



SupportPac LA71: IBM WebSphere Operational Decision Management Integration for WebSphere Process Server

Getting started

Copyright

Copyright notice

© Copyright IBM Corp. 1987, 2012

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Trademarks

IBM, the IBM logo, ibm.com, WebSphere and ILOG are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

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

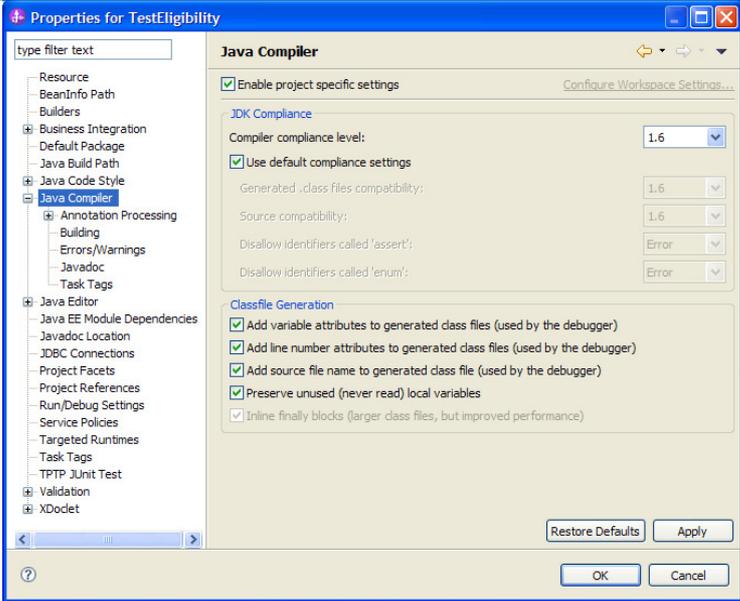
UNIX is a registered trademark of The Open Group in the United States and other countries.

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

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

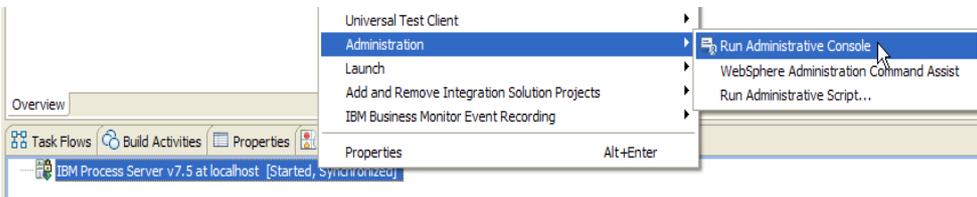
Key reference notes

Passwords, shortcuts, and properties

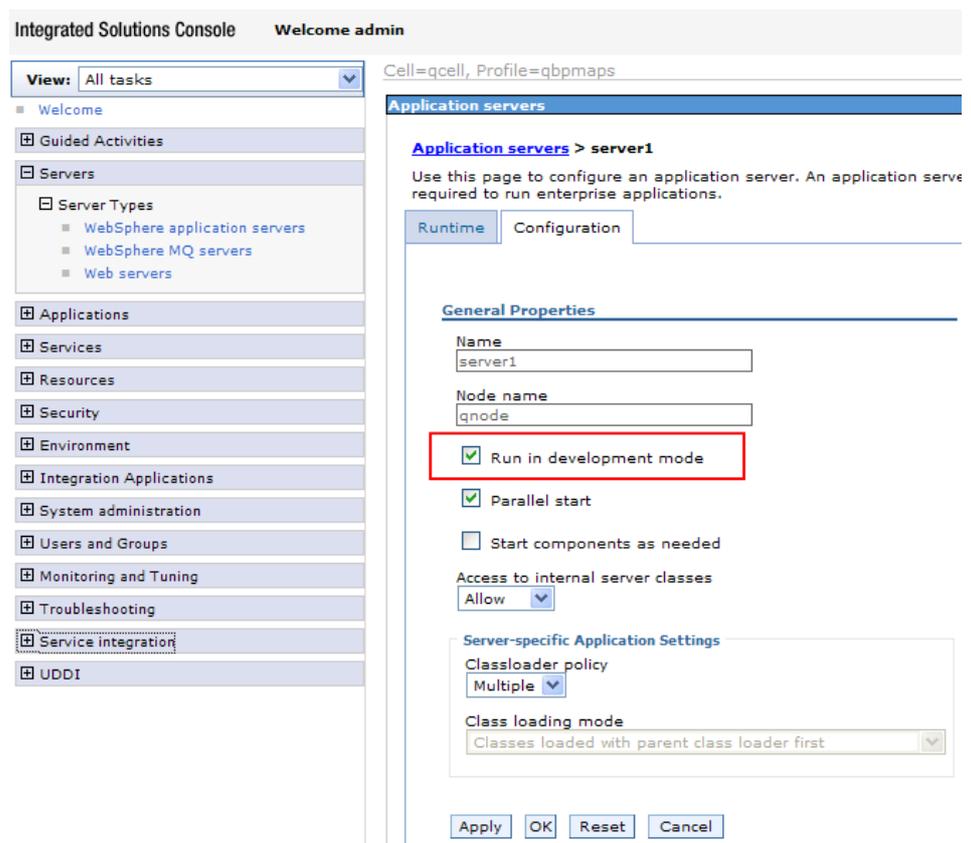
| | |
|--|---|
| Password information (credentials also provided in-line with the tutorial) | |
| Rule Execution Server Console | User: resAdmin Password: resAdmin |
| IBM Process Server | User: admin Password: admin |
| Names and locations of server processes, workspaces, etc. | |
| Tutorial files | [SupportPac LA71 Path]\tutorial\ |
| Completed workspaces for steps | [SupportPac LA71 Path]\tutorial\part_x*_PI.zip (Import into an empty Eclipse workspace using Project Interchange. Note that some directories exist but do not contain a workspace by design. |
| Empty tutorial workspace area | [SupportPac LA71 Path]\tutorial\workspace\ Example: C:\Program Files\IBM\WODM75\SupportPacLA71v1.7\tutorial\workspace\ |
| Properties | |
| Java compiler properties for the IID project containing artifacts from SupportPac LA71 |  |

Ensure that the instance of IBM Process Server is running in development mode

Run the Admin console



Navigate to this panel and check the box for development mode:



Ensure that the CEI event data store is enabled

Run the Admin console

Navigate to *Service integration > Common Event Infrastructure > Event service > Event services > Default Common Event Infrastructure event server* and check the box for enable event data store:

Cell=qcell, Profile=qbpmaps

Event service

[Event service](#) > [Event services](#) > **Default Common Event Infrastructure event server**

These settings define the properties for the event service.

Configuration

General Properties

* Scope
cells:qcell:nodes:qnode:servers:server1

* Name
Default Common Event Infrastructure event server

* JNDI name
com/ibm/events/configuration/event-server/Default

Description
The profile of the event server shipped with the Common Event Infrastructure.

Category

Enable event distribution

Enable event data store

Event data store EJB JNDI name
ejb/com/ibm/events/datastore/impl/DefaultDataStoreEJBLocalHome

Apply OK Reset Cancel

Initialize the Rule Execution Server (RES) databases

In a Web browser, open RES Console with the following URL <http://localhost:9080/res>

Login the RES console with **resAdmin** password **resAdmin**.

Initialize RuleApp persistence:

In **Step 1 – Welcome** – check the details and Click **Next**.

In **Step 2 – Database Schema** – ensure **derby** is selected and click **Next**.

In **Step 3 - Review Schema** – ensure Create SQL schema “ilog” is selected and click **Execute**.

In **Step 4 – Installation Manager Report** – click **Finish**.

Initialize Java XOMs persistence:

In **Step 1 – Welcome** – check the details and click **Next**.

In **Step 2 – Choose the database Schema** - ensure **derby** is selected and click **Next**.

In **Step 3 – Review Schema** – click **Execute**.

In **Step 4 – Report** – click **Finish**.

Tutorial objectives

There are many starting points when using WebSphere Operational Decision Management (WODM) with BPM and IBM Process Server in particular to build flexible and dynamic BPM applications. You can start in the Rule Developer role and capture the business logic in business rules or you can start in the IT Developer Role and create the business process. In this tutorial, we explore how to bring a rule project and an integration project together at different points in their life cycle. You will start as the Rule Developer, building business rules and testing. You will then take on the IT Developer role and define the process and integrate the rules into the business process. This will empower one to know how to make good decision based on needs at the time. In more general terms, tutorial participants should walk away with the following skills:

1. How to identify the required artifacts to build a rule project.
2. How to establish the best times to integrate a rule project (WODM) with an integration project (BPM).
3. How to take steps to make the process of integration easier.
4. How to establish clear criteria for integration choices—APIs, wizards, standards and the like.
5. How to engineer for reuse and minimize the cost of refactoring.
6. How to bring a rule project into an integration project using perspectives.
7. How to do end-to-end testing from IBM Integration Designer (IID) into a deployed WODM decision.
8. How to use CEI events that contain Decision Service traces for basic debugging requirements.
9. How to bring business users into the process life-cycle using WODM tooling.

Prerequisite knowledge

- General knowledge of WebSphere Operational Decision Management, IBM Integration Designer, and WebSphere Business Process Management
- General knowledge of business rules

Introduction

Business Process Management (BPM) and Decision Management (DM) are two technologies that are used to improve the agility, flexibility and efficiency of operational processes. Many people have questions about the differences between the two, or use the terms interchangeably—there are, however, clear differences in terms of the functionality and value of each. BPM is focused on defining, orchestrating and monitoring long running processes that are comprised of both people- and system-based activities. DM is focused on defining, maintaining and executing decision logic that is used at specific points within a process or as part of automated decisions within business systems. Overall, the complement, or synergy, between BPM and DM is about reaching further—expanding the breadth of problems that can be solved with a single solution. Both technologies share goals such as improving the efficiency and visibility of business processes, but they do so at different levels in a solution.

While it is true that BPM thrives on Service Oriented Architecture (SOA) and open architectures, it cannot by itself replace many of the systems it orchestrates. At the same time, the drive for transparency and encapsulation of business logic still remains for those existing systems. DM can open up existing systems, provide extended levels of transparency and allow both BPM and existing applications to share more than ever before. The synergy relies on the difference in orientation between BPM and DM.

First, consider the following comparison:

BPM orchestrates and improves business processes

- Flow orientation
- Human orientation
- Crosses system and organization boundaries
- Process-oriented transparency—driving awareness and improvement of business processes to an increased set of stakeholders
- Long and short running

DM expresses and automates business decisions

- Data orientation
- Encapsulates to a single boundary of a decision
- Promotes reuse for any client (BPM and otherwise)
- Decision-oriented transparency—increasing visibility of decisions driving critical business applications and processes
- Straight through processing

While BPM focuses on the overall business process, DM automates and encapsulates business decisions at every level of a solution. When they work together, BPM can resolve problems related to silos and inefficient human interactions with systems and each other, while DM employs data validation, transformation and selection to automate tasks or make a step in a process (or application) more productive and predictable.

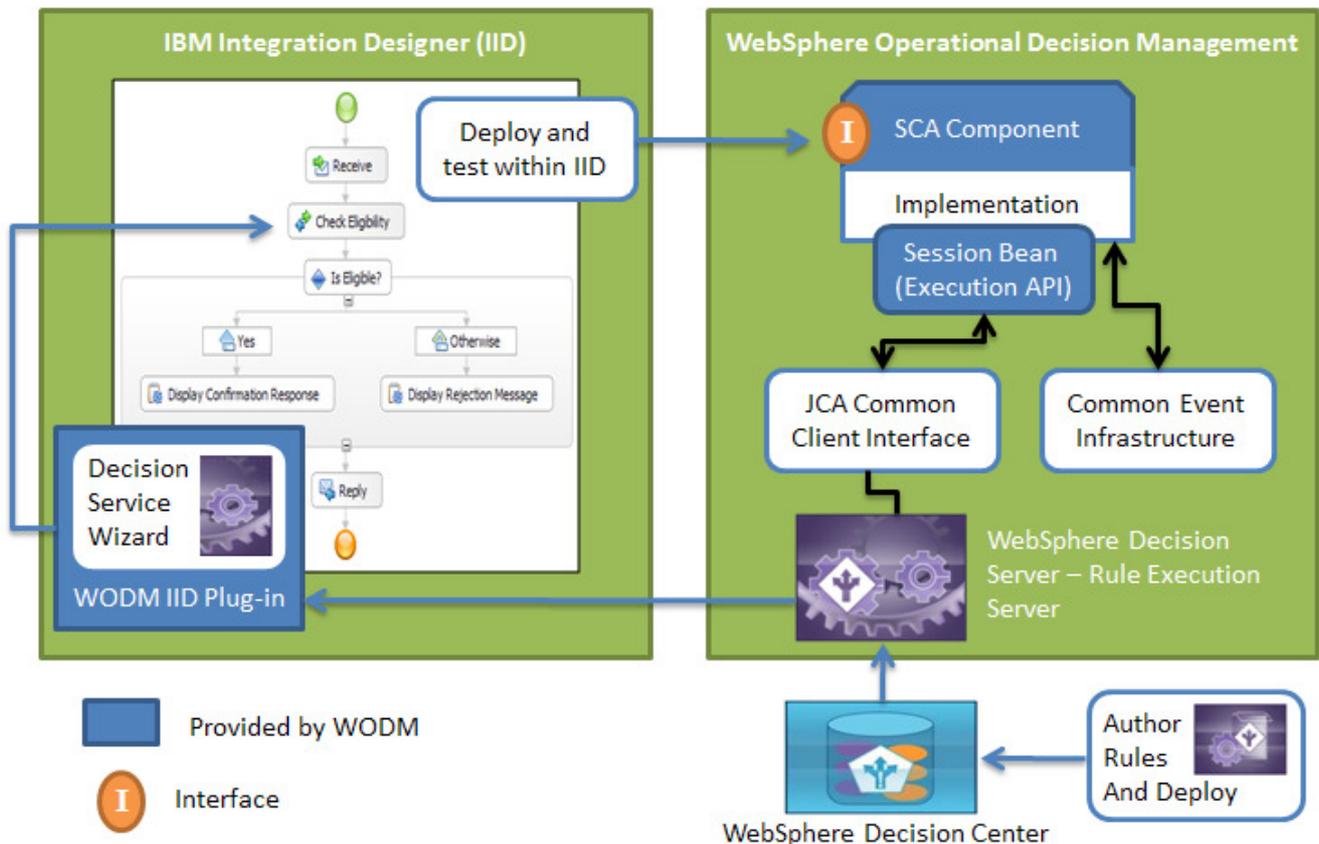
While the orientations of the technologies may be orthogonal, they depend upon a contractual relationship that is typical in an SOA. For example, in the case of SOAP, the service signature is

defined by a WSDL, an XML schema (XSD) and a policy. The XSD may in turn be shared among all upstream and downstream integration points within a business process, and is a critical artifact in the integration between BPM and a DM.

Managing a decision and authoring rules is primarily focused on data. At design time, a DM presents decision content to business users for authoring and management, since they understand the content directly and make the better owner. They also benefit from separating the DM life cycle from the BPM life cycle to support rapid changes to decisions that are owned by the business. Business process models tend to change less frequently than decisions, and the overall business goals may be maintained while operationally the frequency of change continues at different rates between all of the participating life cycles.

For integrating WODM and business process created with IBM BPM, there are two options. The first option is to integrate through Web Services. All ruleapps are exposed on the Rule Execution Server (RES) as Web Services with WSDL interfaces and these interfaces can be used directly in BPM applications with a Web Service import. The second option is to use the support provided by WebSphere Operational Decision Management LA71 SupportPac. With the SupportPac, the specific rulesets of the ruleapps can be selected to be exposed as SCA components where they can then be wired to other components.

The scenario for this tutorial is to build an insurance quote process that uses WODM to capture the business logic in determining whether a driver is approved for insurance. While the quote process and the eligibility rules are fairly simplistic, they show the integration and can be expanded for more complex processes and rules.

