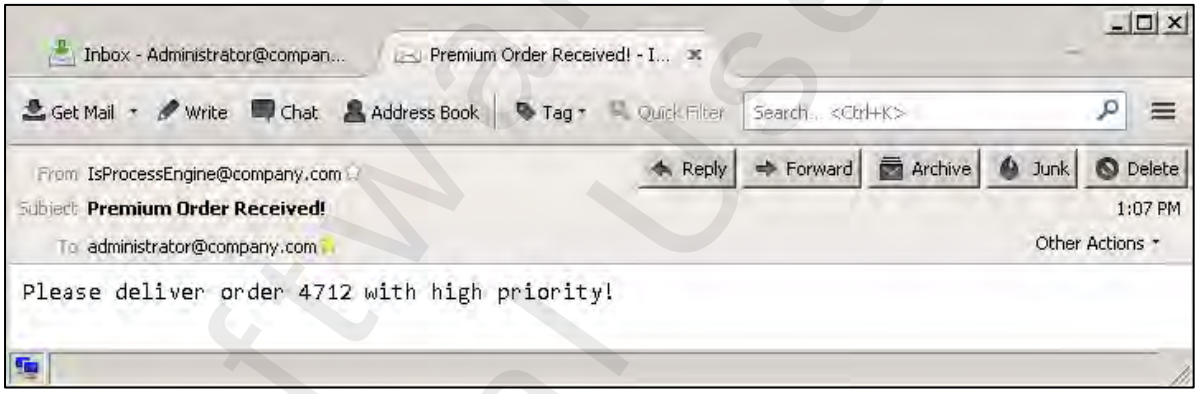
g. Use the Trace view to step over (F6) all activities until the process ends. By using the **Trace** and **Pipeline Data** views verify that your Decision Tree has been invoked successfully.



15. Additionally, open a browser tab, use URL **http://localhost:5555** to connect to the IS Administration console, and login as **Administrator | manage**. Visit **Logs** > **Server** and check for ***** Rule Results ***** written by your process:



16. Repeat debugging by running the configured Debug configuration **InvokeDecisionTreeDebugConfiguration** from the drop-down next to the debug icon. Perform debug session with "premium" and "non-premium" order data. For all premium orders, email recipient **Administrator@company.com** should additionally receive an email in Thunderbird:



This page intentionally left blank

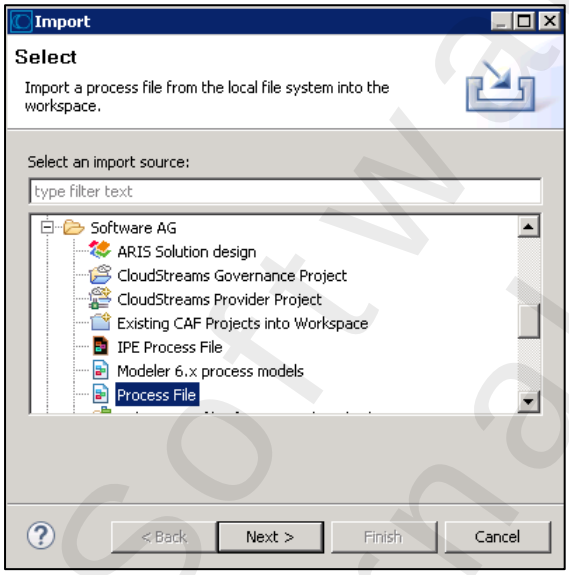
Exercise 12:
Process Invokes Rule Set

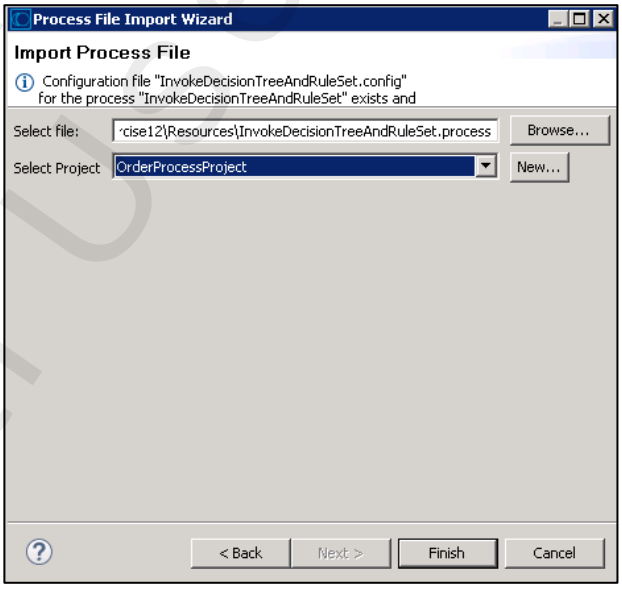
Objectives

In this exercise, you will enhance a preconfigured process model by adding two Rule Task Activities. The first Rule Task Activity has to be configured to invoke your Rule Set CustomerRuleSet, the second Rule Task Activity should invoke your Decision Tree DetectPremiumOrder. You will debug the process to see the returned results of your invoked Rule entities.

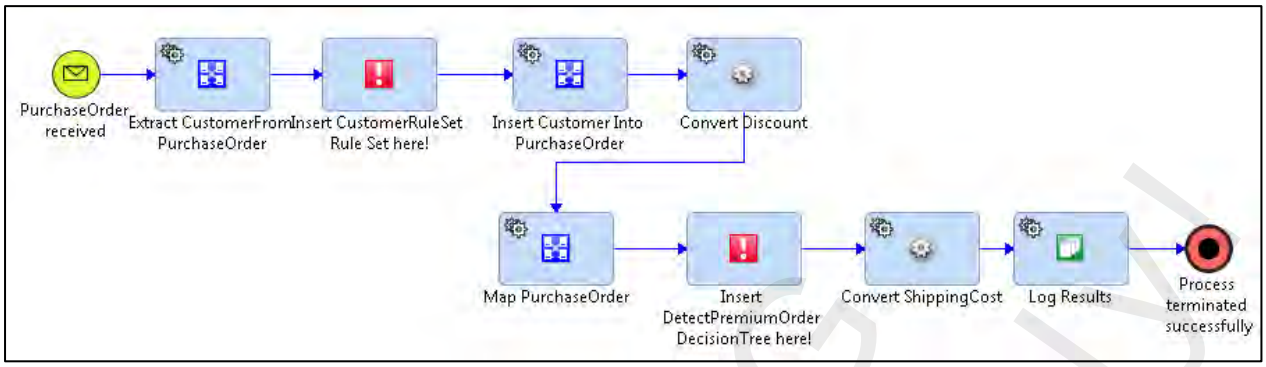
Steps

1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. Start **Designer** and switch to the **Process Development** perspective. Ensure you are working in **Process Developer** mode .
4. Select **File** > **Import** to import the Process File **<*workshop-dir*>\Exercise12\Resources\InvokeDecisionTreeAndRuleSet.process** into your process project **OrderProcessProject**.



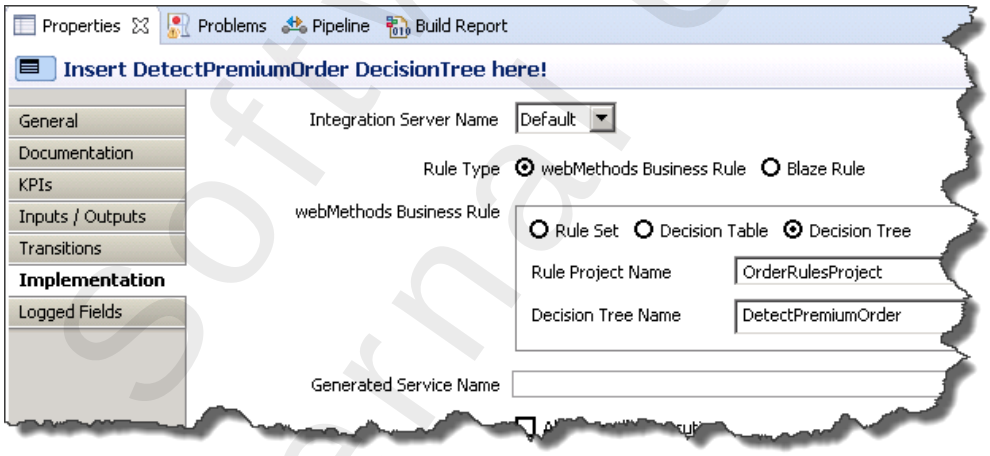


5. Open the process model **InvokeDecisionTreeAndRuleSet** in the process editor. It should look like this:

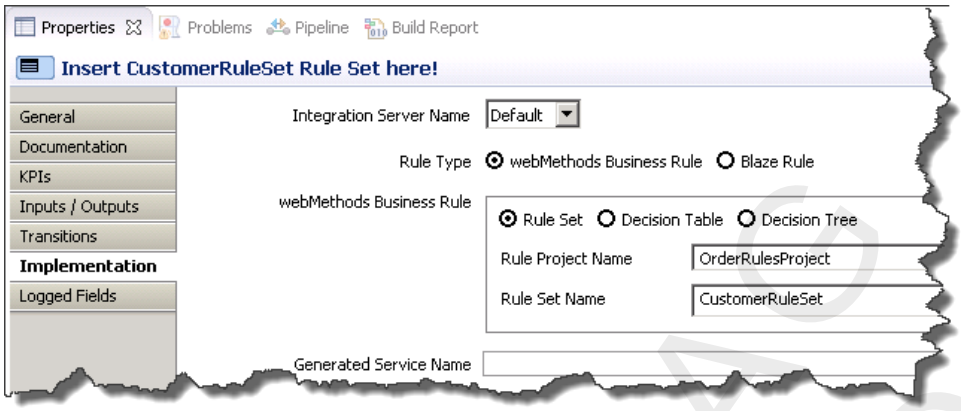


Remarks: The process model extends the model of the previous exercise. It was enhanced to invoke the Rule Set **CustomerRuleSet** before invoking the Decision Tree **DetectPremiumOrder** like in the last exercise. The Abstract Task Activities **Insert CustomerRuleSet Rule Set here!** and **Insert DetectPremiumOrder DecisionTree here!** have to be updated to invoke the correct Rule Set and respective Decision Tree. Moreover the data mapping of several steps has to be configured. **LogRuleResults** will log all rule results into the IS Server log.

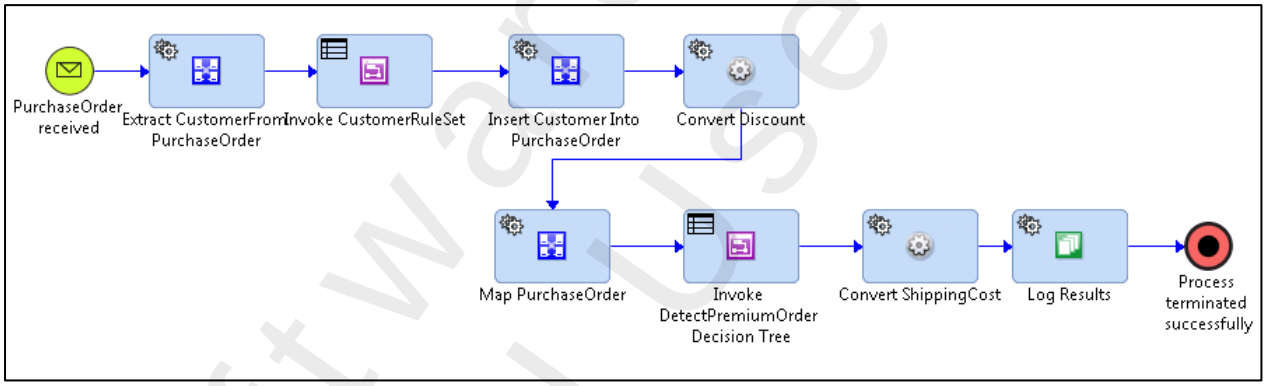
6. Use the Rules Explorer view and drag the Decision Tree **DetectPremiumOrder** contained in your rule project **OrderRuleProject** onto the Activity **Insert DetectPremiumOrder Decision Tree here!** in the process editor.
Double check the Properties of the Activity Task: Dragging should have adjusted the Task Activity type to **Rule** and the rule type to **webMethods Business Rule**, sub type **Decision Tree**:



7. Use the Rules Explorer view and drag the Rule Set **CustomerRuleSet** contained in your rule project **OrderRuleProject** onto the Activity **Insert CustomerRuleSet Rule Set here!** in the process editor. Double check the Activity Properties view: Dragging should have adjusted the Task Activity type to **Rule** and the rule type to **webMethods Business Rule**, sub type **Rule Set**:

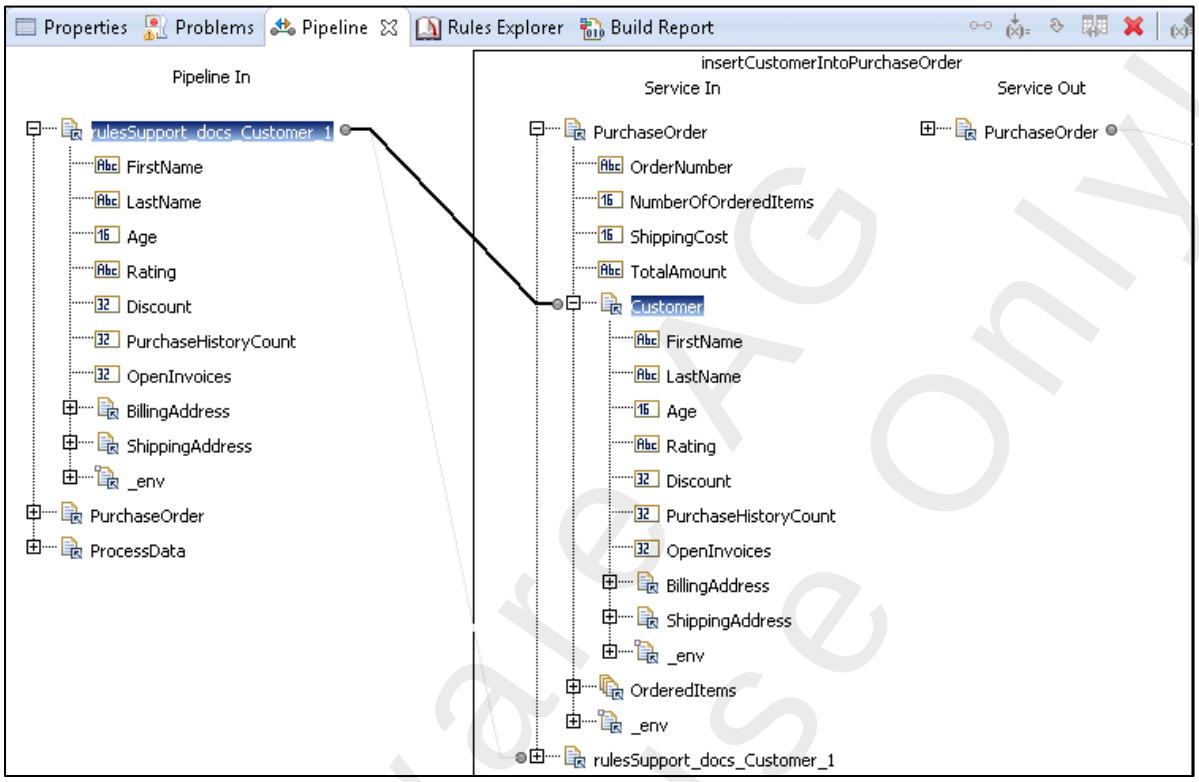


8. Rename the Rule Task Activity **Insert DetectPremiumOrder Decision Tree here!** to **Invoke DetectPremiumOrder Decision Tree** and **Insert CustomerRuleSet Rule Set here!** to **Invoke CustomerRuleSet.** Adjust the images to fit to the following screen shot:

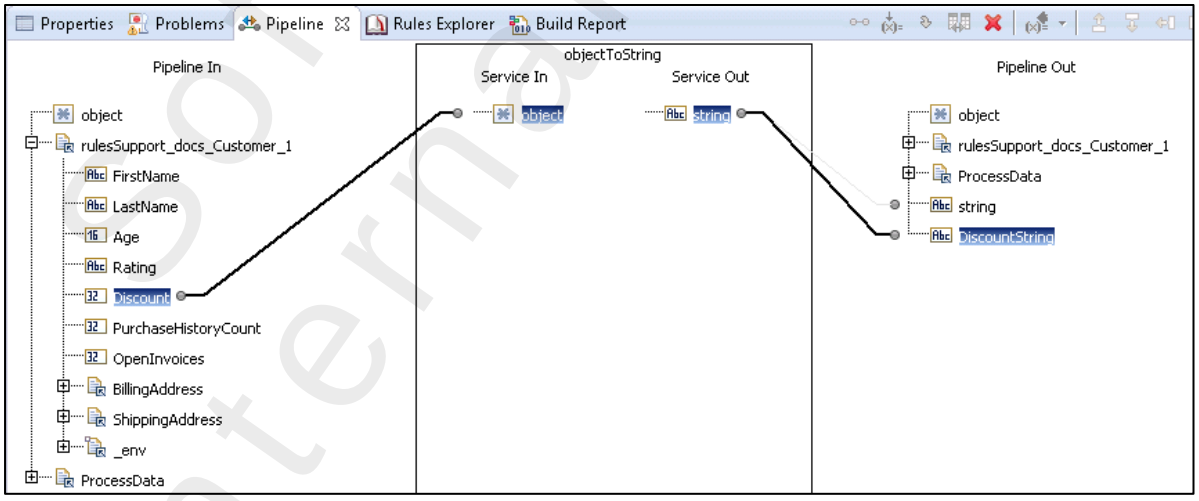


9. Save your process model.

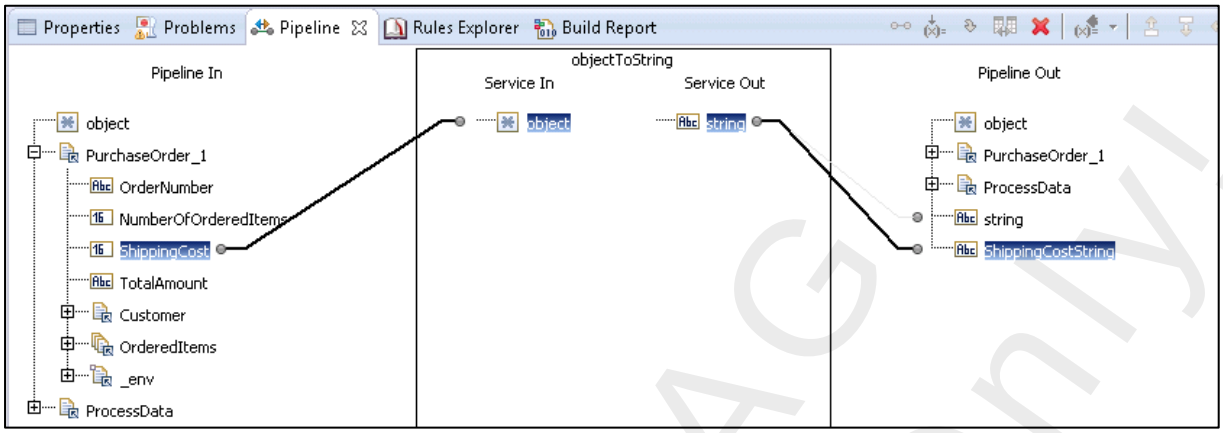
10. Right-click Task Activity **Insert Customer Into PurchaseOrder** in the process editor and choose **Edit Data Mapping**. Select the invocation of the IS service **rulesSupport.services:insertCustomerIntoPurchaseOrder** in the Tree view and adjust the corresponding mapping in the Pipeline view to fit to the following screen shot:



11. Right-click Task Activity **Convert Discount** in the process editor and choose **Edit Data Mapping**. Select the invocation of the IS service **pub.string.objectToString** in the Tree view and adjust the corresponding mapping in the Pipeline view to fit to the following screen shot:



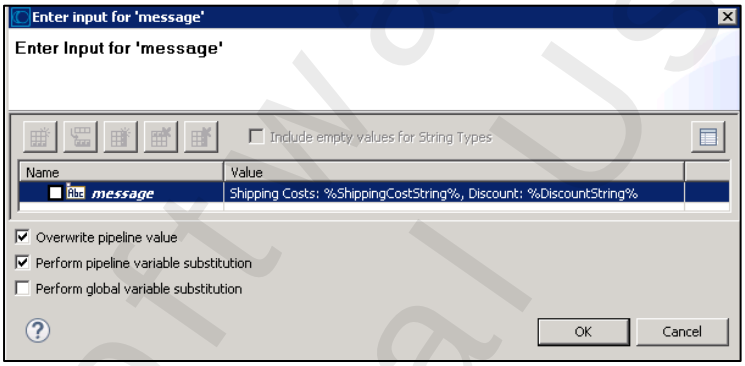
12. Right-click Task Activity **Convert ShippingCost** in the process editor and choose **Edit Data Mapping**. Select the invocation of the IS service **pub.string.objectToString** in the Tree view and adjust the corresponding mapping in the Pipeline view to fit to the following screen shot:



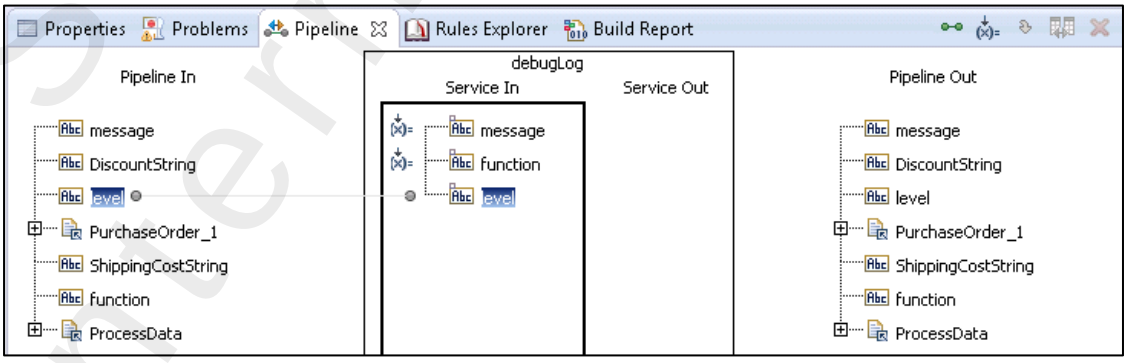
13. Right-click Task Activity **Log Results** in the process editor and choose **Edit Data Mapping**.

    a. Select the invocation of the IS service **pub.flow.debugLog** in the Tree view. Assign value ***** Rule Results ***:** to **function** (within **Service In**).

    b. Assign value Shipping Costs: **%ShippingCostString%, Discount: %DiscountString%** to **message** and check **Perform pipeline variable substitution**:

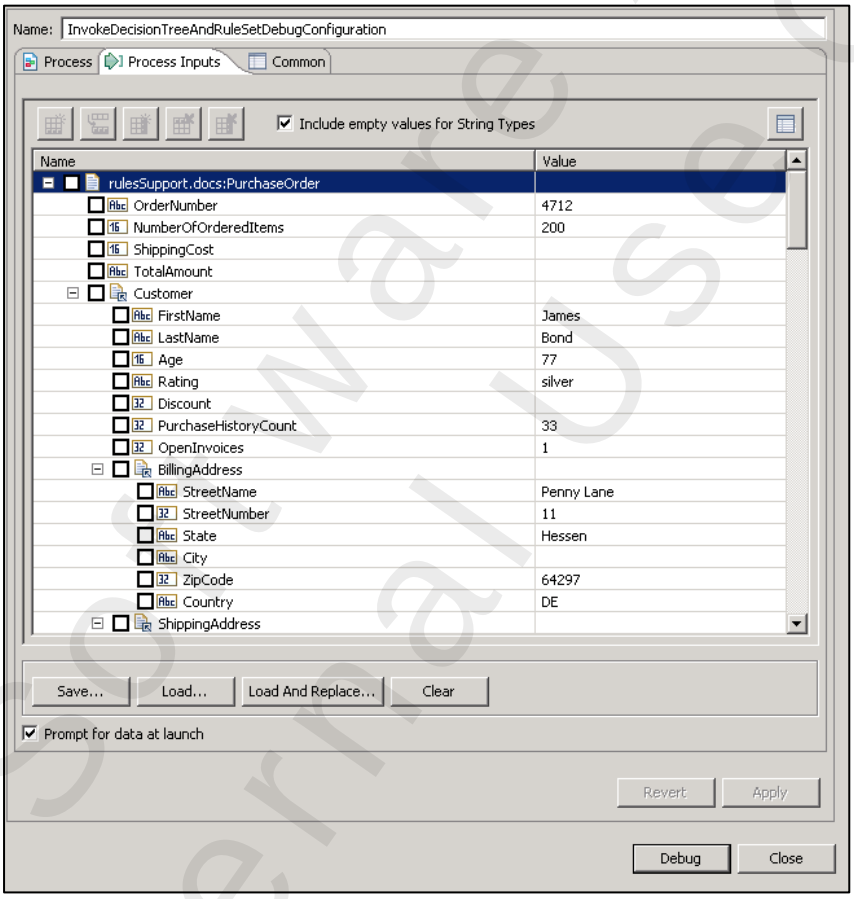


    c. Adjust the mapping in the Pipeline view (if necessary) to fit to the following screen shot:



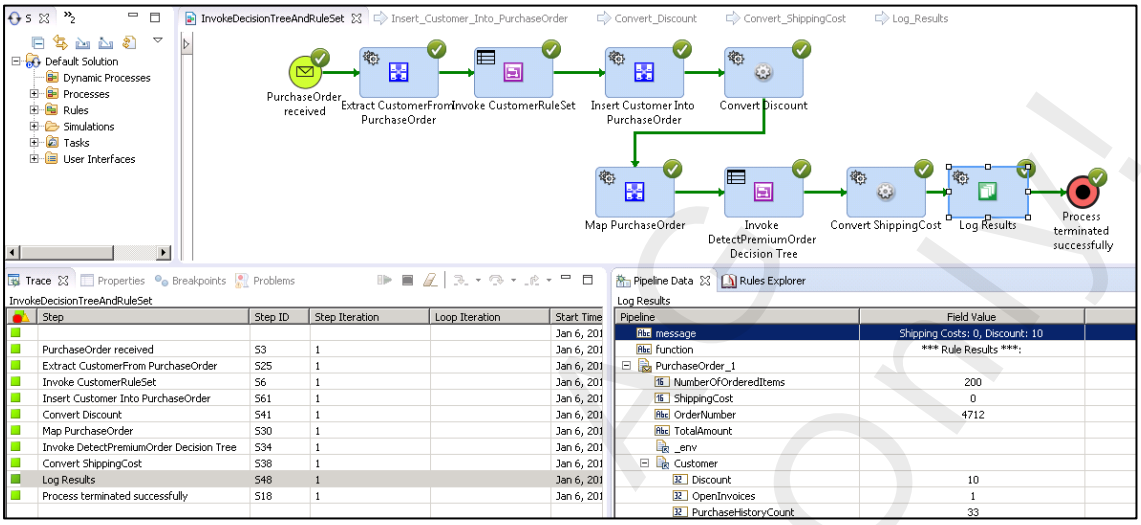
14. Save all your changes.

15. Build and Upload your process model **InvokeDecisionTreeAndRuleSet**. If asked to automatically enable your process model for execution, click **Yes**.
16. Switch to the Process Debug perspective.
17. As done in exercise 11, create, configure and run a Debug Configuration to test your process:
    a. On the **Process** tab of your new Debug Configuration, specify **InvokeDecisionTreeAndRuleSetDebugConfiguration** as Run Configuration name and select process model **InvokeDecisionTreeAndRuleSet** within process project **OrderProcessProject**.
    b. On the **Process Inputs** tab, provide input data for the Start Message Event and check to get prompted to type/alter those data when debugging starts.
    Instead of typing you can load input data from:
    <*workshop-dir*>\Exercise12\Resources\InvokeDecisionTreeAndRuleSetDebugConfigInput.xml.
    Click **Apply** to save the Debug Configuration.
    c. Then click **Debug** to start process debugging.



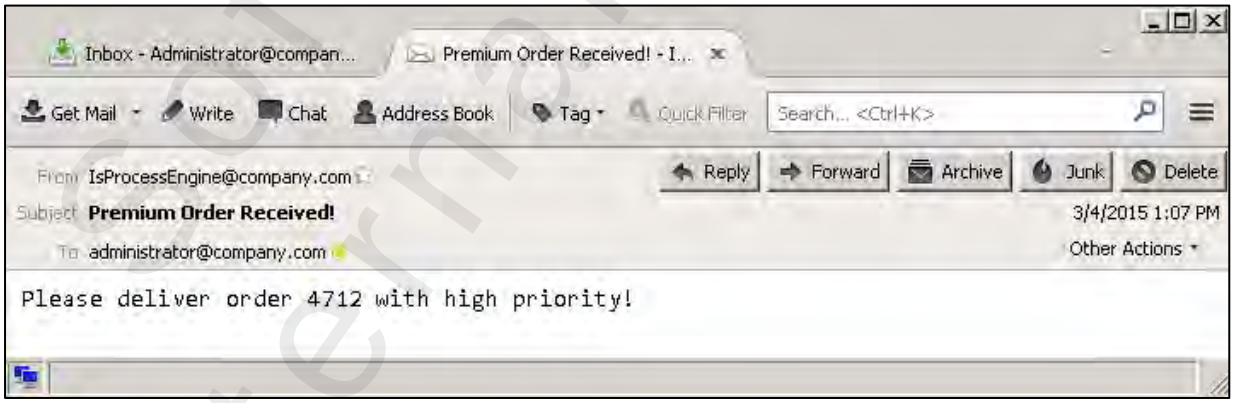
    d. Because of the settings you made in the Debug Configuration you get prompted for the input data having your data specified as in the Debug Configuration preloaded. (Same) sample input can optionally be loaded from:
    <*workshop-dir*>\Exercise12\Resources\InvokeDecisionTreeAndRuleSetDebugInput.xml.
    Check the box beside **Include empty values for String Types** and click **OK**.

e. Use the Trace view to step over (F6) all activities until the process ends. By using the **Trace** and **Pipeline Data** views verify that all Rule assets within Rule Set **CustomerRuleSet** and Decision Tree **DetectPremiumOrder** have been invoked successfully.



18. Open a browser tab, use URL http://localhost:5555 to connect to the IS Administration console, and login as **Administrator | manage**. Visit **Logs** > **Server** and check for ***** Rule Results ***** written by your process:

19. Repeat debugging with premium and non-premium order data. For all premium orders, you should receive an email additionally.



This page intentionally left blank

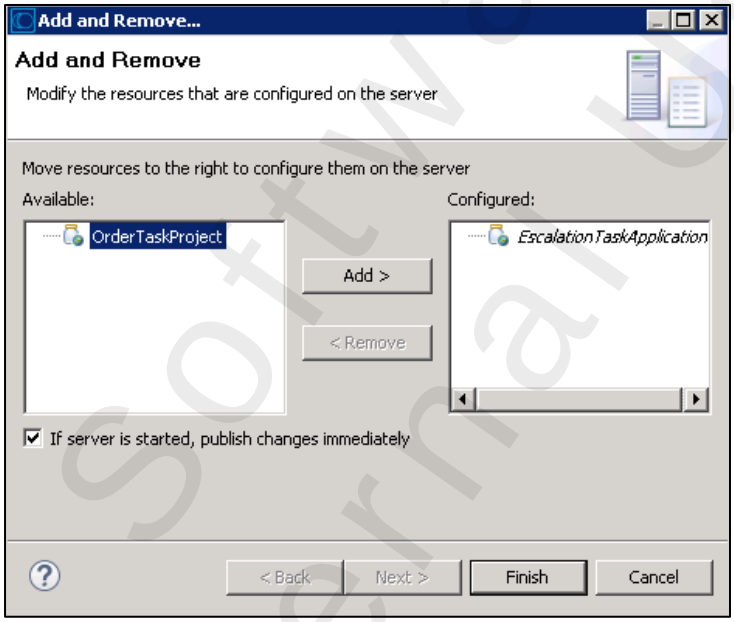
Exercise 13:
Start Process Action

Objectives

In this exercise, you will create a Process Action to start a process instance by publishing Alert documents. The Process Action will be invoked by a Decision Table catching Alert documents.
The process to be started contains a User Task Activity to show and confirm the mapped alert data.

