

JBOSS BUSINESS LOGIC DEVELOPER WORKSHOP

LAB INSTRUCTIONS

Table of Contents

| | | |
|----------|--|-----------|
| 1 | Preface | 4 |
| 1.1 | About This Document | 4 |
| 1.2 | Audience..... | 4 |
| 1.3 | Background..... | 4 |
| 2 | Labs..... | 5 |
| 2.1 | Lab 1 – Installation and Analysis | 5 |
| 2.1.1 | Software installation instructions | 5 |
| 2.1.2 | Analyze rules | 6 |
| 2.1.3 | Further analysis | 7 |
| 2.1.4 | Text description version 2..... | 7 |
| 2.2 | Lab 2 – DRL Rules | 8 |
| 2.2.1 | Create a new Drools Project..... | 8 |
| 2.2.2 | Add fact Objects to the project | 9 |
| 2.2.3 | Author rules | 9 |
| 2.2.4 | Test the rules..... | 10 |
| 2.3 | Lab 3 – RETE..... | 11 |
| 2.3.1 | RETE viewer..... | 11 |
| 2.3.2 | Audit log..... | 11 |
| 2.3.3 | Debug | 11 |
| 2.4 | Lab 4 – BRMS Guided Rule Editor | 12 |
| 2.4.1 | Add rule categories..... | 12 |
| 2.4.2 | Add a rule package..... | 12 |
| 2.4.3 | Fact model..... | 12 |
| 2.4.4 | Create exclusion rules | 12 |
| 2.5 | Lab 5 – BRMS Decision Table | 14 |
| 2.5.1 | Create the web-based decision table | 14 |
| 2.5.2 | Add condition columns | 14 |
| 2.5.3 | Add action columns | 14 |
| 2.5.4 | Add the No Loop attribute..... | 15 |
| 2.5.5 | Add rules to the decision table | 15 |
| 2.6 | Lab 6 – BRMS Test Scenarios | 16 |
| 2.6.1 | Create test scenarios..... | 16 |
| 2.7 | Lab 7 – IDE BRMS Integration | 17 |
| 2.7.1 | Upload rules to BRMS | 17 |
| 2.7.2 | Create test scenarios..... | 17 |
| 2.8 | Lab 8 – Domain Specific Language | 18 |
| 2.8.1 | Author rules | 18 |
| 2.8.2 | Create test scenarios..... | 18 |
| 2.9 | Lab – 9 Advanced Rules | 19 |
| 2.9.1 | Author rules | 19 |
| 2.9.2 | Add the rule to the package in BRMS Guvnor..... | 19 |
| 2.9.3 | Create test scenarios..... | 20 |
| 2.10 | Lab 10 – Ruleflow | 21 |
| 2.10.1 | JBoss Business Process Modeling (jBPM) Preparation | 21 |
| 2.10.2 | Create a new business process | 22 |
| 2.10.3 | Set properties for the diagram | 22 |
| 2.10.4 | Add the start event node and task..... | 23 |

| | | |
|-------------------------|--|-----------|
| 2.10.5 | Conditional business process flow | 23 |
| 2.10.6 | Additional Business Rule Tasks | 24 |
| 2.10.7 | Save the process image | 25 |
| 2.10.8 | Add a rule to trigger the process | 25 |
| 2.10.9 | Add ruleflow groups | 25 |
| 2.10.10 | Create test scenarios | 26 |
| 2.11 | Lab 11 – Sub-Process | 28 |
| 2.11.1 | Lab preparation | 28 |
| 2.11.2 | Add the start event node and script task | 29 |
| 2.11.3 | Add a reusable sub-process | 31 |
| 2.12 | Lab 12 – Human Tasks | 33 |
| 2.12.1 | Create a user task | 33 |
| 2.12.2 | Add Users and Reviewer group | 34 |
| 2.12.3 | Create Data Input Sets and Data Output sets | 34 |
| 2.12.4 | Capture the Edited Price | 35 |
| 2.12.5 | Add Task to Reject Facts | 36 |
| 2.12.6 | Generate a Process diagram image | 37 |
| 2.12.7 | Add Users and Roles to the Configuration | 37 |
| 2.12.8 | Save your progress | 37 |
| Lab 13 | – Forms | 38 |
| 2.12.9 | Add a process start form using Forms Designer | 38 |
| 2.12.10 | Examine the process Input form | 38 |
| 2.12.11 | Check out the Task Form | 39 |
| 2.12.12 | Disable the start process placeholder rule | 41 |
| 2.12.13 | Build package | 41 |
| 2.12.14 | Test the process using the jBPM Console | 41 |
| 2.12.15 | View the status of the process | 42 |
| 2.12.16 | Review the quote | 42 |
| 2.12.17 | Additional test data | 42 |
| 2.13 | Lab 14 – Service Tasks | 44 |
| 2.13.1 | Add the icon for the task | 44 |
| 2.13.2 | Create a new Work Item Definition for the Task | 44 |
| 2.13.3 | Use the new Task in the Designer | 44 |
| 2.13.4 | Assign input data | 44 |
| 2.13.5 | Create a Work Item Handler to execute the task | 45 |
| 2.13.6 | jBPM5 only class | 46 |
| 2.13.7 | Tell the process engine what to do with the AuditReviewTask TaskType | 46 |
| 2.13.8 | Re-test the process in BPM-console | 47 |
| Appendices | | 48 |
| 2.14 | Appendix A: Troubleshooting – Corrective Actions | 49 |
| | Add the Web Designer to the server | 49 |

1 PREFACE

1.1 About This Document

Labs are core to the BRMS Workshop. The instructions within this document guide the student thru a nearly real solution scenario to provide hands-on practice to the concepts presented in the class. Each lab builds on knowledge gained from previous labs and lessons. As the labs progress, steps for material already covered are given less detail to encourage the student to both recall and understand the material and the techniques.