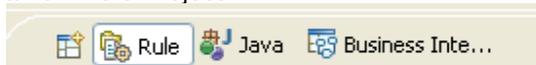


Part 1: Creating a rule project with an existing XSD model

The existence of a reusable model is one of the single more important artifacts for both a project in Rule Designer and IBM Integration Designer. This data represents a contract for many integration points. Moreover, the more mature a model is, then the more refactoring of project content can be avoided.

Let's start by creating a rule project from an existing XSD:

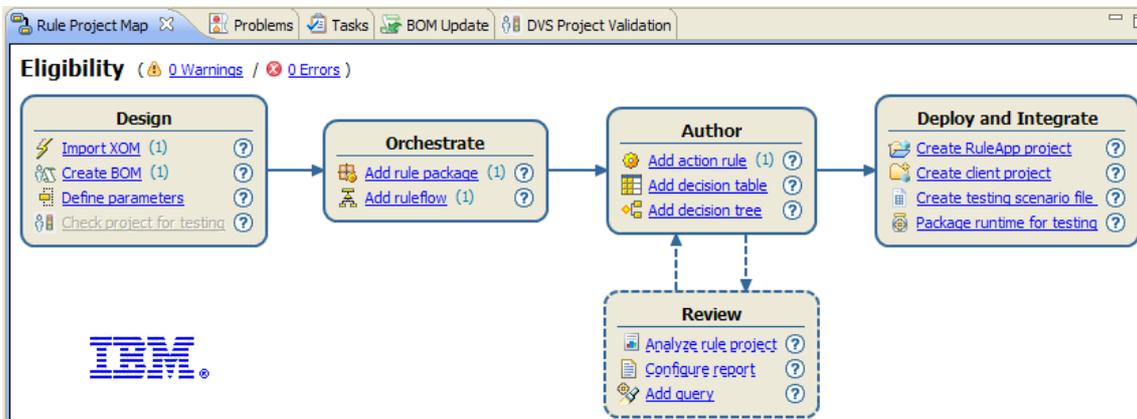
1. Creating a new Rule Project:



- a. Click **Rule** and go to **Rule Perspective**.
- b. Go to **File > New > Rule Project**.
- c. Select **Rule Project with a BOM**
- d. Enter **Eligibility** as the project name. Click **Next** two times.
- e. In the Rule Project XOM settings wizard page, click **Add External XSD**. Point to **[SupportPac LA71 Path]\tutorial\part_1\InsuranceSampleObjectModel.xsd**. Click **Finish**.

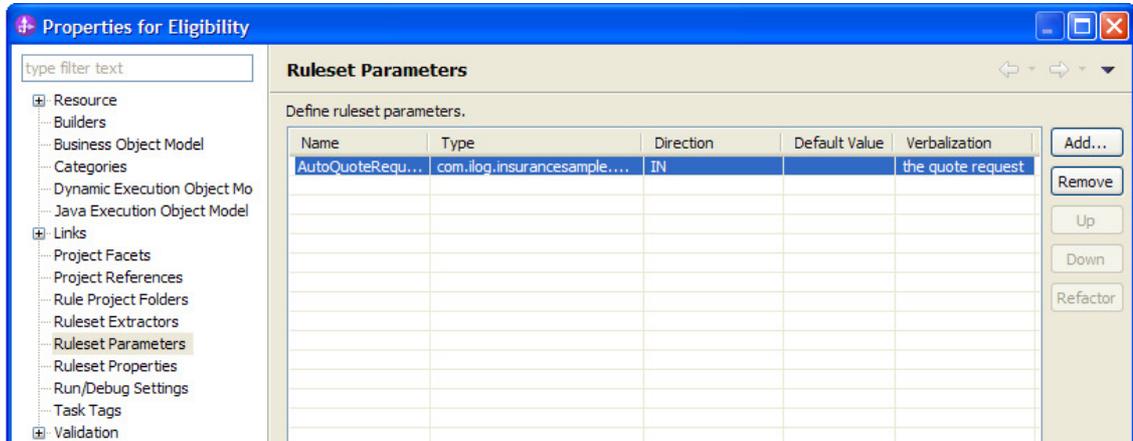
You should see a newly created project named Eligibility. This Rule Project corresponds to a Decision Service Eligibility which is consumed by the business process.

2. Select **Eligibility** Rule Project in the Rule Explorer. You will see the Project Map:



- a. Click **Define parameters** link under the Design box. You will define the signature of the Eligibility service in this step.
- b. Click **Add**.
- c. Enter **AutoQuoteRequest** as the name of the Ruleset Parameter.
- d. Enter **Select com.ilog.insurancesample.AutoQuoteRequest** as the type. This type corresponds to the AutoQuoteRequest XML type defined in the schema.
- e. Select **IN** as Direction to indicate this is the input of the service.

- ___ f. Enter **the quote request** as the verbalization. You will write a rule to access the parameter with the term.



- ___ g. Click Add again. Enter **EligibilityResponse** as the name of the Ruleset Parameter.
 ___ h. Enter Select **com.ilog.insurancesample EligibilityResponse** as the type.
 ___ i. Select **OUT** as Direction to indicate this is the decision result.
 ___ j. Enter **new com.ilog.insurancesample.EligibilityResponse()** as the Default Value to initialize the output parameter.
 ___ k. Enter **the eligibility response** as the verbalization.

| Ruleset Parameters | | | | |
|----------------------------|--|-----------|--|--------------------------|
| Define ruleset parameters. | | | | |
| Name | Type | Direction | Default Value | Verbalization |
| AutoQuoteRequest | com.ilog.insurancesample.AutoQuoteRequest | IN | | the quote request |
| EligibilityResponse | com.ilog.insurancesample.EligibilityResponse | OUT | new com.ilog.insurancesample.EligibilityResponse() | the eligibility response |

- ___ l. Enter the following rule in **businessRule** artifact with BAL editor. It looks like the follows:

```

definitions
set driver to a driver in the drivers of 'the quote request' ;

if
  the number of accidents of driver is more than 2
then
  make it true that 'the eligibility response' is eligible ;
  set the main message of 'the eligibility response' to
  "The driver is eligible for the high-risk program";
  
```

Business Rule: businessRule

General Information

Name : businessRule

Category Filter

Categories: Any. [Edit](#)

Documentation

Code

```
definitions
set driver to a driver in the drivers of 'the quote request' ;

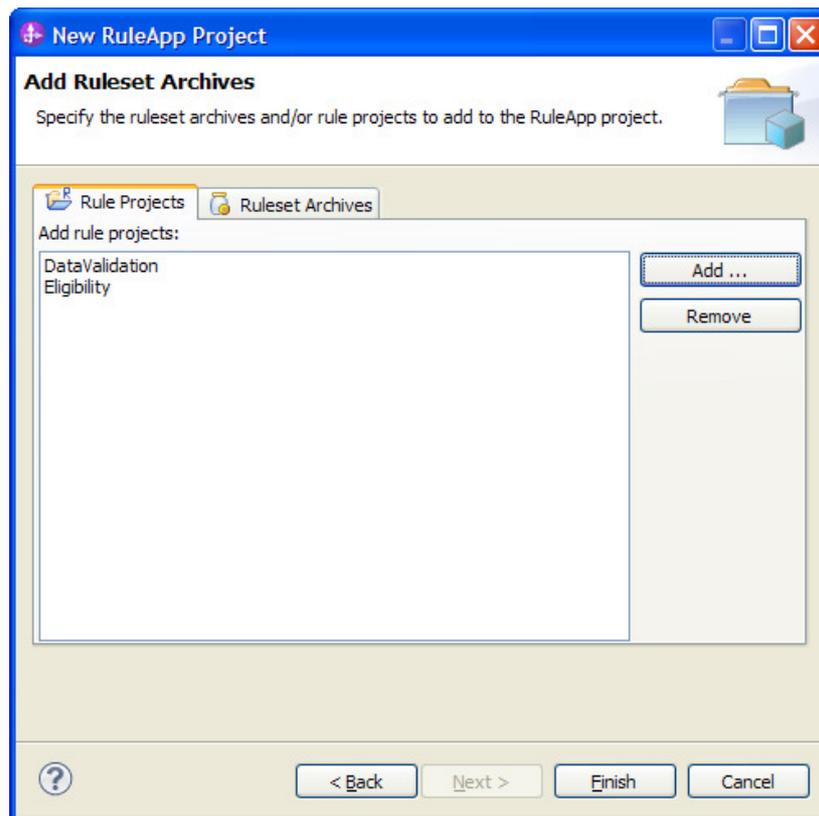
if
  the number of accidents of driver is more than 2
then
  make it true that 'the eligibility response' is eligible ;
  set the main message of 'the eligibility response' to "The driver is eligible for the high-risk program";
```

- ___ 3. Now, create DataValidation Rule Project. Repeat Step 1a to Step 1e but enter **DataValidation** as the project name.
- ___ a. Click on DataValidation Project. Click Define ruleset parameters in the Rule Project Map.
 - ___ b. Add **AutoQuoteRequest** ruleset parameter.
 - ___ c. Enter Select **com.ilog.insurancesample.AutoQuoteRequest** as the type.
 - ___ d. Select **IN** as Direction to indicate this is the input of the service.
 - ___ e. Enter **the quote request** as the verbalization.
 - ___ f. Add **DataValidationResponse** output ruleset parameter.
 - ___ g. Enter Select **com.ilog.insurancesample.ValidationResponse** as the type.
 - ___ h. Select **OUT** as Direction to indicate this is the input of the service.
 - ___ i. Enter **new com.ilog.insurancesample.ValidationResponse()** as the default value.
 - ___ j. Enter **the validation response** as the verbalization.

- ___ 4. Enter the following rule in the **businessRule** artifact in the **DataValidation** project.

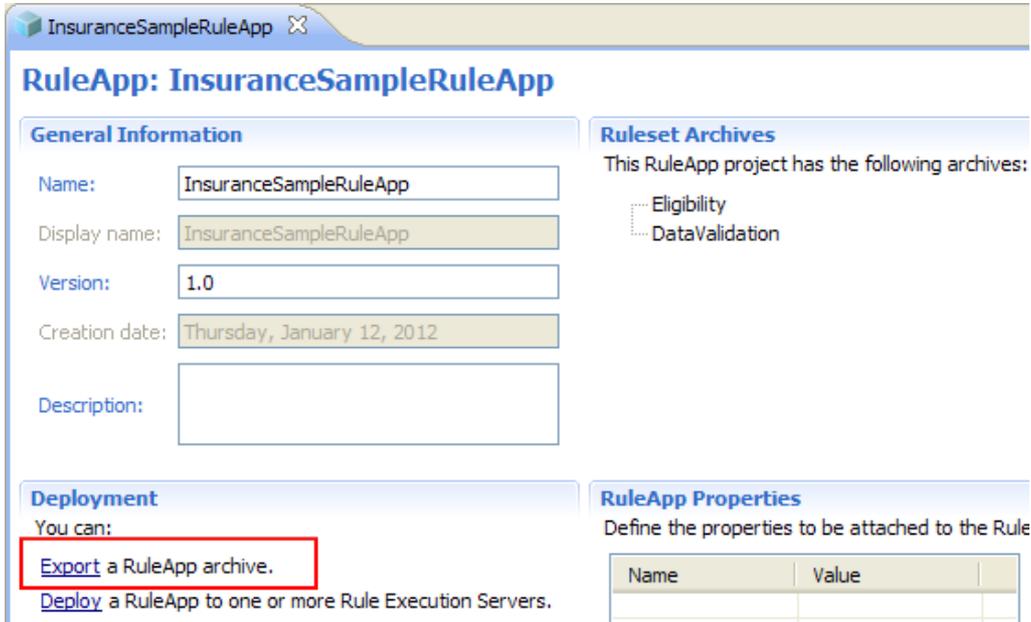
```
definitions
  set driver to a driver in the drivers of 'the quote
  request' ;
if
  the number of accidents of driver is less than 0
then
  make it false that 'the validation response' is
  validated ;
  set the main message of 'the validation response' to
  "The number of accidents should not be negative" ;
```

- ___ 5. Now, create a RuleApp Project for deployment.
- ___ a. Click **File > New > Project...**
 - ___ b. Select **RuleApp project**.
 - ___ c. Enter **InsuranceSampleRuleApp** for the RuleApp project name.
 - ___ d. In the Add ruleset archive wizard page, click **Add**.
 - ___ e. Select both **DataValidation** and **Eligibility** Rule Project. Press **Ctrl** for multiple select.
 - ___ f. Click **OK**.

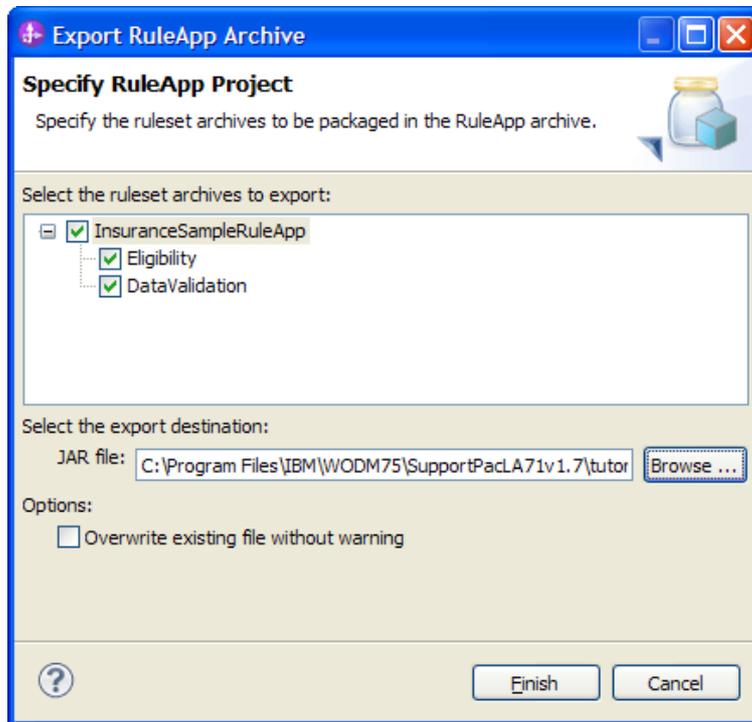


- ___ g. Click **Finish**.

- ___ 6. Export the RuleApp archive for integration with the business process.
 - ___ a. Click **Export** a RuleApp archive.



- ___ b. Enter **[SupportPac LA71 Path]\tutorial\part_2\InsuranceSampleRuleApp.jar** for export destination.



- ___ c. Click **Finish**.

Part 2: Deploy the archive to Rule Execution Server (RES)

A rule application is deployed to Rule Execution Server as an archive—a JAR file that usually bears the name of the rule project. The archive is a portable representation of the decision and it can be used to move the decision between environments and/or be used as a hand-off artifact to System Administrators where there is a separation of responsibility between developers and those that have direct access to production servers.

- ___ 1. Start the server.
 - ___ a. In IBM Integration Designer, select the **Servers** view and select **IBM Process Server v7.5**. The Rule Execution Server has already been configured on the IBM Process Server test server.
 - ___ b. Right-click and select **Start**.

- ___ 2. Deploy the RuleApp with the business rules.
 - ___ a. In a Web browser, open RES Console with the following URL <http://localhost:9080/res>.
 - ___ b. Login with **Username** resAdmin and **Password** resAdmin. Click **Sign In**.
 - ___ c. Click **Deploy RuleApp Archive**.