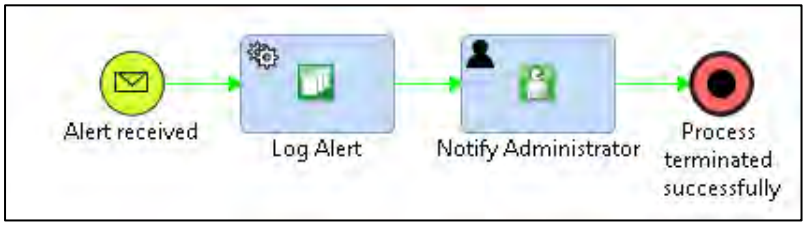
Steps

1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. Start **Designer** and switch to the **Process Development** perspective. Ensure you are working in **Process Developer** mode  .
4. Select **File** > **Import** > **Software AG** > **Existing CAF Projects into Workspace** to import a prepared Task project from the archive file **<*workshop-dir*>\Exercise13\Resources\OrderTaskProject.zip** into your workspace.
5. Right-click into the whitespace of the Servers view (open it via **Window** > **Show View**, if missing) and choose **Add and Remove…** to add the **OrderTaskProject** User Task project to your MWS.
When prompted for MWS authentication, provide **Sysadmin | manage** as user credentials.



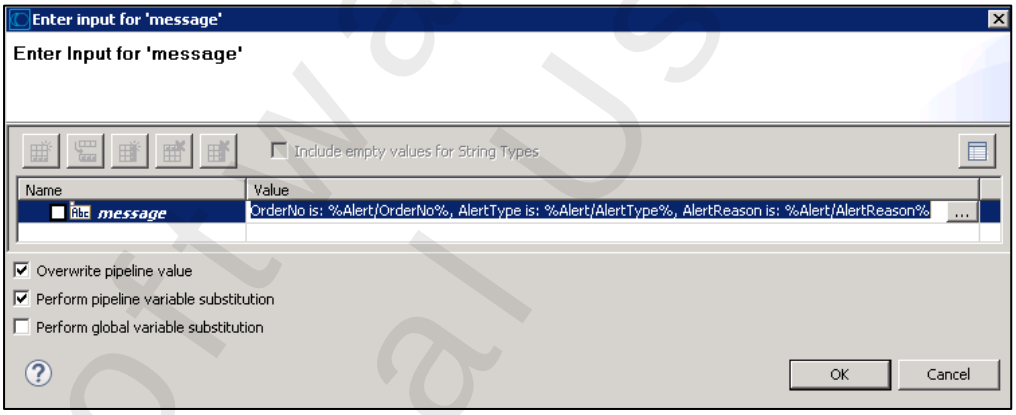
6. Select **File** > **Import** > Software AG > Process File to import the Process File **<*workshop-dir*>\Exercise13\Resources\AlertHandling.process** into your process project **OrderProcessProject**.

7. Open the process model **AlertHandling** in the process editor. It should look like this:

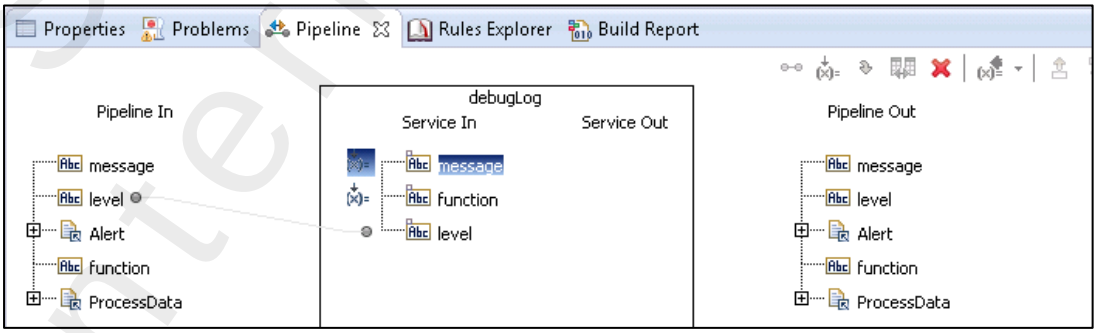


Remarks: The process model receives **Alert** documents. Incoming Alert documents will be logged to the IS Server log (Service Task Activity **Log Alert**) and shown on a Task UI.
The corresponding User Task Activity **Notify Administrator** invokes the User Task **AlertNotificationTask** contained in your imported Task project **OrderTaskProject**.
Process instances of this type could be started from a Decision Table working on Alerts.

8. Right-click the Service Task Activity Log Alert in the process editor and choose Edit Data Mapping.
   a. Select the invocation of the IS service **pub.flow.debugLog** in the Tree view.
   b. In the Pipeline view, assign value ***** Incoming Alert ***:** to **function** (within **Service In**).
   c. In the Pipeline view, assign value **OrderNo is: %Alert/OrderNo%, AlertType is: %Alert/AlertType%, AlertReason is: %Alert/AlertReason%** to **message** (within **Service In**) and check **Perform pipeline variable substitution**:



   d. Adjust (if necessary) the mapping in the Pipeline view to fit to the following screen shot:



9. Save all your changes. Build and Upload the process model **AlertHandling**. When prompted, allow to enable the process model for execution.
10. Right-click **rulesSupport.docs:Alert** in the Package Navigator view and select **Sync Document Type** to push the document type to the Messaging Provider.

11. Test the process using the Process Debugger in Designer:

    a. Switch to the **Process Debug** perspective.

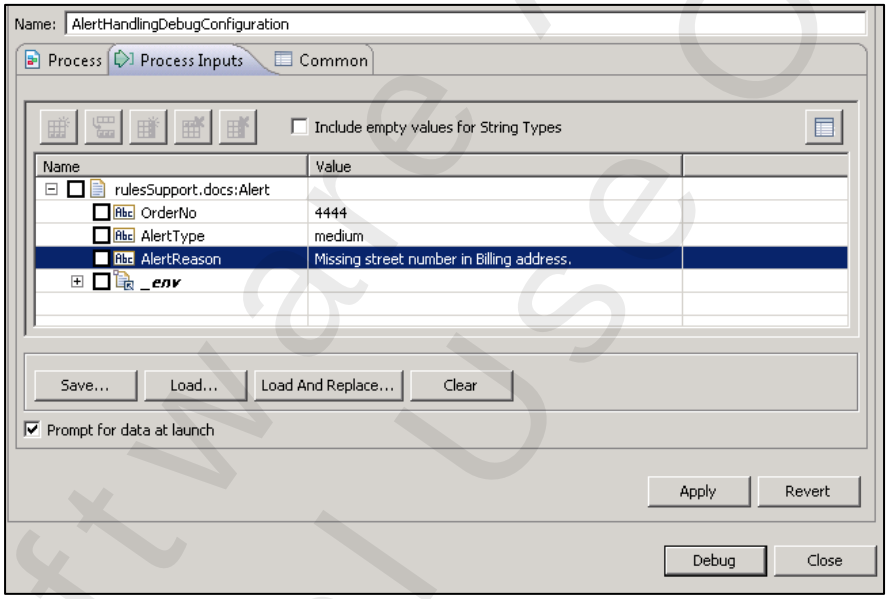
    b. As done in exercise 11, create, configure and run a Debug Configuration for the **AlertHandling** process:

        i. On the **Process** tab of your new Debug Configuration, specify **AlertHandlingDebugConfiguration** as Run Configuration name and select process model **AlertHandling** within process project **OrderProcessProject**.

        ii. On the **Process Inputs** tab, provide input data for the Start Message Event and check to get prompted to type/alter those data when debugging starts.
        Instead of typing you can load input data from:
        ***<workshop-dir>*\Exercise13\Resources\AlertHandlingDebugConfigInput.xml**

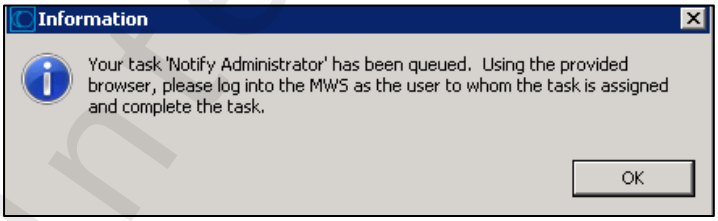
        iii. Click **Apply** to save the Debug Configuration.

        iv. Then click **Debug** to start process debugging.

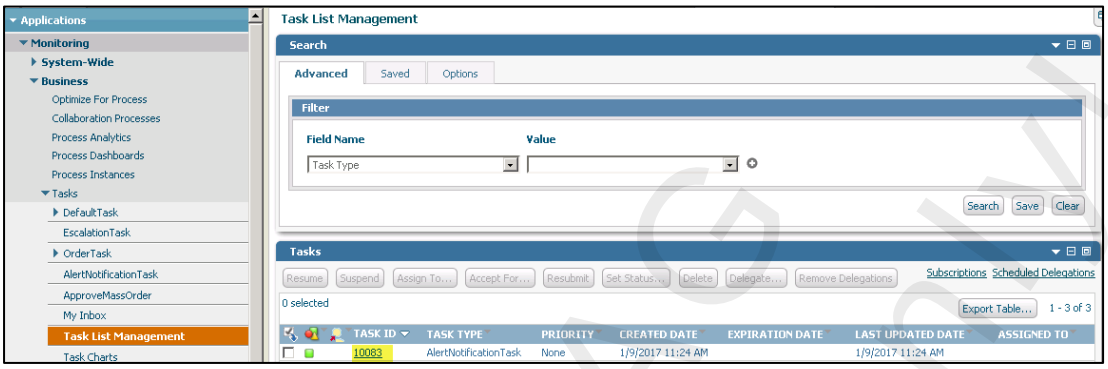


    c. Because of the settings you made in the Debug Configuration you get prompted for the input data having your data specified as in the Debug Configuration preloaded. (Same) sample input can optionally be loaded from:
    ***<workshop-dir>*\Exercise13\Resources\AlertHandlingDebugConfigInput.xml**
    Check the box beside **Include empty values for String Types** and click **OK**.

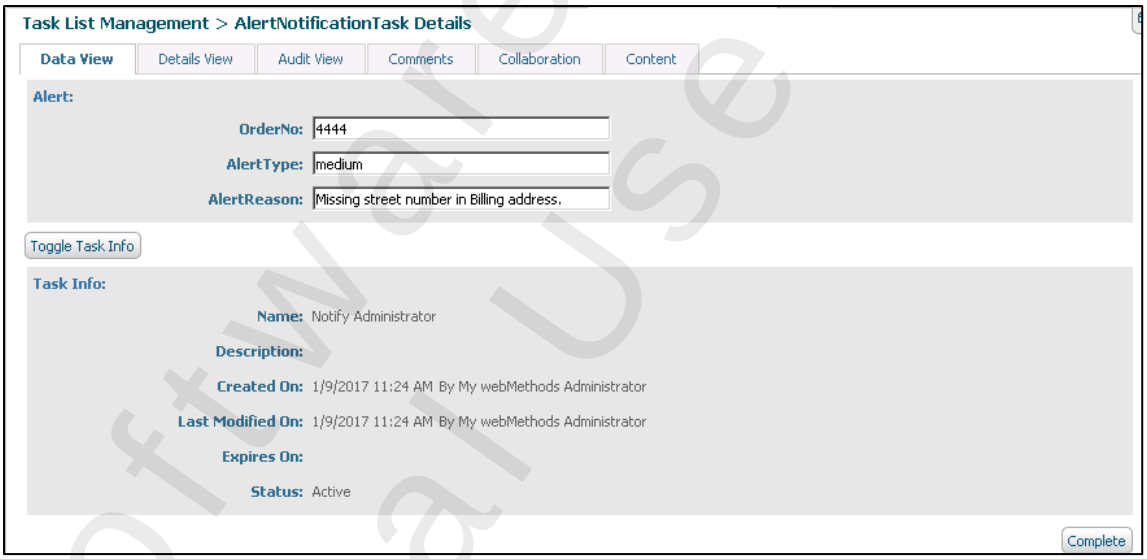
    d. Use the Trace view to step over (F6)  all activities until Task step Notify Customer.
    This step should bring up a new User Task instance.



e. In a browser tab, login to My webMethods (http://localhost:8585) as **Administrator** | **manage**. Navigate to **Applications** > **Monitoring** > **Business** > **Tasks** > **Task List Management**. Hit **Search** to refresh the Tasks list. Open the User Task instance of Task type **AlertNotificationTask**.

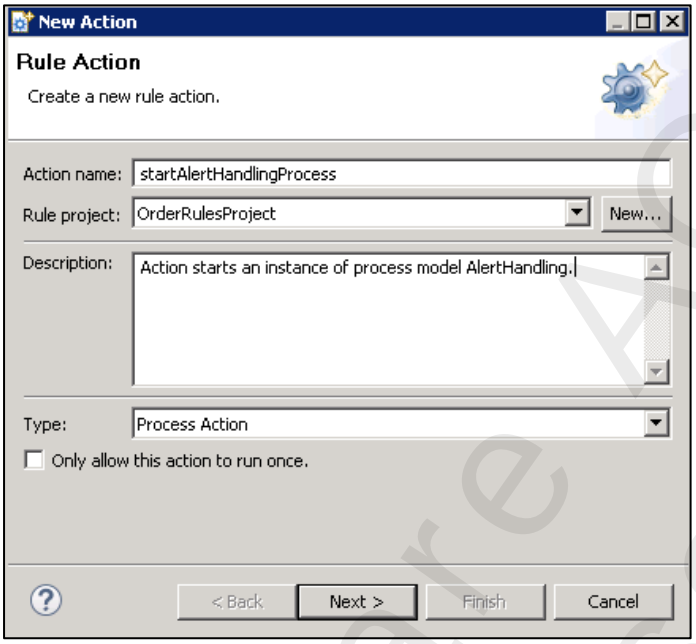


f. Click the Task ID link of User Task instance of Task type **AlertNotificationTask** to open it. Ensure Alert data is shown properly on the Task UI. **Complete** the User Task.

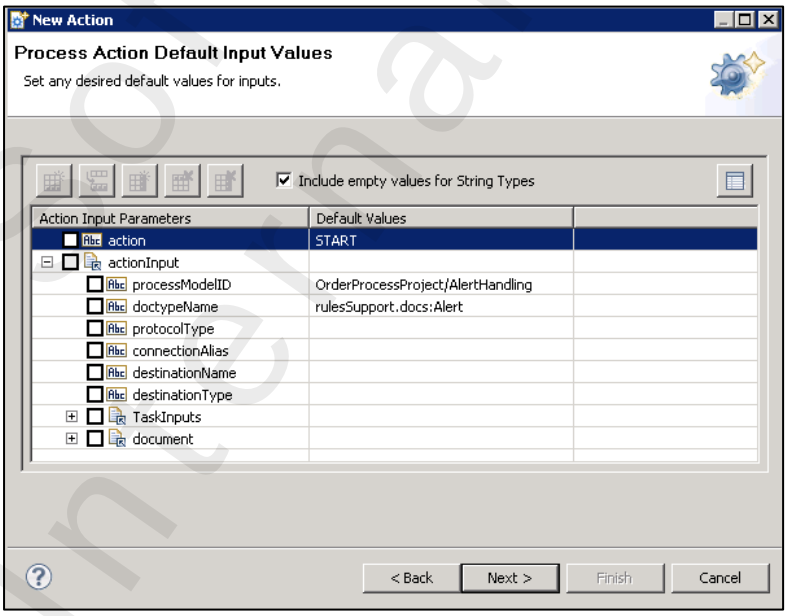


g. Switch back to the Process Debugger session in Designer and complete the process.

12. In Designer, switch to the **Rules Development** perspective.
13. Use the Rules Explorer view to add another Action:
    a. Create an Action of type **Process Action** in your Rule project **OrderRulesProject**. Name it **startAlertHandlingProcess,** provide a description text as shown on the screen shot below, and click **Next**.



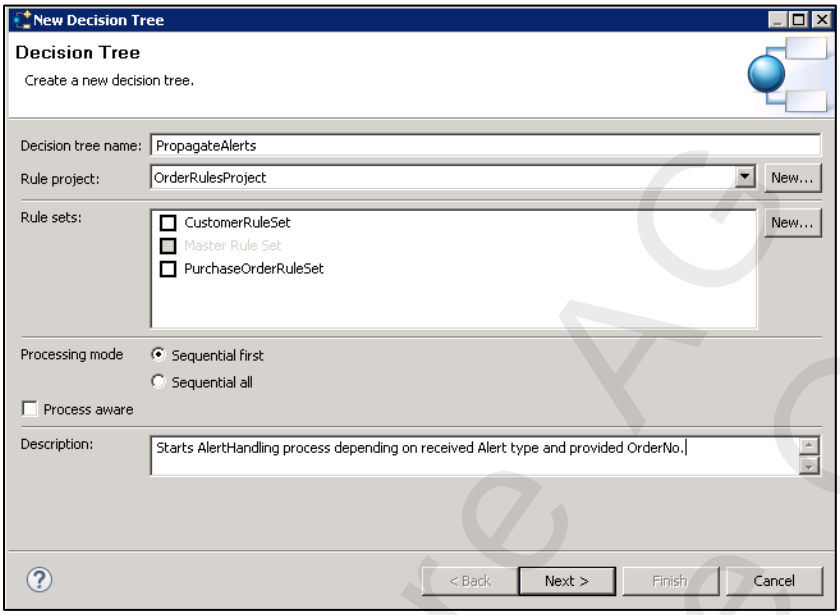
    b. On the next panel, choose Process Action type **Start a new process instance**. Click **Next**.
    c. On the subsequent panel, select process model **AlertHandling**. Click **Next**.
    d. On the next panel, select document type **rulesSupport.docs:Alert** to be published for process instantiation. Click **Next**.
    e. Leave the Process Action Default Input Values unchanged and check the box beside **Include empty values for String Types**. Click **Next**.



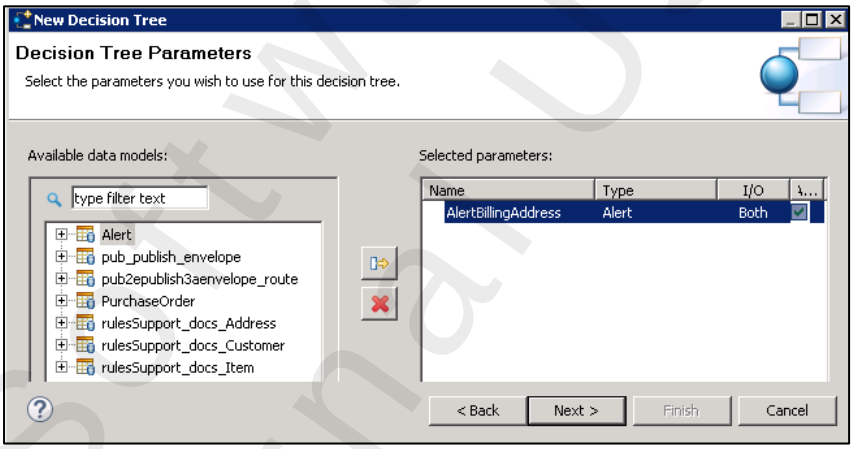
    f. On the last panel, check **outputMessage** as action return value. Click **Finish** to create the action.

14. Use the Rules Explorer view to add another Decision Tree:

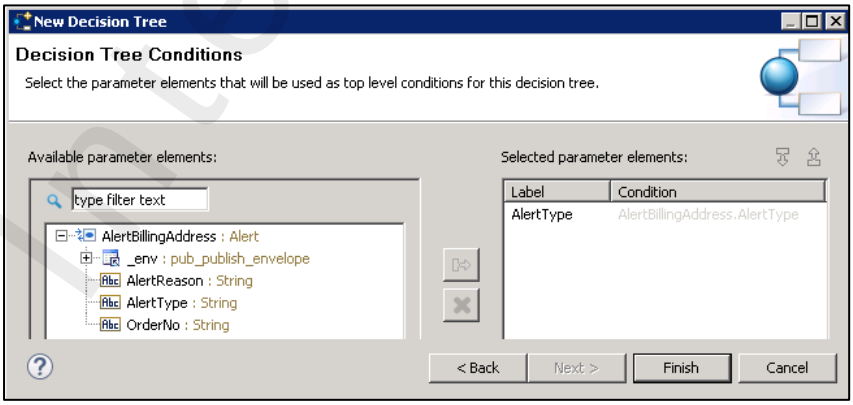
    a. Add a Decision Tree named **PropagateAlerts** to your **OrderRulesProject** project. Do NOT add this Decision Tree to any Rule Set yet (except for the Master Rule Set). Select Processing mode **Sequential first** and provide a description text as shown on the screen shot below. Click **Next**.



    b. On the subsequent panel, select Data Model **Alert** as Decision Table parameter with I/O type **Both**. Check **Any** and rename the parameter to **AlertBillingAddress**. Click **Next**.

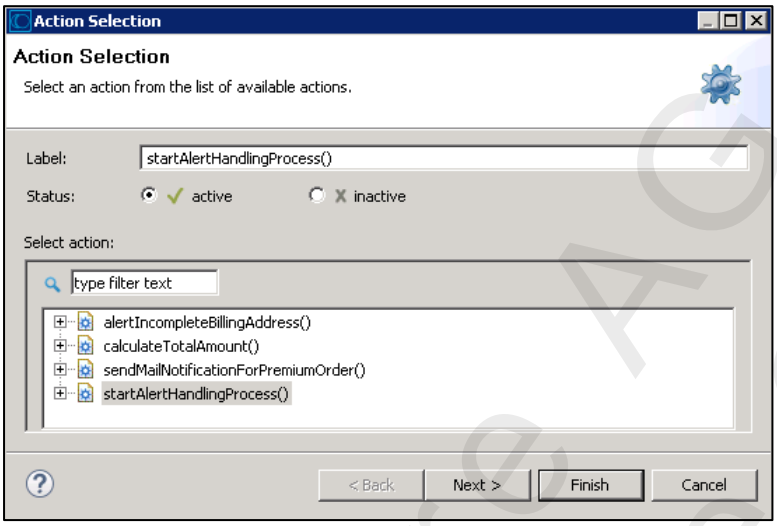


    c. On the next panel, select the parameter element **AlertType** to be used in the top level rule condition of the Decision Tree. Click **Finish**.

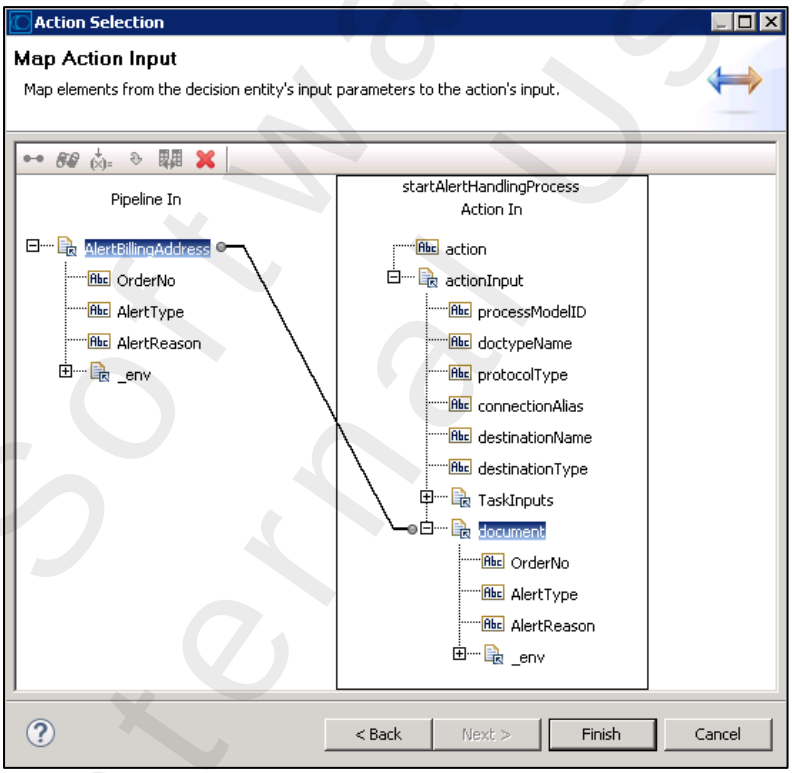


d. The Decision Tree will be opened in the Decision Tree Editor pane. Right-click parameter element **AlertType** and select **Add Condition** to add **OrderNo** as second parameter to be used in a subsequent condition.

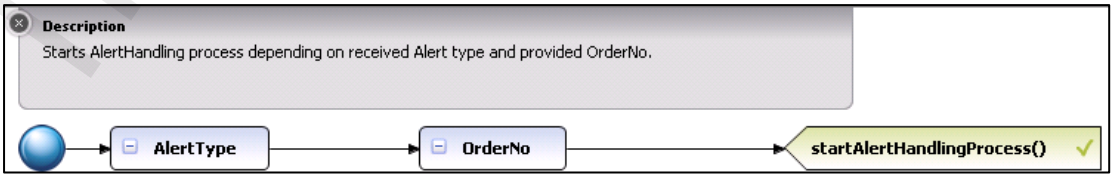
e. Right-click parameter element **OrderNo** and select **Add Action** to add **startAlertHandlingProcess()** as an action to be invoked as a result. Choose Status **active** to enable the invocation for this rule. Click **Next**.



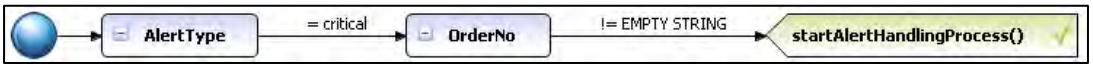
f. On the next panel map the Decision Tree input parameters to the action's input like this:



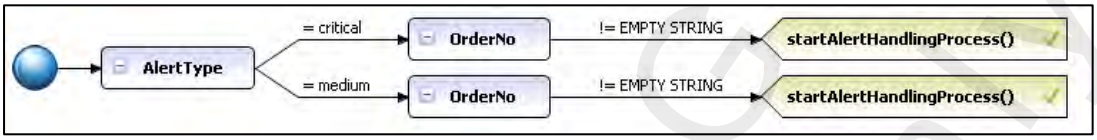
g. Click **Finish**. Your Decision Tree should now look like this:



h. To complete the first rule, configure the transition from **AlertType** to **OrderNo** to express the condition **AlertType = critical.** Next modify the transition from **OrderNo** to **startAlertHandlingProcess()** to express the condition **OrderNo |= EMPTY STRING**.



i. To create the second rule, copy condition node **OrderNo** and paste it onto condition node **AlertType**. Adjust the condition values of last second rule to reflect the screen shot below:

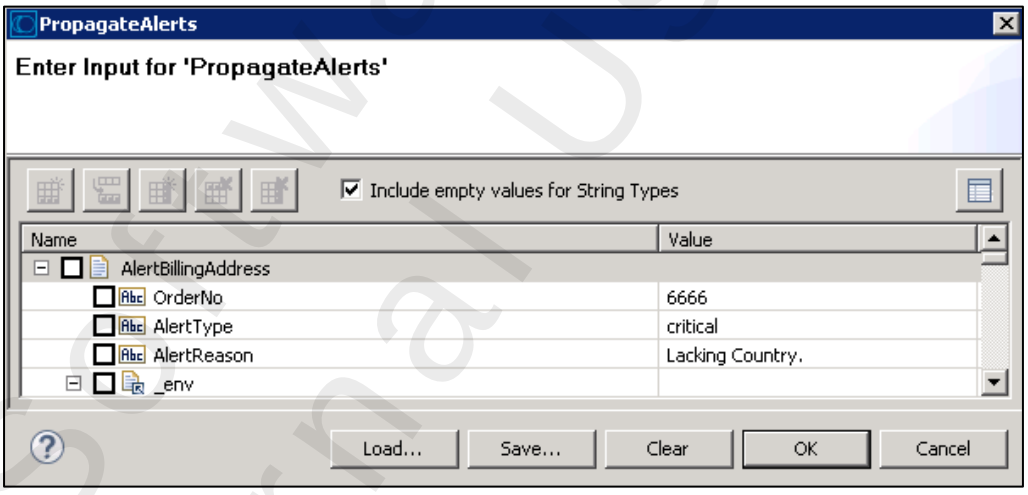


j. Save your Decision Tree.

15. Re-deploy (export) your Rule Project to your Integration Server runtime and MWS Repository.

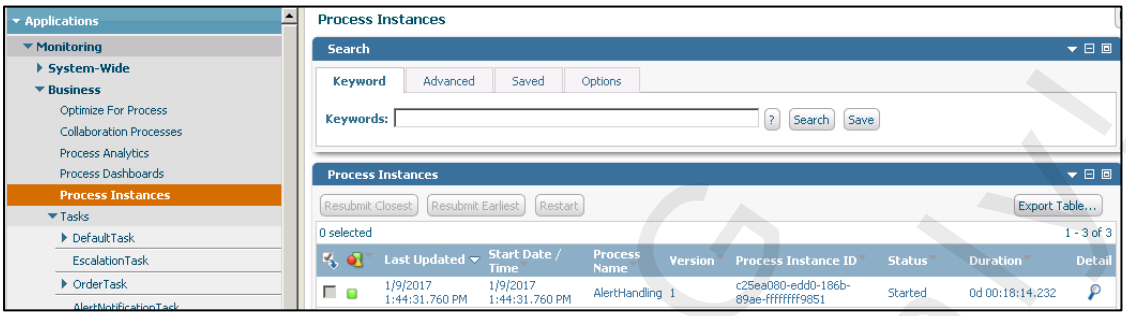
16. Test your Decision Tree and Process Action:

a. Right-click Decision Tree **PropagateAlerts** in the Rules Explorer and select **Run As > Run Decision Tree**. On the appearing panel, specify input values for all fields of **AlertBillingAddress**. Instead of typing, you can load the data from the provided sample input file **<*workshop-dir*>\Exercise13\Resources\AlertDecisionTreeInput.xml**.
Check the box beside **Include empty values for String Types** and hit **OK**.