1.2 Audience

This document is intended for students attending a BRMS Workshop, provided by a Professional Instructor or Senior Consultant.

1.3 Background

IT organizations search for greater flexibility to improve business process agility. An agile application and business services portfolio has become one of the top focus items for businesses of all sizes. A more stringent regulatory environment drives IT to better manage critical business logic and rules, enabling superior business process automation as well as application and business process audit readiness. Achieving greater agility and transparency leads organizations to improve the modularity and accessibility of their business policies and rules by separating these business policies and rules from business process and presentation logic into a business rules management system.

JBoss® Enterprise BRMS (Business Rules Management System) provides an open source business rules management system that enables active decisions with easy business policy and rules development, access, and change management. A business rules management system helps an organization capture and manage business knowledge and policies. These business rules form the basis for decision-making in the organization.

JBoss Enterprise BRMS allows businesses to reduce the development time needed to update applications, SOA deployments, and business processes with the latest business rules and policies. This enables enterprises to make active decisions driven by business strategies and environment changes. The enterprise can anticipate and respond rapidly to competitive and business challenges. Something as simple as implementing a new pricing scheme or a policy change driven by regulatory changes can be rolled into production in a matter of hours, with little to no impact to the current application infrastructure. Additionally, JBoss Enterprise BRMS allows IT and business analysts to ensure that the application, SOA, and/or BPM deployment implements the business policies correctly, reducing costs and ensuring compliance with regulations.

2 LABS

2.1 Lab 1 – Installation and Analysis

Goal: Install JBoss Business Rule Manager and JBoss Developer Studio. Analyze a set of rules to calculate a policy quote price.

Pre-requisites: use JDK 1.6, version 16 or later.

2.1.1 Software installation instructions

2.1.1.1 Install BRMS

1. Create a folder **brms-workshop**
2. Copy **brms-cd** from CD / flash drive into **brms-workshop**
3. Create a folder under **brms-workshop** called **software**
4. Unzip **jboss-brms-standalone-5.3.x.zip** into the **software** folder
5. Open **brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/conf/props/brms-users.properties** and uncomment **#admin=admin**

Note: If a default server is not available then make a copy of the **production** server and rename the copy to **default**.

2.1.1.2 Make the log more useful

As a convenience, the following steps are helpful for reducing the volume of excess logs sent to the console.

1. In an editor open the **brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/conf/jboss-log4j.xml** file.
2. Add the two categories illustrated below:

```
<!-- ===== -->
<!-- Limit categories -->
<!-- ===== -->
<!-- Exclude jackrabbit Session open and close messages -->
<category name="org.apache.jackrabbit.core">
  <priority value="ERROR"/>
</category>
<!-- Limit Mina Logging -->
<category name="org.apache.mina.filter.logging">
  <priority value="ERROR" />
</category>
```

2.1.1.3 Install JBoss Developer Studio

7. Create a folder under **software** called **jbds**
8. Open a command line. Change directories to your **brms-cd** folder and execute **java -jar jbdevstudio-
<os_and_arch_version_id>.jar** (on Linux or Mac equivalent) to run the installer

Warning: If your JBDS workspace becomes corrupted during the course of the workshop, you may be required to rebuild your workspace. If this should occur, repeat the steps provided in 2.1.1.3 and 2.1.1.4 below.

9. Use the **brms-workshop/software/jbds** folder as the installation path
10. Use a version 1.6 JVM or higher
11. For the **Select Platforms and Servers** dialog, select the **jboss-as** folder of your BRMS installation
12. Click through the wizard screens to complete the installation of JBDS

FYI: JBoss Developer Studio 5 is the next version of our Eclipse based IDE targeted for development on JBoss supported platforms.

This milestone, we've focused on migrating to Eclipse 3.7, JBoss AS 7/EAP 6 support and separated out the SOA tooling so JBoss Developer Studio 5 will **not** include Teiid, jbpmp, drools, ESB and Smooks by default - these are planned to be available from our Extra's update site going forward.

Source: <https://community.jboss.org/en/tools/blog/2011/07/05/jboss-developer-studio-5-early-access>

As of March 27, 2012, version 4.1.2 is used in this lab.