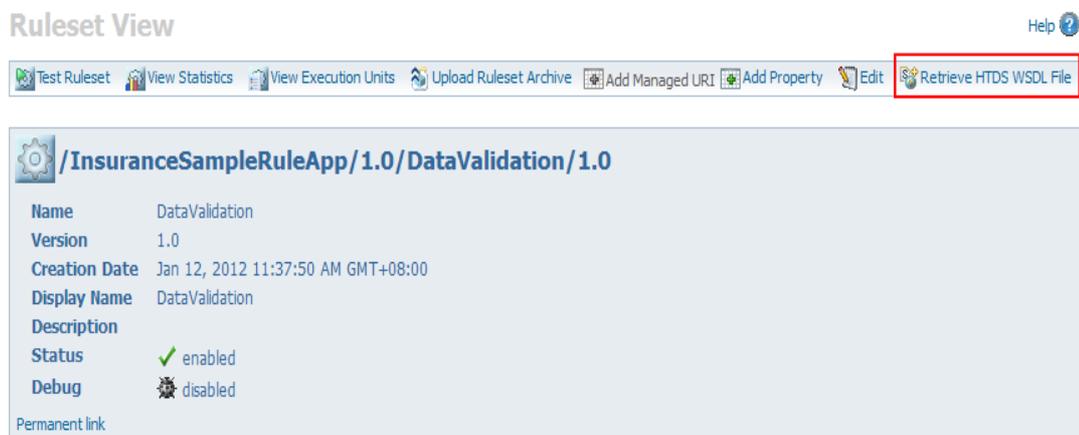


- ___ d. Click **Browse** and select **[SupportPac LA71 Path]\tutorial\part_2\InsuranceSampleRuleApp.jar**.
- ___ e. Click **Deploy** and then click **OK**.

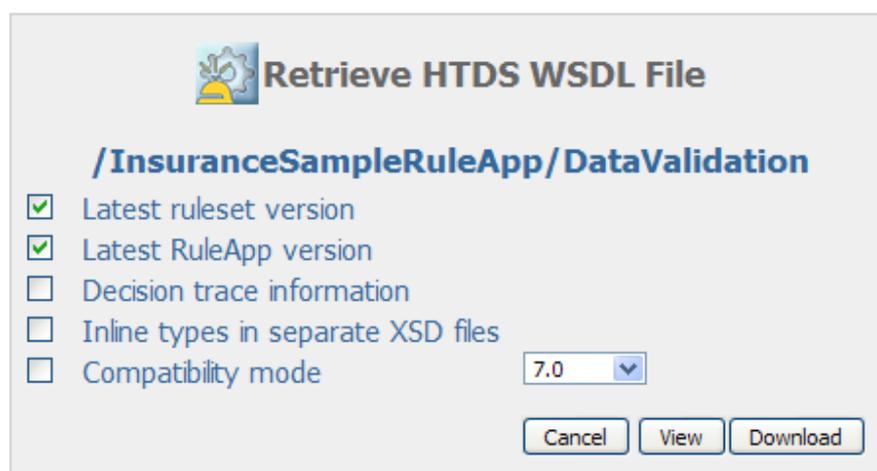
Part 3: Assemble WODM SCA component with Web Services

In this part, the task is to generate an SCA component from a web service and test. The purpose of the test is to establish a feedback-loop for observing the behavior of the decision and make required modifications. The test will be done from IBM Integration Designer (IID) using Hosted Transparent Decision Services (HTDS) provided by Rule Execution Server (RES).

- ___ 1. Log into Rule Execution Server (<http://localhost:9080/res/> User: resAdmin Password: resAdmin)
 - ___ a. Click **Explorer**.
 - ___ b. Click **InsuranceSampleRuleApp**.
 - ___ c. Click **DataValidation**.
 - ___ d. Click on **Retrieve HTDS WSDL File**.



- ___ e. Select **Latest ruleset version** and **Latest RuleApp version**. Then click **Download** to save the WSDL to the local file system. You may also find a copy of this already in **[SupportPac LA71 Path]\tutorial\part_3**.

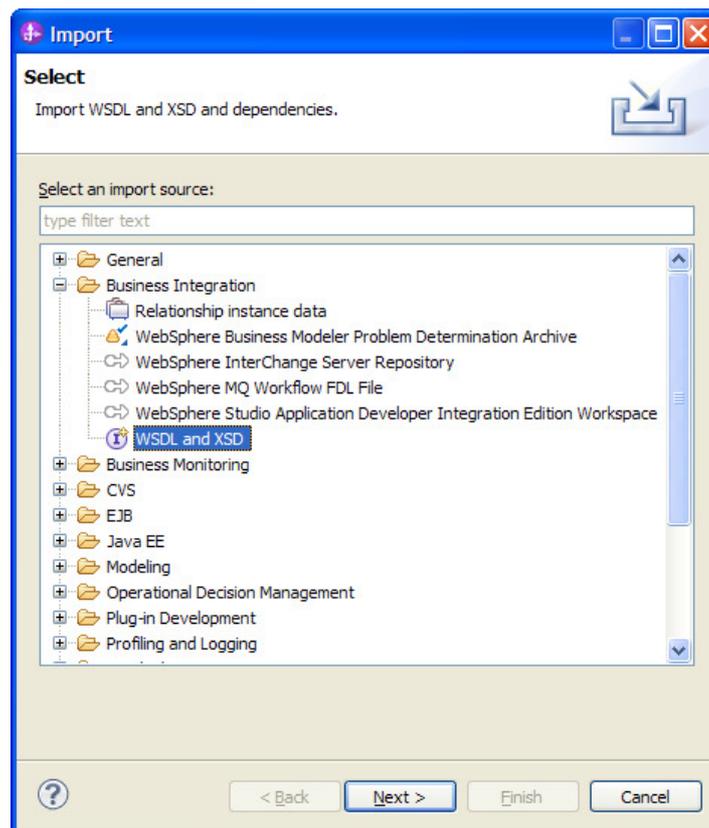


- ___ 2. Follow the steps in (1) above for “Eligibility”.

- ___ 3. Change the perspective within IID to “Business Integration”.
 - ___ a. From the top menu, click **Window** → **Open Perspective** → **Other**.
 - ___ b. Select **Business Integration** from the list.

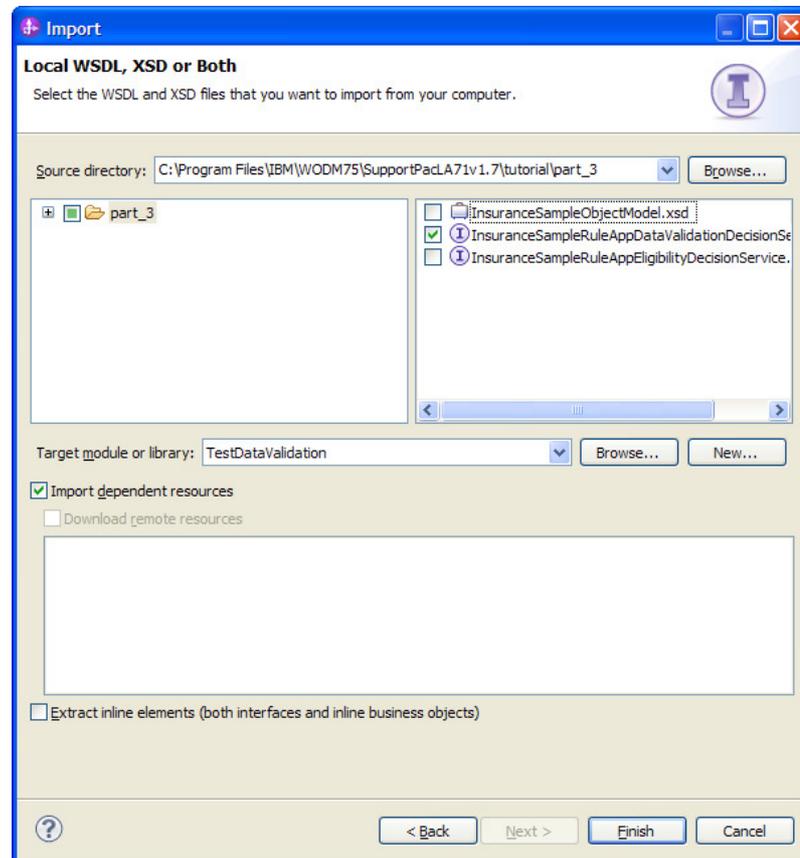
- ___ 4. Create a test module for DataValidation.
 - ___ a. Click **File** → **New** → **Module** from the menu.
 - ___ b. Give the module the following name: **TestDataValidation**.
 - ___ c. Click **Finish**.

- ___ 5. Import the WSDL files and XSD.
 - ___ a. From the top menu click **File** → **Import**.
 - ___ b. Select **Business Integration** → **WSDL and XSD**.

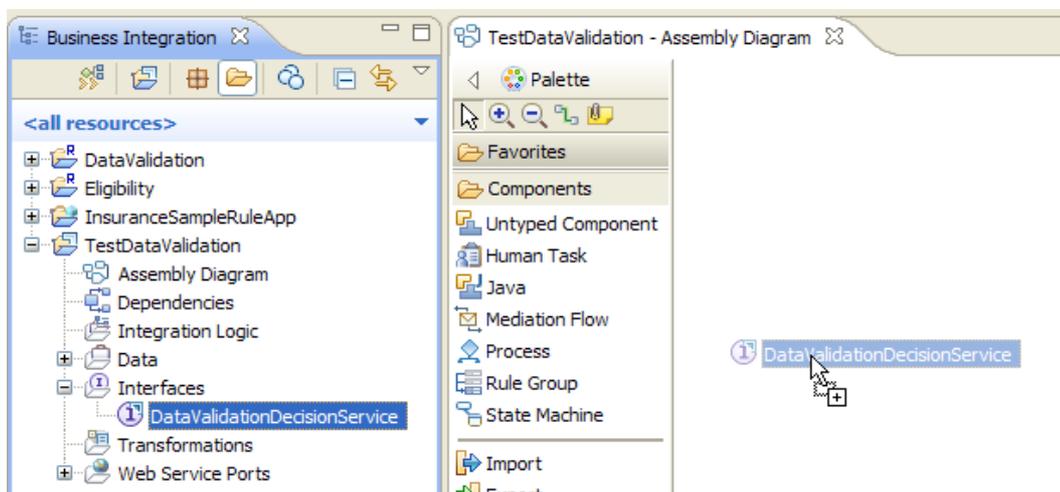


- ___ c. Click **Next**.
- ___ d. Select **Local WSDL or XSD file, or both**.
- ___ e. Click **Next**.

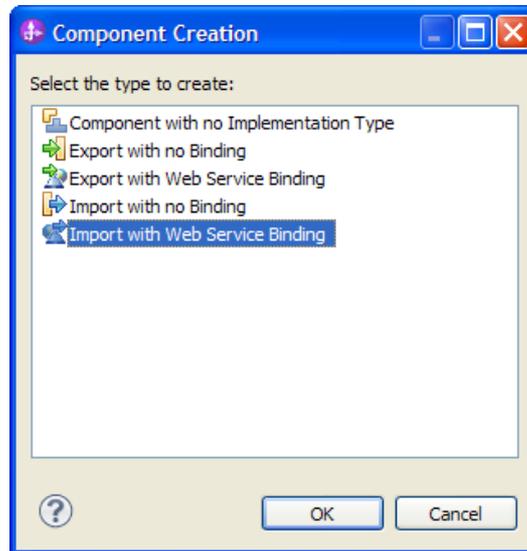
- ___ f. Provide the path to your WSDL and XSD files. You can use the artifacts provided by the tutorial: **[SupportPac LA71 Path]tutorial\part_3**.
- ___ g. Select the DataValidation WSDL file, InsuranceSampleRuleAppDataValidationDecisionService.wsdl.
- ___ h. Select the module: TestDataValidation. Click **Finish**.



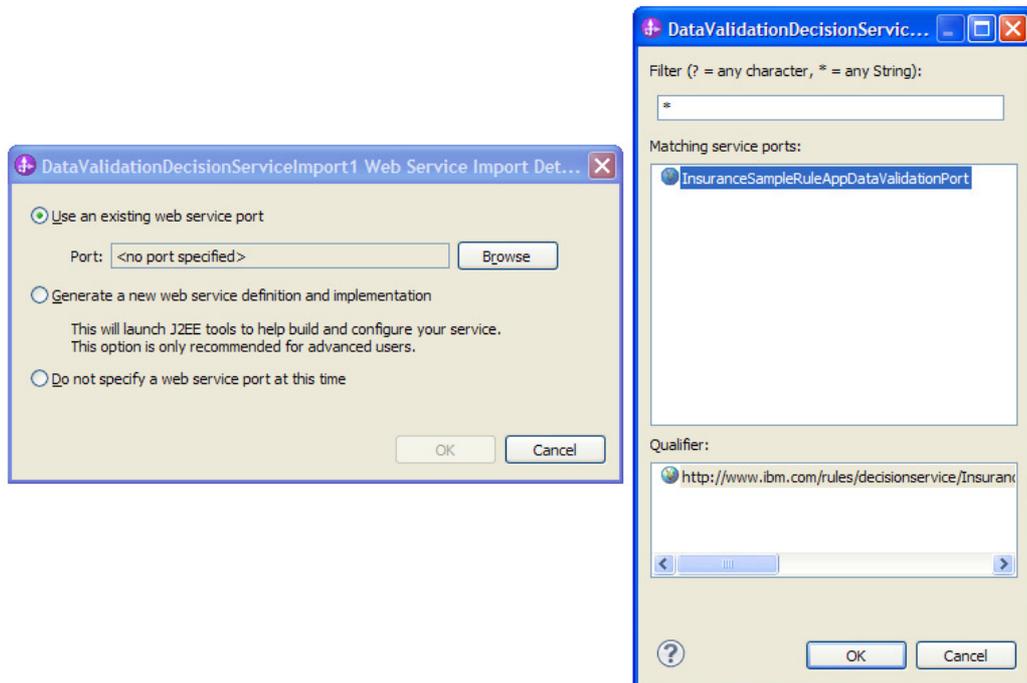
- ___ 6. Implement the web service import.
 - ___ a. Drag the DataValidationDecisionService interface onto the assembly diagram (you may need to expand the tree to see it).