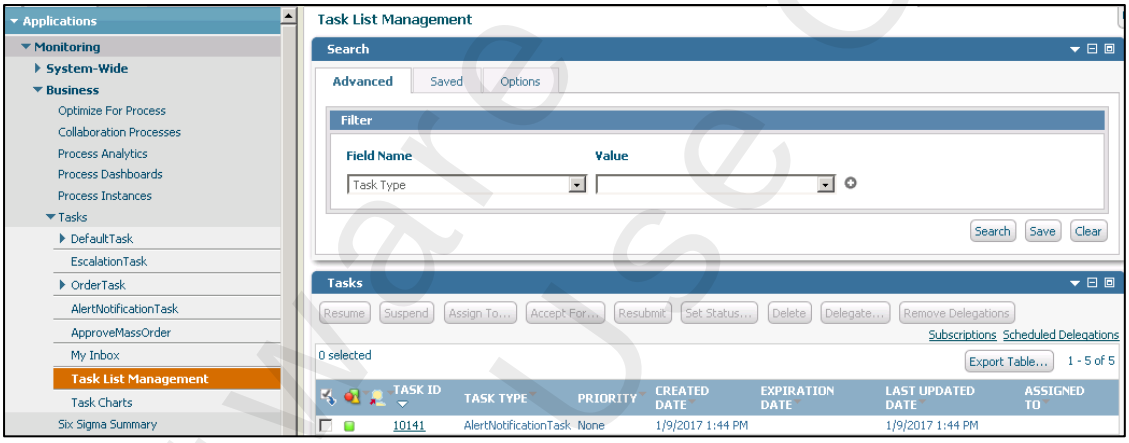


b. Open a browser tab and connect to the IS Administration console using URL http://localhost:5555. Specify **Administrator | manage** to authenticate.
Select **Logs > Server** to check whether the process has been started and if the process has logged the provided alert data:

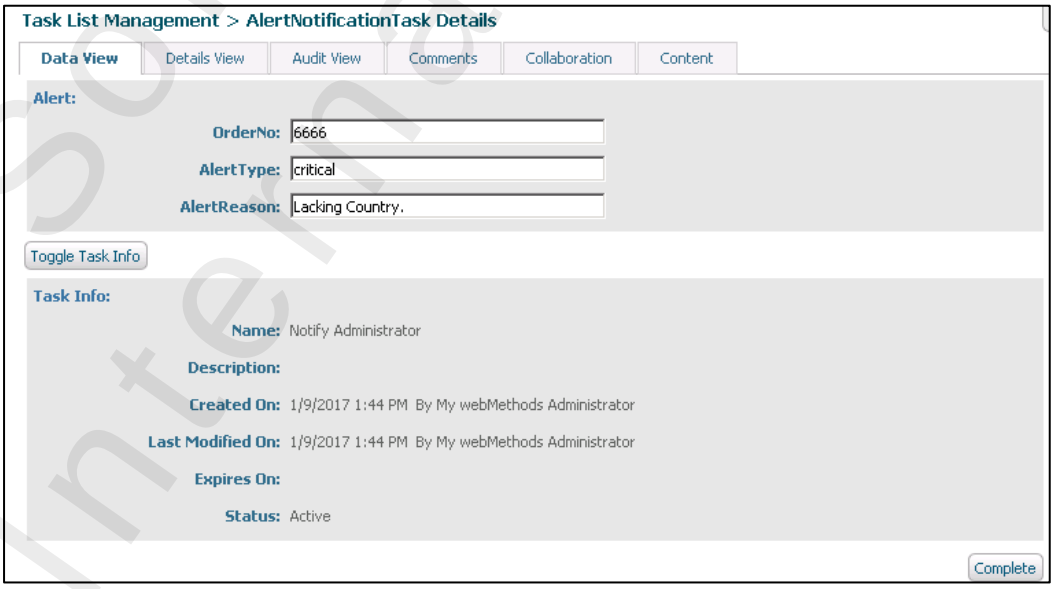
c. In a second browser tab, use URL **http://localhost:8585** to connect to My webMethods. Use **Administrator | manage** for authentication.
Drill down to **Applications > Monitoring > Business > Process Instances** to look for a process instance of type **AlertHandling** with status **Started**:



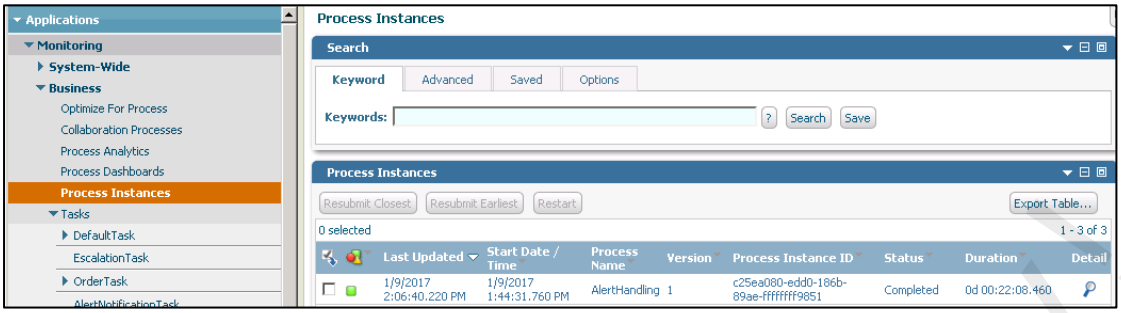
d. In My webMethods, navigate to **Applications > Monitoring > Business > Tasks > Task List Management**. Hit **Search** to refresh the Tasks list and look for a new User Task instance of User Task type **AlertNotificationTask**.



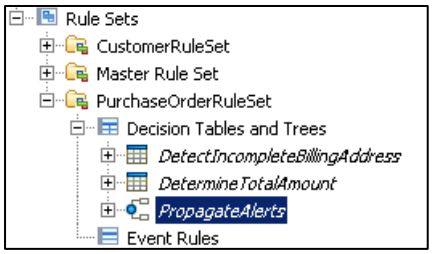
e. Select the User Task instance to open its UI. On the UI, have a look for the Alert data and complete the User Task:



f. On the Process Instances page, double-check that your process terminated successfully:

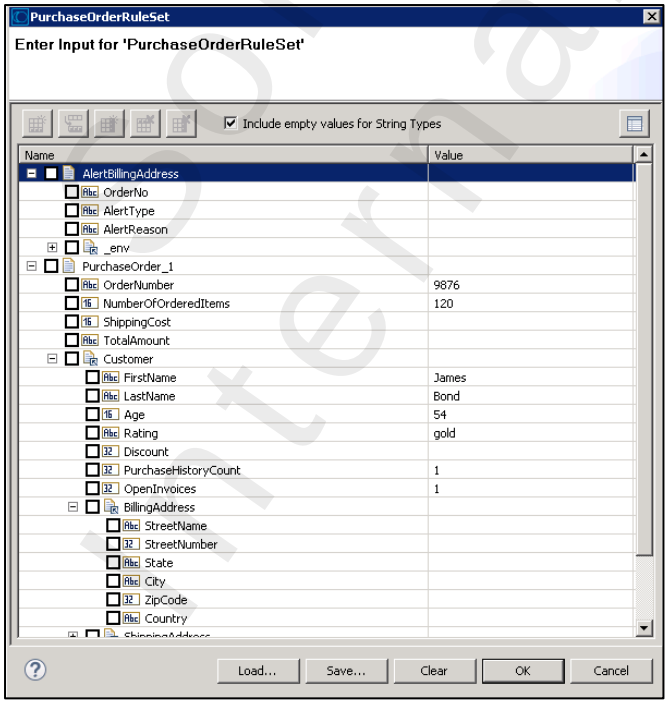


17. In Designer, right-click Decision Tree **PropagateAlerts** in the Rules Explorer view and add it to the Rule Set **PurchaseOrderRuleSet**. The Rule Set **PurchaseOrderRuleSet** has a processing mode of **Sequential**, so ensure that Decision Tree **PropagateAlerts** is executed as the last Decision Entity.



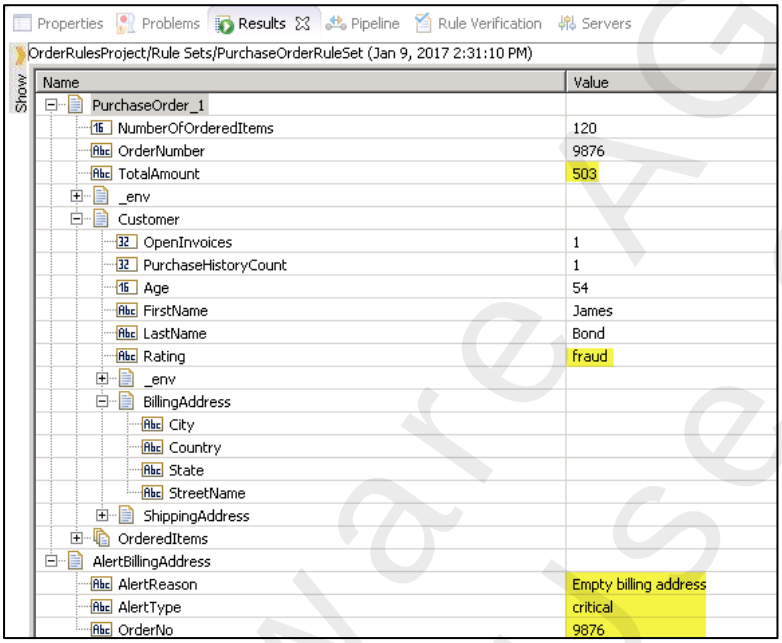
Hint: As an alternative you can also drag the Decision Tree **PropagateAlerts** in the Rules Explorer view onto the Rule Set **PurchaseOrderRuleSet** to perform this task.

18. Right-click the Rule Set **PurchaseOrderRuleSet** in the Rules Explorer view and select **Run As > Run Rule Set**. On the appearing panel provide input values for the parameter elements of **PurchaseOrder_1** including **Customer** and **OrderItems** data, but leave all fields of **BillingAddress** empty to enforce that your process action gets fired. Do NOT provide values for **AlertBillingAddress**. Instead of typing you can load the data from the provided sample input file ***<workshop-dir>*\Exercise13\Resources\PurchaseOrderRuleSetInput.xml**.
Check the box beside **Include empty values for String Types** and hit **OK**.

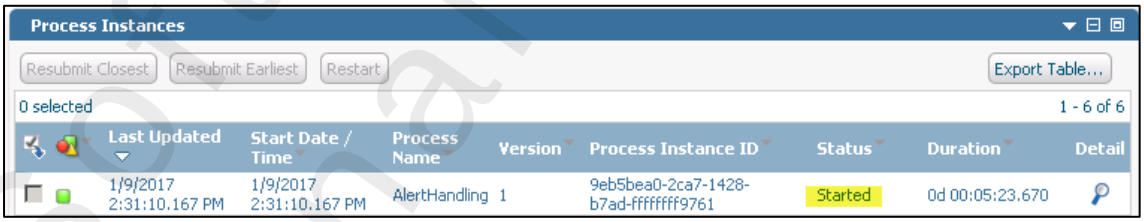


19. Check the results in the Results view and in My webMethods.
    If you used the provided sample data, the Rule Set should have

    - detected the empty BillingAddress and set the **Rating** to **fraud**
      (Decision Table **DetectIncompleteBillingAddress**)
    - created an **Alert** document
      (Decision Table **DetectIncompleteBillingAddress**)
    - calculated the **TotalAmount** of all order items
      (Decision Table **DetermineTotalAmount**)



    - started a new instance of the **AlertHandling** process
      (Decision Tree **PropagateAlerts**)



This page is intentionally left blank.

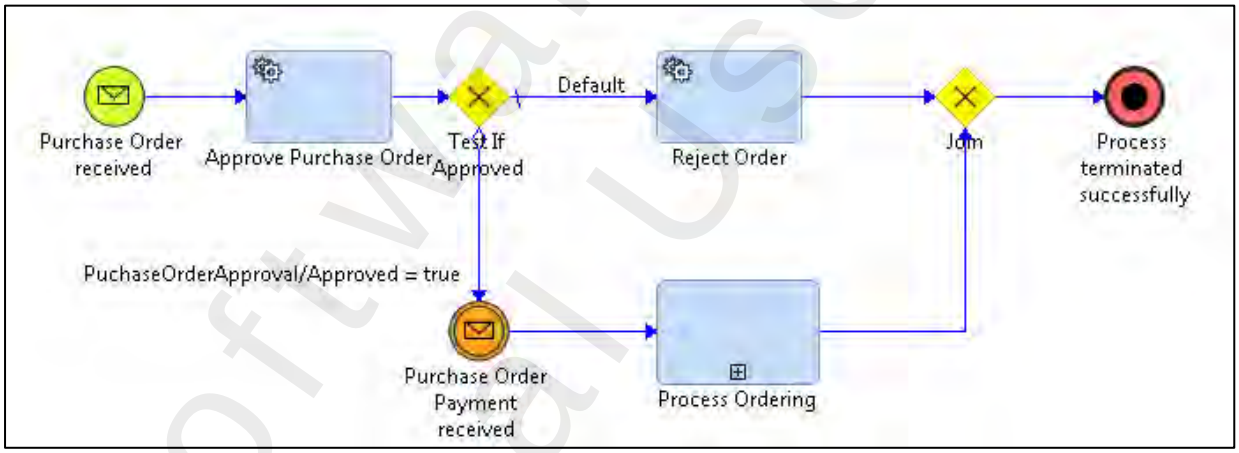
Exercise 14:
Join Process Action

Objectives

In this exercise, you will create a Process Action to join a process instance by publishing a PurchaseOrderPayment document. The action will be invoked by a Decision Table. The process has to be started before you run your Decision Table to wait for the incoming missing PurchaseOrderPayment document. Joining depends on a matching correlation ID.

Steps

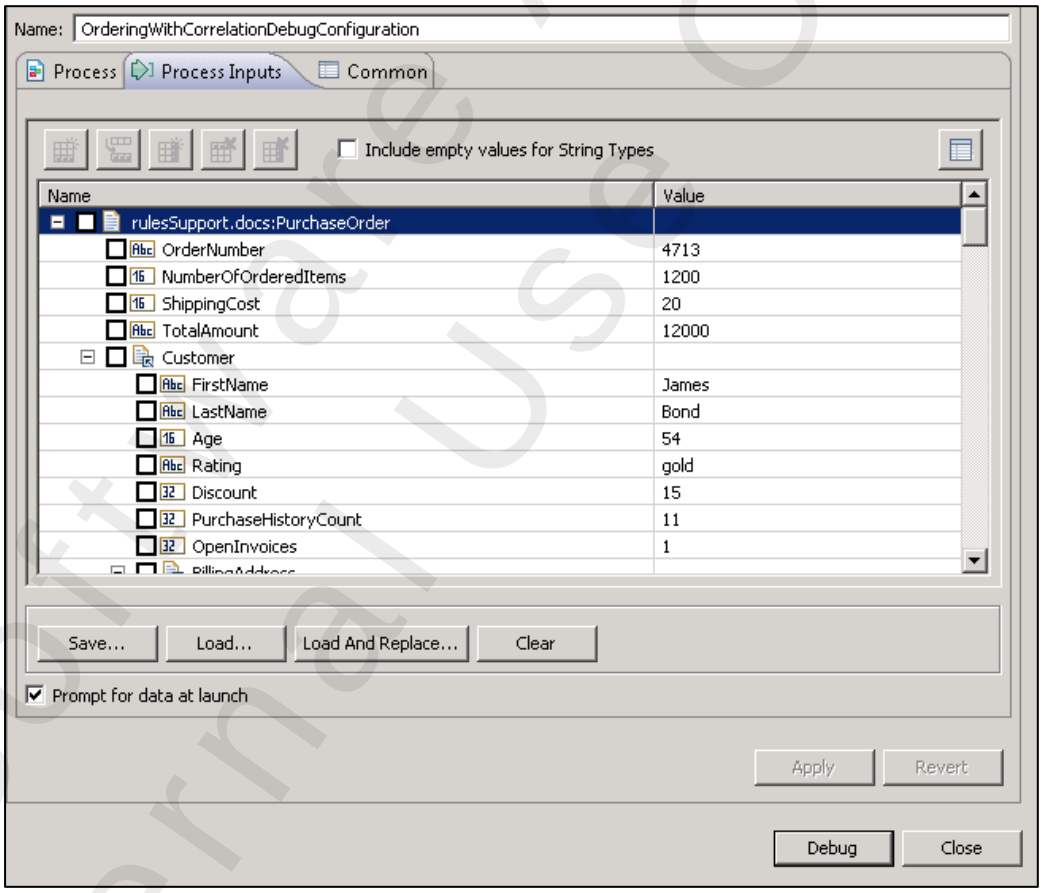
1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. Start Designer and switch to the **Process Development** perspective. Ensure you are working in **Process Developer** mode  .
4. Select **File** > **Import** > **Software AG** > **Process File** to import the Process File **<*workshop-dir*>\Exercise14\Resources\OrderingWithCorrelation.process** into your process project **OrderProcessProject**.
5. Open the process model **OrderingWithCorrelation** in the process editor. It should look like this:



Remarks: The process model receives **PurchaseOrder** documents. Incoming orders will be approved by the system (Service Task Activity **Approve PurchaseOrder**). If approved, the process waits for an additional document of type **PurchaseOrderPayment** containing additional payment data. This is implemented by an intermediate catching Message Event. Correlation is based on the order number contained in both documents. Moreover, correlation is configured with a timeout of 10 minutes. Finally, subprocess **Process Ordering** is performed before the workflow terminates successfully.

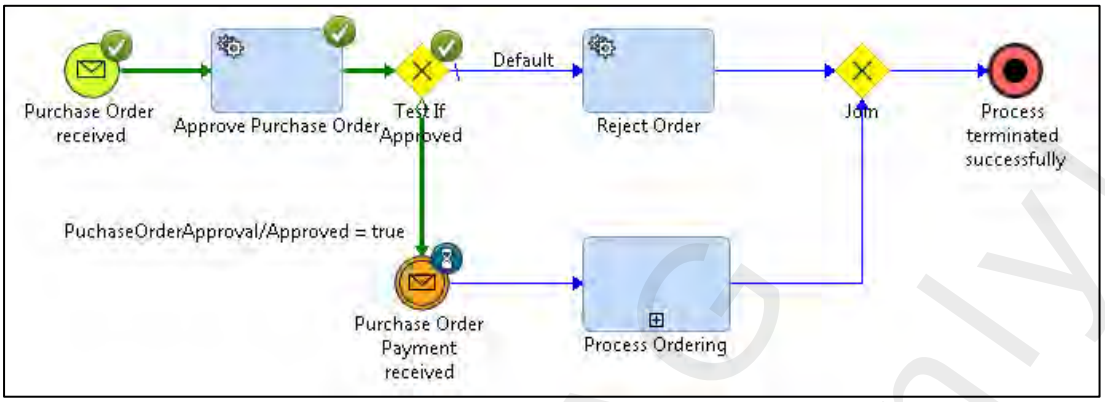
6. Build and upload process model **OrderingWithCorrelation**. If asked, enable the process for execution.
7. Right-click Document Type **rulesSupport.docs:PurchaseOrder** in the Package Navigator view and select **Sync Document Type** to push the document type to the Messaging Provider. Do the same for Document Type **rulesSupport.docs:PurchaseOrderPayment**.

8. Test the process using the Process Debugger in Designer:
    a. Switch to the **Process Debug** perspective.
    b. As done in exercise 11, create, configure and run a Debug Configuration for the OrderingWithCorrelation process:
        i. On the **Process** tab of your new Debug Configuration, specify **OrderingWithCorrelationDebugConfiguration** as Run Configuration name and select process model **OrderingWithCorrelation** within process project **OrderProcessProject**.
        ii. On the **Process Inputs** tab, provide input data for the Start Message Event and check to get prompted to type/alter those data when debugging starts.
        Instead of typing you can load input data from:
        <*workshop-dir*>\Exercise14\Resources\OrderingWithCorrelationDebugConfigInput.xml
        iii. Click **Apply** to save the Debug Configuration.
        iv. Then click **Debug** to start process debugging.



    c. Because of the settings you made in the Debug Configuration you get prompted for the input data having your data specified as in the Debug Configuration preloaded. (Same) sample input can optionally be loaded from:
    <*workshop-dir*>\Exercise14\Resources\OrderingWithCorrelationDebugInput.xml
    Check the box beside **Include empty values for String Types** and click **OK**.

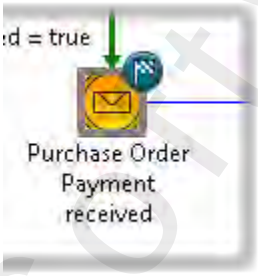
d. Use the Trace view to step over (F6) all activities until step **Join and Purchase Order Payment received** are in a waiting status.



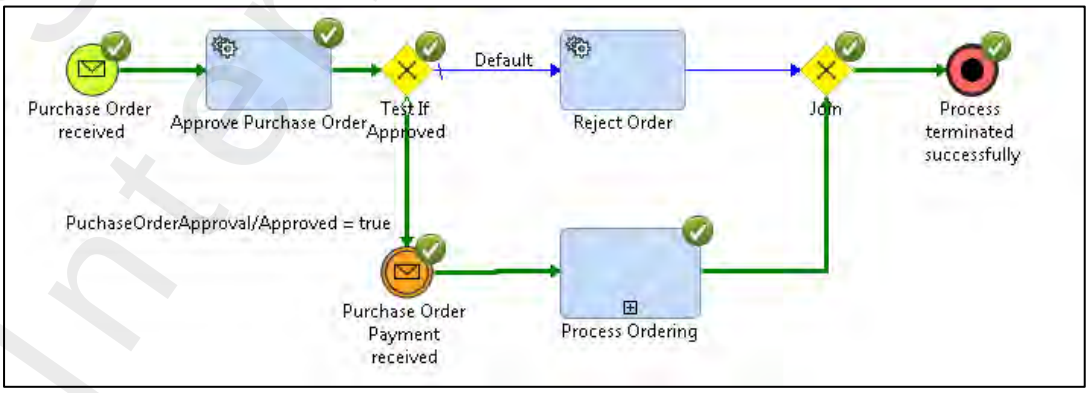
e. To provide input for the waiting intermediate Message Event **Purchase Order Payment received**, switch temporary to the **Process Development** perspective. In the Package Navigator view, right-click **rulesSupport.docs:PurchaseOrderPayment** and select **Run As > Publishable Document**. Key in or load sample input from file **<*workshop-dir*>\Exercise14\Resources\PurchaseOrderPaymentInput.xml**.

> *Important*: To allow correlation, the value specified in your input data for field **OrderNumber** of the PurchaseOrderPayment document must match the value you specified for field **OrderNumber** in your PurchaseOrder document input from above.

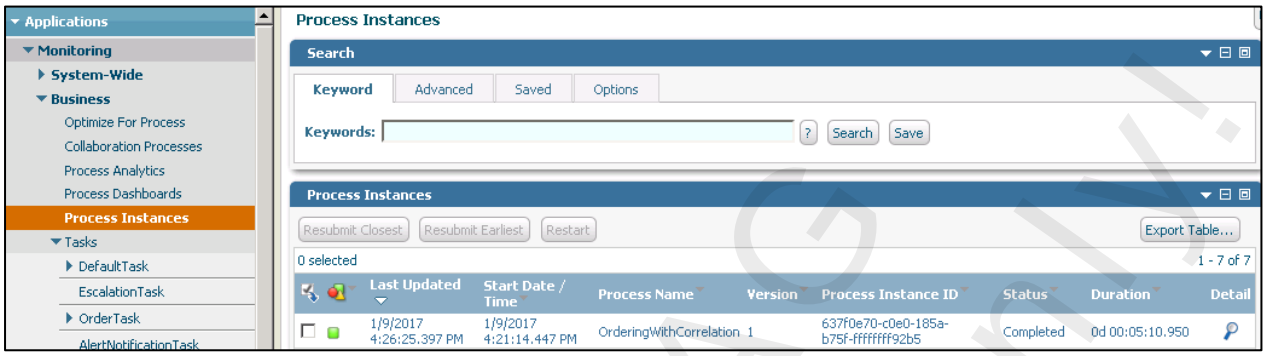
f. Switch back to the **Process Debug** perspective.
Note that the status of the intermediate Message Event **Purchase Order Payment received** changed.



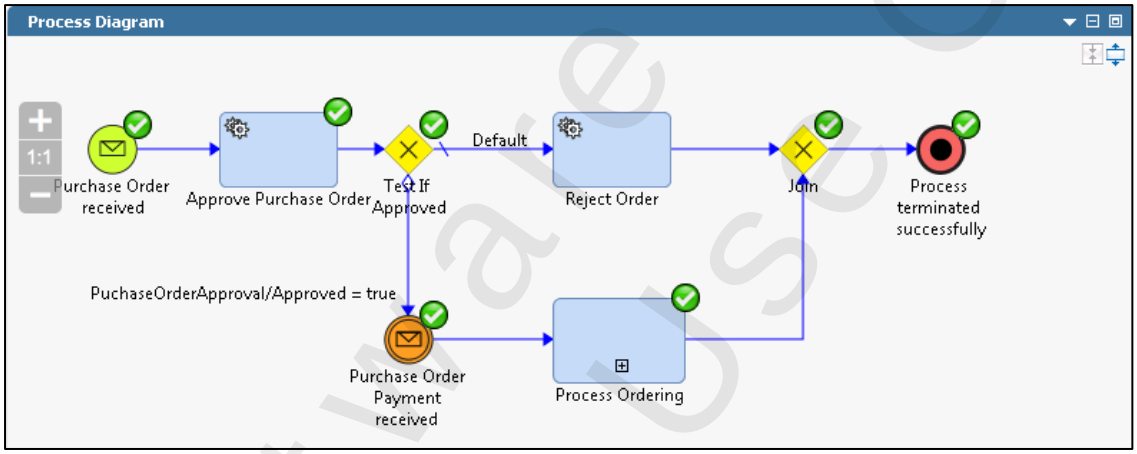
Execute the residual steps in Debugger until the process terminates successfully.



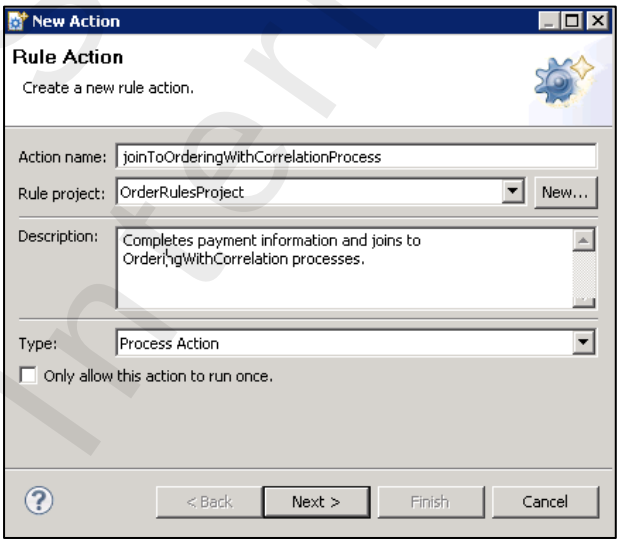
9. Open a browser tab, use URL http://localhost:8585 to connect to My webMethods. Use **Administrator| manage** for authentication.
   Drill down to **Applications** > **Monitoring** > **Business** > **Process Instances**. Search and look for the (debugged) process instance of type **OrderingWithCorrelation**. Confirm it has completed successfully.



   Click on the magnifying glass and open the process diagram. Inspect the used trail:



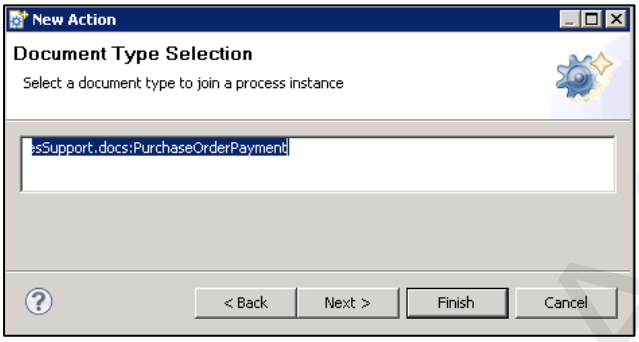
10. Switch back to Designer and open the **Rules Development** perspective.
11. Use the Rules Explorer view to add another Action:
    a. Create an Action of type **Process Action** in your Rule project **OrderRulesProject**. Name it **joinToOrderingWithCorrelationProcess** and provide a description text as shown on the screen shot below. Click **Next.**



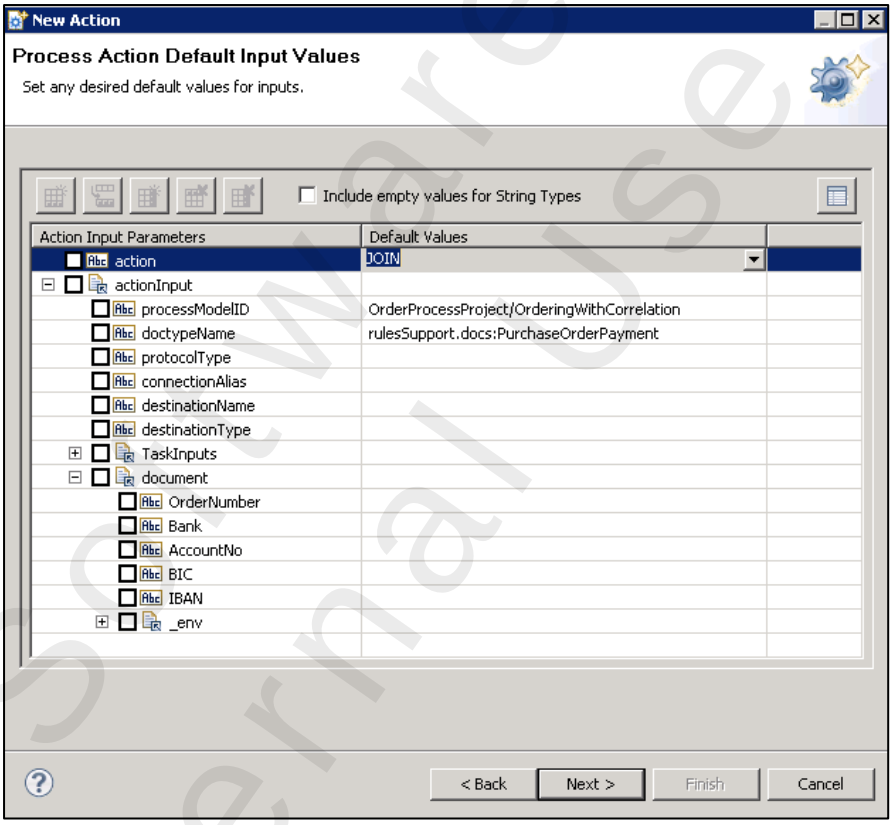
b. On the next panel, choose Process Action type **Join a running process instance**. Click **Next**.

c. On the subsequent panel, select process model **OrderingWithCorrelation**. Click **Next**.

d. On the next panel, select document type **rulesSuport.docs:PurchaseOrderPayment** to be published for process joining. Click **Next**.



e. Leave the Process Action Default Values unchanged and check the box beside **Include empty values for String Types**. Click **Next**.

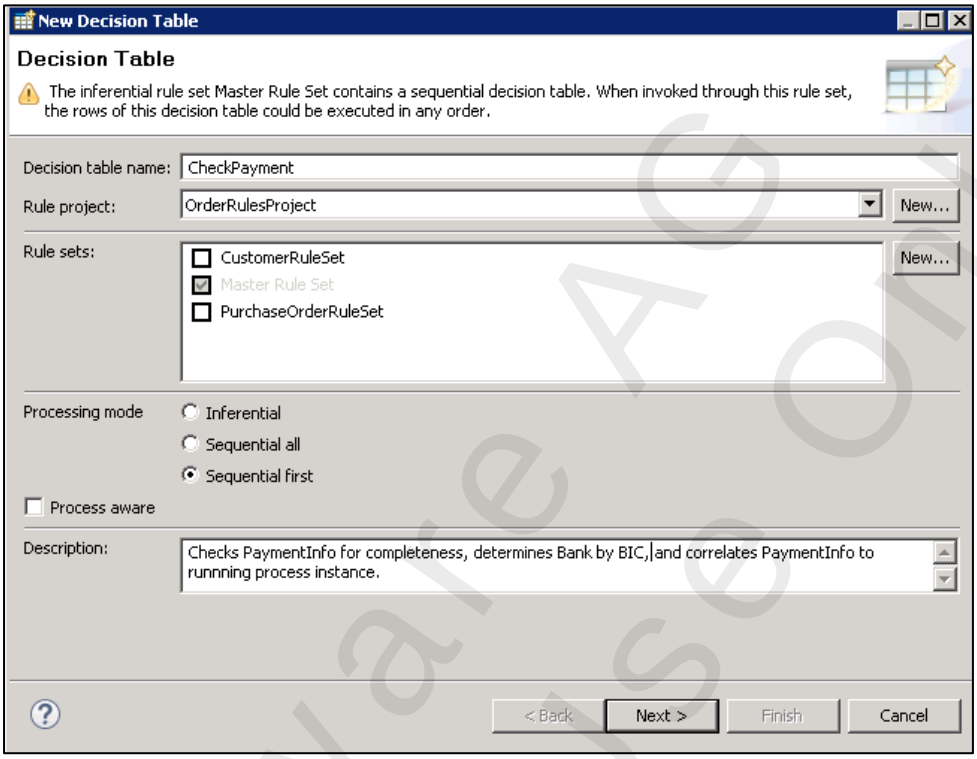


f. On the last panel, check **outputMessage** as action return value. Click **Finish** to create the action.