2.1.1.4 Start JBDS

13. Create a folder under **software** called **workspace**
14. Start **JBDS** with the **JBDS** icon and set your **workspace** to **/brms-workshop/software/workspace**
15. Close the welcome screen

2.1.1.5 Start the server

16. Save your changes (if any) and start the server using the **Servers View Start** option. The server is started once you see a **Started in xxxx ms** message similar to the following:

```
09:20:02,481 INFO [Http11Protocol] Starting Coyote HTTP/1.1 on http-localhost%2F127.0.0.1-8080
09:20:02,854 INFO [AjpProtocol] Starting Coyote AJP/1.3 on ajp-localhost%2F127.0.0.1-8009
09:20:02,861 INFO [ServerImpl] JBoss (Microcontainer) [5.1.0 (build: SVNTag=JBAPP_5_1_0 date=201009150134)] Started in 1m:29s:532ms
```

Figure 2-1. Server Started log

Note: If you are using a server configuration that is a copy of production, then you will not see the **server started** message on the console log. Instead you will need to view the file log. In a new browser window, navigate to the log folder of server configuration and enter the command: **tail -f server.log**

17. Optionally, you may choose to run the JBoss server from the command line. On some machines this option provides better JBDS performance. To do this, open a command window to **brms-workshop/software/brms-standalone-5.3.0/jboss-as/bin** and type the command:

```
==> ./run.sh -c default -b localhost
```

You can use **ctrl+c** to stop the server.

FYI: The **-c** option is used to specify the desired server configuration. The **-b** option is used to specify the network interface that the server will use to listen for messages.

On windows use **run.bat** instead of **run.sh**

17. Logon to the BRMS Guvnor by opening a browser and entering the URL <http://localhost:8080/jboss-brms> in a browser window and entering **admin / admin** for the **userId** and **password**
18. When asked **Would you like to install a sample repository?**, Select **No Thanks**

2.1.2 Analyze rules

Create a text file and express the following rules as a series of **If / Then** statements.

- Safe Youths pay 450

- Risky Youths pay 700
- Safe Adults pay 120
- Risky Adults pay 300

2.1.3 Further analysis

After further analysis, the following is determined. Make the appropriate changes:

- Youths are between 18 and 24
- Adults are over 24
- Safe drivers have no accidents and less than two tickets
- Risky drivers have one or more accidents or two or more tickets
- Also, a fact model has been created for Driver and Policy facts (see next section)

2.1.4 Text description version 2

1. Open the **org.acme.insurance.Driver** and **Policy** classes from the **labs/lab01-Installation and Analysis/solution** folder
2. Review these fact model objects
3. Re-state the rules using the fact model and the respective definitions of Youth, Adult, etc

2.2 Lab 2 – DRL Rules

Goal: Create a set of rules to calculate the quote price.

2.2.1 Create a new Drools Project

1. In JBoss Developer Studio (JBDS) open the **Drools Perspective** (**Windows** → **Open Perspective** → **Other...** → **Drools**)

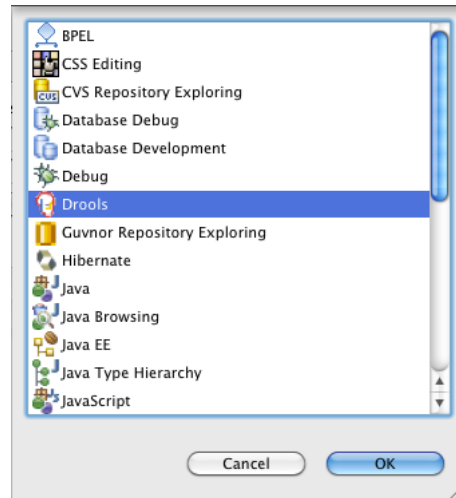


Figure 2-2. Drools Perspective

2. Use the Drools Icon (or the File menu) to create a **New Drools Project**
3. Name it **policyquote-rules**
4. **Click Next**
5. Uncheck the boxes in the window to create a sample **HelloWorld** rule as well as the others:

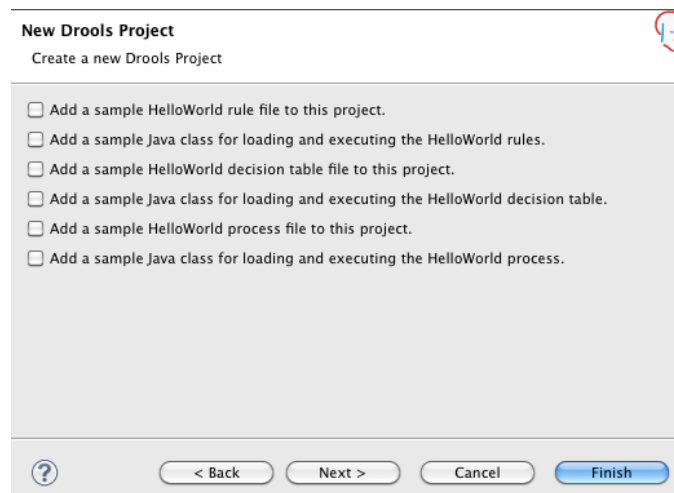


Figure 2-3. Skip the Samples

6. **Click Next**
7. You will need to create a new Drools Runtime here. Select **Configure Workplace Settings...**

8. Select the **Add**, name it **BRMS Runtime**, and browse to `brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/deploy/jboss-brms.war/WEB-INF/lib`
9. **Click OK**
10. Select the **BRMS Runtime** check box
11. **Click OK**
12. Select the **Drools 5.1 or above** option for **Generate code compatible** with dropdown
13. **Click Finish**

2.2.2 Add fact Objects to the project

1. Recursive copy (`cp -r`) in the the **org** folder and it's contents (with the **Driver**, **Policy**, and **Rejection** classes) from the `labs/lab02-drl-rules/solution/src/main/java` folder into the project's `src/main/java` folder
2. Review these domain objects
3. Also review the solution to the **Lab 01 Rule Analysis**

2.2.3 Author rules

2.2.3.1 Create a Rule Package

1. Select the `src/main/rules` folder
2. **Right+Click → New → File**
3. Name it `policyquote.package`
4. **Click Finish**
5. Open the file and add the following:

```
package org.acme.insurance.pricing  
  
import org.acme.insurance.Driver;  
import org.acme.insurance.Policy;  
import org.acme.insurance.Rejection;
```

2.2.3.2 Create four rules resource files

Use the rule creation wizard to create four individual rule files in the `src/main/rules` folder.