___ b. Select **Import with Web Service Binding** in the dialog box:

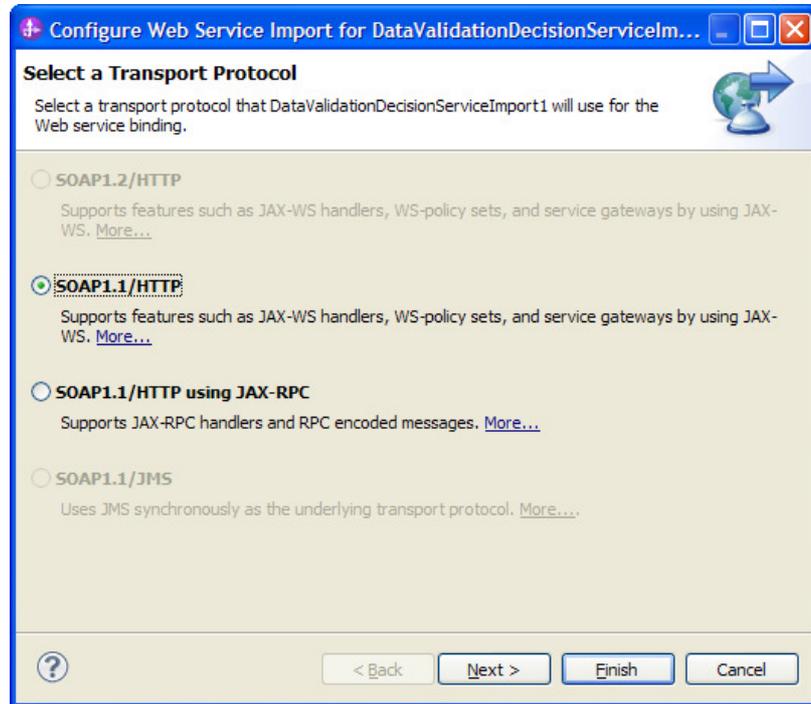


- ___ c. Select **Use an existing web service port**.
- ___ d. Click **Browse**.
- ___ e. Select the existing port in the window like the following:



___ f. Click **OK**.

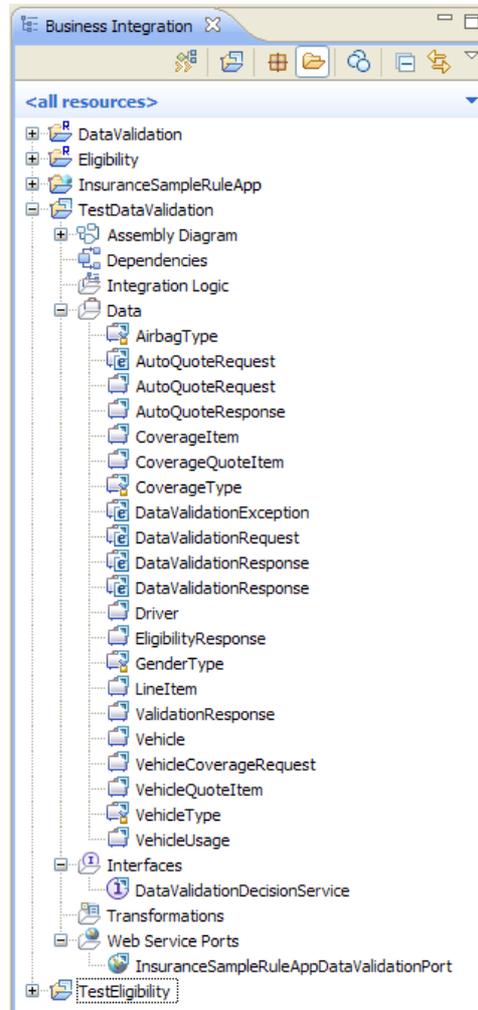
___ g. Select **SOAP1.1/HTTP using JAX-RPC**



___ h. Click **Finish**.

___ i. Click **OK**.

- ____ 7. Repeat the steps for a new module called “TestEligibility”. At this point, the project should contain several modules and look something like the following:



- ____ 8. Run the test of a module containing a decision:
____ a. Right click on the module “TestEligibility” and click on **Test → Test Modules**.

__ b. For the Initial request parameters section, enter **15** for the driver **Age**

Initial request parameters:

Value editor XML editor

| Name | Type | Value |
|--------------------|----------------|---------|
| AutoQuoteRequest | AutoQuoteRe... | [ab] |
| Driver * | Driver[] | 60 |
| Driver[0] | Driver | [ab] |
| DriverID * | int | [ab] 0 |
| FirstName * | string | [ab] |
| LastName * | string | [ab] |
| Age * | int | [ab] 15 |
| StateOfResidence * | string | [ab] |

__ c. For the Initial request parameters section, enter **3** for the driver **NumberOfAccidents**.

Initial request parameters:

Value editor XML editor

| Name | Type | Value |
|-----------------------------|---------|------------|
| CompletedDriversEdCourse | boolean | [ab] false |
| VehicleVandalizedOrStolen * | boolean | [ab] false |
| LicenseSuspendedOrRevoked * | boolean | [ab] false |
| DUI * | boolean | [ab] false |
| NumberOfAccidents * | int | [ab] 3 |
| NumberOfTrafficTicket * | int | [ab] 0 |

__ d. Under **VehicleCoverage**, right-click on **Vehicle** and select **Set To > Default**.

Initial request parameters:

Value editor XML editor

| Name | Type | Value |
|-------------------------|-------------------|------------|
| DUI * | boolean | [ab] false |
| NumberOfAccidents * | int | [ab] 3 |
| NumberOfTrafficTicket * | int | [ab] 0 |
| VehicleCoverage * | VehicleCoverage[] | [ab] |
| VehicleCoverage[0] | VehicleCoverage | [ab] |
| Vehicle * | Vehicle | [ab] |
| AnnualMileage * | int | [ab] 0 |

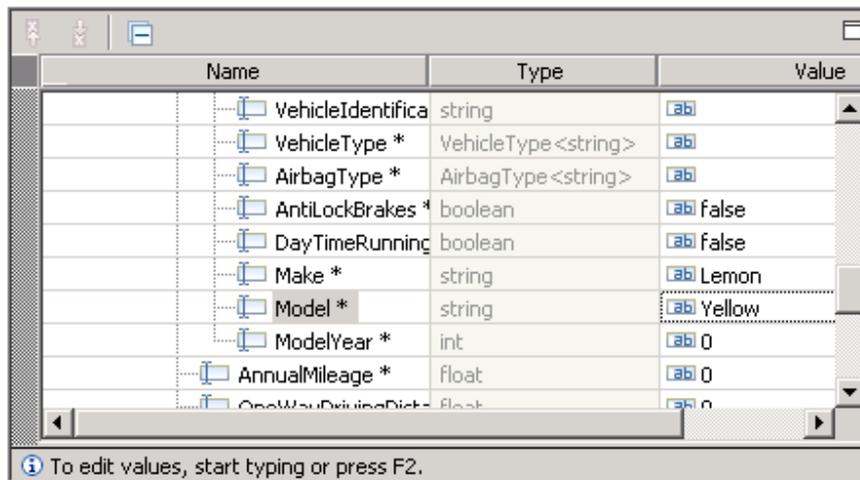
Context menu for Vehicle:

- Select All
- Set To
 - Value...
 - Default
 - Unset
 - Null
- Set Required to Default
- Add Value to Pool...
- Use Value from Pool...
- Import from File...
- Export to File...
- Use Derived Type...
- Select Element...

___ e. Under Vehicle, for the Make enter **Lemon** and for the Model enter **Yellow**.

Initial request parameters:

Value editor XML editor



The screenshot shows a 'Value editor' window with a table of request parameters. The table has three columns: Name, Type, and Value. The parameters are listed as follows:

| Name | Type | Value |
|-------------------|---------------------|-------------|
| VehicleIdentifica | string | [ab] |
| VehicleType * | VehicleType<string> | [ab] |
| AirbagType * | AirbagType<string> | [ab] |
| AntiLockBrakes * | boolean | [ab] false |
| DayTimeRunning | boolean | [ab] false |
| Make * | string | [ab] Lemon |
| Model * | string | [ab] Yellow |
| ModelYear * | int | [ab] 0 |
| AnnualMileage * | float | [ab] 0 |
| OneWayDrivingDist | float | [ab] 0 |

At the bottom of the window, there is a note: **To edit values, start typing or press F2.**

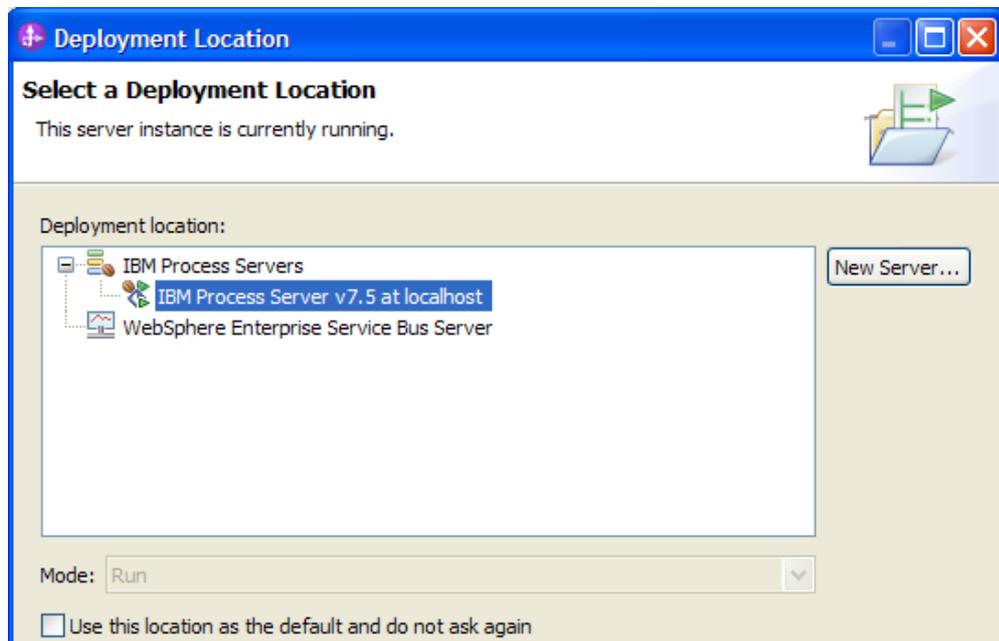
___ f. Click the **Continue** button.

Events

This area displays the events in a test trace. Select an General Properties and Detailed Properties sections. [M...](#)



___ g. Insure the Deployment location is **IBM Process Server v7.5 at localhost** and click **Finish**.



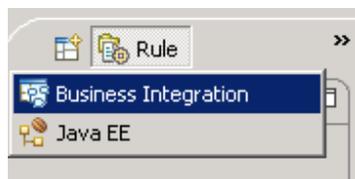
___ h. For the User Login screen, accept the existing values and click **OK**.

___ i. Inspect the data returned for eligibility information and the main message. Repeat the step for the TestDataValidation module if desired.

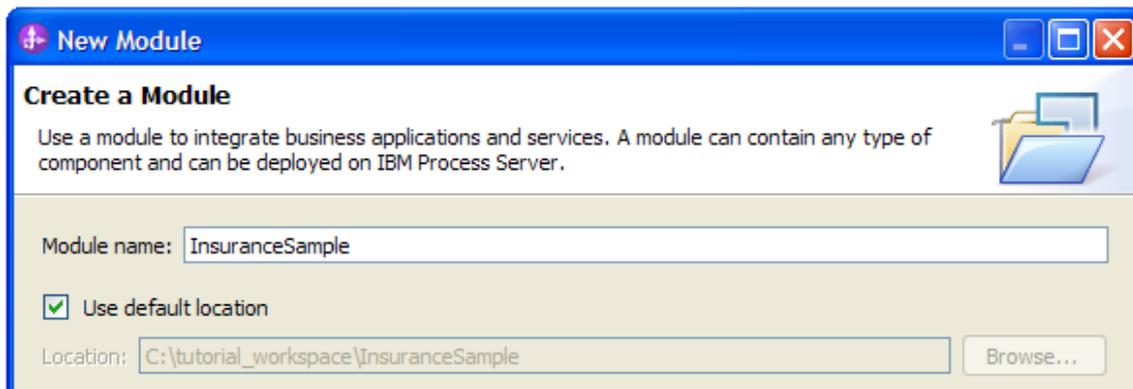
Part 4: Assemble WODM SCA component with SupportPac LA71

For this exercise we are going to integrate using the SCA component. It is natural for the testing of the Decision Service to be done with the integrated Test Component in preparation of integrating with the business process.

- ___ 1. Start by creating a module for the SCA components. This module will also be used for the business process.
 - ___ a. Select **Window > Open Perspective > Business Integration** or **Business Integration** from the quick view.



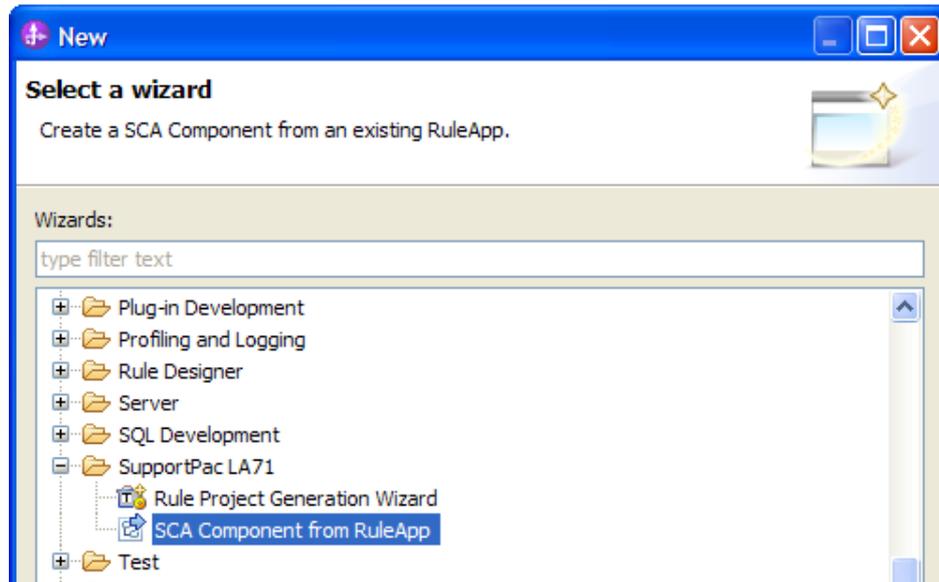
- ___ b. Select **File > New > Module**.
- ___ c. For the Module name, enter **InsuranceSample**.



- ___ d. Click **Finish**.

- ___ 2. Create the SCA Component for the WODM.
 - ___ a. Select **File > New > Others**.

___ b. Expand **SupportPac LA71** and select **SCA Component from RuleApp**. Click **Next**.

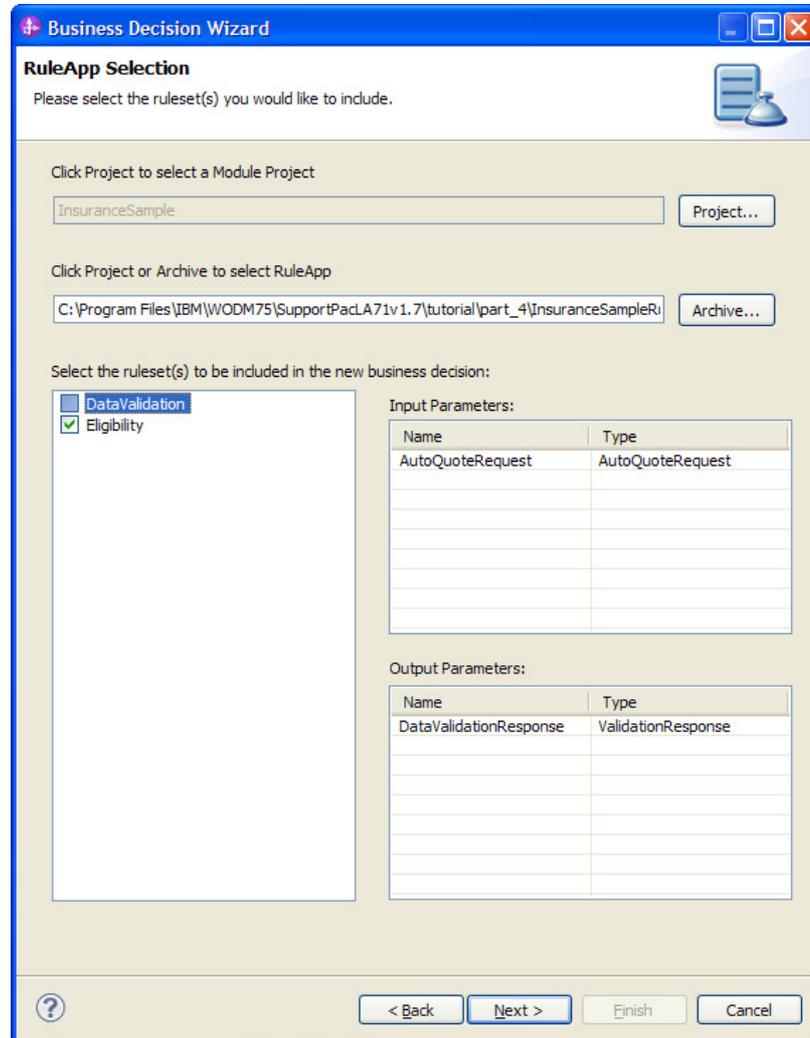


___ c. In the Decision Service Wizard, if the InsuranceSample module project was not selected click **Project...** and select **InsuranceSample**. Click **OK**.

The RuleApp must be provided as input for creation of the SCA Component. Click **Archive...** and select the RuleApp JAR you exported in Part 2 or pick up the jar at **[SupportPac LA71 Path]\tutorial\part_4\InsuranceSampleRuleApp.jar** provided for convenience.