12. Use the Rules Explorer view to create a Data Model named **PurchaseOrderPayment** in your Rule Project **OrderRulesProject**. Select **rulesSupport.docs:PurchaseOrderPayment** as related Document Type. Click **Finish**. Because the **PurchaseOrderPayment** document also contains **pub.publish_envelope.datamodel**, click **Select All** on the appearing pop up to allow overwriting of existing Data Models in your project.

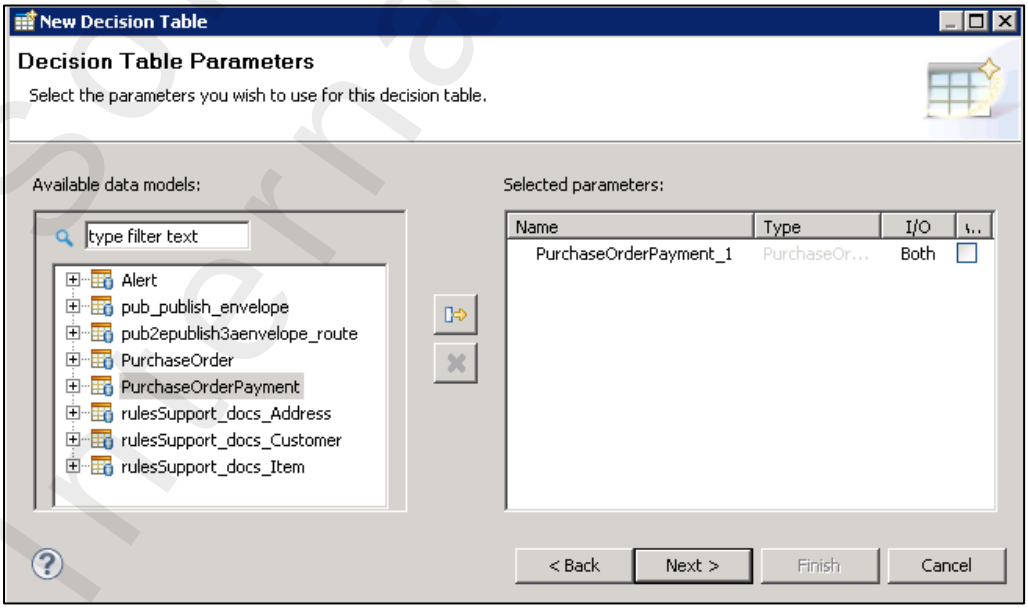
13. Use the Rules Explorer view to add another Decision Table:

    a. Add a Decision Table named **CheckPayment** to your **OrderRulesProject** project. Do NOT add this Decision Table to any Rule Set (except for the Master Rule Set).
    Select Processing mode **Sequential first** and provide a description text as shown on the screen shot below. Click **Next**.

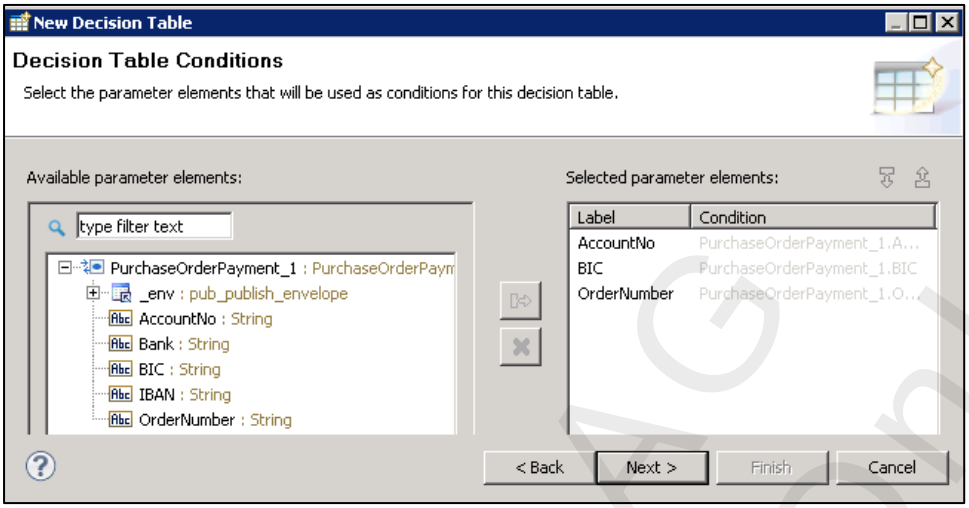


Note: Accept the warning about the inferential Processing mode of the Master Rule Set.

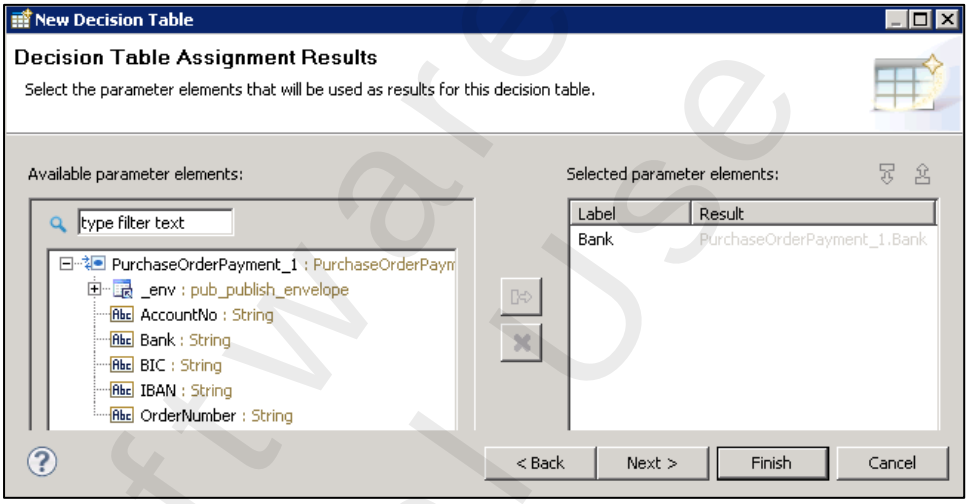
    b. On the subsequent panel, select Data Model **PurchaseOrderPayment** as Decision Table parameter with I/O type **Both**. Keep the default parameter name and click **Next**.



c. On the next panel, select the parameter elements **AccountNo**, **BIC**, and **OrderNumber** to be used in your rule conditions. Click Next.



d. On the following panel, select **Bank** as result parameter element:



e. Click **Finish** to complete your Decision Table structure.

14. Use the Decision Table Editor for your Decision Table **CheckPayment** to insert five rules. Your Decision Table should look like this:

Description

Checks PaymentInfo for completeness, determines Bank by BIC, and correlates PaymentInfo to runnning process instance.

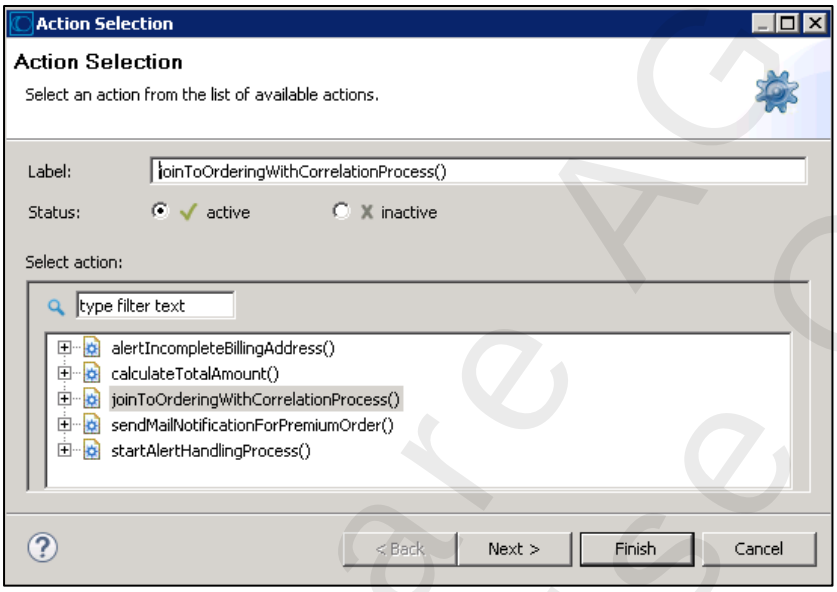
	AccountNo	BIC	OrderNumber	Bank
1	!= EMPTY STRING	= 4711US11	!= EMPTY STRING	= Leeman Sisters
2	!= EMPTY STRING	= 4712UK22	!= EMPTY STRING	= UMG DuNa
3	!= EMPTY STRING	= 4713LP33	!= EMPTY STRING	= Royal Bank of Lampukistan
4	!= EMPTY STRING	= 4714DE44	!= EMPTY STRING	= Willi Wuff Bank24
5	!= EMPTY STRING	= 4715FR55	!= EMPTY STRING	= Crazy Credits

15. Save and test your Decision Table **CheckPayment**. As sample input you can load: **<*workshop-dir*>\Exercise14\Resources\CheckPaymentDecisionTableInput.xml**

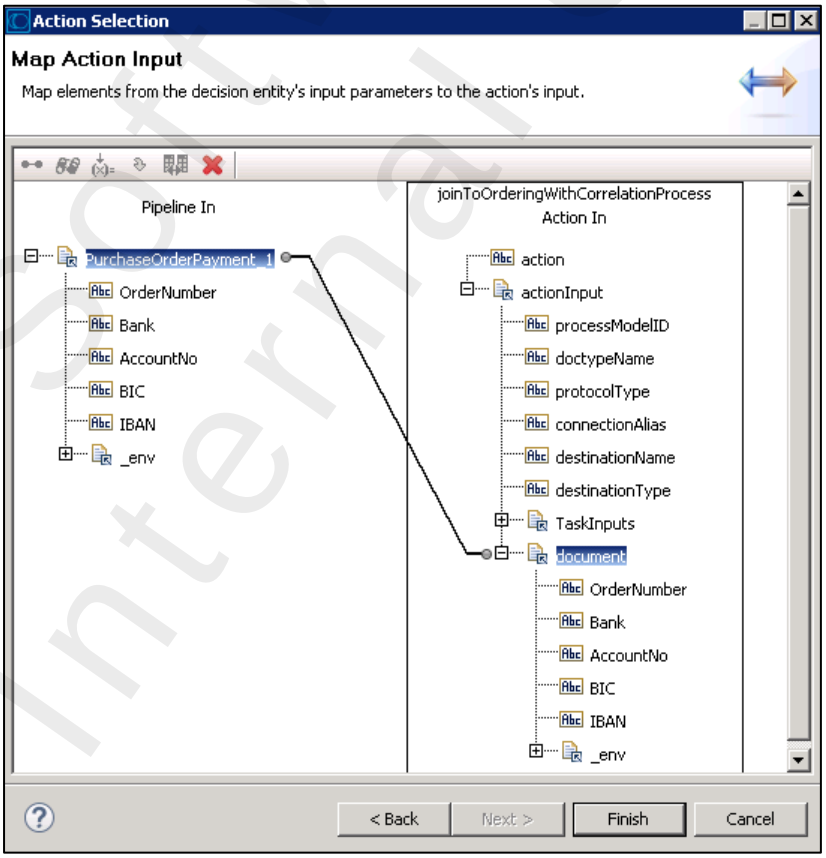
16. Enhance your Decision Table **CheckPayment** by an Action result column:

    a. To do so, drag Action **joinToOrderingWithCorrelationProcess** from the Rules Explorer next to column **Bank** in the Decision table Editor pane.

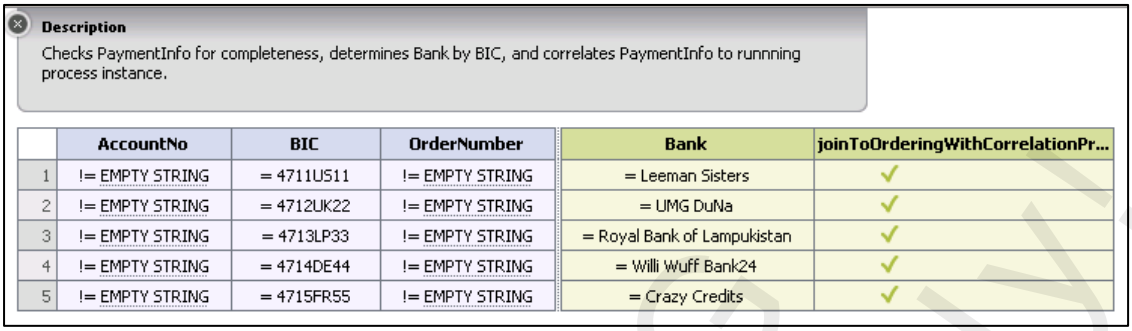
    b. In the appearing wizard, confirm the pre-selected action and select **active** as default for all rules. Click **Next**.



    c. Map the Decision Table parameter **PurchaseOrderPayment_1** to the parameter document of your joining Process Action. Finally click **Finish**.



Your Decision Table **CheckPayment** should now look like this:



Description
Checks PaymentInfo for completeness, determines Bank by BIC, and correlates PaymentInfo to runnning process instance.

	AccountNo	BIC	OrderNumber	Bank	joinToOrderingWithCorrelationPr...
1	!= EMPTY STRING	= 4711US11	!= EMPTY STRING	= Leeman Sisters	✓
2	!= EMPTY STRING	= 4712UK22	!= EMPTY STRING	= UMG DuNa	✓
3	!= EMPTY STRING	= 4713LP33	!= EMPTY STRING	= Royal Bank of Lampukistan	✓
4	!= EMPTY STRING	= 4714DE44	!= EMPTY STRING	= Willi Wuff Bank24	✓
5	!= EMPTY STRING	= 4715FR55	!= EMPTY STRING	= Crazy Credits	✓

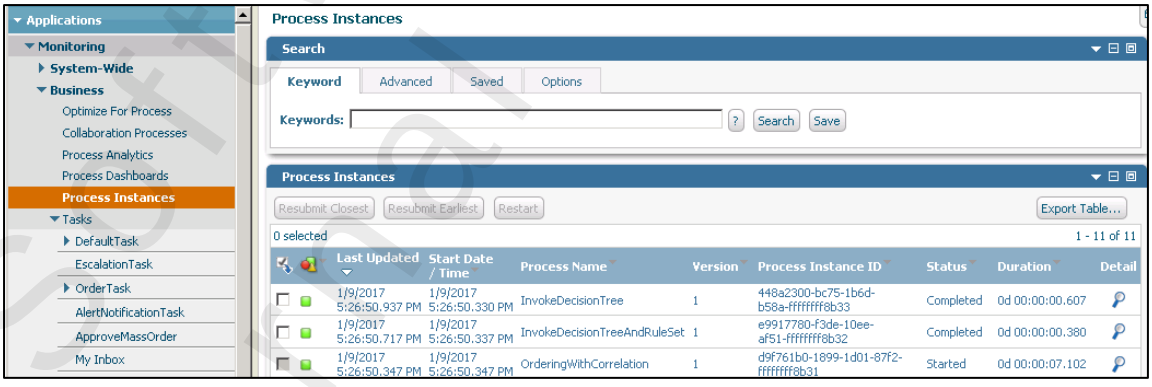
d. Save your Decision Table.

17. Re-deploy (export) your Rule Project to your Integration Server runtime and MWS Repository.
18. Test your Decision Table and Process Action:

a. Use the Package Navigator view and right-click the Document Type **rulesSupport.docs:PurchaseOrder**. Choose **Run As > Publishable Document**.
Load sample input data from file **<*workshop-dir*>\Exercise14\Resources\PurchaseOrderInput.xml.**
Do NOT check the box beside **Include empty values for String Types** and hit **OK.**

Important: For each test cycle, always use a different value for field **OrderNumber**.

b. Switch back to My webMethods still opened in your browser tab. Drill down to **Applications** > **Monitoring** > **Business** > **Process Instances**. Search and look for a new process instance of type **Invoke Decision Table**. Confirm it is in status **Started**.



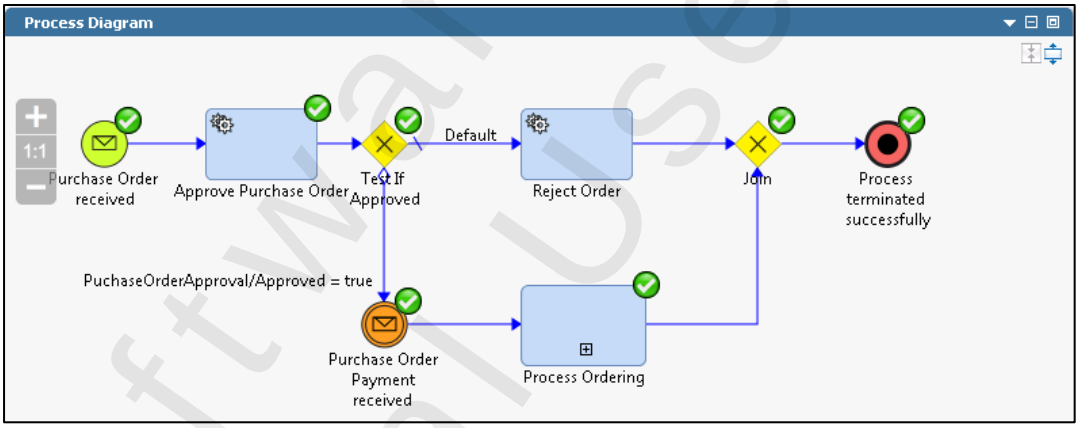
Note: As process models of type **InvokeDecisionTable** and **InvokeDecisionTableAndRuleSet** also have a subscription to Document Type **PurchaseOrder**, another two processes have been started (and completed successfully).

c. Go back to Designer. Switch to the **Rules Development** perspective.
Use the Rules Explorer view and run your Decision Table **CheckPayment**. You can use the provided sample input data as provided in:
<*workshop-dir*>\Exercise14\Resources\CheckPaymentDecisionTableInput.xml
Do NOT check the box beside **Include empty values for String Types** and hit **OK.**

> *Important*: To allow correlation, you must assign the same value for field **OrderNumber** as provided in step 18a.

d. Switch back to My webMethods in your browser tab. Refresh the Process Instances page to look for the status of your process instance of type **OrderingWithCorrelation**. Confirm it is in status **Completed** now.

e. Open the Process Diagram and inspect the used trail:



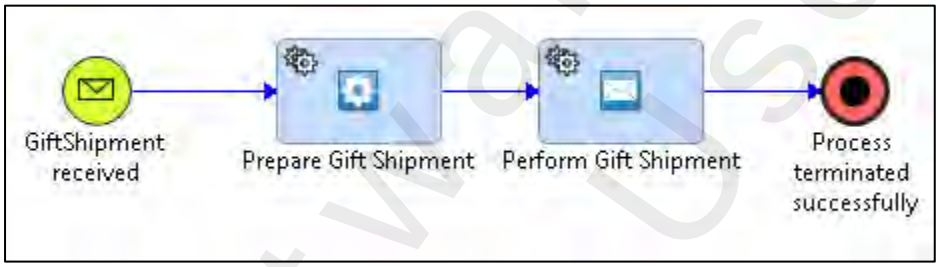
Exercise 15:
Cancel Process Action

Objectives

In this exercise, you will create a Process Action that cancels running process instances of a certain model type and a matching condition based on logged process data fields. The action will be invoked conditionally by a Decision Table.

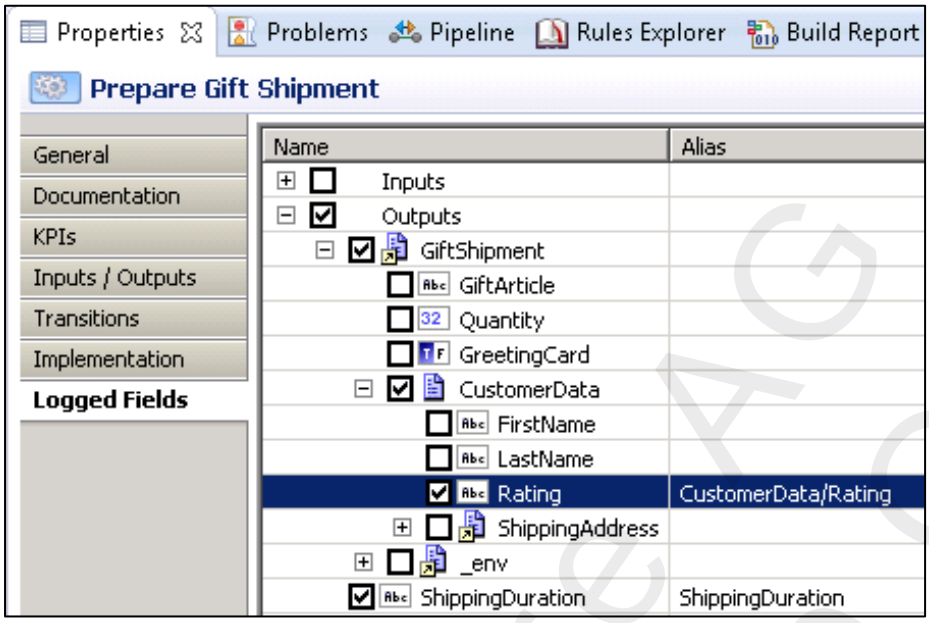
Steps

1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. Start Designer and switch to the **Process Development** perspective. Ensure you are working in **Process Developer** mode  .
4. Select **File** > **Import** > **Software AG** > **Process File** to import the Process File **<*workshop-dir*>\Exercise15\Resources\GiftShipment.process** into your process project **OrderProcessProject**.
5. Open the process model **GiftShipment** in the process editor. It should look like this:

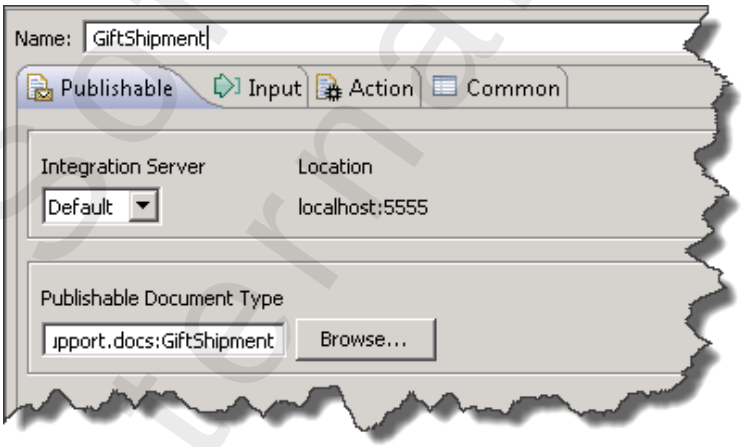


Remarks: The process model receives **GiftShipment** documents. Shipment is performed by two Service Task Activities named **Prepare Gift Shipment** and **Perform Gift Shipment**. Activity **Prepare Gift Shipment** invokes the existing flow service **rulesSupport.services:prepareGiftShipment**, activity **Perform Gift Shipment** invokes the flow service **rulesSupport.services:performGiftShipment**. Because of the (intended) internal implementation, the execution of Activity Perform Gift Shipment will take between 2 and 10 minutes to complete, giving you a long running process instance.

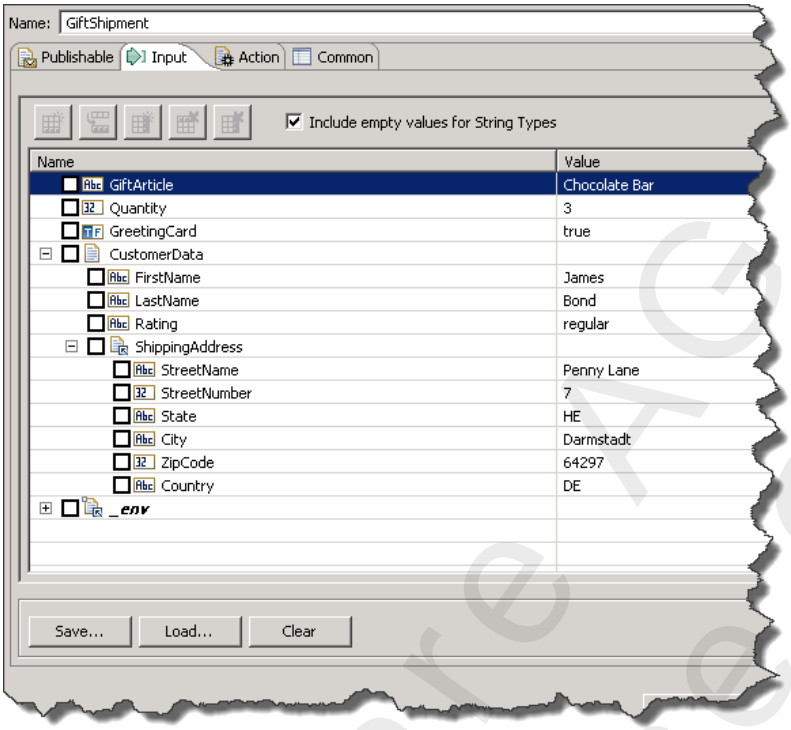
6. Select the Service Task Activity **Prepare Gift Shipment** in the process editor and open the Logged Fields tab in its Properties view. Select **GiftShipment/CustomerData/Rating** and **ShippingDuration** as logged Output fields.



7. Save, build and upload the process model **GiftShipment**. If asked, enable the process for execution.
8. Right-click rulesSupport.docs:GiftShipment in the Package Navigator view and select **Sync Document Type** to push the document type to the Messaging Provider.
9. Test the **GiftShipment** process:
    a. Right-click **rulesSupport.docs:GiftShipment** in the Package Navigator view and select **Run As > Run Configurations…** . Click  to create a new Run Configuration.
    b. On the **Publishable** tab, browse for **rulesSupport.docs:GiftShipment** as Publishable Document Type and specify **GiftShipment** as configuration name.



c. On the **Input** tab, you can load the provided sample input from file **<*workshop-dir*>\Exercise15\Resources\GiftShipmentConfigInput.xml**. Check the box beside **Include empty values for String Types**.



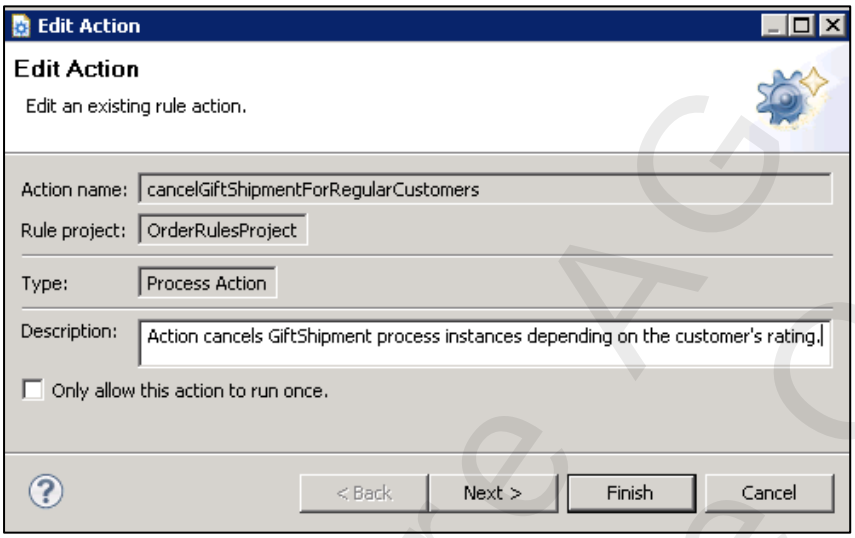
d. On the **Action** tab, select **Publish to the Provider** and hit **Apply**.

e. Finally click **Run** to publish a document instance and to start a process instance.

10. Use a browser tab to logon to My webMethods (https://localhost:8585) as **Administrator | manage**. Go for **Applications** > **Monitoring** > **Business** > **Process Instances**. You should see an instance of your **GiftShipment** process instance. Monitor the status of started process in the process instances. The execution will last at least 2 minutes and may last 10 minutes as a maximum.

11. Go back to Designer and switch to the **Rules Development** perspective.
12. Use the Rules Explorer view to add another Action:

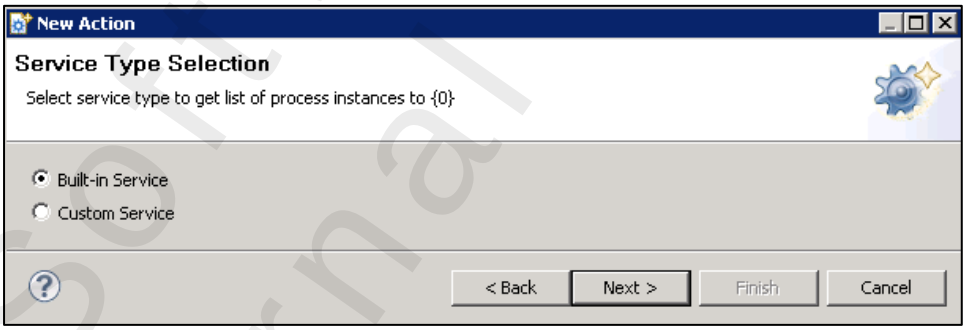
    a. Create an Action of type **Process Action** in your Rule project **OrderRulesProject**. Name it **cancelGiftShipmentForRegularCustomers** and provide a description text as shown on the screen shot below. Click **Next**.



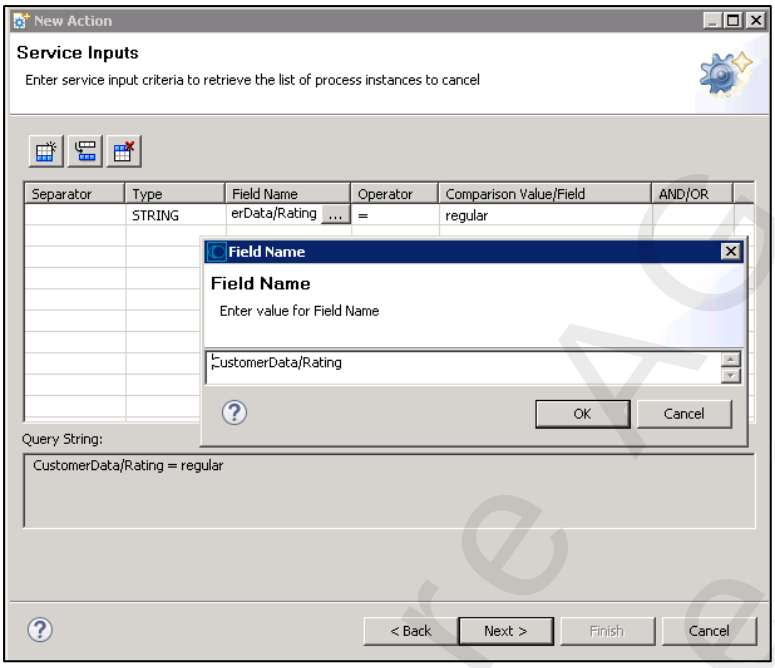
    b. On the next panel, choose Process Action type **Cancel running process instance(s)**. Click **Next**.

    c. On the subsequent panel, select process model **GiftShipment**. Click **Next**.

    d. On the next panel, select **Built-in Service** as service type to retrieve the list of process IDs to be canceled at runtime. Click **Next**.