6. Select the `src/main/rules` folder
7. **Right+Click→New → Rule Resource**
8. **Select New Rule (individual rule)** for the **Type of rule resource**
9. Name the first file `safeyouths.drl`, and the package `org.acme.insurance.pricing`
10. **Click Finish**
11. Name the rule `safeyouths`. (For the purpose of these lab, please use the exact names provided here)
12. Create a separate file for each rule; this makes integration with the Guvnor easier. Name the others `riskyouths`, `safeadults`, and `riskyadults`.
13. In each of the four rules add the following import statements under the package declaration:

```
import org.acme.insurance.Driver;  
import org.acme.insurance.Policy;
```

2.2.3.3 DRL rule authoring in JBDS

14. Write the LHS of each rule. It should look something like:

```
when  
  
    #conditions
```

```
driver : Driver(age > ???, numberOfAccidents > ?? || numberOfTickets >= ??)
policy : Policy(price == 0, policyType == "AUTO")
```

15. Use a modify statement in the RHS of the rule to set the price and update the object. You can use either Java or MVEL dialect (if you use MVEL remember to set the dialect attribute to "mvel" under the rule name).

MVEL Syntax:

```
modify(policy) {price = 300}
```

Java Syntax:

```
modify(policy) {setPrice(300)}
```

16. Express each rule from the **Lab 1 Rule Analysis** solution in DRL syntax.
17. Also add a `println` statement for debugging purposes:

```
System.out.println("fired rule " + kcontext.getRule().getName());
```

2.2.4 Test the rules

1. Copy `com/sample/PolicyQuoteRulesTest.java` from `lab2-drl-rules/solution` to `src/main/java` in order to test the rules.
2. Examine the source code. You may have to add `JUnit.jar` to the java build path. Use the latest one provided with JBDS (Eclipse). Use **Project** → **properties** → **Java Build Path** → **Libraries** → **Add Library...** → **Next** → **JUnit4** → **Finish** → **Ok**
3. Create a folder at the root of the project called `log` (we will talk more about the `log` in a later lab)
4. Run the test class as a JUnit test. Select `PolicyQuoteRulesTest.java` → **Right+Click** → **Run As** → **JUnit Test**
5. The **JUnit** view will display a green bar if all the tests passed correctly. If any of the tests failed, make corrections before proceeding.

2.3 Lab 3 – RETE

Goal: Examine the policy quote rules using the RETE Viewer and the Agenda log.

2.3.1 RETE viewer

1. Open the `policyquote-rules/src/rule/safeadults.drl`
2. Click on the **Rete Tree** tab at the bottom of the editor
3. Observe the RETE View
4. Note that the colors are different from the presentation:
 - Red == ObjectTypeNode
 - Blue == Alpha Node
 - Yellow == leftinput adapter
 - Green == Join Node
 - Black == Terminal Node
5. Click on a node; it will show some info in the Properties panel (as in what the constraint is, type etc)

Note: Sometimes the properties panel in JBDS can act “sticky”. When this occurs you can often work-around the problem by minimizing and maximizing the panel to force it to refresh.

6. Identify each type of node by clicking on it. If you click on the black node, you will see the rule name
7. Remove the constraint on the Driver fact for `age > 24`
8. Then switch to the **Rete Tree** and view it
9. You should see one of the Alpha Nodes was removed
10. Add the `age > 24` constraint back in

2.3.2 Audit log

1. If it is not already there, add a folder at the root of the project called `log`
2. Open `PolicyQuoteRulesTest.java` and look for the `logger =` statement
3. The engine will log its actions to a log file in that directory (`log/policyquote`)
4. Rerun the test class
5. Open the audit view, **Window → Show View → Other → Drools → Audit**
6. Click on the “Open Log” button in the audit view window
7. Browse to the directory mentioned above, and open the log file
8. This will show a view of rule engine events from the last run (in the order in which they happened)
9. Clicking on an “Activation executed” node will show what caused it (you can also right click and “show cause”). This is normally the result of an object assertion (creates the activation).

Note: Recall that an activation is a rule that is ready to be fired for a tuple of facts. When a rule consequence is fired, you will see that as “Activation executed” in the view. Any resulting working memory actions (e.g., assertions) will be shown.

2.3.3 Debug

1. Next set a breakpoint on the consequence of one of the rules, e.g., on the modify statement. This is achieved in jbds with a double click in the margin. Note the blue dot in the illustration below.

```

then
    #actions
    modify(policy) {setPrice(300)}
    System.out.println("fired rule " + kcontext.getRule().getName());
  
```

2. Then debug the test class as a **Drools JUnit Test**.
 - Select `PolicyQuoteRulesTest.java` → Right+Click → Debug As → Drools JUnit Test

2.4 Lab 4 – BRMS Guided Rule Editor

Goal This exercise uses the BRMS Guided Rule Editor to create a few new rules.