The wizard automatically detects the rulesets on the left panel and they are checked by default. It also reads input and output parameters for the rule application.

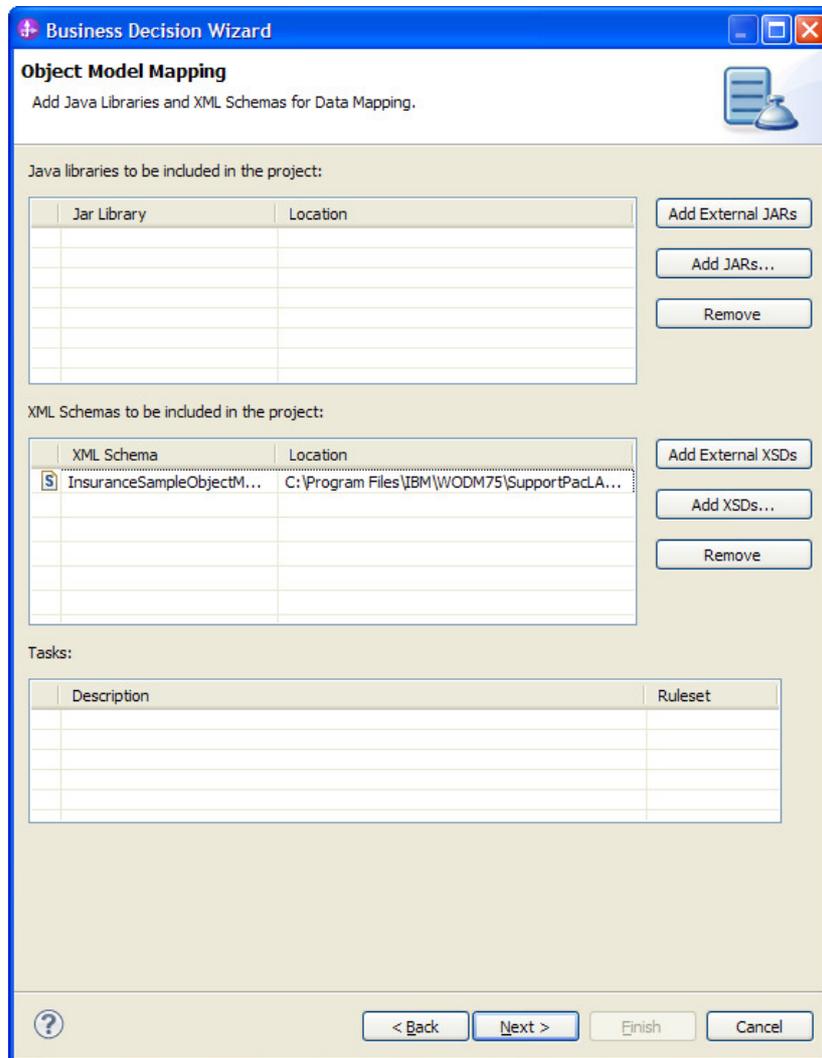
___ d. Select the “**Eligibility**” decision. In addition, make sure that “InsuranceSample” is the project.



___ e. Click **Next**.

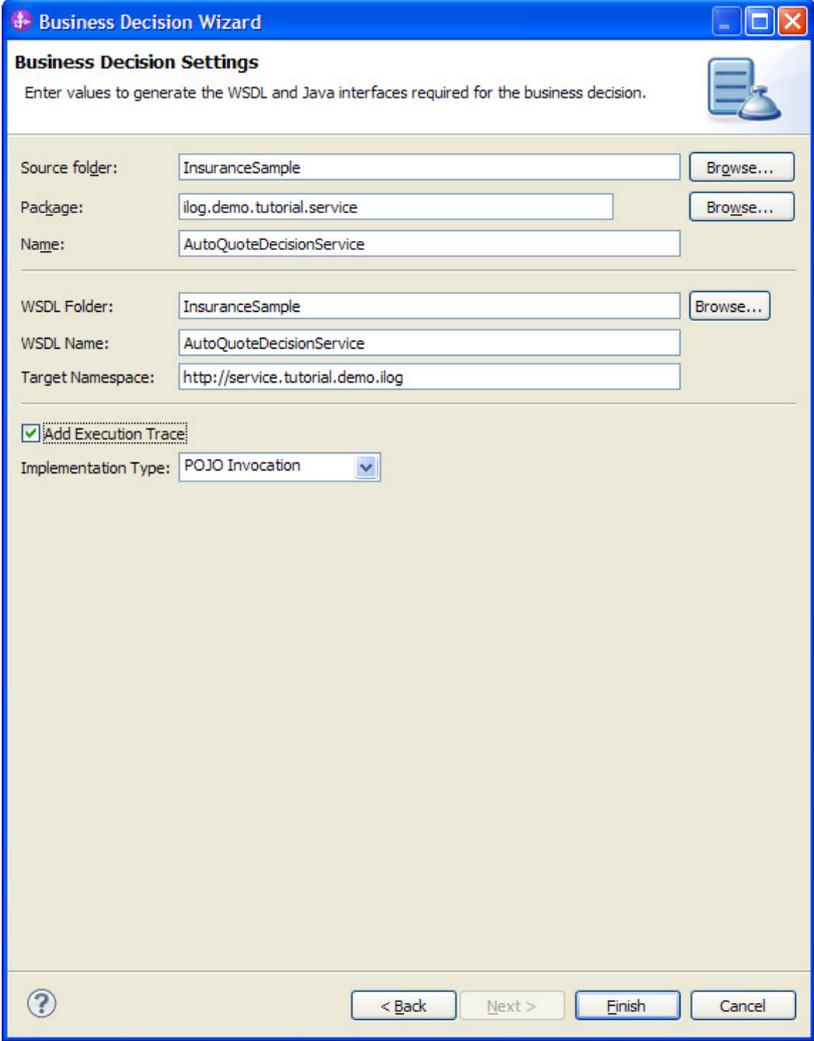
___ f. In this page of the wizard, there are some warnings in the bottom Task List due to unresolved types in the ruleset signatures. In some cases you need to provide java libraries as well as XML schemas used in the Rule Projects; however, for the tutorial you will use the same XML schema for the creation of the component signature that you used for the Execution Object Model (XOM) and Business Object Model (BOM). Click **Add External XSD** and browse to **[SupportPac LA71 Path]\tutorial\part_4\InsuranceSampleObjectModel.xsd**, then click **OK**.

The warnings are resolved after object model is defined.



- ___ g. Click **Next**.
- ___ h. In the final page of the wizard, for **Package** enter **ilog.demo.tutorial.service** and for the **Name** enter **AutoQuoteDecisionService**.
- ___ i. Check **Add Execution Trace**. With this option enabled when the rulesets are executed, detailed audit trails are logged through Common Event Infrastructure (CEI).

- ___ j. Insure **POJO Invocation** is selected for the Implementation Type. It is recommended to use the POJO invocation approach for this component since we have the Rule Execution Server installed on IBM Process Server and can take advantage of the proximity to make a direct Java call rather than going through the EJB Container.



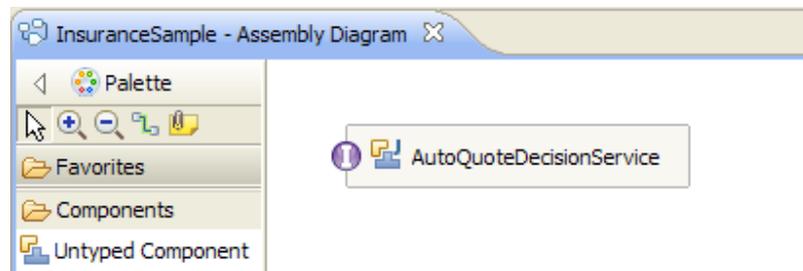
The screenshot shows the 'Business Decision Wizard' dialog box with the following settings:

- Source folder:** InsuranceSample
- Package:** ilog.demo.tutorial.service
- Name:** AutoQuoteDecisionService
- WSDL Folder:** InsuranceSample
- WSDL Name:** AutoQuoteDecisionService
- Target Namespace:** http://service.tutorial.demo.ilog
- Add Execution Trace**
- Implementation Type:** POJO Invocation

At the bottom of the dialog, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'. A help icon (?) is also present in the bottom left corner.

- ___ k. Click **Finish**.

- ___ I. Note that SCA component AutoQuoteDecisionService has been added to the module. If you open up the Assembly Diagram you will see the SCA component.



- ___ 3. With the AutoQuoteDecisionService SCA Component created, you can now test the calls to the RuleApp.
 - ___ a. Open the Assembly Diagram if it is not open already.
 - ___ b. Right-click on the **AutoQuoteDecisionService** and select **Test Component**.
 - ___ c. For the operation, select **Eligibility**.

▼ **Detailed Properties**

Specify the component, interface, operation, and input parameter values for the Invoke event, then click the Continue icon in the Events area to run the test. [More...](#)

Configuration: Default Module Test

Module: InsuranceSample

Component: AutoQuoteDecisionService

Interface: AutoQuoteDecisionService

Operation: Eligibility

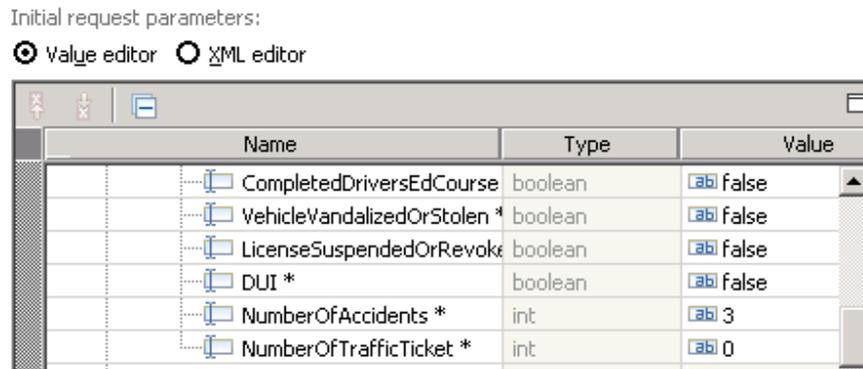
- ___ d. For the Initial request parameters section, enter **15** for the driver **Age**

Initial request parameters:

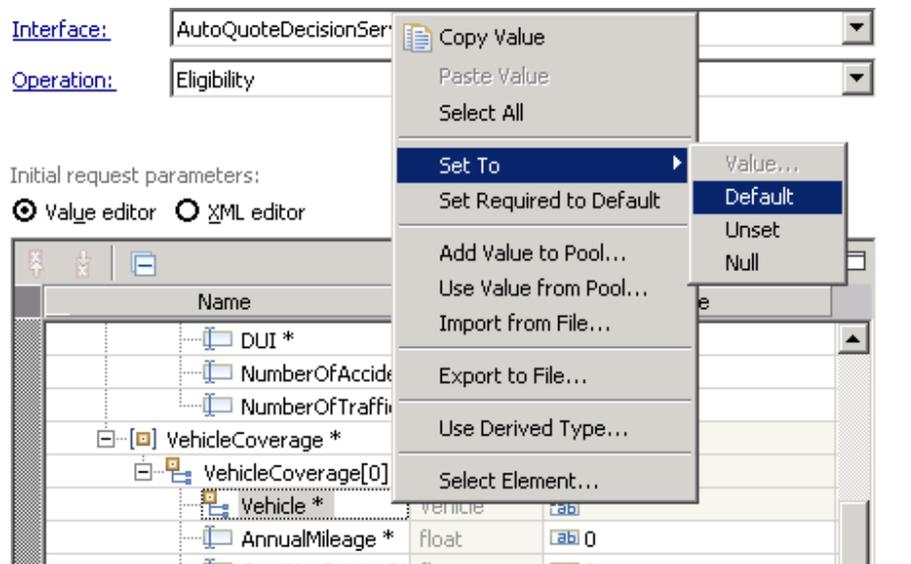
Value editor XML editor

| Name | Type | Value |
|--------------------|----------------|---------|
| AutoQuoteRequest | AutoQuoteRe... | [Ab] |
| Driver * | Driver[] | 60 |
| Driver[0] | Driver | [Ab] |
| DriverID * | int | [Ab] 0 |
| FirstName * | string | [Ab] |
| LastName * | string | [Ab] |
| Age * | int | [Ab] 15 |
| StateOfResidence * | string | [Ab] |

___ e. For the Initial request parameters section, enter **3** for the driver **NumberOfAccidents**.



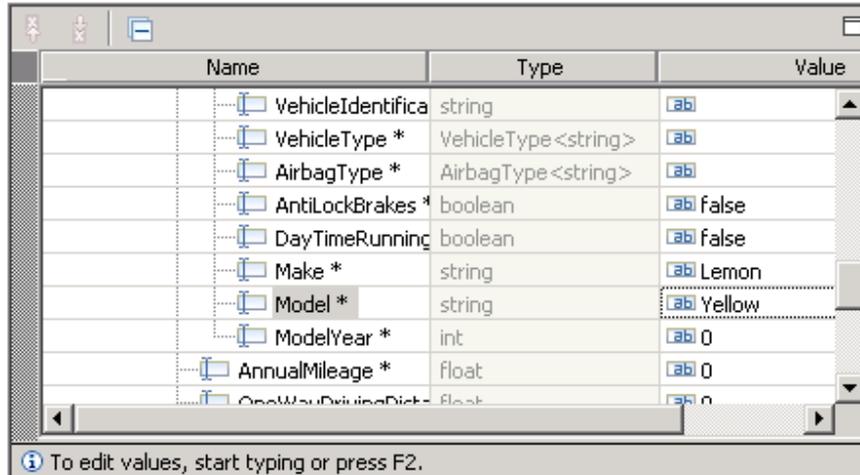
___ f. Under **VehicleCoverage**, right-click on **Vehicle** and select **Set To > Default**.



___ g. Under Vehicle, for the Make enter **Lemon** and for the Model enter **Yellow**.

Initial request parameters:

Value editor XML editor



| Name | Type | Value |
|-----------------------|----------------------|--------|
| VehicleIdentification | string | |
| VehicleType * | VehicleType <string> | |
| AirbagType * | AirbagType <string> | |
| AntiLockBrakes * | boolean | false |
| DayTimeRunning | boolean | false |
| Make * | string | Lemon |
| Model * | string | Yellow |
| ModelYear * | int | 0 |
| AnnualMileage * | float | 0 |
| OneWayDrivingDist | float | 0 |

To edit values, start typing or press F2.

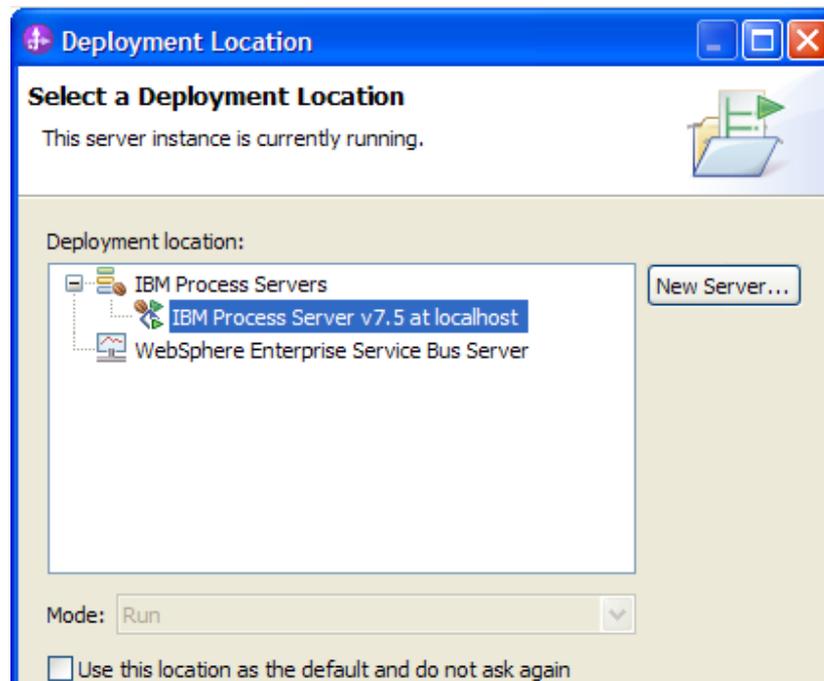
___ h. Click the **Continue** button.