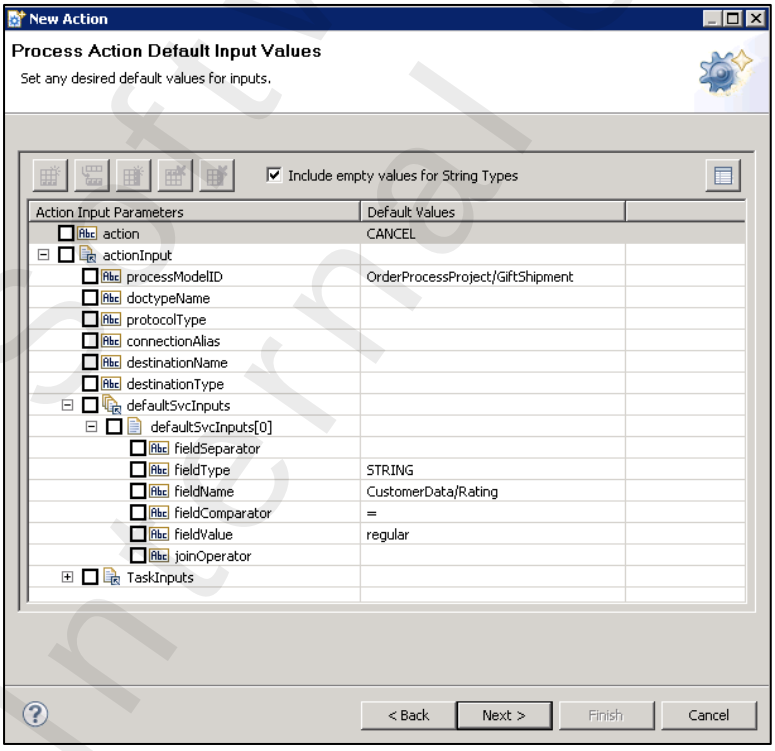


e. On the Service Inputs panel for the built-in service, click on the Add icon to create a filter criteria for processes to be canceled. The action should only cancel process instances of process model GiftShipment with value of **regular** in the logged field **CustomerData/Rating.** Your query should look like this:



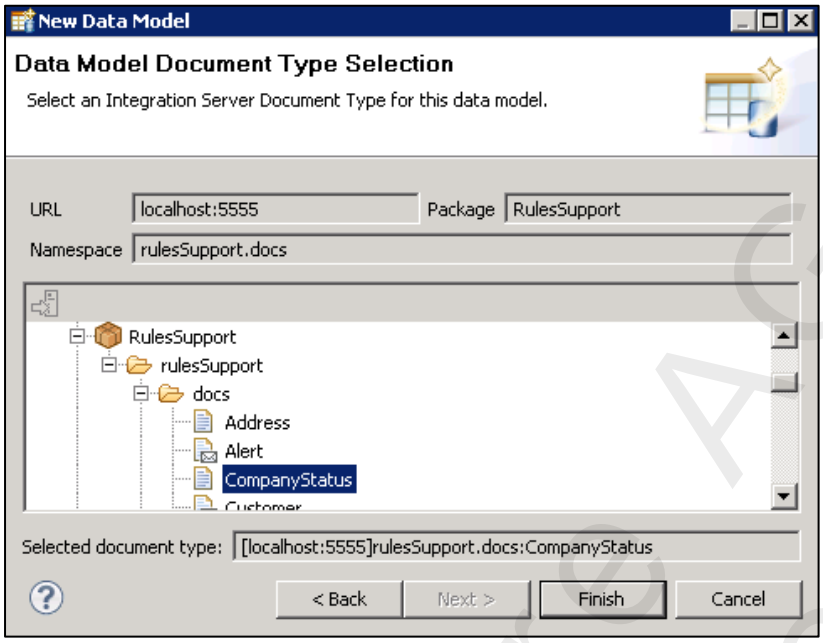
Click **Next**.

f. On the Process Action Default Input Values panel, leave all default values and check the box beside **Include empty values for String Types**. Click **Next**.



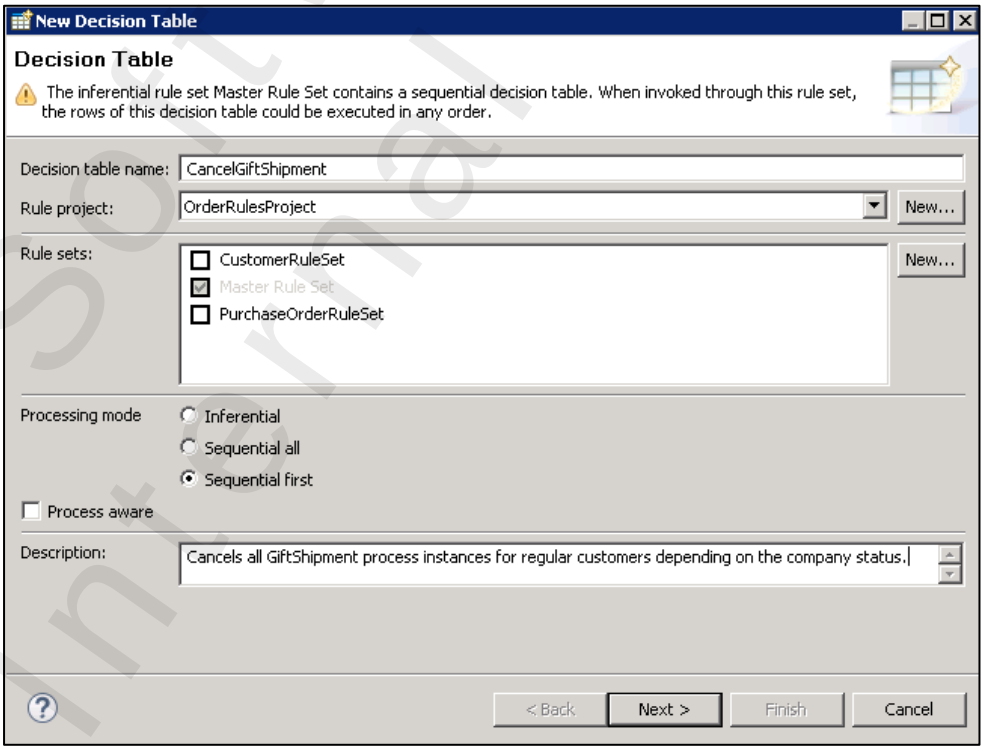
g. On the last panel, check **outputMessage** as action return value. Click **Finish** to create the action.

13. Use the Rules Explorer view to add a Data Model named **CompanyStatus** in your Rule Project **OrderRulesProject**. Select **rulesSupport.docs:CompanyStatus** as related Document Type. Click **Finish**.

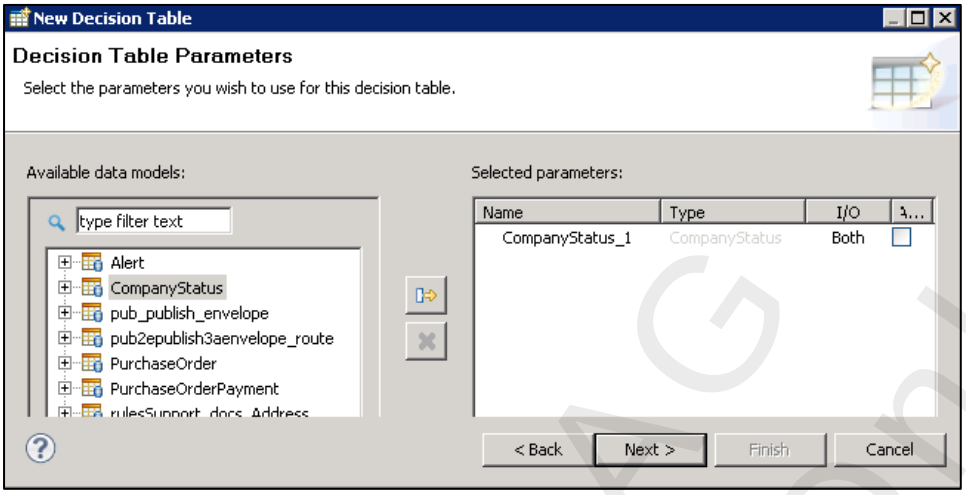


14. Use the Rules Explorer view to add another Decision Table:

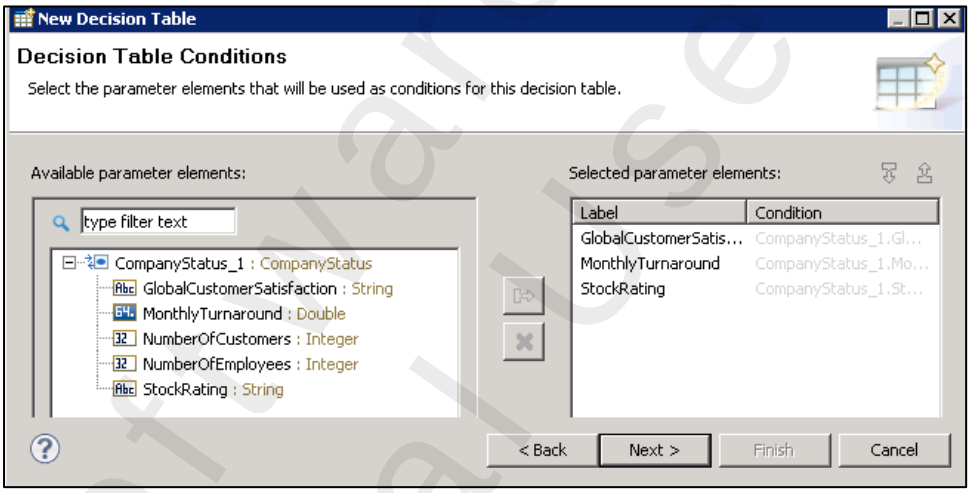
    a. Add a Decision Table named **CancelGiftShipment** to your **OrderRulesProject** project. Do NOT add this Decision Table to any Rule Set yet (except for the Master Rule Set). Select Processing mode **Sequential first** and provide a description text as shown on the screen shot below. Click **Next**.



b. On the subsequent panel, select Data Model **CompanyStatus** as Decision Table parameter with I/O type **Both**. Click **Next**.

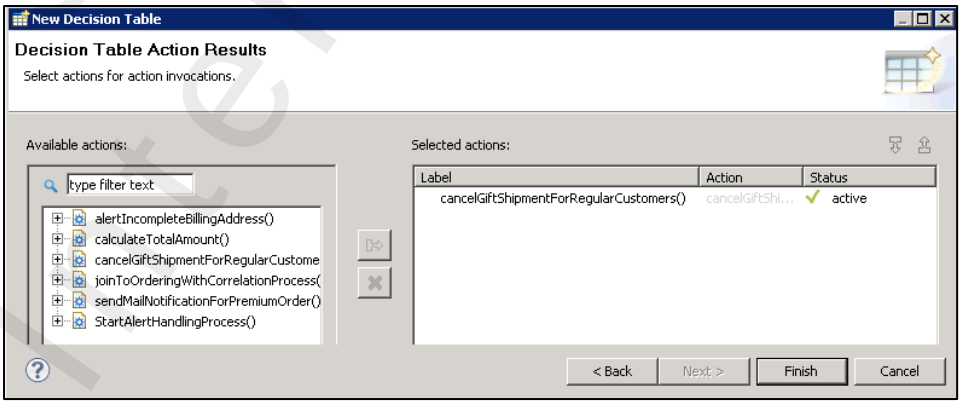


c. On the next panel, select the parameter elements **GlobalCustomerSatisfaction**, **MonthlyTurnaround**, and **StockRating** to be used in your rule conditions. Click **Next**.



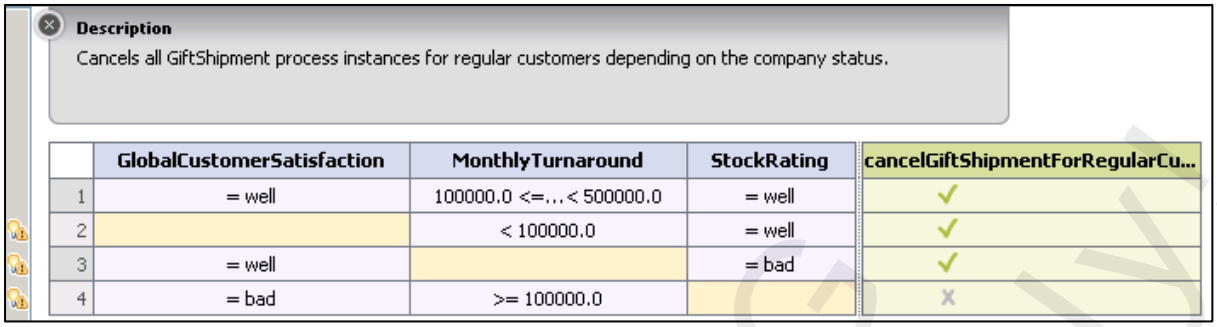
d. On the following panel, click **Next** to skip specifying Decision Table Assignment Results.

e. On the last panel select action **cancelGiftShipmentForRegularCustomers** as a result action with default status **active** for all rules.



f. Click **Finish** to complete your Decision Table structure.

15. Use the Decision Table Editor for your Decision Table **CancelGiftShipment** to insert four rules. Your Decision Table should look like this:



Description

Cancels all GiftShipment process instances for regular customers depending on the company status.

	GlobalCustomerSatisfaction	MonthlyTurnaround	StockRating	cancelGiftShipmentForRegularCu...
1	= well	100000.0 <=...< 500000.0	= well	✓
2		< 100000.0	= well	✓
3	= well		= bad	✓
4	= bad	>= 100000.0		✗

16. Suppress warnings for empty cells and save your Decision Table.

17. Re-deploy (export) your Rule Project to your Integration Server runtime and MWS Repository.

18. Start a least four **GiftShipment** process instances.

> *Important*: Ensure that at least half of them have been started with a customer rating equals **regular**.

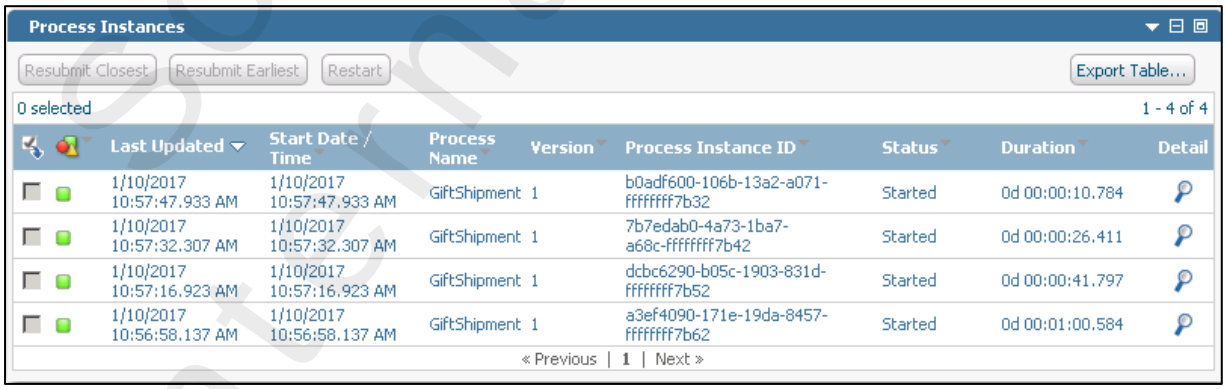
To do so:

a. Right-click **rulesSupport.docs:GiftShipment** in the Package Navigator view and select **Run As** > **Publishable Document**.

b. For launching you can reuse and modify the provided sample input as configured in your Run Configuration.

c. Repeat steps a) and b) three more times.

19. Revisit the Process Instances page in My webMethods. Select Search to refresh the process instances list. Ensure all your process instances have a status of **Started**.



Process Instances

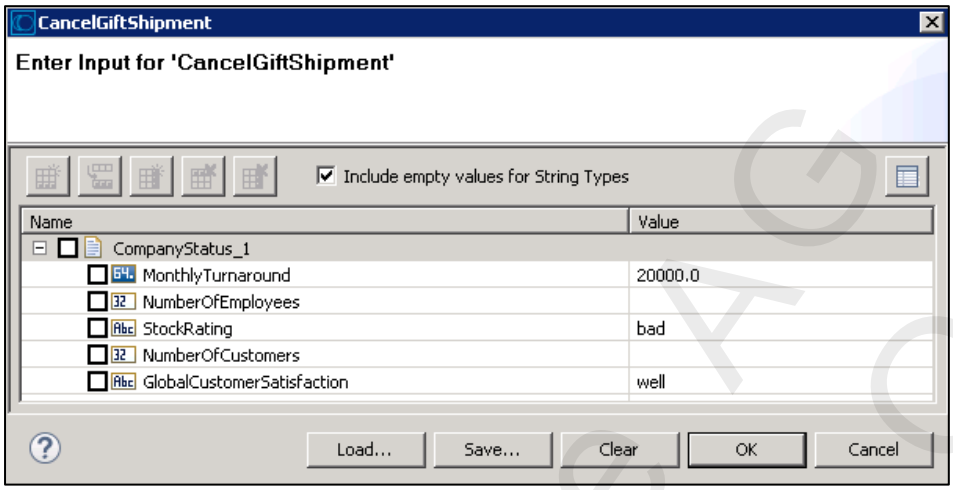
Resubmit Closest | Resubmit Earliest | Restart | Export Table...

0 selected | 1 - 4 of 4

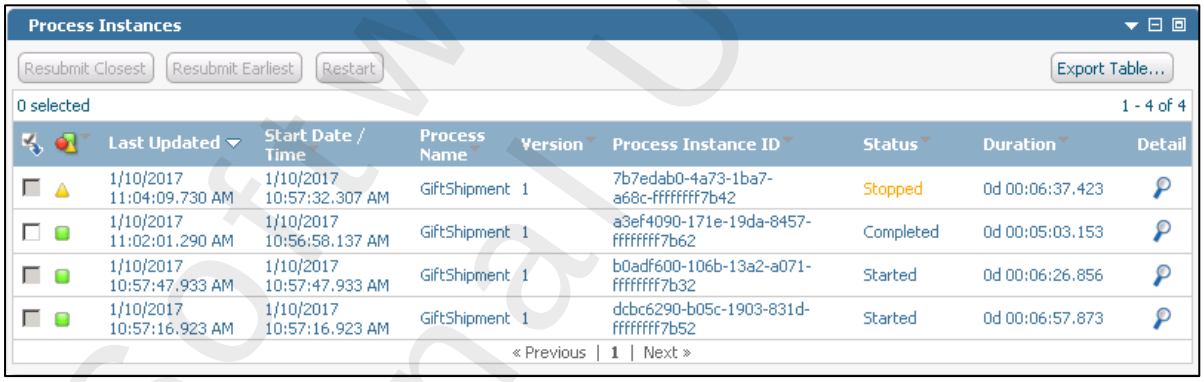
Last Updated	Start Date / Time	Process Name	Version	Process Instance ID	Status	Duration	Detail
1/10/2017 10:57:47.933 AM	1/10/2017 10:57:47.933 AM	GiftShipment	1	b0adf600-106b-13a2-a071-ffffffff7b32	Started	0d 00:00:10.784	
1/10/2017 10:57:32.307 AM	1/10/2017 10:57:32.307 AM	GiftShipment	1	7b7edab0-4a73-1ba7-a68c-ffffffff7b42	Started	0d 00:00:26.411	
1/10/2017 10:57:16.923 AM	1/10/2017 10:57:16.923 AM	GiftShipment	1	dcbc6290-b05c-1903-831d-ffffffff7b52	Started	0d 00:00:41.797	
1/10/2017 10:56:58.137 AM	1/10/2017 10:56:58.137 AM	GiftShipment	1	a3ef4090-171e-19da-8457-ffffffff7b62	Started	0d 00:01:00.584	

« Previous | 1 | Next »

20. Right-click Decision Table **CancelGiftShipment** in the Rules Explorer and select **Run As** > **Run Decision Table**. On the appearing panel, specify some input values that will enforce the execution of the process action **cancelGiftShipmentForRegularCustomers** as a result of a fired rule. As sample input you can load input form file **<*workshop-dir*>\Exercise15\Resources\CancelGiftShipmentDecisionTableInput.xml**. Check the box beside **Include empty values for String Types** and click **OK**.



21. Switch back to My webMethods in your browser tab. Refresh (click **Search** in the upper Search Portlet) and check the status of your **GiftShipment** process instances. Some of them should be canceled (displayed as **Stopped**). Depending on the elapsed time, some could show up as **Completed** also.



This page is intentionally left blank.

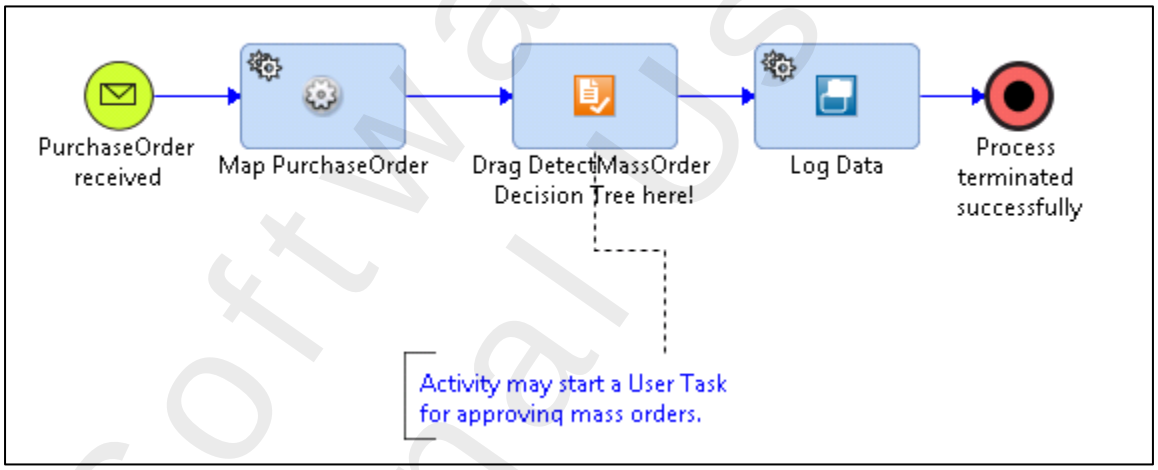
Exercise 16:
Manual Process Action

Objectives

In this exercise, you will create a Manual Process Action to start a User Task. The Action will be invoked by a Decision Tree which will be called by a Rule Task Activity in a business process. This scenario shows how to a Decision Tree from a business process to dynamically assign User Tasks depending on the rule conditions.

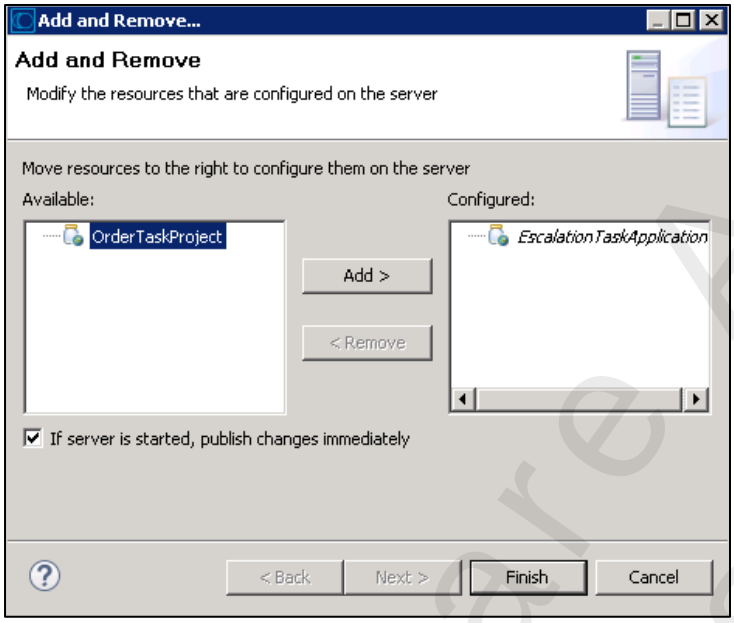
Steps

1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. Start Designer and switch to the **Process Development** perspective. Ensure you are working in **Process Developer** mode  .
4. Select **File** > **Import** > **Software AG** > **Process File** to import the Process File **<*workshop-dir*>\Exercise16\Resources\Ordering.process** into your process project **OrderProcessProject**.
5. Open the process model **Ordering** in the process editor. It should look like this:

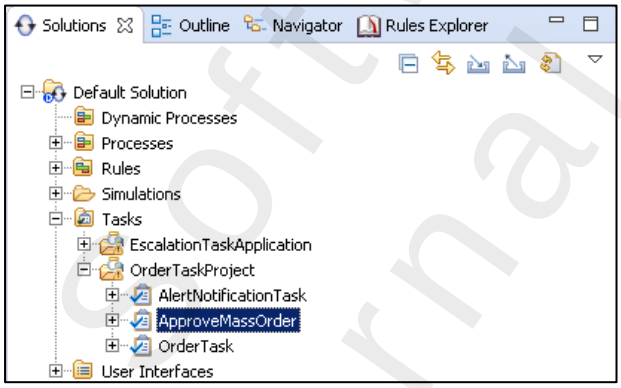


Remarks: The process model **Ordering** also receives **PurchaseOrder** documents. The first activity just maps **PurchaseOrder** to **PurchaseOrder_1**. The second activity has to be modified later to become a Rule Task Activity that invokes a Decision Tree that you will develop. The Decision Tree will receive two documents (**PurchaseOrder_1** and **ProcessData**) and returns the **PurchaseOrder_1** document and a document named **PurchaseOrderApproval_1**.
Approval data will either be created by the Decision Tree or by a User Task dynamically inserted into the workflow.

6. If you did exercise 13, skip this step and continue with step 7.
   Otherwise, select **File** > **Import** > **Software AG** > **Existing CAF Task Projects into Workspace** to import the Task project from the archive file **<*workshop-dir*>\Exercise16\Resources\OrderTaskProject.zip** into your workspace.
   Right-click in the whitespace of the Servers view (if missing, open it via **Window** > **Show View**) and choose **Add and Remove…** to add the **OrderTaskProject** Task project to your MWS. When prompted for MWS authentication, provide **Sysadmin | manage** as user credentials.



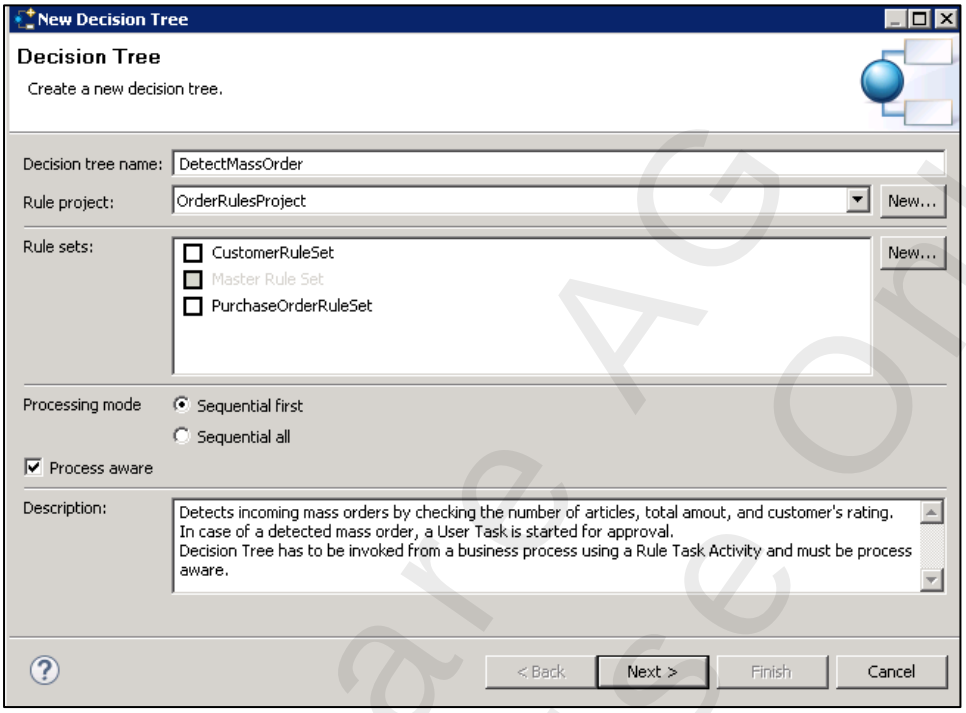
7. At this point you should have a published User Task project named **OrderTaskProject** in the Solutions view, and the User Task project should contain the User Task type **ApproveMassOrder**.



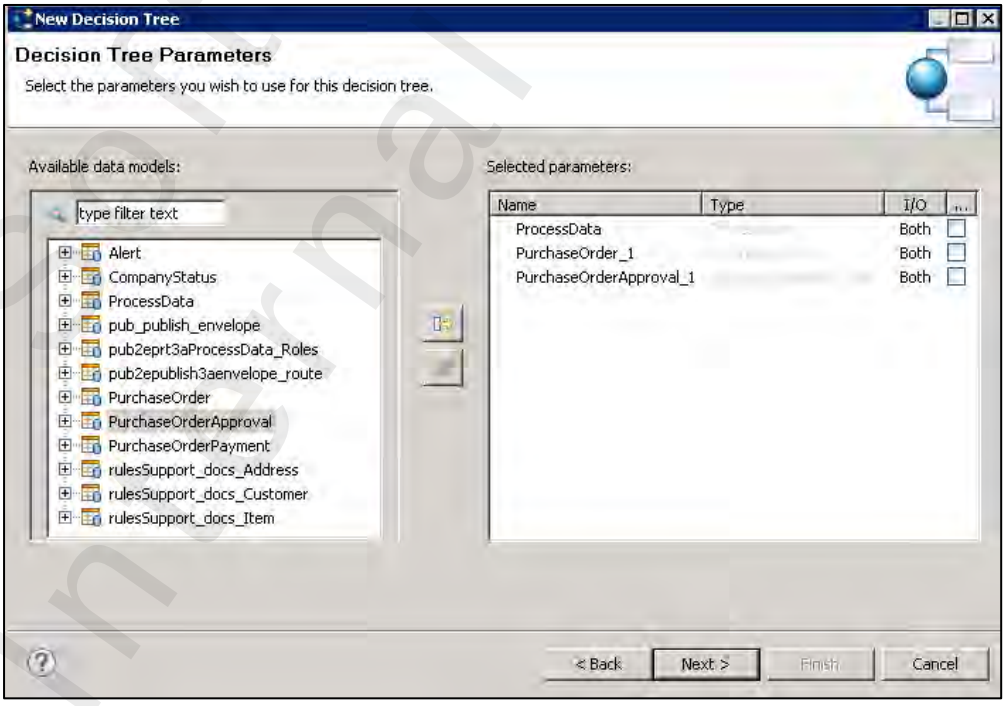
8. Switch to the **Rules Development** perspective.
9. Use the Rules Explorer view to create another Data Model named **PurchaseOrderApproval** in your Rule Project **OrderRulesProject**. Select **rulesSupport.docs:PurchaseOrderApproval** as related document type. Click **Finish**.

10. Add another Decision Tree:

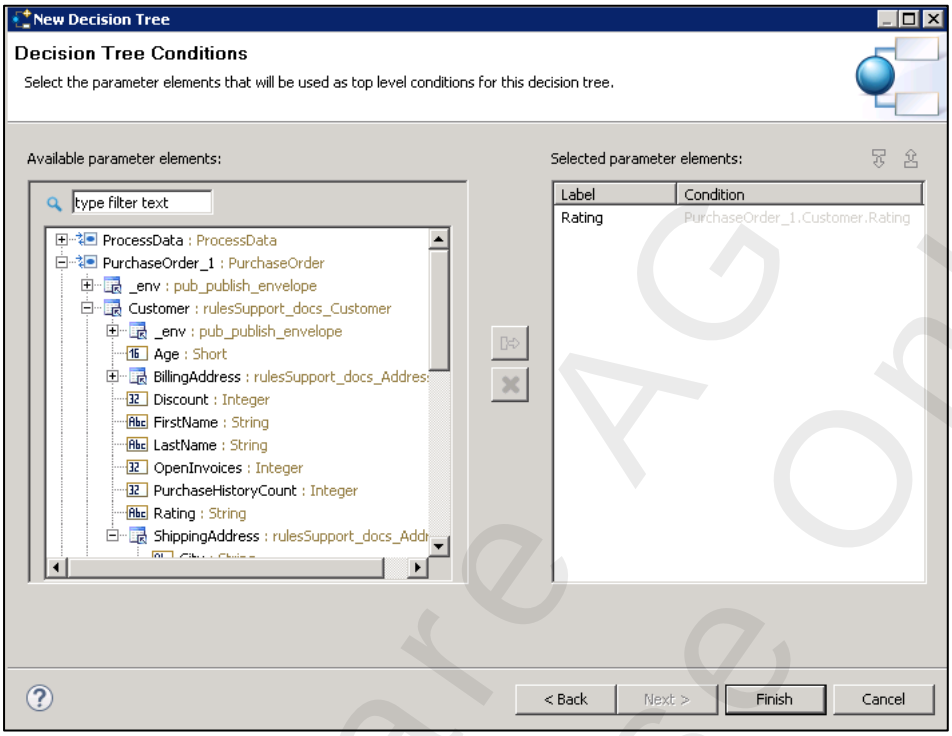
    a. Create a Decision Tree named **DetectMassOrder** in your Rule project **OrderRulesProject**. Do NOT add this Decision Tree to any Rule Set yet (except for the Master Rule Set). Select Processing mode **Sequential first** and provide a description text as shown on the screen shot below. Check the Decision Tree to be **Process aware**. Click **Next**.



    b. On the subsequent panel, add parameters of Data Model types **PurchaseOrder** and **PurchaseOrderApproval** to your Decision Table. Keep the default parameter names and do NOT remove the generated parameter **ProcessData**. Click **Next**.