2.4.1 Add rule categories

Categories enable assets to be grouped according to a purpose as identified by the category name.

1. If it is not already running, restart your server.
2. Open a browser to <http://localhost:8080/jboss-brms>
3. Sign on as **admin** with password **admin**
4. Click on **Administration** → **Category**
5. Add the hierarchy of categories **Insurance** → **Policy** → **Pricing**. (Highlight the higher level category and then select **New category**)

2.4.2 Add a rule package

The Rule Package is the deployment unit for a set of related rules.

1. To create a new package, go to the Package management feature (**Knowledge Bases**) and Click on **Create New** → **New Package**
2. Give it the name `org.acme.insurance.pricing`
3. Click **Create Package**

2.4.3 Fact model

Rules need a fact model (object model) to work with. To upload a model, use the `insurancepolicy.jar` at the root of the `lab04-brms-gre` directory).

1. Click on **Upload POJO Model Jar** from the **Knowledge Bases** → **Create New** menu
2. From the **New model archive dialog**, choose the `org.acme.insurance.pricing` package name from the list.
3. Enter `PricingFactModel` for the **Name** and Click **OK**
4. Click **Choose File** to select the jar file and then **Upload**
5. Click **File** → **Save and close**; On the dialog click **checkin**

FYI: Don't be confused by the use of the word "**checkin**". Although the BRMS Guvnor repository will save snapshots of the package, it is not designed to be a source code repository. I.e. it will not do branches, merges, version comparisons and other functions that are required for source code management.

2.4.4 Create exclusion rules

We want to use the **Business Rule Editor** to create several exclusion rules. There are multiple rule "formats", but from the BRMS point of view, they are all "**assets**".

You create a rule by clicking the **Create New** → **New Rule**. Enter a Name. You will also have to choose one category (choose **Insurance** → **Policy** → **Pricing**). Categories provide a way of viewing rules that is distinct from packages (and you can make rules appear in multiple packages). Also the package will have defaulted to the current package (`org.acme.insurance.pricing`).

Note that our fact model contains a Rejection class with a reason code. We do not care how many reasons we reject a driver, so whichever reason occurs first we can stop evaluating rules. So we need to add a constraint to each rule that tests if a Rejection already exists.

The rules will look like the following:

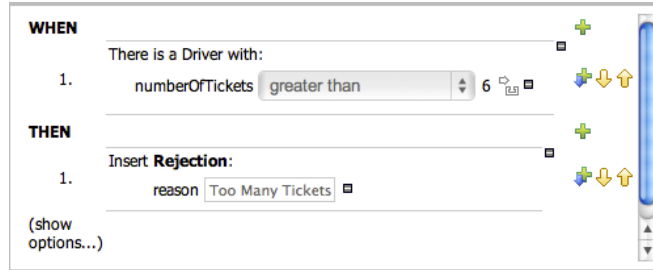


Figure 2-4. Guided Rule Editor

Create rules for the following:

| Rule Descriptions: 4 Exclusion Rules |
|--|
| <p>Rule "TooYoung"</p> <p>If the driver age is less than 16,</p> <p>Then reject with reason Too Young</p> |
| <p>Rule "TooOld"</p> <p>If the driver age is greater than 100,</p> <p>Then reject with reason Too Old</p> |
| <p>Rule "TooManyAccidents"</p> <p>If the driver numberOfAccidents is greater than 4,</p> <p>Then reject with reason Too Many Accidents</p> |
| <p>Rule "TooManyTickets"</p> <p>If the driver numberOfTickets is greater than 6,</p> <p>Then reject with reason Too Many Tickets</p> |

Table 2-5. Exclusion Rules

1. Once you have finished each rule, **Select Source** → **View source**, then **Source** → **Validate**, then **File** → **Save changes** on each rule.
2. After you have created the four rules, build the package; open the **org.acme.insurance.pricing** package, and build the whole package by **Selecting Edit (Edit tab not the Edit menu)** → **Build package** as illustrated below.

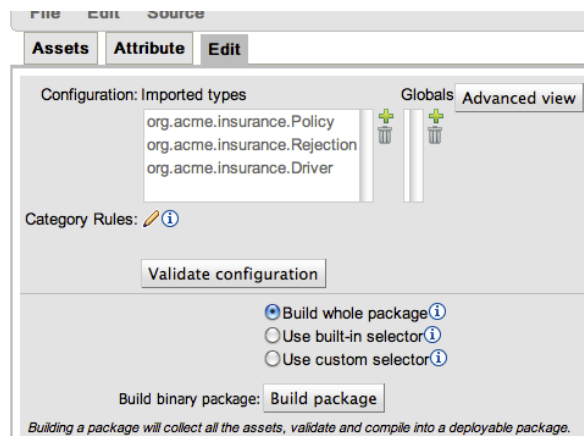


Figure 2-6. Build Package

2.5 Lab 5 – BRMS Decision Table

Goal: This exercise uses the BRMS Web Decision Table Editor to create a few new decision table rules.

2.5.1 Create the web-based decision table

We want to use the **Web Decision Table Editor** to create several discount rules.