Events

This area displays the events in a test trace. Select an General Properties and Detailed Properties sections. [M](#)



___ i. Insure the Deployment location is **IBM Process Server v7.5 at localhost** and click **Finish**.



___ j. For the User Login screen, accept the existing values and click **OK**.

___ k. The AutoQuoteDecisionService is called and the Eligibility ruleset is also called. The response is returned indicating the driver IS eligible.

Value Editor XML Source

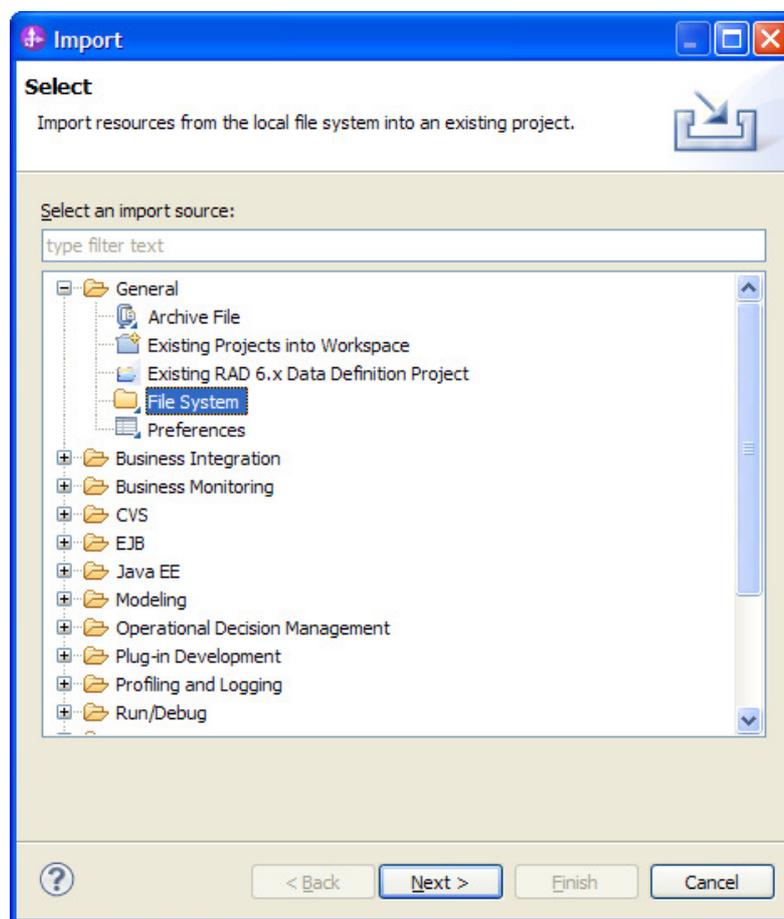
| Name | Type | Value |
|-------------------------------|----------------------------|---|
| EligibilityExecutionResult... | EligibilityExecutionResult | [ab] |
| EligibilityResponse * | EligibilityResponse | [ab] |
| Eligible * | boolean | [ab] true |
| MainMessage * | string | [ab] The driver is eligible for the high-risk program |
| ErrorMessages | string[] | 60 |

___ l. Close the Integration Test Client without saving.

Part 5: Create Business Process

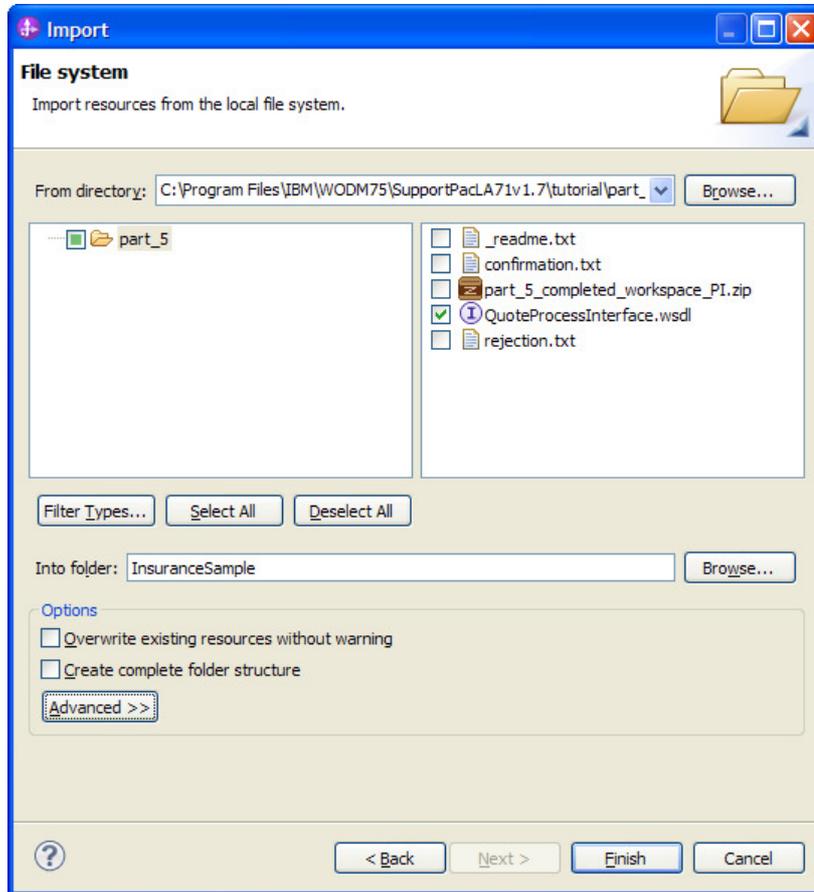
With the business rules created, the next step is to utilize this business logic in business processes. In our story here, a rule project (business decision) already exists and now a business process may take advantage of it. In this part you will create the business process with calls to the business rules. You will link the business process to the business rules in the next section.

- ___ 1. Except for InsuranceSample, remove all other modules from your workspace completed from **Part 4**.
- ___ 2. The business process will have an interface which can be called to start the quote process. Import the WSDL interface into the project.
 - ___ a. Select **File > Import....**
 - ___ b. **General** and select **File System**.



- ___ c. Click **Next**.
- ___ d. For the From directory, click **Browse...** and select **[SupportPac LA71 Path]\tutorial\part_5**.

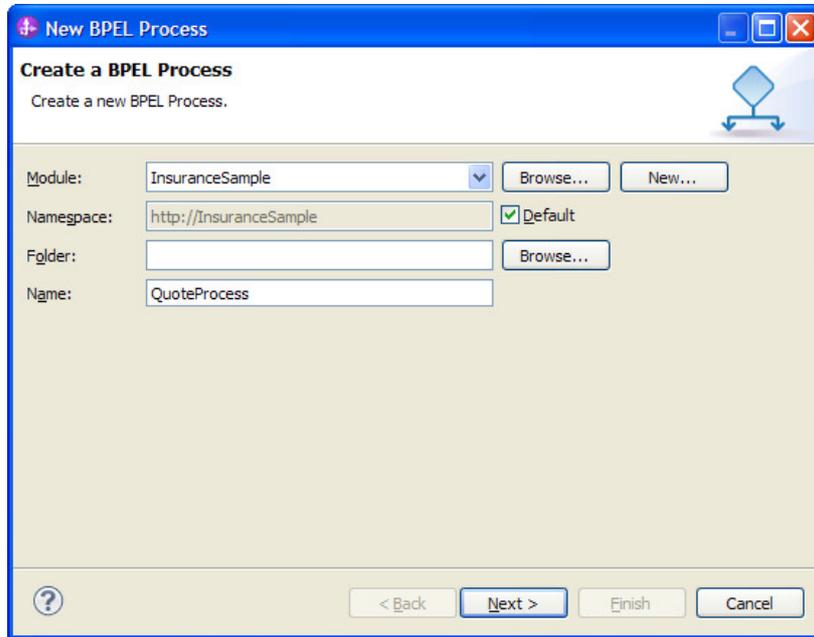
___ e. Check the box for **QuoteProcessInterface.wsdl**.



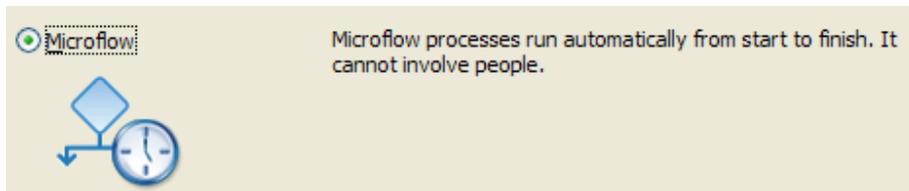
___ f. Click **Finish**.

- ___ 3. Create business process for quotes.
___ a. Select **File > New > BPEL Process**.

___ b. For the Name, enter **QuoteProcess**.

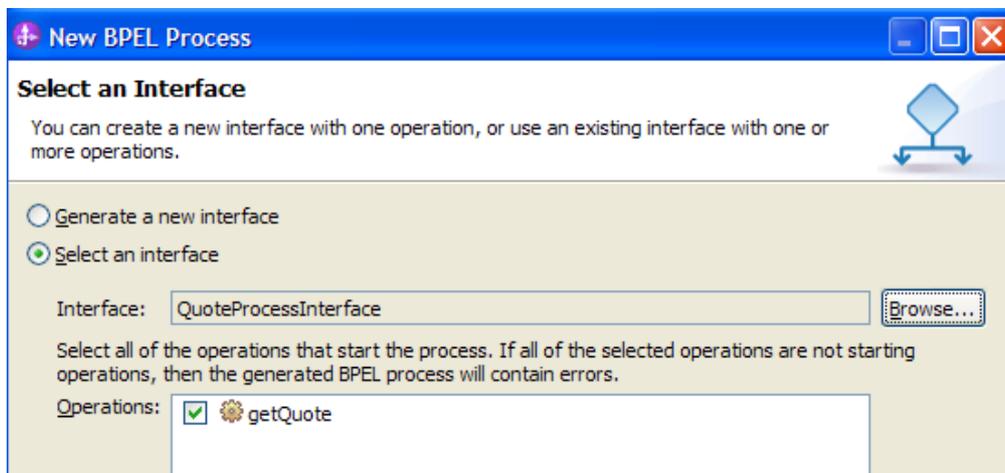


___ c. On the Business Process Type Selection screen, select **Microflow**. Select **Next**.



___ d. Select the radio button **Select an interface**.

___ e. Click **Browse...** and select **QuoteProcessInterface**. Click **OK**.



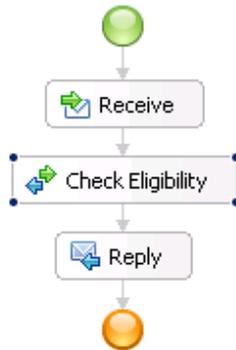
___ f. Click **Finish**.

- ___ 4. Add logic for checking eligibility before confirming the quote.
- ___ a. In the process editor, from the Palette under Basic Actions, select **Invoke**.
 - ___ b. Click beneath the **Receive** activity.

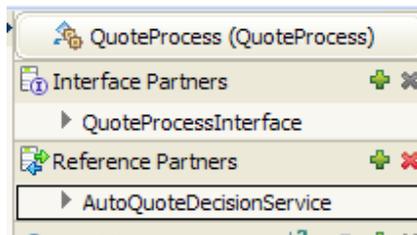


___ c. Select the Invoke activity and select the **Properties** view.

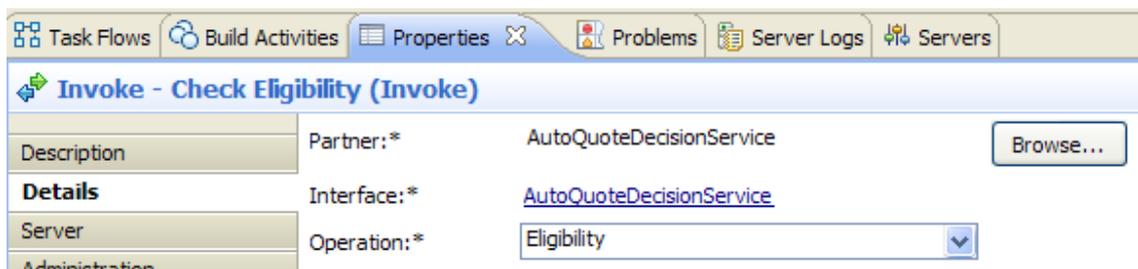
- ___ d. Change the Display name to **Check Eligibility**. This node will make the call to the Check Eligibility Decision Service to determine if the individual is eligible for insurance.



- ___ e. The Decision Service is called through a Reference Partner defined for the process. Select the add symbol for **Reference Partners**.
- ___ f. For the name enter **AutoQuoteDecisionService**.
- ___ g. In the Matching interface select **AutoQuoteDecisionService**. Click **OK**.



- ___ h. Select the Check Eligibility node again and the Properties view, select the **Details** tab.
- ___ i. For the Partner, select **Browse...** and select **AutoQuoteDecisionService**. Click **OK**.
- ___ j. Change the Operation to be **Eligibility**.



- ___ k. Scroll down the Details tab. For the Input, under the Read from Variable column, click (**none**).
- ___ l. Select **quoteRequest**.
- ___ m. For the Output, under the Read from Variable column, click (**none**).

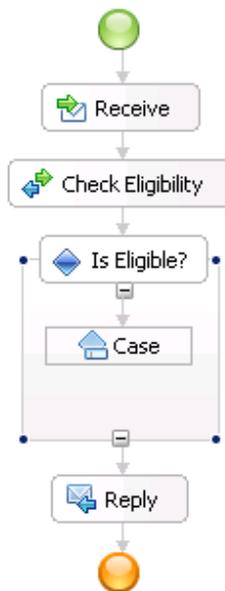
___ n. Click **New** and for the Variable Name accept the value **EligibilityExecutionResult**. Click **OK**.

| | Name | Type | Read from Variable |
|---------|----------------------------|----------------------------|----------------------------|
| Inputs | AutoQuoteRequest | AutoQuoteRequest | quoteRequest |
| | Name | Type | Store into Variable |
| Outputs | EligibilityExecutionResult | EligibilityExecutionResult | EligibilityExecutionResult |

___ 5. Add logic for using the eligibility result to compute price and specify confirmation message.

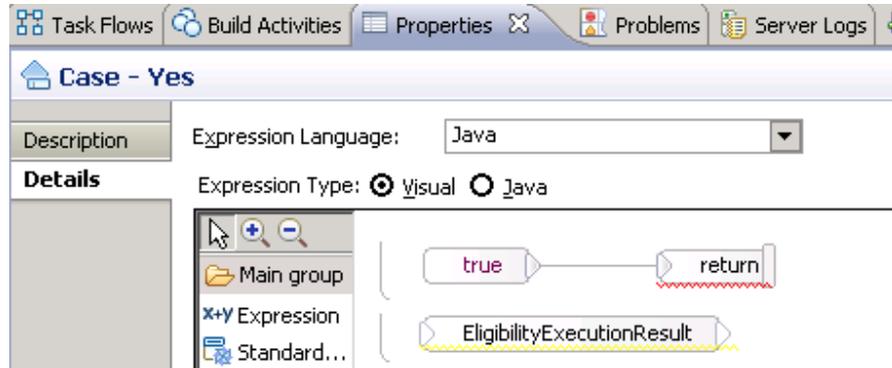
Combining a call to an **Invoke** (representing a decision) followed a **Choice** node is a common design pattern which reduces the amount of hard-coded logic. The decision is reused from the WODM but the “Choice” enforces the decision in the context of the process. The logic can be captured in business rules with the result checked by the Choice and different cases.