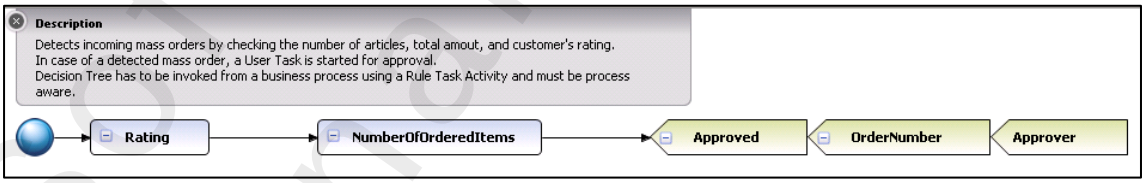


c. On the next panel select the parameter element **Rating** (contained in **PurchaseOrder_1/Customer**) to be used as top level rule condition in the Decision Tree. Click **Finish**.

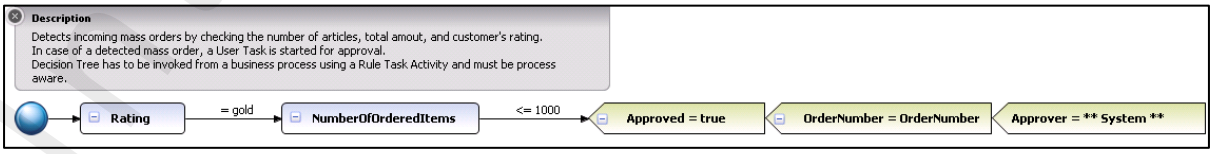


d. The Decision Tree will be opened in the Decision Tree Editor pane. Right-click parameter element **Rating** and choose **Add Condition** to select **NumberOfOrderedItems** (contained in **PurchaseOrder_1**) as second parameter to be used in a subsequent condition.

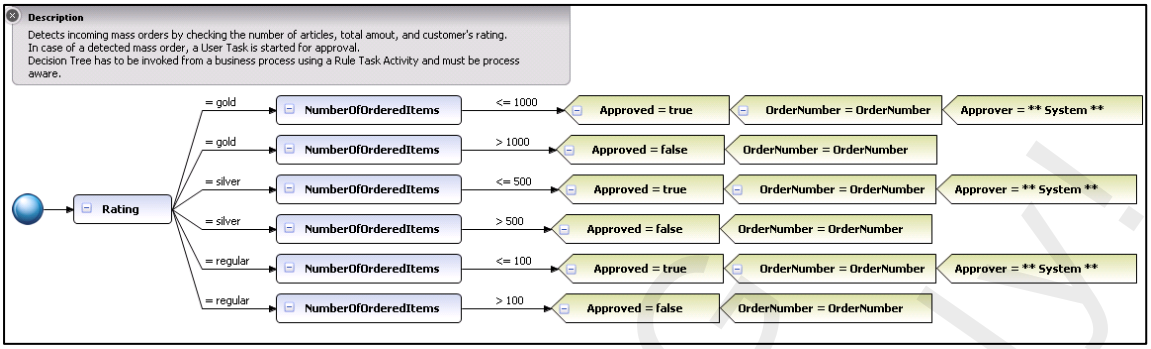
e. Right-click the ending tree element or use the palette to add three result **Assignments** to your tree structure: parameter elements **Approved**, **OrderNumber,** and **Approver** (all of parameter **PurchaseOrderApproval_1).** Your tree structure should look like this:



f. To complete the first rule, configure the transition from **Rating** to **NumberOfOrderedItems** to express the condition **Rating = gold.** Next modify the transition from **NumberOfOrderedItems** to assignment **Approved** to express the condition **NumberOfOrderedItems <= 1000**. Assign **true** to **Approved,** value of **OrderNumber** as contained in **PurchaseOrder_1** to **OrderNumber** of **PurchaseOrderApproval_1**, and **** System **** to Approver.



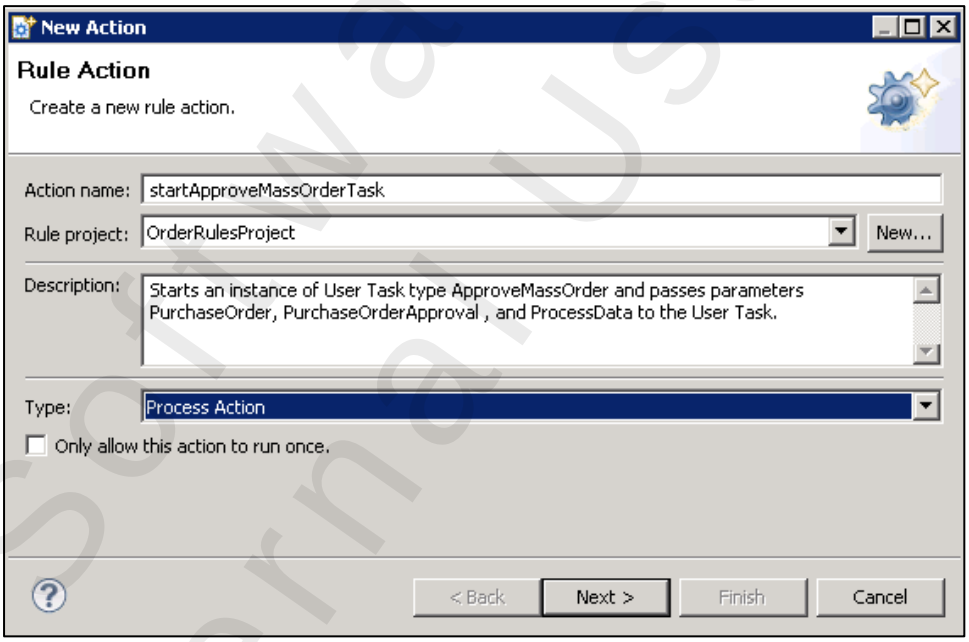
g. Copy node **NumberOfOrderedItems** and paste it five times to node **Rating**. Adjust/delete the conditions and assignments in the pasted five rules to match the following screen shot:



11. Save and test your Decision Tree. For testing you can load the provided sample input from file: **<*workshop-dir*>\Exercise16\Resources\DetectMassOrderDecisionTreeInput.xml**

12. Add another Action to your Rule project:

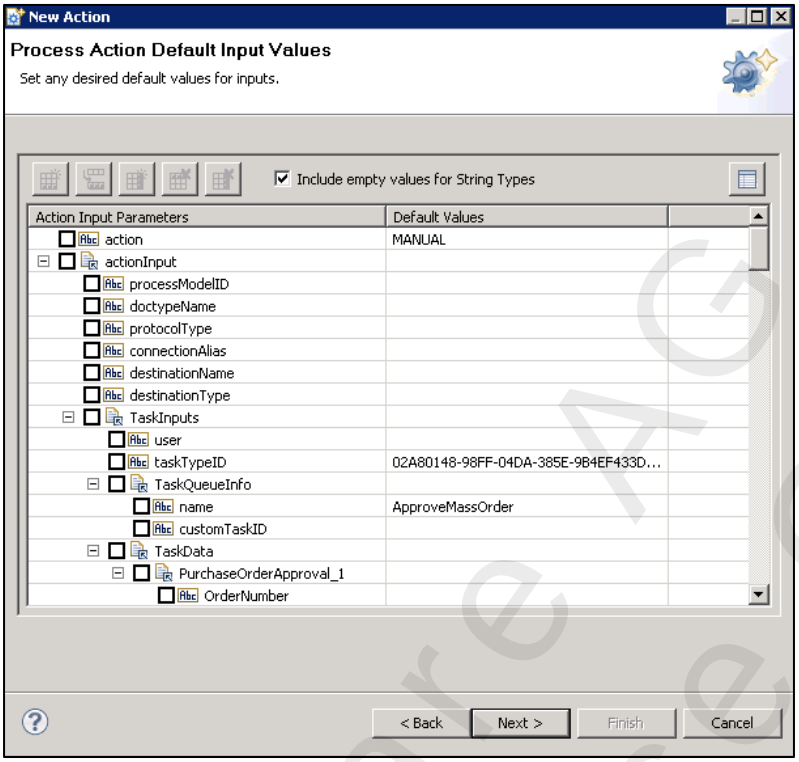
    a. Create an Action of type **Process Action** in your Rule project **OrderRulesProject**. Name it **startApproveMassOrderTask** and provide a description text as shown on the screen shot below. Click **Next**.



    b. On the next panel, choose Process Action type **Manual decision**. Click **Next**.

    c. On the Select task panel, select the User Task type **ApproveMassOrder**. Click **Next**.

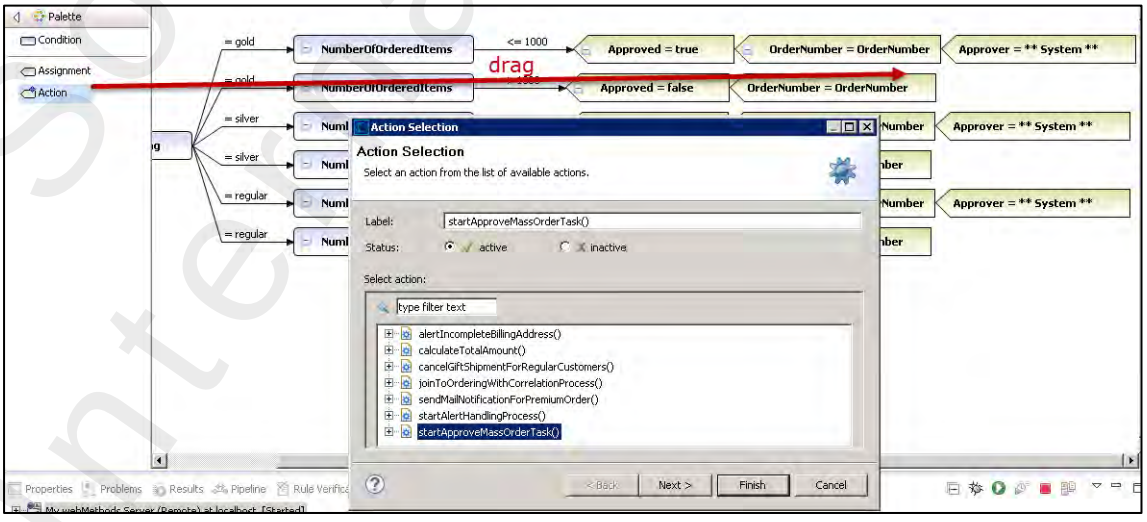
d. On the Process Action Default Input Values panel, leave all default values unchanged and check the box beside **Include empty values for String Types**. Click **Next**.



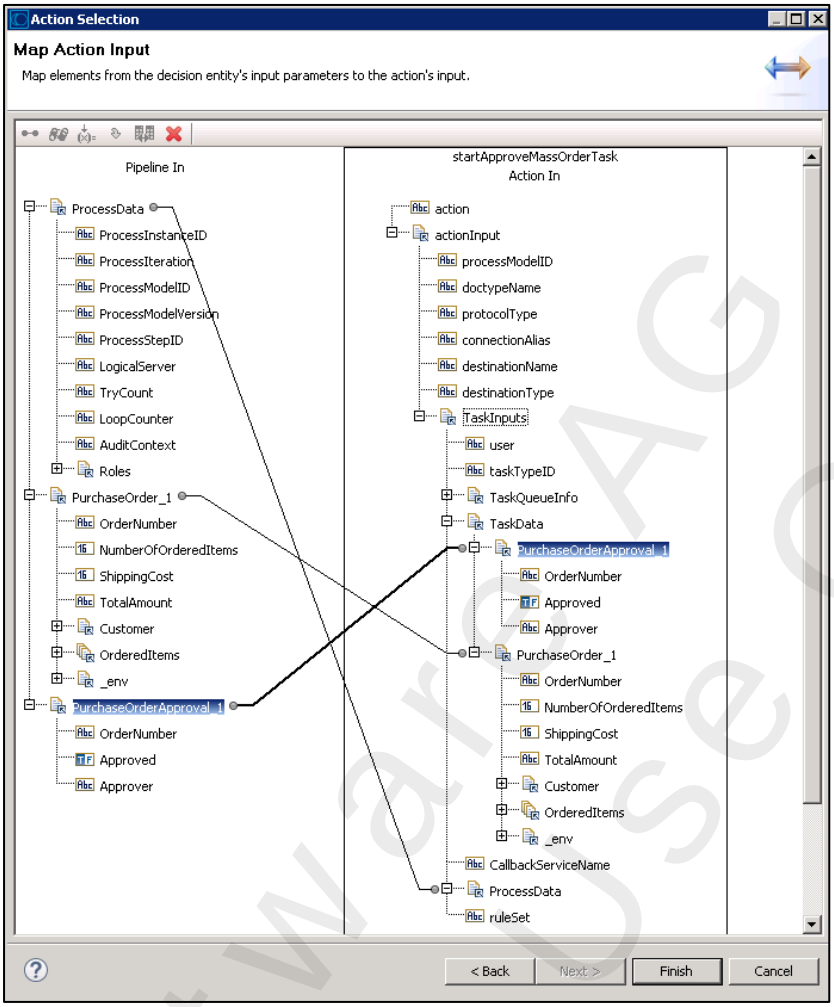
e. On the last panel, check **outputMessage** as action return value. Click **Finish** to create the action.

13. Open the Decision Tree **DetectMassOrder** contained in your **OrderRulesProject** in the Decision Tree Editor.

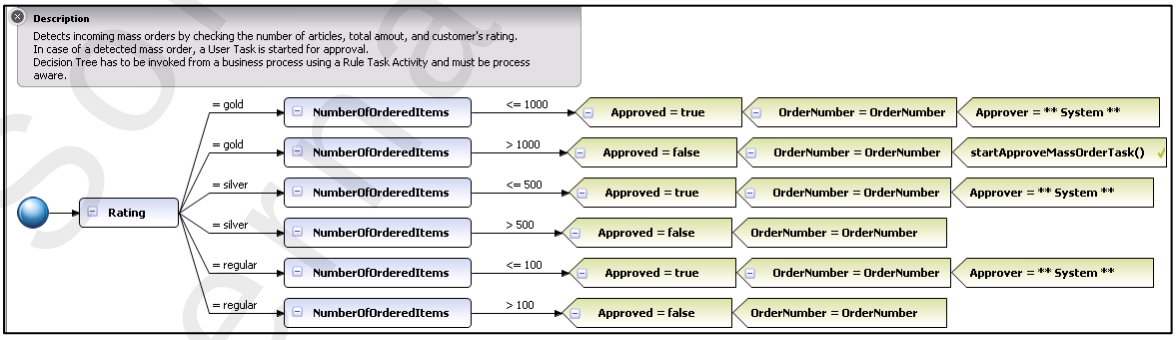
a. Use the Palette to add an Action at the end of the first rule with a lacking Approver assignment. On the action Selection panel select Action **startApproveMassOrderTask** with a Status of **active** and click **Next**.



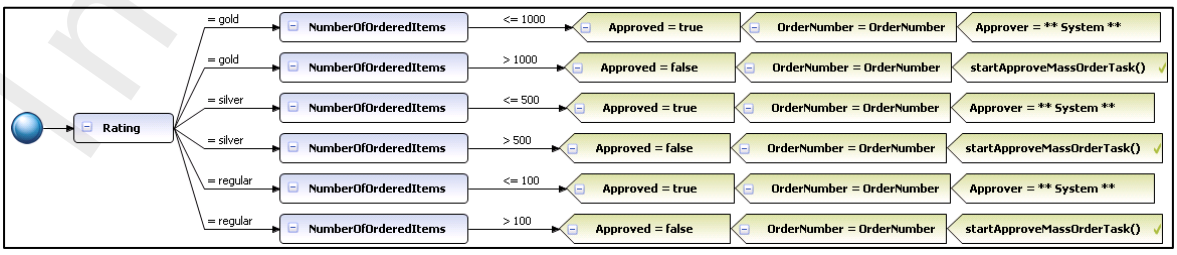
b. On the next panel, map the Decision Tree parameter elements to your Action inputs as shown below and click **Finish**.



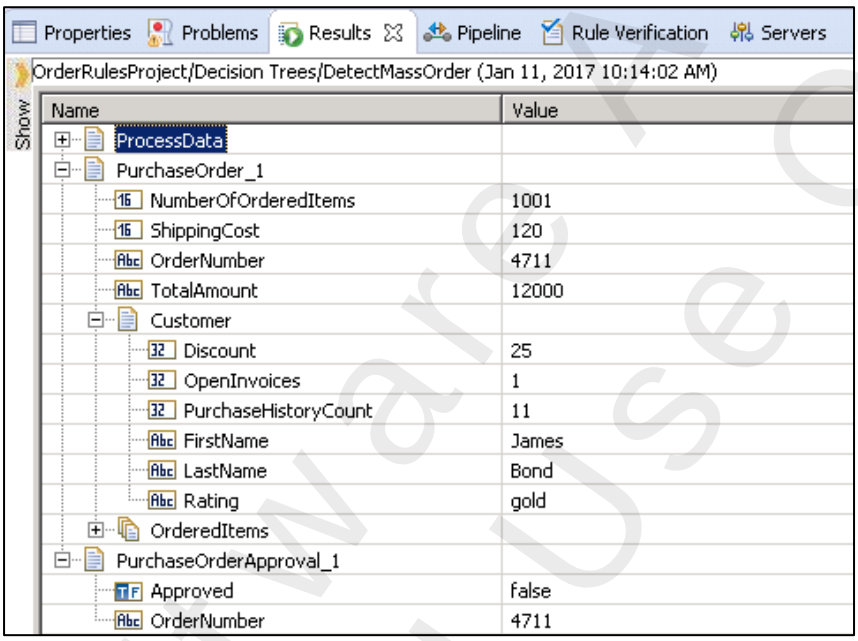
Your Decision Tree should now look like this:



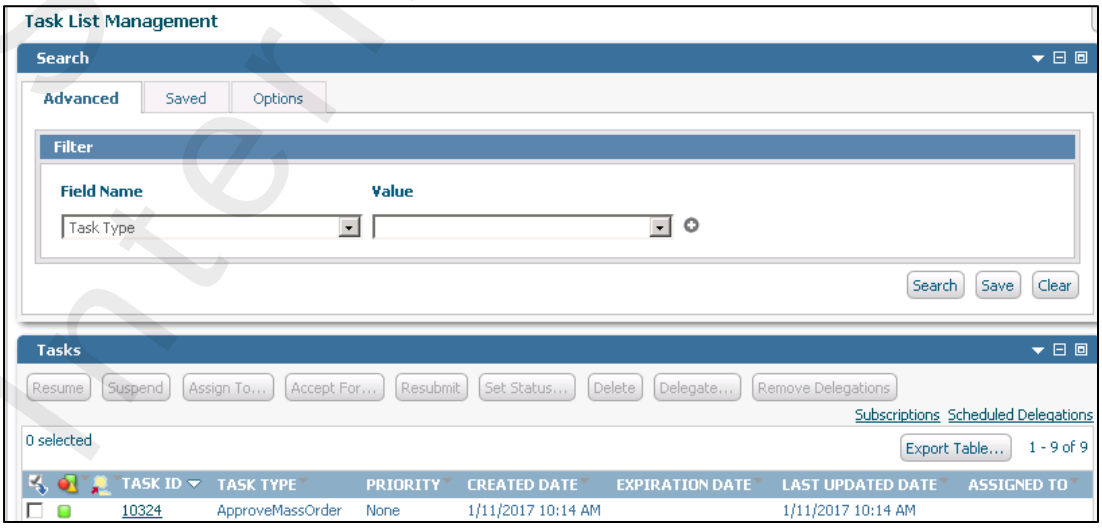
c. Copy the Action node **StartApproveMassOrderTask()** added in the previous step and paste it to the end of the other two rules with a lacking Approver assignment.



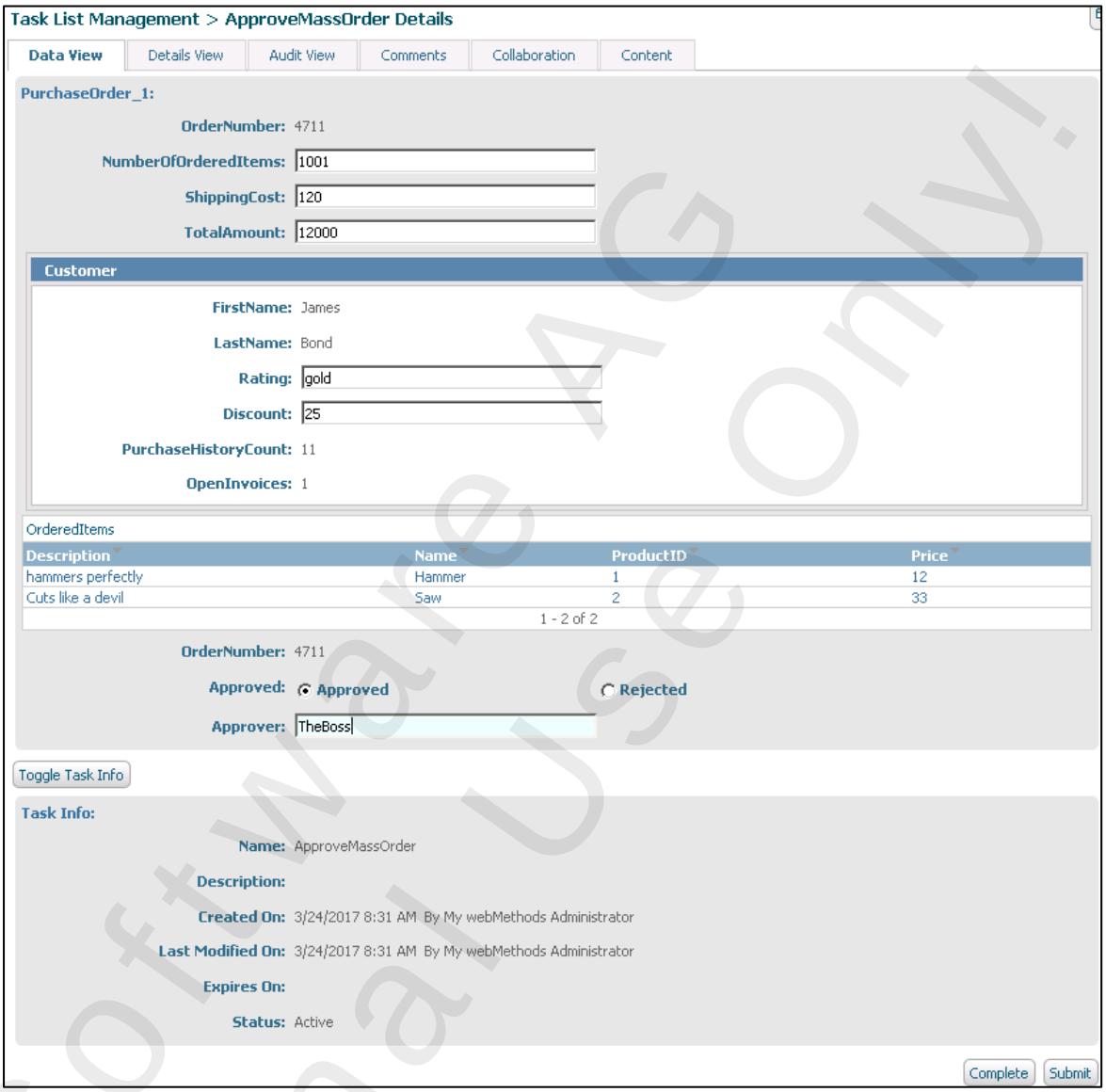
14. Save your Decision Tree.
15. Re-deploy (export) your Rule Project to your Integration Server runtime and MWS Repository.
16. Test your Decision Tree:
    a. Right-click Decision Tree **DetectMassOrder** in the Rules Explorer and select **Run As > Run Decision Tree**. On the appearing panel, specify/modify input values that will fire a rule and execute the action **startApproveMassOrderTask**. Instead of typing, just load the provided sample input from file: **<*workshop-dir*>\Exercise16\Resources\DetectMassOrderDecisionTreeInput.xml**. In the Results view, confirm **PurchaseOrderApproval_1\Approved** returns **false** and **PurchaseOrderApproval_1\OrderNumber** is set to the value of **PurchaseOrder_1\OrderNumber**:



    b. Open a browser tab and login to My webMethods (http://localhost:8585) as **Administrator | manage**.
    Navigate to **Applications** > **Monitoring** > **Business** > **Tasks** > **Task List Management**. Click **Search** to refresh the Tasks list. Have a look for a new queued User Task instance of type **ApproveMassOrder**.



c. Open the User Task instance. In the User Task UI, you should see your **PurchaseOrder** data as well as the assigned order number. Moreover, the pre-selected value of Approved should be **Rejected**. Enter your name as **Approver**, set Approved to **Approved** and **complete** the User Task instance.



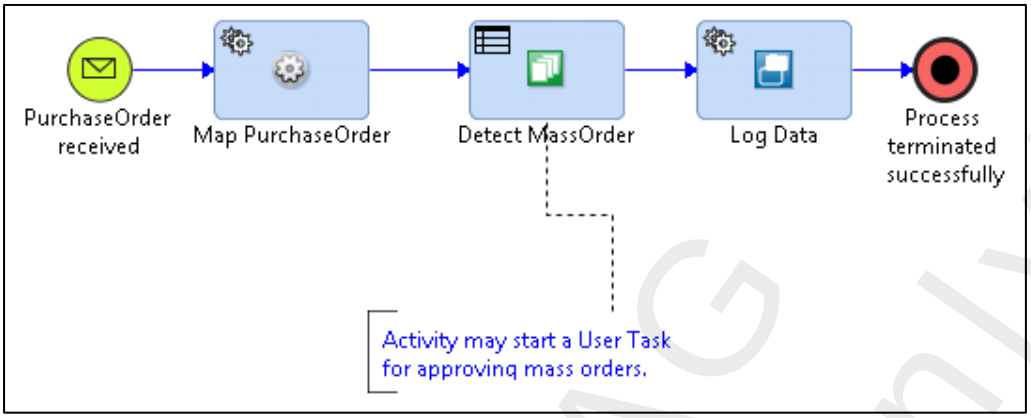
17. Switch back to Designer and open the **Process Development** perspective.
18. Open the imported process model **Ordering** as contained in the process project **OrderProcessProject** in the process editor pane:

    a. Ensure the Rules Explorer view is visible. If the Rules Explorer view is not visible, choose **Window** > **Show View** > **Other…** and select the Rules Explorer view to be displayed in the Process Development perspective.

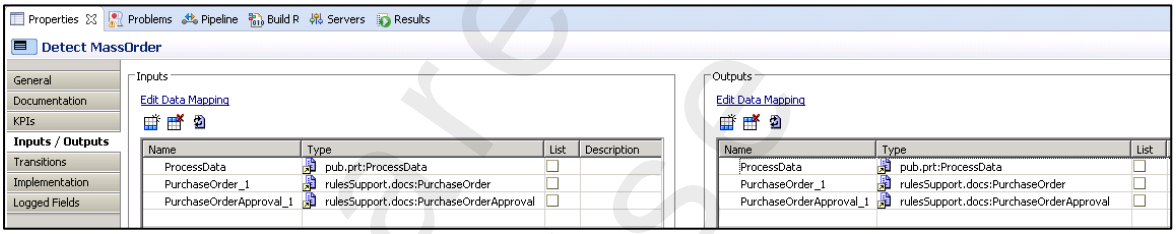
    b. Use the Rules Explorer view and drag the Decision Tree **DetectMassOrder** contained in your Rule project **OrderRuleProject** onto the Abstract Task Activity named **Drag DetectMassOrder Decision Tree here!** in the process editor pane. Select the modified Task Activity and inspect its Properties view: dragging should have set the Task type to Rule and the rule type should be webMethods Business Rule of sub type Decision Tree.

    c. Rename the Rule Task Activity to **Detect MassOrder**.

d. Adjust the image of the Rule Task Activity **Detect MassOrder** to fit to the following screen shot. Your process should now look like this:

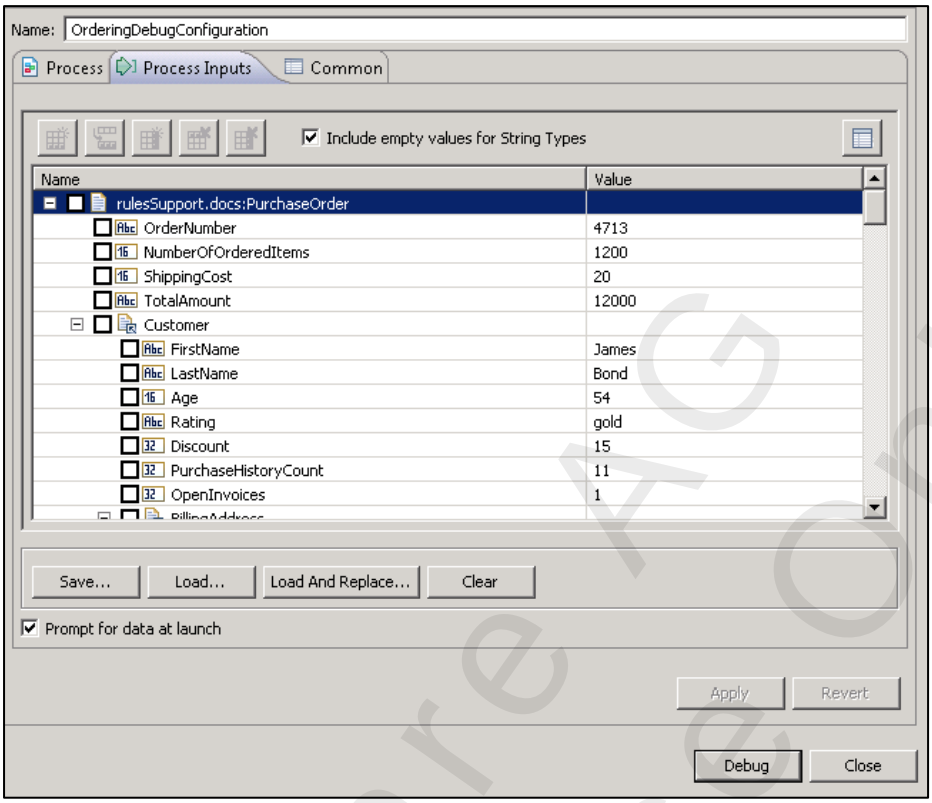


e. Select the Rule Task Activity **Detect MassOrder** and inspect its Inputs/Outputs in the Properties view. Ensure input and output parameters are set like this (otherwise use the Refresh icons):



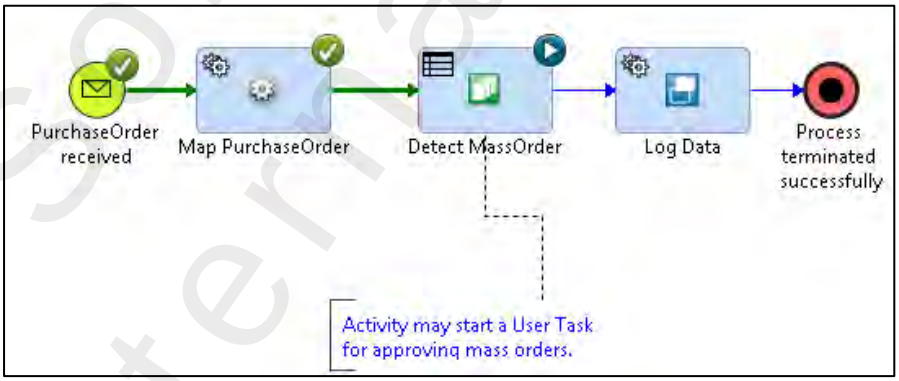
19. Save, Build and Upload your process model.
20. If not done before, right-click **rulesSupport.docs:PurchaseOrder** in the Package Navigator view and select **Sync Document Type** to push the document type to the Messaging Provider.
21. Switch to the **Process Debug** perspective.
22. As done in exercise 11, create, configure and run a Debug Configuration to test your Ordering process:
    a. On the **Process** tab of your new Debug Configuration, specify **OrderingDebugConfiguration** as Run Configuration name and select process model **Ordering** within process project **OrderProcessProject**.
    b. On the **Process Inputs** tab, provide input data for the Start Message Event and check to get prompted to type/alter those data when debugging starts. Instead of typing, you can load input data from file **<*workshop-dir*>\Exercise16\Resources\OrderingDebugConfigInput.xml**.
    c. Check the box beside **Include empty values for String Types** and click **Apply** to save the Debug Configuration.

d. Click **Debug** to start process debugging.