1. If it is not already running, restart your server.
2. Open a browser to <http://localhost:8080/jboss-brms>
3. Sign on as **admin** with password **admin**
4. Add the category **Insurance** → **Policy** → **Pricing** → **Discount**
5. Create a new rule by clicking **Knowledge Bases** → **Create New** → **New rule**
6. Enter the Name **CreditScoreDiscount**
7. You will also have to choose one category (choose **Insurance** → **Policy** → **Pricing** → **Discount**)
8. From the **Type (format) of rule:** **select Decision Table (Web – guided editor)** from the drop-down
9. **Select org.acme.insurance.pricing** for the package
10. **Click OK**

FYI: If your next screen is a file upload screen then you accidentally selected **Decision Table (Spreadsheet)** instead of **Decision Table (Web – guided editor)**. In this case select **Delete** from the File menu and try again.

2.5.2 Add condition columns

1. **Open Decision Table** (small plus sign to the left)
2. Add the following **Condition columns**:

| Pattern | Calculation Type | Field | Operation | From Entry Point | Value List | Column Header | Default Value | Hide Column |
|-------------------------|------------------|-------------|-----------------------|------------------|-------------|---------------|---------------|-----------------|
| Driver (bind to driver) | Literal value | creditScore | Greater than | MinScore | Leave Blank | Min Score | Leave Blank | Leave Unchecked |
| Driver (bind to driver) | Literal value | creditScore | Less than or equal to | MaxScore | Leave Blank | Max Score | Leave Blank | Leave Unchecked |
| Policy (bind to policy) | Literal value | price | Greater than | Price | Leave Blank | Price | 0 | Check |

Table 2-7. Condition Columns

Note: If the Fact type dropdown is blank, then return to the model asset and import the insurancepolicy.jar model file from the lab04 solution again.

2.5.3 Add action columns

1. **Open the Action columns**
2. Add the following Action column:
3. **Select Set the value of a field** for the **Type of action column**:

| Fact | Field | Value list | Column Header | Update Engine with changes | Default Value |
|--------|---------------|-------------|----------------|----------------------------|---------------|
| policy | priceDiscount | Leave blank | Price Discount | check | Leave Blank |

Table 2-8. Action Columns

2.5.4 Add the No Loop attribute

1. Return to **Decision table** → (**options**), Add the no-loop attribute
2. Do not hide the column; do not try to set a default value

2.5.5 Add rules to the decision table

1. Click **Add row...** at the bottom of the screen
2. Now use the green plus signs to add 3 more rows:

| Description | MinScore | MaxScore | No Loop | PriceDiscount |
|------------------|----------|----------|---------------|---------------|
| bad credit | 0 | 400 | Check the box | 0 |
| good credit | 400 | 600 | Check the box | 10 |
| very good credit | 600 | 700 | Check the box | 25 |
| excellent credit | 700 | 800 | Check the box | 50 |

Table 2-9. Decision Table Data

- 3.
4. Once you have finished adding the rows, select **View source**, then **Validate**, then **Save changes**.
5. After you have created the four rules, click on the package, open the **org.acme.insurance.pricing** package, and build the whole package.

2.6 Lab 6 – BRMS Test Scenarios

Goal This exercise uses the BRMS Test Scenarios to create a few test scenarios for our rules.

2.6.1 Create test scenarios

1. Open a browser to <http://localhost:8080/jboss-brms>
2. Sign on as admin / admin for username / password
3. You will create a Test Scenario for each row in the table below by selecting the **Knowledge Bases** → **Create New** → **New Test Scenario**
4. Enter the name and the package
5. Use the following to create each test scenario:

| Test Name | Given | Expect |
|----------------------|----------------------------|--|
| TooYoungTest | Driver age 15 | Rejection with ReasonCode "TooYoung" |
| TooOldTest | Driver age 101 | Rejection with ReasonCode "TooOld" |
| TooManyAccidentsTest | Driver numberOfAccidents 5 | Rejection with ReasonCode "Too Many Accidents" |
| TooManyTicketsTest | Driver numberOfTickets 7 | Rejection with ReasonCode "Too Many Tickets" |

Table 2-10. Test Scenarios

6. Run each Test Scenario individually. Note you can use the configuration options to specify that only the specific rule being tested fires.
7. Open the **QA** pallet, select the **org.acme.insurance.pricing** package and click **Run All Test Scenarios**

2.7 Lab 7 – IDE BRMS Integration

Goal This exercise adds the technical rules created in lab2-drl-rules to the BRMS using the Drools IDE Eclipse Guvnor Integration feature.

2.7.1 Upload rules to BRMS

1. Open JBoss Developer Studio
2. Open the **policyquote-rules** project
3. **Window** → **Open Perspective** → **Other...** → **Guvnor Repository Exploring**
4. Add a new repository. **File** → **New** → **Other** → **Guvnor** → **Guvnor repository location**
5. Enter **admin** for the **User Name** and **Password** fields
6. **Click Finish**
7. Return to the Drools perspective

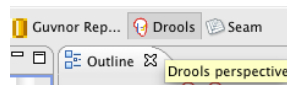


Figure 2-11. Drools Perspective Selection

8. In the Package Explorer panel, **Right+Click** on each DRL rule, **Select Guvnor**→**Add...** →**Use an existing Gufnor repository connection** →**Next>**
9. Expand the folders on the dialog. **Select** the **org.acme.insurance.pricing** package →**Finish**
10. Note that a Guvnor checkin timestamp appears next to the file name in JBDS
11. In the BRMS Guvnor UI open the **org.acme.insurance.pricing** package and open the Technical Rules tab. Open each rule and validate them. If the Technical Rule assets do not appear, the do a **Save changes** in Guvnor.