- ___ a. From the Palette under Structures, select **Choice**.
- ___ b. Click beneath the Check Eligibility node.
- ___ c. Select the Choice node and in the Properties view change the Name to **Is Eligible?**.

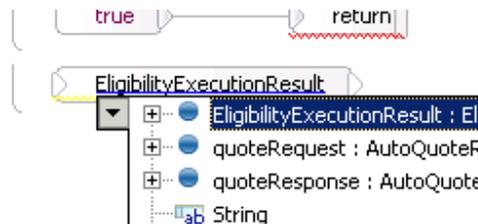


- ___ d. A case is created by default. Change the Display name in the Properties view to **Yes**.
- ___ e. The “Yes” condition is executed if the result from the Check Eligibility invoke is true. To specify the condition, select the Details tab.
- ___ f. For the Expression Language select **Java**.

___ g. From the list of Variables, drag and drop **EligibilityExecutionResult**.

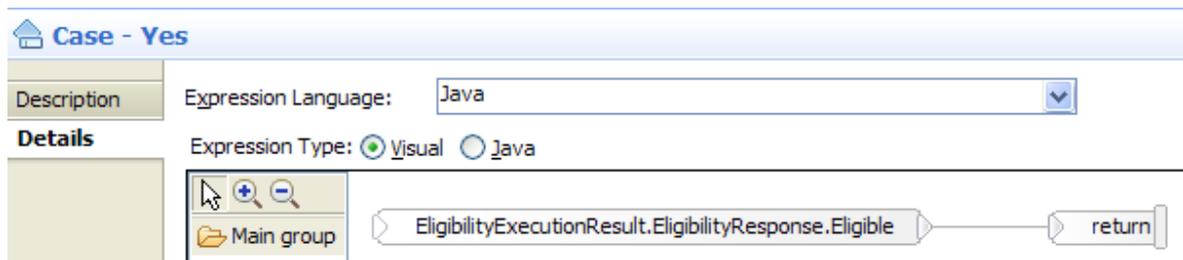


___ h. Click EligibilityExecutionResult to see the attributes for the variable.

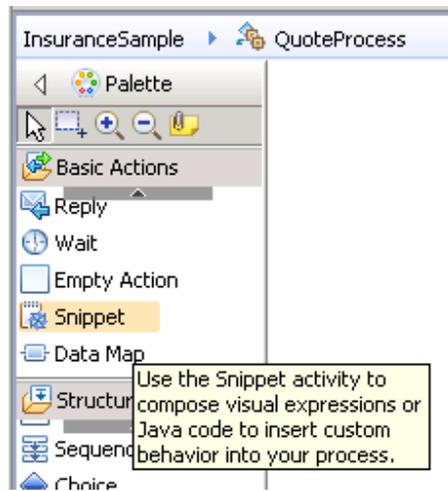


___ i. Expand and select **EligibilityExecutionResult.EligibilityResponse.Eligible**.

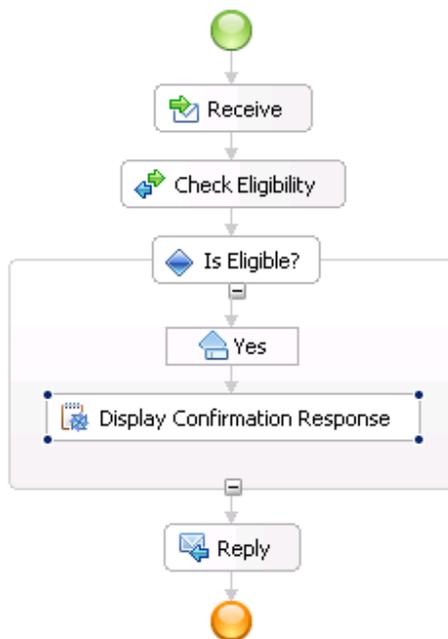
___ j. Delete the true value and wire the **EligibilityExecutionResult** variable to the **return**. Note that the wire will not operate until the fully qualified value is used.



- ___ k. A confirmation needs to be returned to the customer. For this process, a snippet will be used to specify a confirmation message. From the Palette, select **Snippet** from the list of Basic Actions.



- ___ l. Click beneath the **Yes** case node.
- ___ m. Change the Display name to **Display Confirmation Response**.



- ___ n. The confirmation message will be set with a small amount of Java code. In the Properties view, select the **Details** tab.
- ___ o. Select the Java radio button. Select **Yes** to the warning message.

___ p. Type in the following code or copy from the file [SupportPac LA71 Path]\tutorial\part_5\confirmation.txt, copy and paste the code into the Java window.

```
commonj.sdo.DataObject eligibilityResp =
this.EligibilityExecutionResult.getDataObject("EligibilityResponse");

com.ibm.websphere.bo.BOFactory boFactory = (com.ibm.websphere.bo.BOFactory)
com.ibm.websphere.sca.ServiceManager.INSTANCE

    .locateService("com/ibm/websphere/bo/BOFactory");

this.quoteResponse = boFactory.create("http://www.ilog.com/InsuranceSample",
"AutoQuoteResponse");

java.util.List messages = new java.util.Vector();

String msg = eligibilityResp.getString("MainMessage");

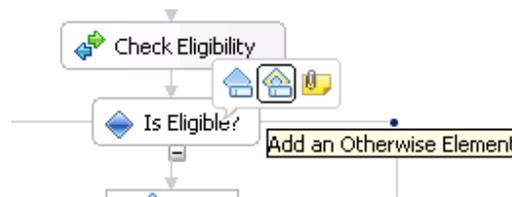
messages.add(msg);

this.quoteResponse.setList("Messages", messages);

String result = "Congratulations! Your application was accepted. We can offer
you the coverage you requested.";

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

- ___ 6. Besides the Yes case, a case is needed to handle a customer that is not eligible. Add logic to produce a rejection message.
 ___ a. Click the **Is Eligible** choice node.

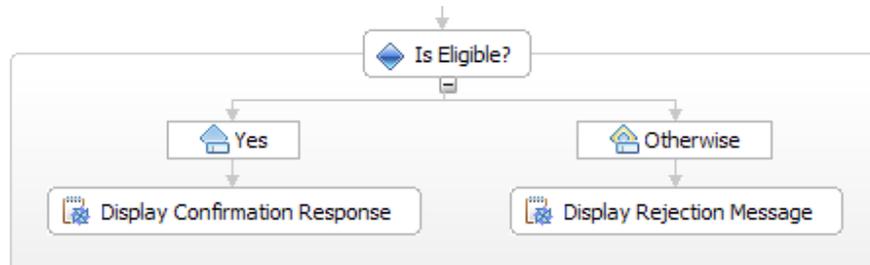


- ___ b. Select the **Otherwise** icon. An otherwise case will be added.



- ___ c. Just as the confirmation message, the logic for the rejection message will be specified with a Snippet. In the palette from the list of Basic Actions, select **Snippet**.

- ___ d. Click beneath the **Otherwise** case.
- ___ e. Change the Display name to **Display Rejection Message**.



- ___ f. The rejection message will also be set with a small amount of Java code. In the Properties view, select the Details tab.
- ___ g. Select the Java radio button. Select **Yes** to the warning message.
- ___ h. From the file **[SupportPac LA71 Path]\tutorial\part_5\rejection.txt**, copy and paste the code into the Java window.

```

commonj.sdo.DataObject eligibilityResp =
this.EligibilityExecutionResult.getDataObject("EligibilityResponse");

com.ibm.websphere.bo.BOFactory boFactory = (com.ibm.websphere.bo.BOFactory)
com.ibm.websphere.sca.ServiceManager.INSTANCE
    .locateService("com/ibm/websphere/bo/BOFactory");

this.quoteResponse = boFactory.create("http://www.ilog.com/InsuranceSample",
"AutoQuoteResponse");

java.util.List messages = eligibilityResp.getList("ErrorMessages");

String errors = "";

if ( messages != null && messages.size() > 0)

{

    this.quoteResponse.setList("Messages", messages );

    int size = messages.size();

    for(int i =0; i<size; i++)

    {

        errors += messages.get(i);

        if(i<size-1)

            errors +="; ";

    }

}
  
```

```

}

else {

    messages = new java.util.Vector();

    errors = eligibilityResp.getString("MainMessage");

    messages.add(errors);

    this.quoteResponse.setList("Messages", messages);

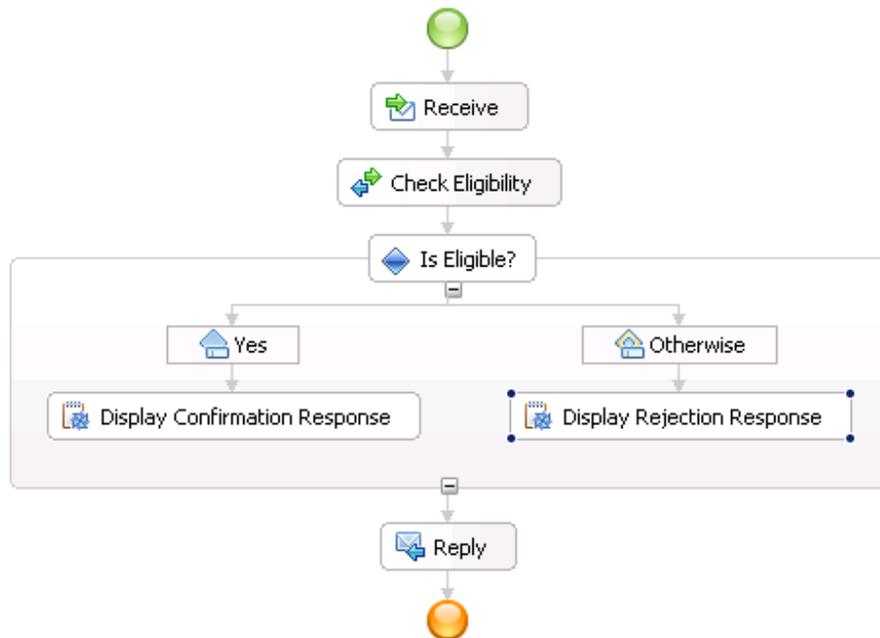
}

String result = "We're sorry, your application was rejected for the following
reason:\n\t" + errors;