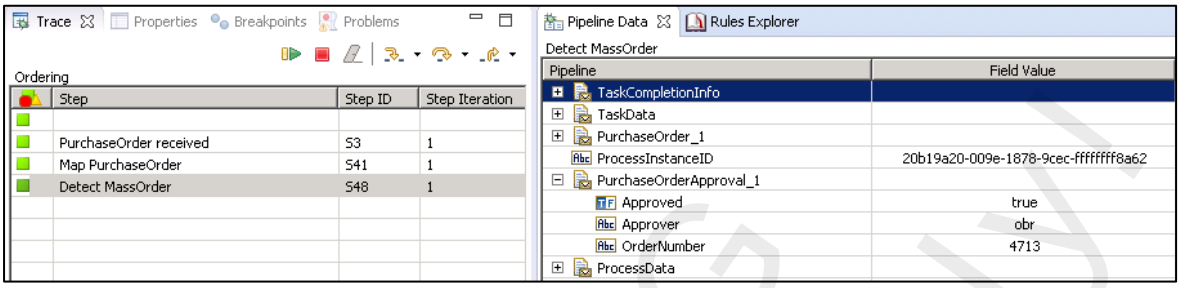
e. Because of the settings you made in the Debug Configuration you get prompted for the input data having your data specified as in the Debug Configuration preloaded. (Same) sample input can optionally be loaded from file:
<*workshop-dir*>\Exercise16\Resources\OrderingDebugInput.xml
Check the box beside **Include empty values for String Types** and click **OK**.

f. Use the Trace view to step over (F6) the process activities and verify that your Rule Task Activity **Detect MassOrder** queues a User Task instance and waits until the User Task has been completed:



g. Use a browser tab to revisit My webMethods as **Administrator | manage**. Like above, look for a new User Task instance of Task type **ApproveMassOrder**, and open the Task instance. In the Task UI, verify the mapped **PurchaseOrder** data, add an **Approver** name, approve or reject the order, and **complete** the User Task instance.

h. Switch back to the Process Debugger in Designer. Select the (completed) step **Detect MassOrder** in the Trace view and double-check in the Pipeline Data view that the **Approver** and **Approved** fields were returned by the User Task into the process pipeline.



i. Finally, complete your process in the Process Debugger.

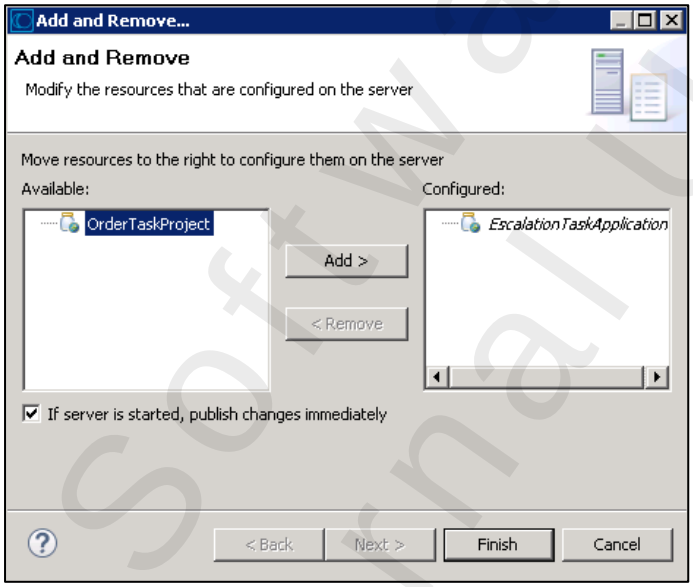
Exercise 17:
User Task Invokes Decision Entity for User Task Assignment

Objectives

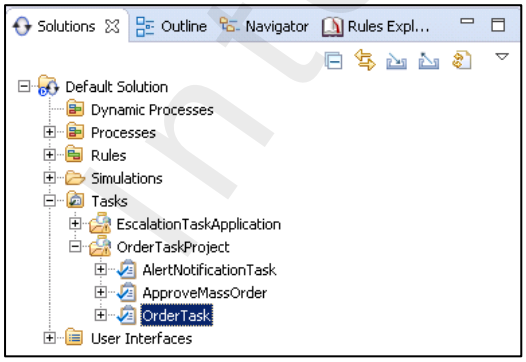
In this exercise, you will invoke a Decision Table from a User Task. The Decision Table will return a string value that contains an assignee for the User Task Assignment. The value can contain a user, group, or role.
User Task and user definitions are already provided in your environment.

Steps

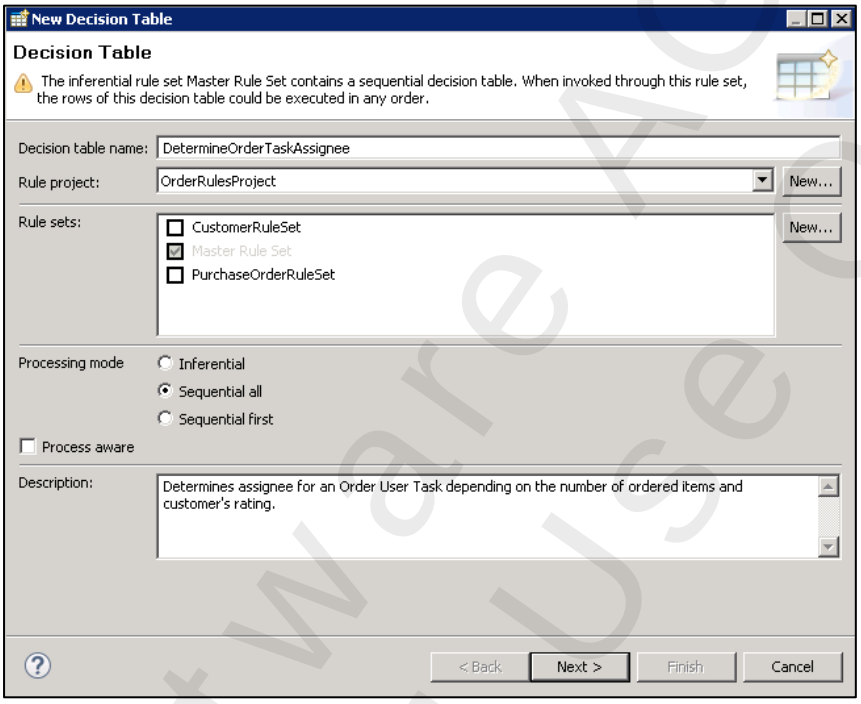
1. Start your **Integration Server** as a Windows service, if not active.
2. Start **MWS** and **Optimize Analytic Engine** as Windows services, if not active.
3. Start **Designer**.
4. If you did exercise 13 and/or 16, skip this step and continue with step 5.
   Otherwise, select **File** > **Import** > **Software AG** > **Existing CAF Task Projects into Workspace** to import the Task project from the archive file **<workshop-dir>\Exercise17\Resources\OrderTaskProject.zip** into your workspace.
   Right-click in the whitespace of the Servers view (if missing, open it via **Window** > **Show View**) and choose **Add and Remove…** to add the **OrderTaskProject** Task project to your MWS. When prompted for MWS authentication, provide **Sysadmin | manage** as user credentials.



5. At this point you should have a published User Task project names **OrderTaskProject** in the Solutions view, and the User Task project should contain the User Task Type **OrderTask**.

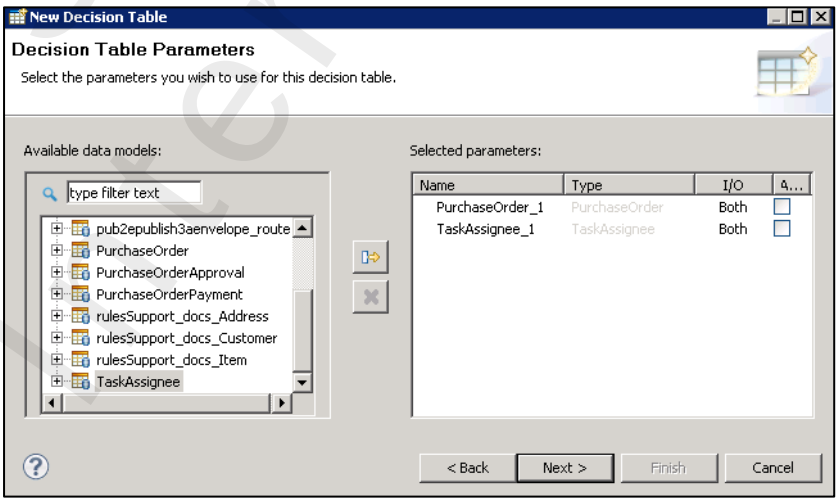


6. Switch to the **Rules Development** perspective.
7. Use the Rules Explorer view to add a Data Model named **TaskAssignee** in your Rule Project **OrderRulesProject**. Select **rulesSupport.docs:TaskAssignee** as related Document Type. Click **Finish**.
8. Add another Decision Table:
    a. Create a Decision Table named **DetermineOrderTaskAssignee** in your Rule project **OrderRulesProject**. Do NOT add this Decision Table to any Rule Set (except for Master Rule Set) and select Processing mode **Sequential all**. Provide a description text as shown on the screen shot below and click **Next**.

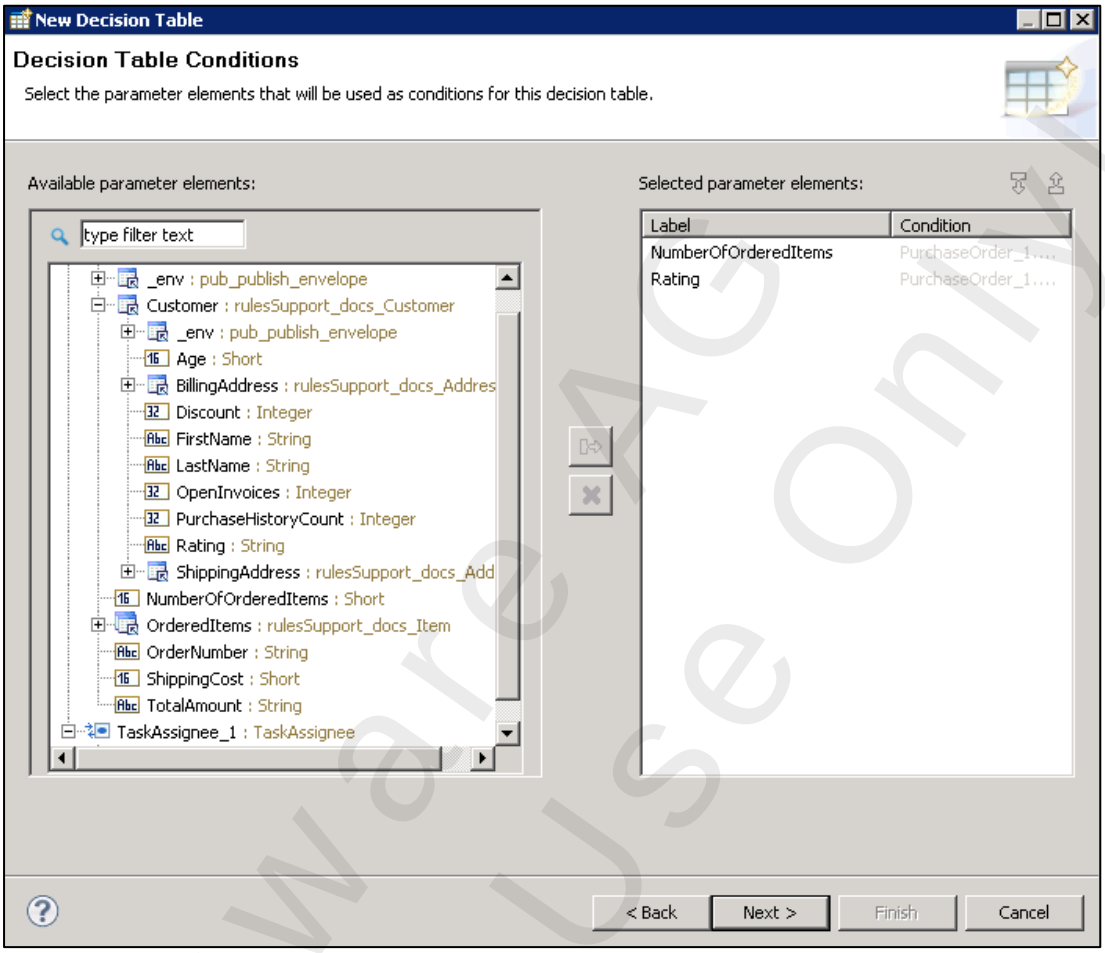


Note: Accept the warning about the inferential Processing mode of the Master Rule Set.

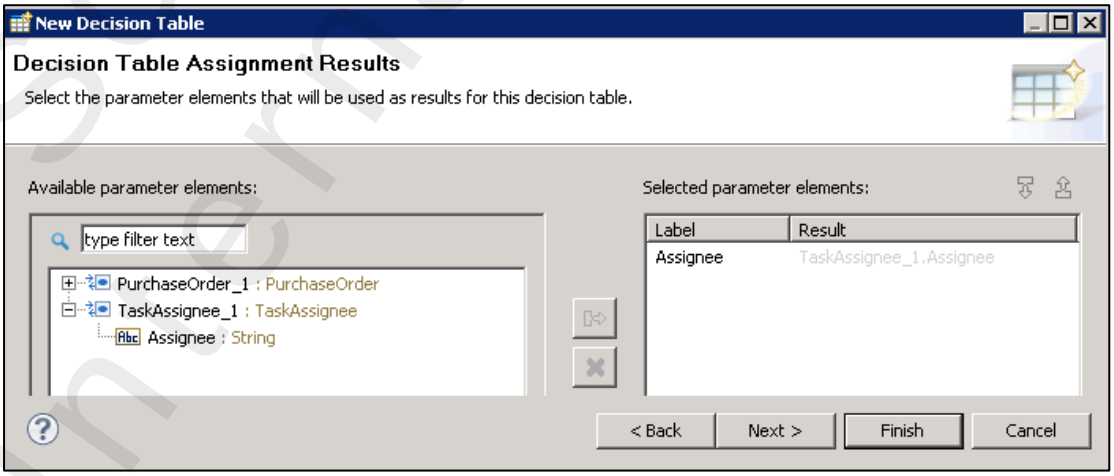
    b. On the subsequent panel, add parameters of Data Model types **PurchaseOrder** and **TaskAssignee** to your Decision Table. Keep the default parameter names and select I/O type **Both**. Click **Next**.



c. On the next panel select the parameter elements **NumberOfOrderedItems** (contained in **PurchaseOrder_1**) and **Rating** (contained in **PurchaseOrder_1/Customer**) to be used later in your rule conditions. Click **Next**.



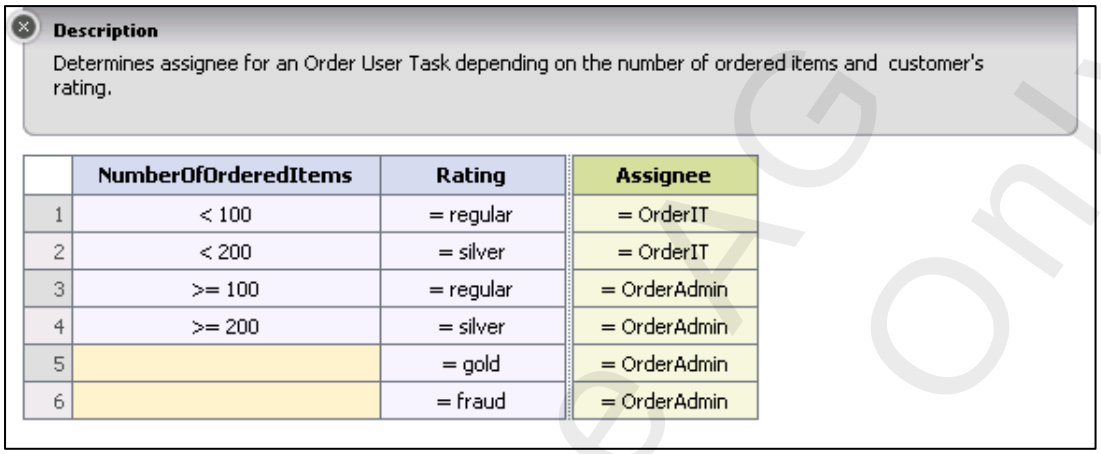
d. Finally specify the parameter elements used as results for your Decision Table. Select the element **Assignee** of parameter **TaskAssignee_1** only. Click **Finish** to complete your Decision Table structure.



9. Use the Decision Editor for your Decision Table **DetermineOrderTaskAssignee** to insert six rules as shown on the screen shot below.

   Hint: The string values assigned in the result column are existing MWS user group names (see picture at step 17).

   Finally, your Decision Table should look like this:

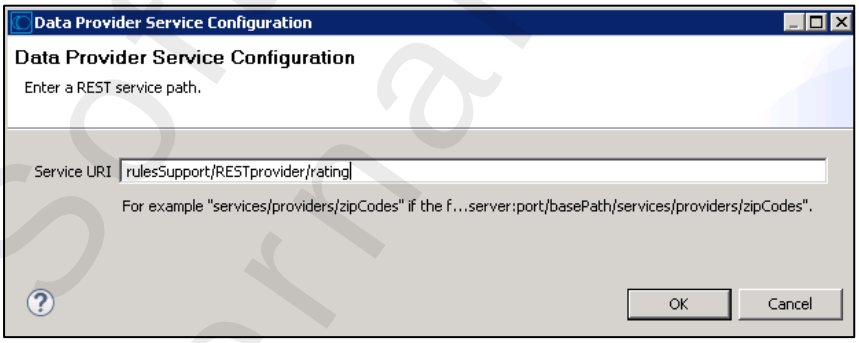


	NumberOfOrderedItems	Rating	Assignee
1	< 100	= regular	= OrderIT
2	< 200	= silver	= OrderIT
3	>= 100	= regular	= OrderAdmin
4	>= 200	= silver	= OrderAdmin
5		= gold	= OrderAdmin
6		= fraud	= OrderAdmin

10. To offer better comfort to Business Users working on the Decision Table in the web-based RMC, configure a Data Provider Service for Rating and enable a Principle Picker for Assignee. To do so:

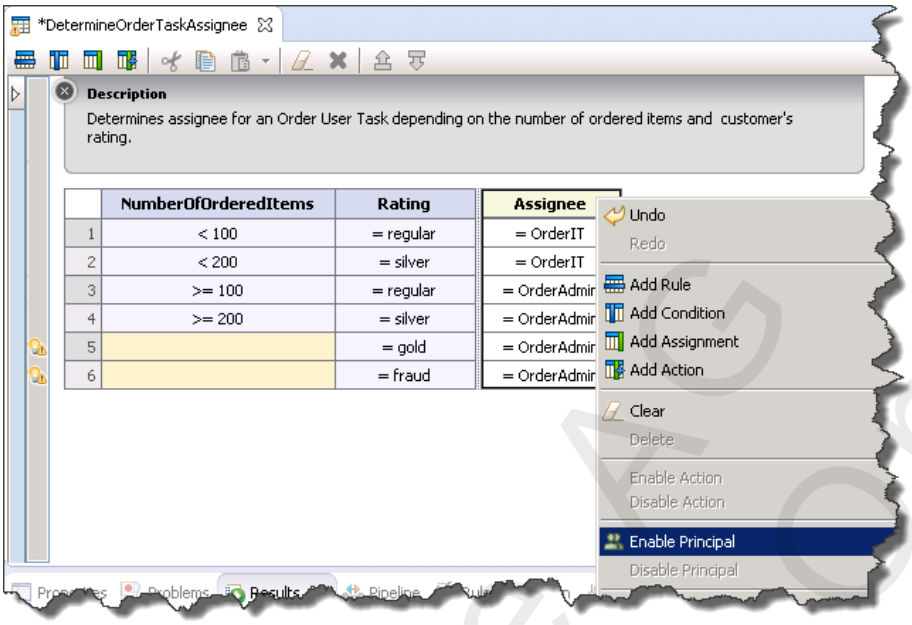
    a. Still in Designer, right-click the column header of condition **Rating** and select **Configure Data Provider Service**.

    b. Your IS package **RulesSupport** already contains a ready-made REST resource that returns valid ratings by a _get method. Specify **rulesSupport/RESTprovider/rating** as the trailing REST service path and click **OK**.

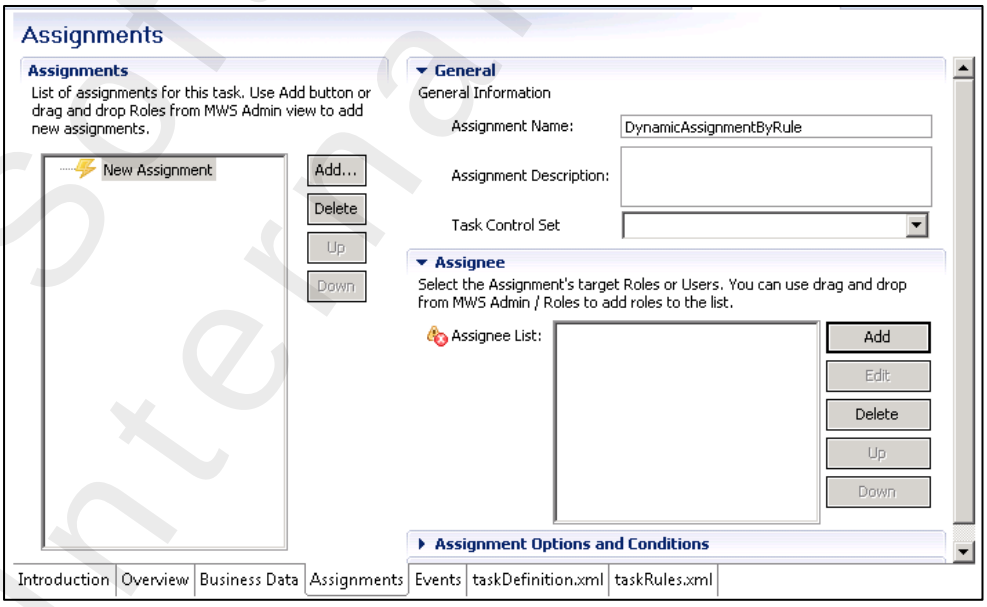


   Note: The leading portion of the Data Provider service URL has already been configured on the My webMethods Business Rules Settings page (see exercise 9).
   Service URI must be formed up using "/". Do NOT use "\".

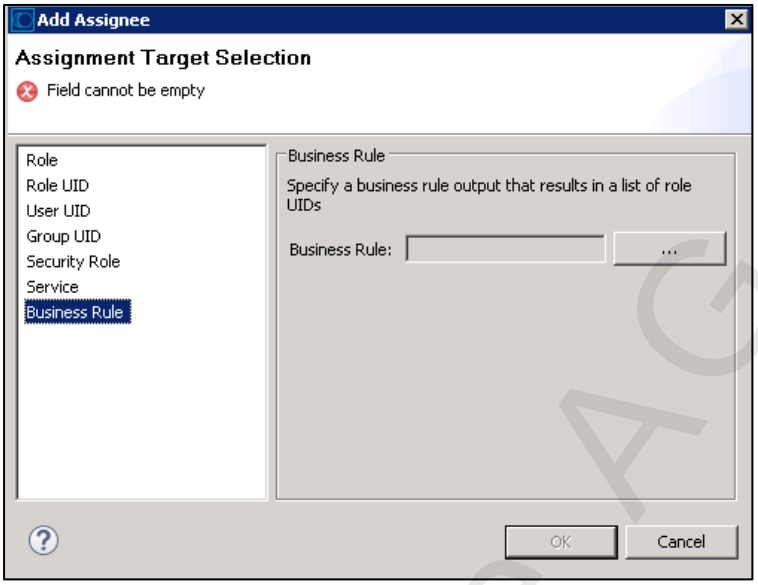
c. Right-click column header of assignment **Assignee** and select **Enable Principal** to annotate the assigned values as MWS principals.



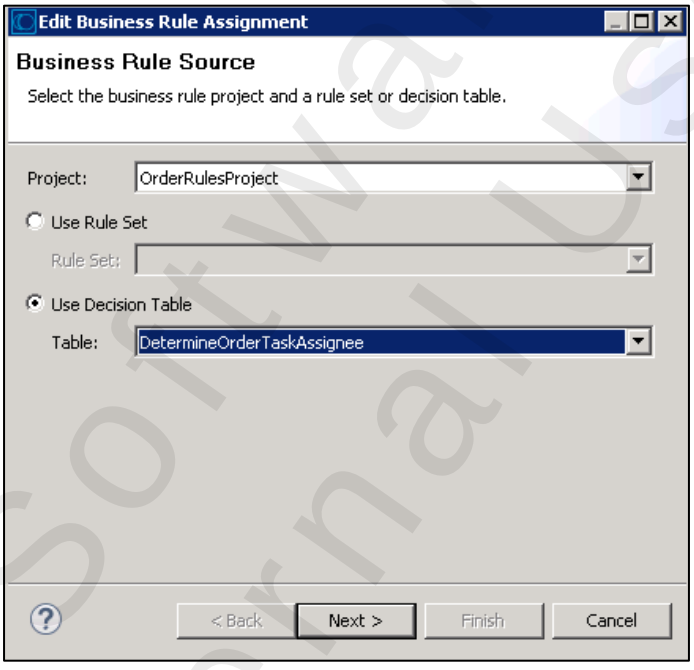
11. Save and test your Decision Table. For testing you can load the provided sample input **<*workshop-dir*>\Exercise17\Resources\DetermineOrderTaskAssigneeDecisionTableInput.xml**.
12. Re-deploy (export) your Rule Project to your Integration Server runtime and MWS Repository.
13. In Designer, switch to the **UI Development** perspective.
14. Modify the existing User Task OrderTask:
    a. Open the User Task **OrderTask** from the Solutions view.
    b. In the Task Editor, open the **Assignments** tab. Add a new Assignment and provide the Assignment Name **DynamicAssignmentByRule**.



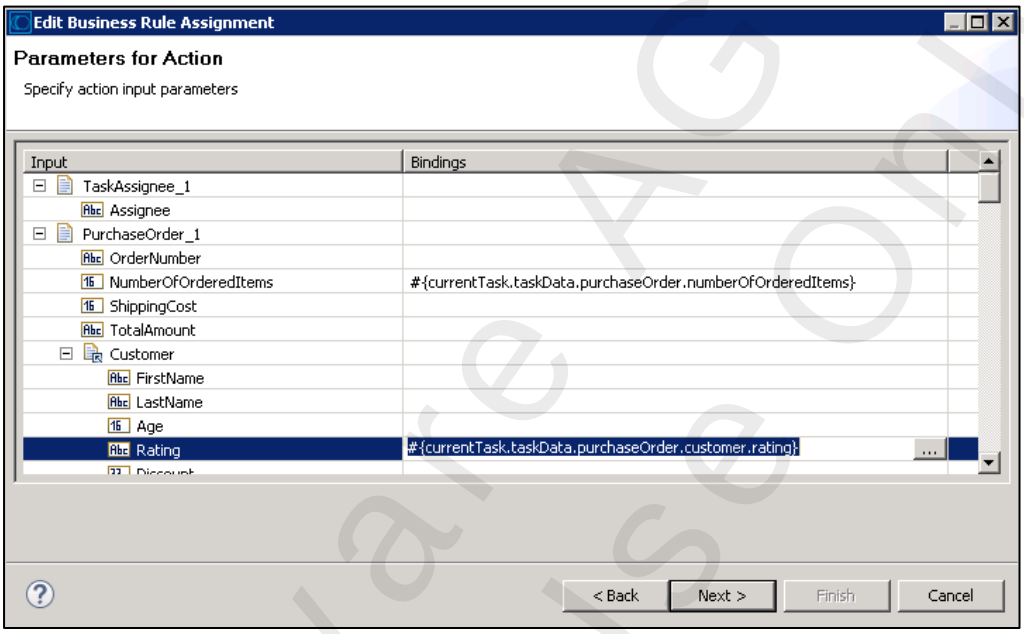
c. In the Assignee section click **Add** to add an assignee. On the Assignment Target Selection panel, choose **Business Rule** and click the **…** button.



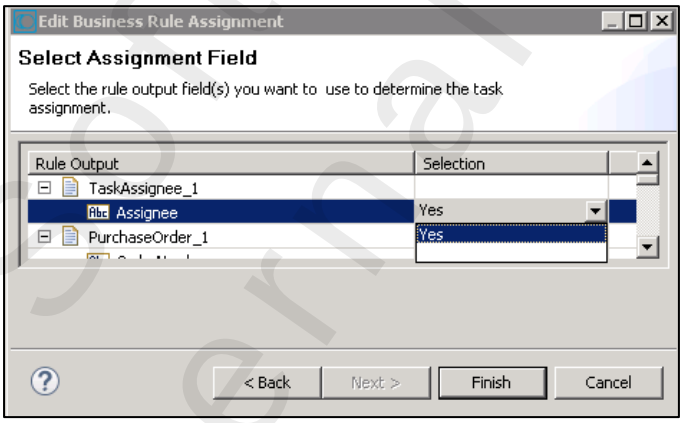
d. Select Project **OrderRulesProject** and then DecisionTable **DetermineOrderTaskAssignee**. Click **Next**.



e. On the Parameters for Action panel, click into the Bindings column for Input parameter **PurchaseOrder_1\NumberOfOrderedItems** and click the **…** button.
On the Expression Binding panel, map the existing Task Business Data field **Business Data\PurchaseOrder\NumberOfOrderedItems** to the required input parameter element of your Decision Table.
In the same way map the existing Task Business Data field **Business Data\PurchaseOrder_1\Customer\Rating** to the required input parameter element **PurchaseOrder_1\Customer\Rating**.
Click **Next**.



On the Select Assignment Field panel select the output parameter element **Assignee** used to deliver the required User Task assignee:



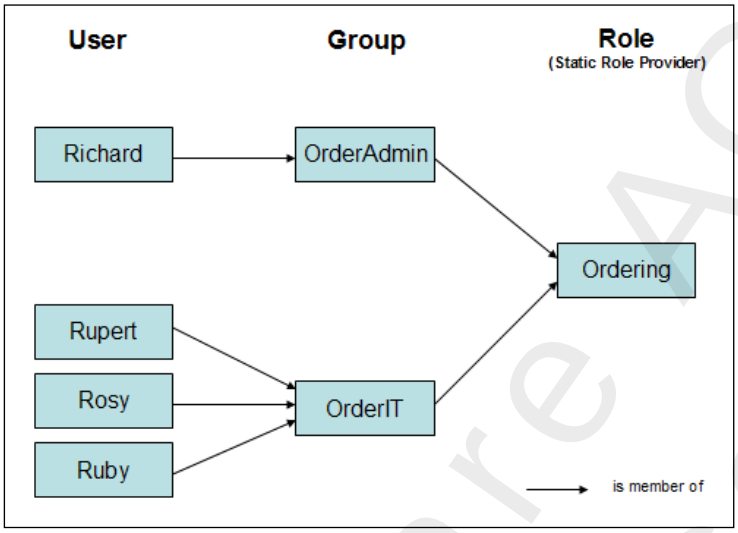
f. Click **Finish** and **OK** to complete the User Task Assignment.