2.7.2 Create test scenarios

1. Now write a Test Scenario to test each rule individually in the BRMS.

2.8 Lab 8 – Domain Specific Language

Goal The goal is to learn how to use a Domain Specific Language.

2.8.1 Author rules

1. Copy `acme.dsl` from `labs/lab08-dsl/` to `src/main/rules/`
2. Open the file and examine them dsl mappings
3. Use the Drools IDE Eclipse Guvnor Tooling to upload `acme.dsl` to the BRMS (right+click →Guvnor →Add..)

FYI: As an alternative this lab can be done solely using the Drools IDE / JBDS

4. Open the BRMS, go into the `org.acme.insurance.pricing` package and add three more business rules using the DSL. To create the rules use **Knowledge Bases →Create New →New Rule**
5. **Select Business Rule (Guided editor)** for the **Type (format) of rule**
6. Create rules using the following table:

| Rule | LHS | RHS |
|-----------------------|--|---------------------------------|
| AccidentSurcharge | If the Driver has numberOfAccidents > 2 and there is a policy with price > 0 | Then price = price + 200 |
| NoviceDriverSurcharge | If the Driver has age < 19 and there is a policy with price > 0 | Then price = price + 250 |
| NewerVehicleSurcharge | If the vehicle year is >= 2004 and there is a policy with price > 0 | Then policy price = price + 100 |

Table 2-12. Rules for DSL

Note: If your DSL mappings are not available in the Add Conditions dialog, Open the `acme.dsl` asset to verify it, then explicitly save and close it.

7. Set the **no-loop** attribute
Setting no-loop to true means those additional attempts to create the Activation for the current set of data will be ignored.
8. Set the **lock-on-active** attribute

FYI: When a ruleflow-group (discussed later) becomes active or an agenda-group receives the focus any rules that have **lock-on-active** set to true cannot place subsequent activations onto the agenda. The rules are matched once and the resulting activations discarded. This is a stronger version of no-loop. It's ideal for calculation rules where you have a number of rules that will modify a fact and you don't want any rule re-matching and firing. In summary fire these currently active rules and only these rules. No matter how the data changes, do not allow any more activations for the rules with the attribute set to true. When the ruleflow-group is no longer active or agenda-group loses the focus those rules with **lock-on-active** set to true can once again add activations onto the agenda.

9. When we add Ruleflow in a later lab we can use the **lock-on-active** attribute to eliminate looping between surcharge rules and allow multiple surcharges to be added on.

2.8.2 Create test scenarios

Create Test Scenarios for each rule, and then run the tests.

2.9 Lab – 9 Advanced Rules

Goal The goal is to learn how to use advanced language features such as accumulate.

2.9.1 Author rules

Note: This lab can also be done within BRMS directly using the Guided Rule Editor as long as any classes used in the result pattern are also included in the fact model.

1. Create a new individual rule in the Drools IDE / JBDS policyquote-rules project
2. Name it `pricemultiplevehicles.drl`
3. Create rules using the following table:

| Rule | LHS | RHS |
|-----------------------|--|--|
| PriceMultipleVehicles | For all vehicles for the same driver A policy with policyType==MASTER must exist in addition to the AUTO policies | Then update a master policy with price = the sum of the individual policy prices |

Table 2-13. Rule Guidelines

Hints:

- Use accumulate with total function - see slides.

```
<result pattern> from accumulate
    (
        <source pattern> ,
        init ( <init code> ),
        action ( <action code> ),
        reverse ( <reverse code> ),
        result ( <result expression> )
    )
```

- Coerce the result to an int. Use either

`total : Integer(intValue > 0)`
 or
`Integer (total : intValue > 0)`
- Use Java dialect
- Use the following attributes

`salience -10`
- Use modify in the consequence

`modify ($policyM) {setPrice(total)}`

2.9.2 Add the rule to the package in BRMS Guvnor

Recall using **right+click** → **Guvnor** → **Add...**

2.9.3 Create test scenarios

1. Create a Test Scenario for this rule
2. Name it **PriceMultipleVehiclesTest**
3. Use the following to build the test scenario:

| Given | Expected |
|---|---|
| Driver d age = 30 numberOfAccidents = 0 numberOfTickets = 0 creditScore=715 | Row 4 CreditScoreDiscount to fire 2 times safeadults to fire 2 times |
| Policy p1 vehicleYear = 2003 policyType = AUTO driver = d | Price = 120 priceDiscount = 50 |
| Policy p2 vehicleYear = 2004 policyType = AUTO driver = d | Price = 120 priceDiscount = 50 |
| Policy pM policyType = MASTER driver = d | Policy pM price = 240 PriceMultipleVehicles to fire at least once |

Table 2-14. Multiple Vehicle Test

4. In the BRMS Test Scenario use =d (or =driver) to establish the relationship between the policies and the driver
5. Run the test scenario

2.10 Lab 10 – Ruleflow

Goal The goal is to learn how to use ruleflow to control the sequence of rule execution. With this and the remaining labs, we transition our focus from rules management to business process modeling (BPMN2) and control. These labs will be done within the Guvnor's Web Designer.

2.10.1 JBoss Business Process Modeling (jBPM) Preparation

Some setup is required to prepare your environment for the remainder of the labs.