System.out.println(result);

```

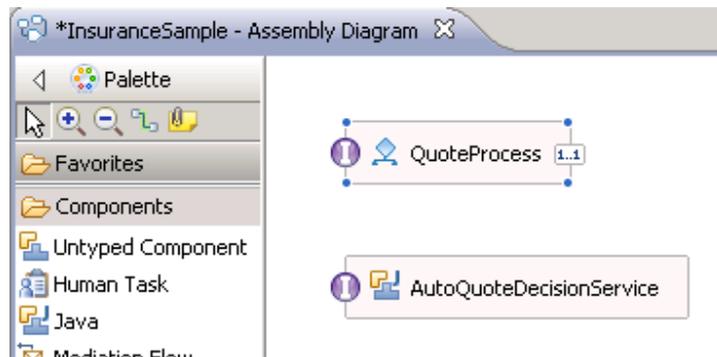
__ i. The process is now complete. Save the process. Check that no errors in the Problems view.



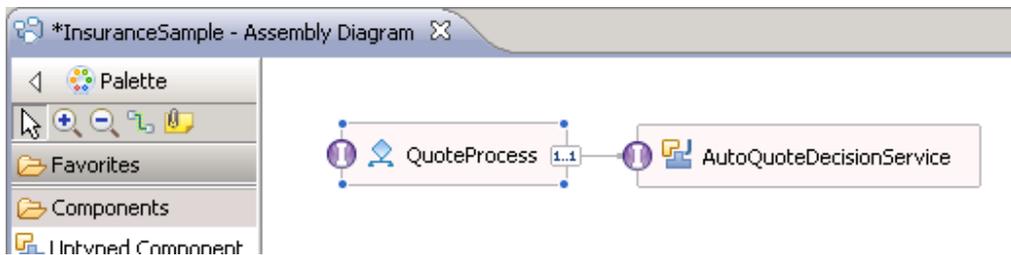
Part 6: Assembling BPM Application

With the business rules and the business process created, the full solution can now be assembled.

- ___ 1. Assemble the business rules and business process.
 - ___ a. Open the **Assembly Diagram** to open the assembly editor if it is not already opened.
 - ___ b. In the Business Integration view, under **InsuranceSample > Integration Logic > BPEL Processes**, select the **QuoteProcess** process.
 - ___ c. Drag and drop the QuoteProcess process onto the assembly editor.



- ___ d. Connect the reference on the **QuoteProcess** to the **AutoQuoteDecisionService** component by dragging the connector from the reference to the interface of the component.



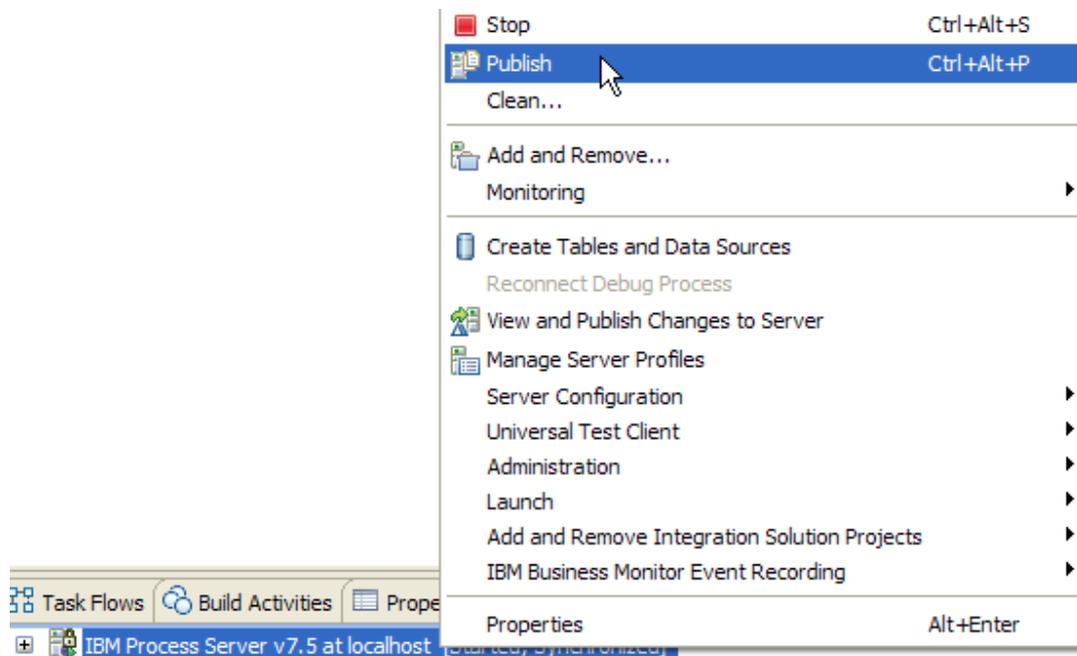
- ___ e. Save the assembly editor. Check for any errors in the Problems view and resolve.

Your application is complete and ready for a full test of the process and rules.

Part 7: Deploy and Test BPM Application

Just as the business rules, the business process can be deployed as an application to the IBM Process Server. Unlike the business rules, deployment is simply installing the application on the server. Once deployed, the process can be tested using the Test Component facility, Business Process Choreographer Explorer, or Business Space. You will test the process and the rules using the Test Component.

- ___ 1. The application deployed to the server needs to be updated with the business process. Republish the application.
 - ___ a. In the Servers view, select **IBM Process Server v7.5 at localhost**.
 - ___ b. Right-click and select **Publish**.

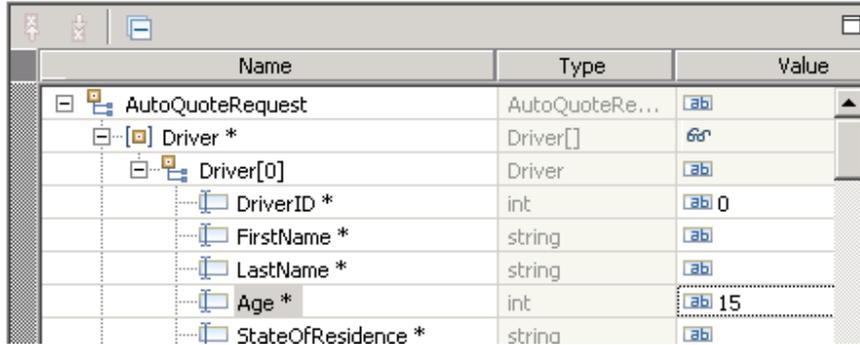


- ___ 2. Test business process using the Test Component.
 - ___ a. In the Assembly Diagram, right-click on the **QuoteProcess** and select **Test Component**.

___ b. For the Initial request parameters section, enter **15** for the driver **Age**

Initial request parameters:

Value editor XML editor

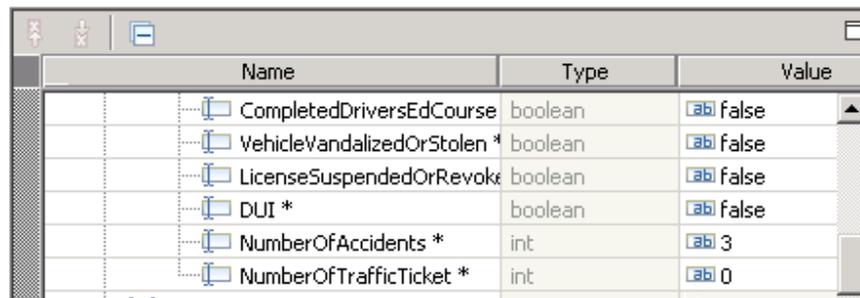


| Name | Type | Value |
|--------------------|----------------|---------|
| AutoQuoteRequest | AutoQuoteRe... | [ab] |
| Driver * | Driver[] | 60 |
| Driver[0] | Driver | [ab] |
| DriverID * | int | [ab] 0 |
| FirstName * | string | [ab] |
| LastName * | string | [ab] |
| Age * | int | [ab] 15 |
| StateOfResidence * | string | [ab] |

___ c. For the Initial request parameters section, enter **3** for the driver **NumberOfAccidents**.

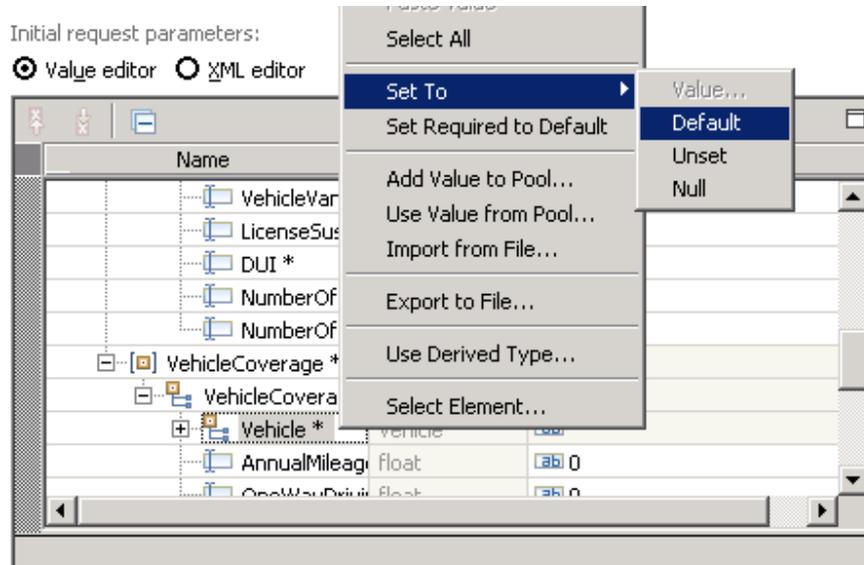
Initial request parameters:

Value editor XML editor

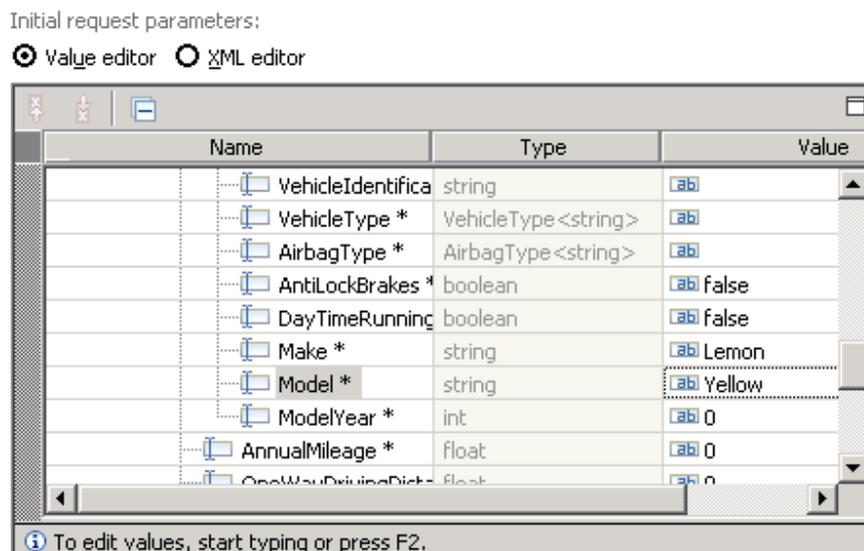


| Name | Type | Value |
|-----------------------------|---------|------------|
| CompletedDriversEdCourse | boolean | [ab] false |
| VehicleVandalizedOrStolen * | boolean | [ab] false |
| LicenseSuspendedOrRevoked * | boolean | [ab] false |
| DUI * | boolean | [ab] false |
| NumberOfAccidents * | int | [ab] 3 |
| NumberOfTrafficTicket * | int | [ab] 0 |

___ d. Under **VehicleCoverage**, right-click on **Vehicle** and select **Set To > Default**.



___ e. Under **Vehicle**, for the **Make** enter **Lemon** and for the **Model** enter **Yellow**.



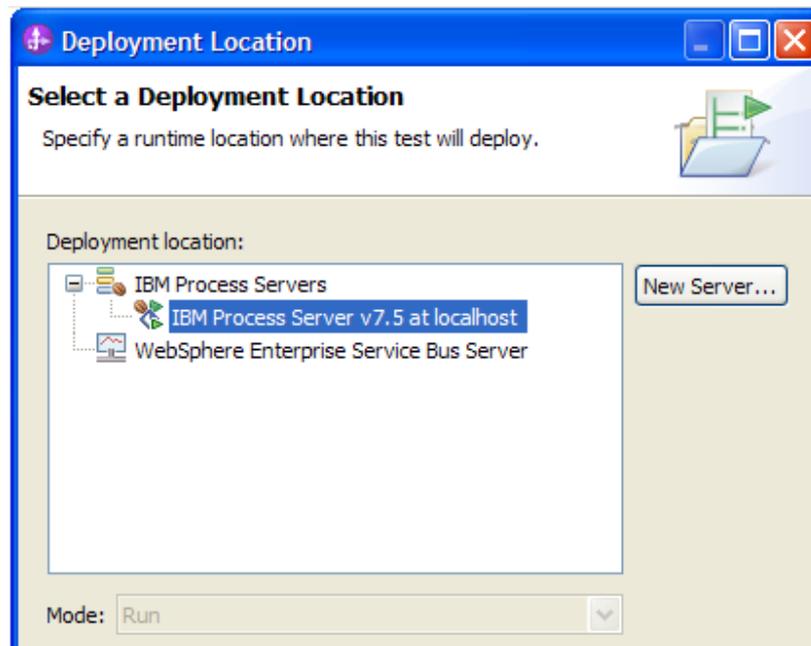
___ f. Click the **Continue** button.

Events

This area displays the events in a test trace. Select an General Properties and Detailed Properties sections. [M...](#)



___ g. Insure the Deployment location is **IBM Process Server v7.5 at localhost** and click **Finish**.

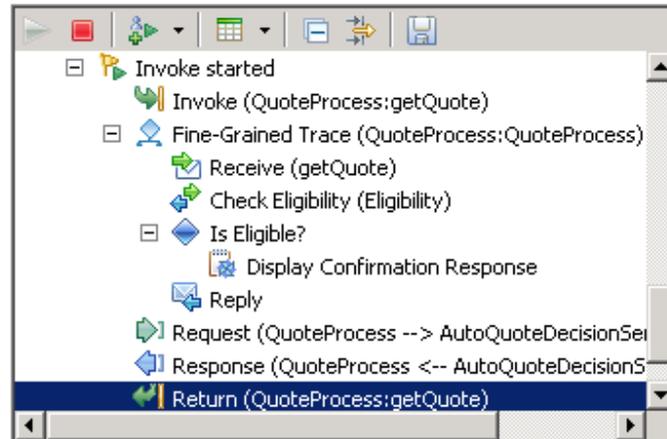


___ h. For the User Login screen, accept the existing values and click **OK**.

- ___ i. The QuoteProcess business process component is called and then the AutoQuoteDecisionService component is called.

Events

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections. [More...](#)



- ___ j. The response is returned indicating that the driver is not eligible for a quote.

Return parameters:

Value Editor XML Source

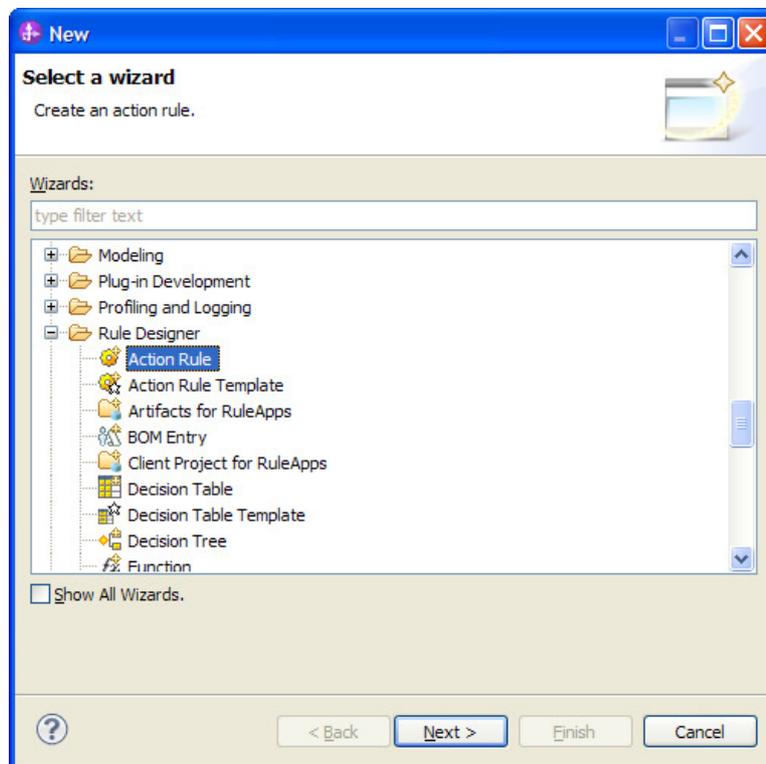
| Name | Type | Value |
|------------------|-------------------|---|
| quoteResponse | AutoQuoteResponse | [ab] |
| VehideQuotes * | VehideQuoteItem[] | 60 |
| GlobalAdjustment | LineItem[] | 60 |
| Messages | string[] | 60 |
| Messages[0] | string | [ab] The driver is eligible for the high-risk program |
| TotalPrice * | float | 60 |

Part 8: (Optional) Create a Business Rule for the Return Message

In this optional part 8, rules will be updated and retested from IID. Most all of the steps to accomplish this have been completed; however, versioning will be introduced.

Since the WSDL used always pulls the latest version of the "Eligibility" ruleset, the updated will be deployed as a ruleset version 1.1 without changing the process.

- ___ 1. Update the "Eligibility" decision with a new rule for enforcing age.
 - ___ a. While still in the Business Integration perspective, navigate to the "Eligibility" module→Rules→Rule Module.
 - ___ b. Right click on the **Rule Package** and navigate to **New→ Other**.
 - ___ c. Select **Rule Designer → Action Rule**



- ___ d. Click **Next**.
- ___ e. In the next window type in the name **EnforceDriverAgeBelow18**.
- ___ f. Accept the defaults and click **Finish**.
- ___ g. Type in the following rule:

```

definitions
set driver to a driver in the drivers of 'the quote request' ;

if
  the age of driver is less than 18
then
  make it false that 'the eligibility response' is eligible ;
  set the main message of 'the eligibility response'
  to "A driver must be 18 or older to qualify for this program." ;

```

- ___ h. Add another rule with the name: **EnforceDriverAgeAbove17** like the following:

```

definitions
set driver to a driver in the drivers of 'the quote request' ;

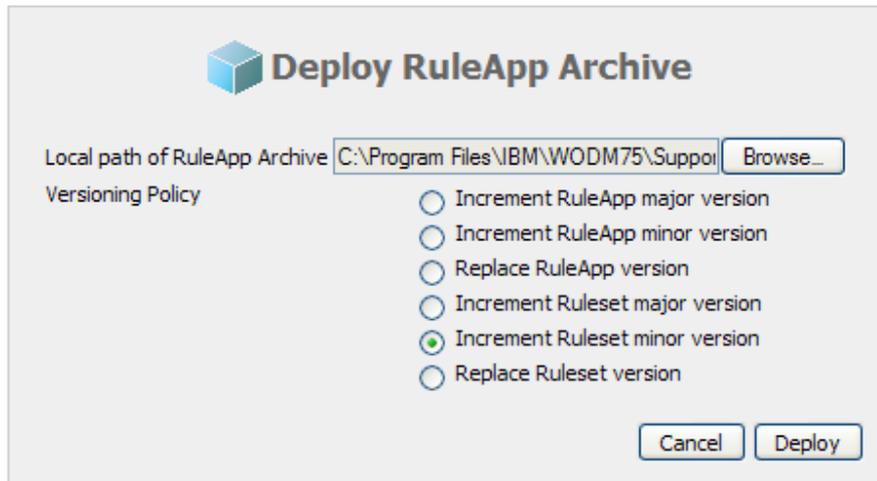
if
  the age of driver is more than 17
then
  make it true that 'the eligibility response' is eligible ;
  set the main message of 'the eligibility response'
  to "The driver is old enough to qualify for this program." ;

```

- ___ i. Navigate to the **InsurnaceSampleRuleApp** module and double click on the **archive.xml** file.
 ___ j. Confirm that the version is 1.0
 ___ k. Export the decision to the file system.

- ___ 2. Load the ruleapp archive to Rule Execution Server
 ___ a. Open the Rule Execution Server console as in previous steps and navigate to the **Explorer** tab.
 ___ b. Click on **Deploy RuleApp Archive**.

___ c. Enter information as follows:



___ d. Click **Deploy**. Since the ruleset already exists, the version policy will increment the ruleset version to a 1.1.

| Details | | |
|-----------------------------|-----------------------------------|-----------------------------|
| Archive content | Operation | Result |
| /InsuranceSampleRuleApp/1.0 | Element updated | /InsuranceSampleRuleApp/1.0 |
| /DataValidation/1.0 | Version changed and element added | /DataValidation/ 1.1 |
| /Eligibility/1.0 | Version changed and element added | /Eligibility/ 1.1 |

___ e. Test the decision from IID as in Part 3 or test the business process using the decision as in Part 6 and add test data related to age.

What you did in this tutorial

In this tutorial, you simulated multiple roles in the development of a highly dynamic business process application. You started as the rule developer, capturing your business logic in business rules. You tested these business rules as decision services. Next you were the IT developer who defined the business process that used the business rules as a means of making your process more flexible. You finished by testing the entire BPM application.