15. Save your changes.

16. Use the Servers view to republish the User Task project **OrderTaskProject** to your MWS. If you are asked to authenticate to MWS, enter Username **Sysadmin** and Password **manage**.

17. Test your User Task:

Remarks: User Task type **OrderTask** already contains a Start portlet (**OrderTaskStart**) which can be used to create a User Task instance without running a process instance. Additionally, the following user/ group/ static role structure has been setup in the MWS user repository on your VM.
Moreover, Static Role **Ordering** has already been granted full access permissions to My webMethods Applications.



a. In a browser tab, login to My webMethods (http://localhost:8585) as **Administrator | manage**.

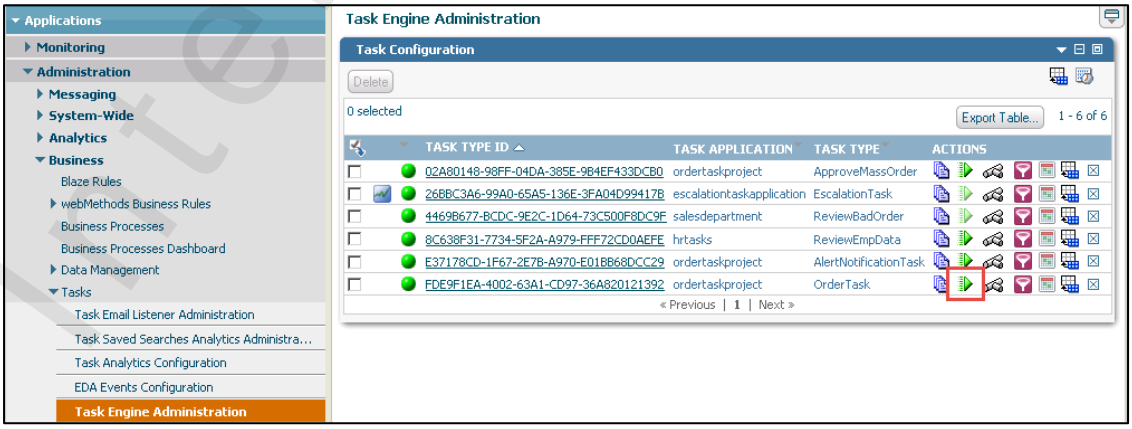
b. Allow users of role **Ordering** to work on **OrderTask** User Tasks. To do so:

   i. Navigate to **Applications** > **Administration** > **System-Wide** > **Permissions Management**.

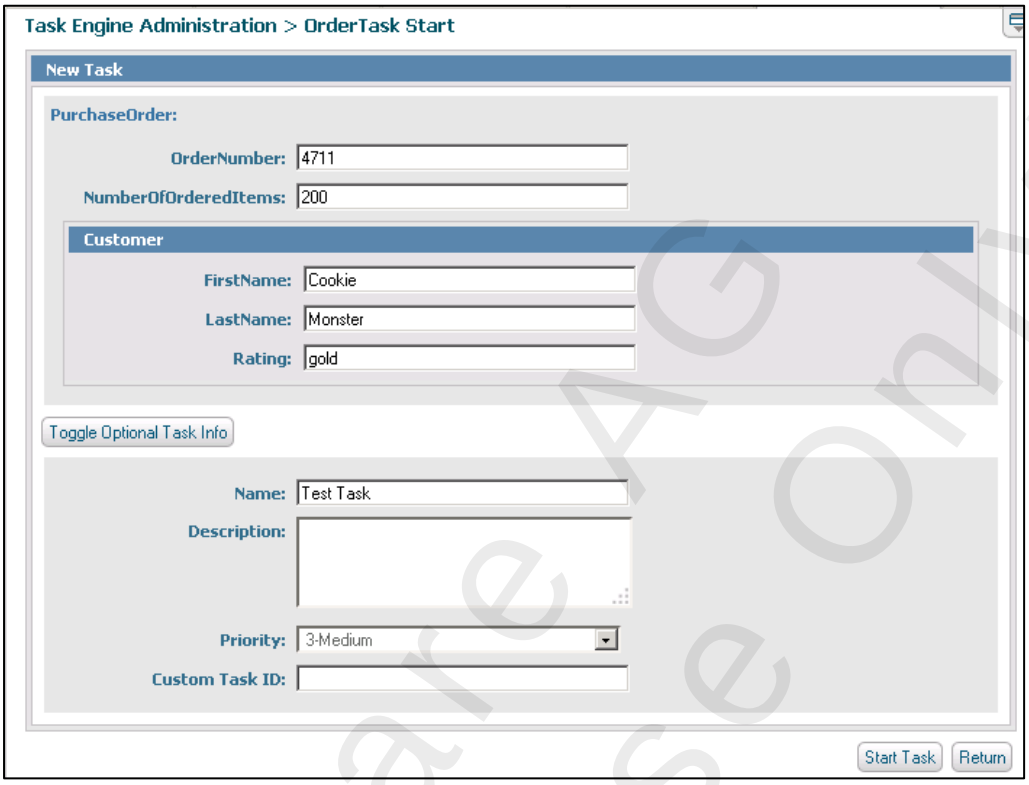
   ii. Select Resource Type **Tasks** and click **Search**. Select User Task type **OrderTask** in the Found list and click the arrow to move it to the Selected list. Click **Next**.

   iii. On the Permissions Management page, ensure that role Ordering has been granted all permissions. If not, **search** and **add** role **Ordering**. On the Permissions page, just press **Grant All** and **OK** twice. After returning to the Permissions Management page, click **Apply**.

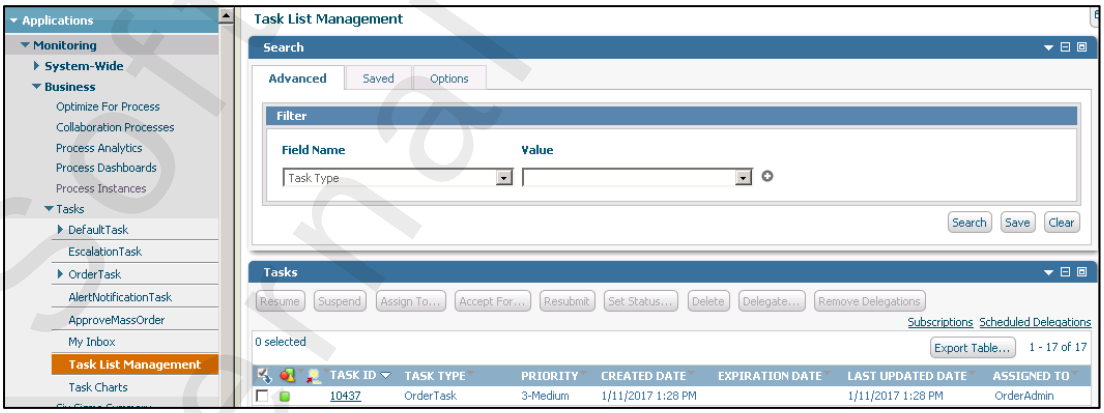
c. In My webMethods, navigate to **Administration** > **Business**> **Tasks** > **Task Engine Administration**. Start a new instance of Task Type **OrderTask** by clicking the green triangle icon to the right of it.



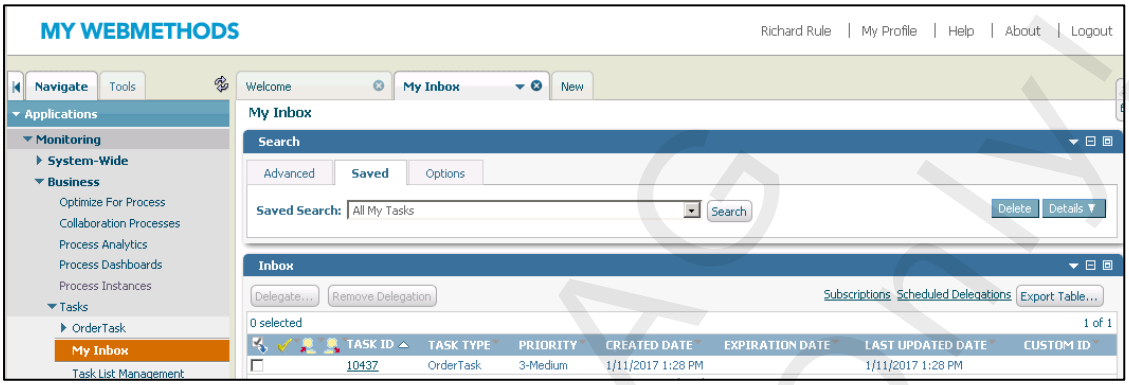
d. On the OrderTask Start portlet UI, enter Purchase Order and Customer test data. At a minimum you have to specify **OrderNumber**, **NumberOfOrderedItems** and **Rating** since they are required by the Decision Table. Then click **Start Task**.



e. Still in My webMethods, navigate to **Applications** > **Monitoring** > **Business** > **Tasks** > **Task List Management**. Click on **Search** to refresh the result list. You should see a new User Task instance assigned to **OrderIT** or **OrderAdmin** depending on the test data you entered in the previous step:



f. Logout from My webMethods. Depending on the assigned group of the previos step, login to My webMethods as **Richard | manage** (assigned to group **OrderAdmin**) or as **Rosy | manage** (assigned to group **OrderIT**).
Navigate to **Applications** > **Monitoring** > **Business** > **Tasks** > **My Inbox**.
Double-check the User Task instance appears in the assignee's Inbox:



Hint: If the user is not able to navigate in My webMethods, login as **Administrator** | **manage** and go to **Administration** > **System-Wide** > **Permissions Management** to grant all permissions for resource type **webMethods Applications** to role **Ordering**.

g. Open and **complete** the User Task.

18. In My webMethods, acting as Business User **Richard** or **Rosie**, use the Rules Management Console:

a. Navigate to **Applications** > **Administration** > **Business** > **webMethods Business Rules** > **OrderRulesProject** > **Decision Tables**.

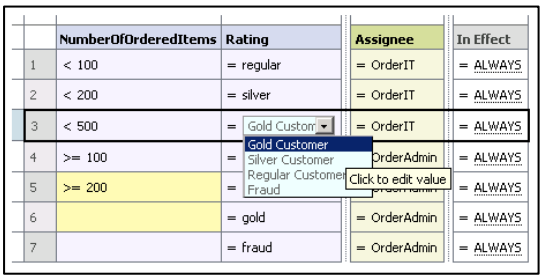
Note: If the project isn't listed in the navigation tree, click in the Navigation pane to refresh the tree structure.

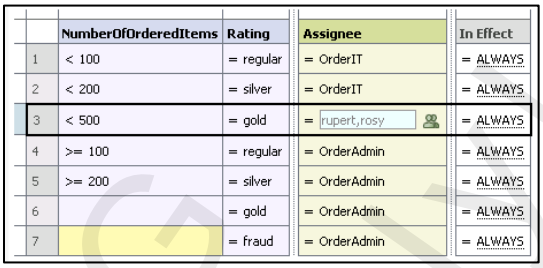
b. Select Decision Table **DetermineOrderTaskAssignee** from the Decision Tables portlet to open it in the RMC Decision Entity Editor.

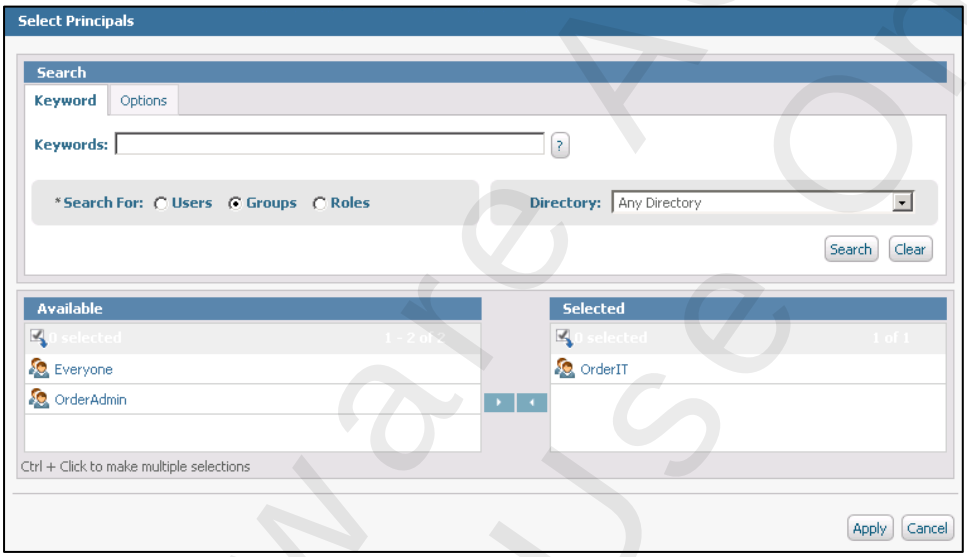
c. **Lock** the Decision Table.

d. **Add** an empty rule to the Decision Table and move it up to become rule number three.

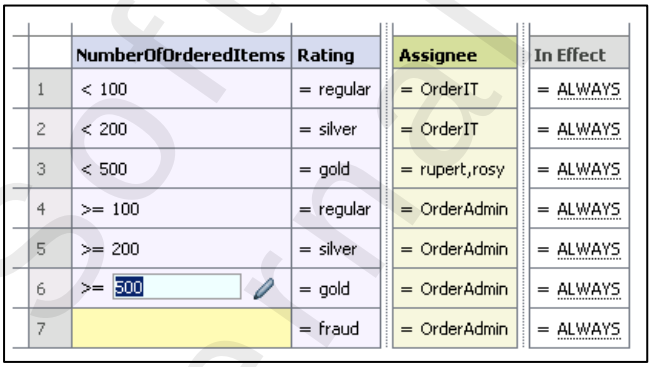
e. Configure the new rule to return MWS group **OrderIT** in case of an order from a **gold** customer with less than **500** ordered items. Make use of the Data Provider Service assigned to column **Rating** and the Principal Picker assigned to result column **Assignee**. For instance, select users **Rupert** and **Rosy** for rule three as assignees instead of group OrderIT.





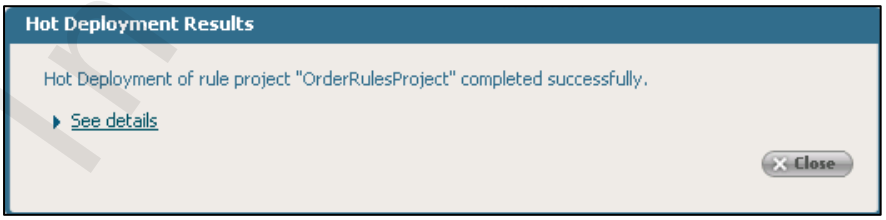


f. Modify Rule six to fire only for **NumberOfOrderedItems >= 500**.



g. **Save** and **unlock** the Decision Table **DetermineOrderTaskAssignee**.

h. Perform a **Hot Deploy** of your Rule project to your IS runtime.



19. Switch back to Designer to reload (synchronize) your modified Rule Project from the MWS repository as done in exercise 10:

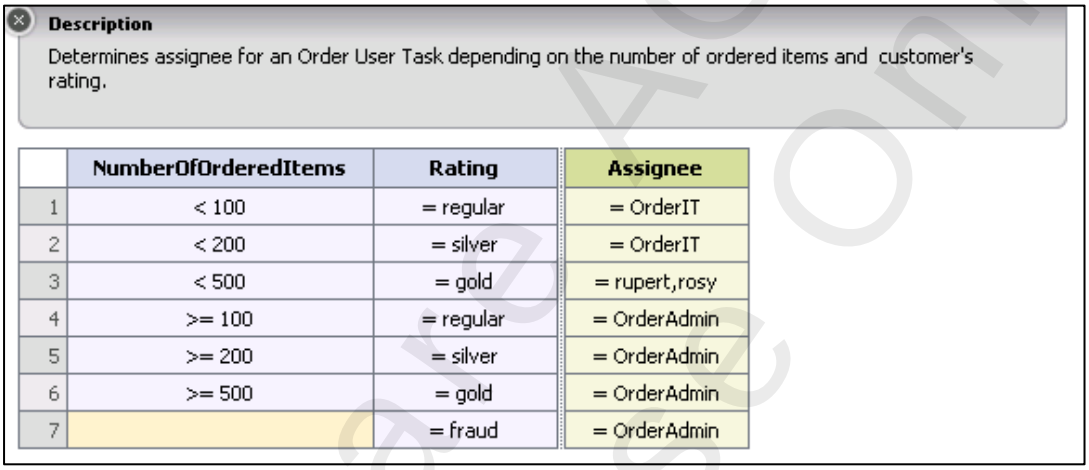
    a. Close all opened rule assets in Designer.

    b. Select **File** - **Import…** from Designer's menu bar. In the appearing wizard select import type **Software AG** - **Rule Project files from My webMethods Server repository**. Click **Next**.

    c. On the next panel select **MWS Server Repository** as source and the rule project **OrderRulesProject** as remote Rule Project.

    d. Click **Finish** to start the import. Allow overwriting.

    e. Check your changes have has been imported:



Description

Determines assignee for an Order User Task depending on the number of ordered items and customer's rating.

	NumberOfOrderedItems	Rating	Assignee
1	< 100	= regular	= OrderIT
2	< 200	= silver	= OrderIT
3	< 500	= gold	= rupert,rosy
4	>= 100	= regular	= OrderAdmin
5	>= 200	= silver	= OrderAdmin
6	>= 500	= gold	= OrderAdmin
7		= fraud	= OrderAdmin

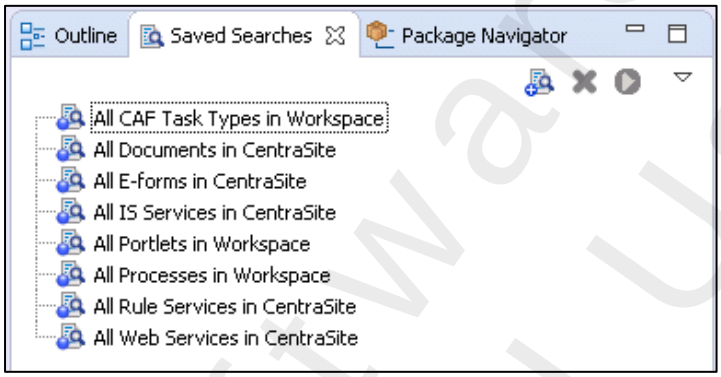
Exercise 18:
Business Rules Metadata

Objectives

In this exercise, you will use Software AG Designer to create and run a saved search against the local metadata.

Steps

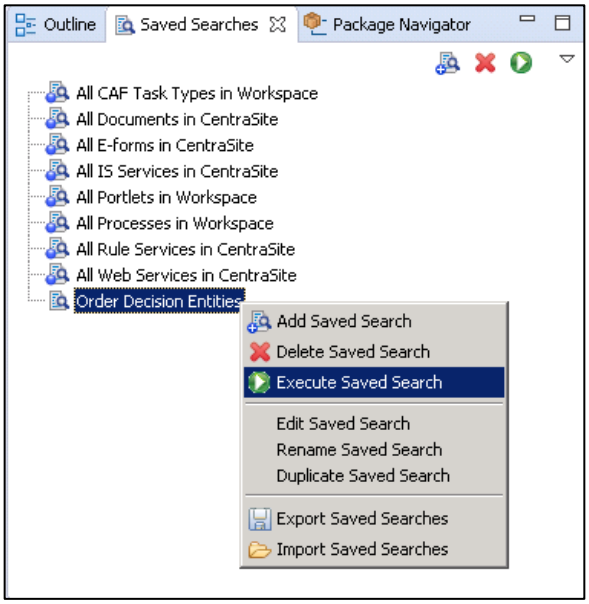
1. Ensure **Integration Server**, **My webMethods Server**, and **Optimize Analytic Engine** are running as Windows services.
2. Launch **Software AG Designer** and ensure you are in the **Rules Development** perspective.
3. In the Designer Preferences (**Window** > **Preferences**), drill down to **Software AG** > **Workspace Index**, ensure that Workspace Indexing is enabled (checkbox is unchecked). If Workspace Indexing was previously disabled, enable it and restart Designer.
4. Show the Saved Searches view in the Rules Development perspective (if not displayed yet, use **Window** > **Show View** > **Other…** > **Software AG** > **Saved Searches** to open it).



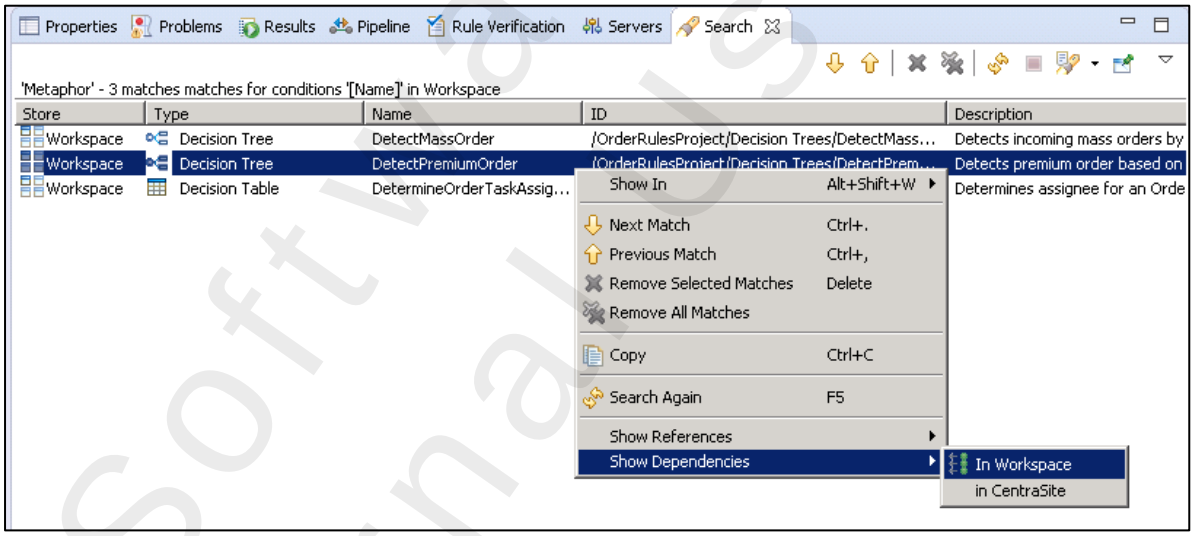
5. In the Saved Searches view, click to add a custom Saved Search. Provide the Saved Search details mentioned in the table below. Finally **save** your definitions.

Data	Value	
Search Name	**Order Decision Entities**	
Asset Type	**Decision Entity**	
Properties	Property	**Name**
	Condition	**contains**
	Value(s)	**order**
Match Condition	**Any**	
Search In	**Workspace**	

6. From the Saved Searches view, execute (run) the Order Decision Entities Saved Search.



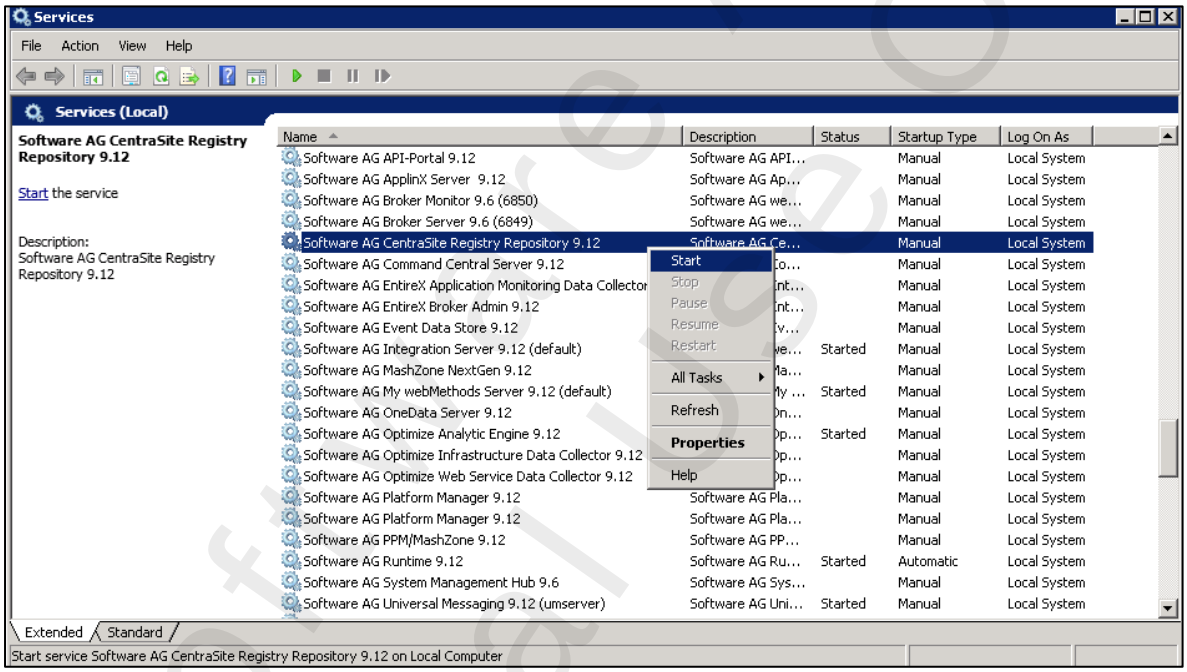
7. Inspect the results in the **Search** view. Select the match which is related to the **DetectPremiumOrder** Decision Tree. Based on the local metadata, retrieve their dependencies in the Workspace (local metadata).



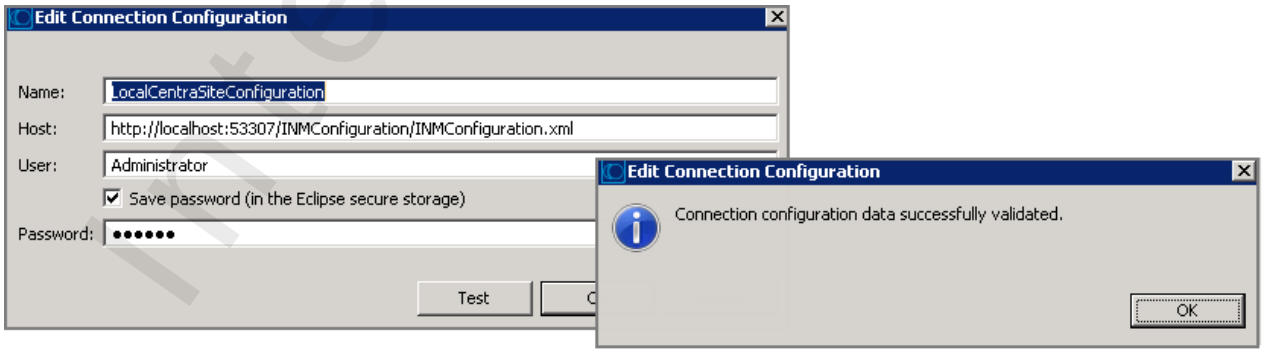
8. *Optional*: Launch the Windows Service Control Panel. Ensure the following Windows services are still up and running:
   - **hMailServer**
   - **Software AG Runtime 9.12**
   - **Software AG Universal Messaging 9.12 (umserver)**
   - **Software AG Integration Server 9.12 (default)**

   Note: In case of running a VM with little main memory available, you can stop the **Software AG My webMethods Server** and **Software AG Optimize Analytic Engine** Windows services for now.

9. Additionally start **Software AG CentraSite Registry Repository** as Windows service:

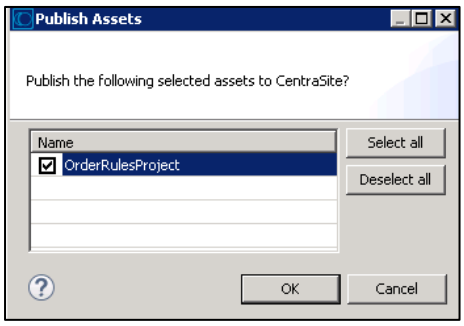


10. Return to Designer. Use **Window** > **Preferences** > **CentraSite** > **Connections** to verify the (pre-configured) connection to your local CentraSite Registry/Repository. Edit the existing connection, provide Administrator | manage as credentials, check to save the password, and use Test to check the connectivity:



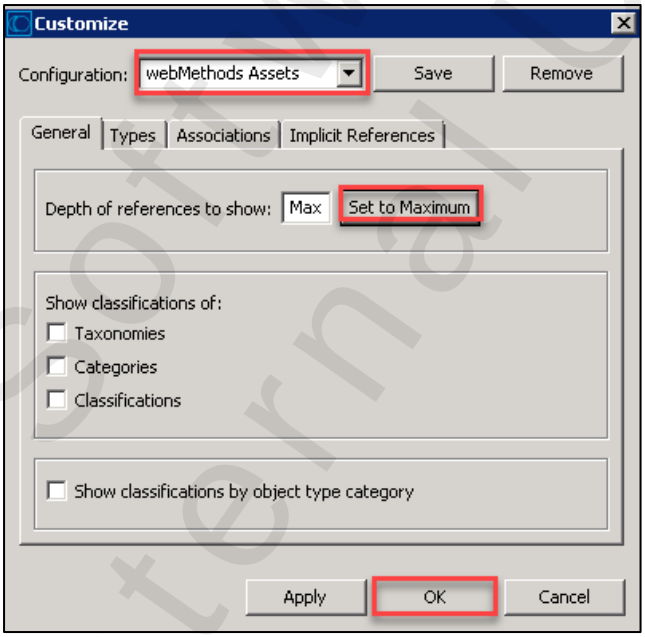
11. In Designer, right-click the Rules project **OrderRulesProject** in the Solutions view or in the Rules Explorer view, and select **Publish** to publish metadata to the CentraSite shared metadata repository.

> *Hint*: Publishing may take a while.



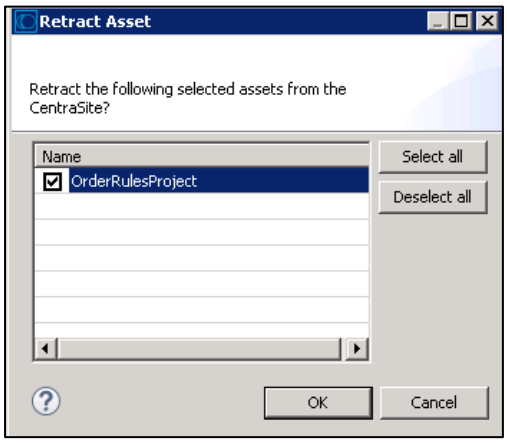
12. Switch to the **CentraSite Search and Browse** perspective.

    a. Open the **Registry Explorer** view. Because user Administrator is defined to belong to the **Default** Organization, your rule assets should be registered below this CentraSite organization. In the Registry Explorer view, drill down to your registered Rule Project **OrderRulesProject**. Right-click the Rule project and choose **Impact Analysis** -> **Customize** from the context menu. Select configuration **webMethods Assets** from the drop-down, set the depth of references to show to **maximum**, and hit **OK**.



    b. Right-click rule project **OrderRulesProject** in the Registry Explorer view again and choose **Impact Analysis** -> **Textual.** Inspect the results of the generated textual impact analysis in the Contents view.

13. Switch back to the **Rules Development** perspective. Right-click the Rules project **OrderRulesProject** in the Solutions view or in the Rules Explorer view, and select **Retract** to remove the meta-data of your Rules project from CentraSite.



14. *Housekeeping*:
Use the Windows Service Control Panel to stop the **Software AG CentraSite Registry Repository** service you started in step 9.
Additionally, start your **Software AG My webMethods Server** and **Software AG Optimize Analytic Engine** services again.