2.10.1.1 Install BRMS

This step is required if you have not performed labs 1 through 9. Otherwise you may skip to section 2.10.1.4 Guvnor access to the domain model.

Pre-requisites: use JDK 1.6, version 16 or later.

1. Create a folder **brms-workshop**
2. Copy **brms-cd** from the CD / flash drive into **brms-workshop**
3. Create a folder under **brms-workshop** called **software**
4. Unzip **jboss-brms-standalone-5.3 x.zip** into the **software** folder

2.10.1.2 Setup user authorizations

5. Open **brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/conf/props/brms-users.properties**
6. Remove the # from the line that reads **#admin=admin**
7. Save and close the file.

2.10.1.3 Make log more useful

As a convenience, the following steps are helpful for reducing the volume of excess logs sent to the console.

8. In an editor open the **brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/conf/jboss-log4j.xml** file.
9. Add the two categories illustrated below:

```
<!-- ===== -->
<!-- Limit categories -->
<!-- ===== -->
    <!-- Exclude jackrabbit Session open and close messages -->
    <category name="org.apache.jackrabbit.core">
        <priority value="ERROR"/>
    </category>
    <!-- Limit Mina Logging -->
    <category name="org.apache.mina.filter.logging">
        <priority value="ERROR" />
    </category>
```

2.10.1.4 Guvnor access to the domain model

10. Navigate to and **select** the **insurancepolicy.jar** file in the root of the **lab04-brms-gre** directory.
11. Copy the file and paste the copy into the **brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/deploy/gwt-console-server.war/WEB-INF/lib** directory.

2.10.1.5 Setup jbpm properties

12. If you have been working through the previous 9 labs, then stop your server prior to proceeding.

13. Navigate to the `brms-standalone-5.3.0/jboss-as/server/default/deploy/jbpm-human-task.war/WEB-INF/classes/org/jbpm/task` directory
14. Create a new directory named `service`
15. Copy `jbpm.usergroup.callback.properties` from `labs/lab10-ruleflow/callback` to `brms-standalone-5.3.0/jboss-as/server/default/deploy/jbpm-human-task.war/WEB-INF/classes/org/jbpm/task/service`

2.10.1.6 Additional configuration

Some early release versions require additional setup steps for the labs to work correctly. Consult your instructor to see if any additional steps are required.

2.10.1.7 Start the server

16. Open a command window to `brms-workshop/software/brms-standalone-5.3.0/jboss-as/bin`
17. Start the server with the following: `./run.sh -c default -b localhost`

FYI: The `-c` option is used to specify the desired server configuration. The `-b` option is used to specify the network interface that the server will use to listen for messages.

On windows use `run.bat` instead of `run.sh`

18. Wait for the server log to display the **Started in Xm:XXs:XXXms** message.
19. Login to Guvnor by opening a browser and entering the URL <http://localhost:8080/jboss-brms>
20. Enter **admin** for both the **userId** and **password**
21. When asked to if you want to import the samples, select **No**

2.10.1.8 Import repository into Guvnor

22. On the panel on the right, select **Administration → Import Export → Choose File**
23. Browse to `brms-cd/labs/lab10-RuleFlow/repo/repository_export.xml`
24. Click the **Import** button

2.10.2 Create a new business process

Create a new business process to manage rule flow using the Guvnor Web Designer.

1. Select **Knowledge Bases** on the left hand pallet.
2. Select **Create New → New BPMN2 Process**. You will get a dialog box.
3. Name the new process `policyquotecalculatationprocess`
4. For the package, select `org.acme.insurance.pricing`
5. Click **Create**

2.10.3 Set properties for the diagram

The Id value for the diagram is important. As you will see in a later lab, the Id value can be used to execute this business process.

1. View the properties panel for the entire diagram by clicking on the canvas and selecting the << icon in the upper right hand corner of the canvas to open the **Properties** panel.
2. Verify that the Id property of the process is set to the following value:
`org.acme.insurance.pricing.policyquotecalculatationprocess`
3. Likewise, verify that the Package property is set to `org.acme.insurance.pricing`

FYI : It is a recommended naming convention to derive the ID by concatenating the package name and the process name separated by a dot.

2.10.4 Add the start event node and task

Event An event is something that “happens” during the course of a process. These events affect the flow of the process and usually have a cause or an impact and in general require or allow for a reaction.

Task A task is an atomic activity in a BPMN2 process.

1. **Select** the “>>” icon in the upper left corner of the drawing pallet to open the **Shape Repository**.
2. **Open** the **Start Events** pallet and **drag** a **Start Event** icon onto the drawing pallet.
3. With the start event icon selected, you will see some shortcut items next to the start node:



Figure 2-15. Shortcut Pallet

4. **Click** the square with rounded corners icon.
5. With the new task node still selected, click on the “<<” icon on the upper right hand corner of the drawing pallet to open the Properties view.
6. Change the **Name** of the Task to **Rejection**
7. For the **Task Type** **Select Business Rule**
8. Under the **Task Type**, **click** the + sign to open the tree of **More Properties**
9. Find the **RuleFlow Group** property and set the **value** to **rejection**

2.10.5 Conditional business process flow

1. **Select** the **Rejection** task on the canvas to bring up the shortcut menu.
2. **Click** the **Exclusive Or** gateway from the pallet to the right.
3. On the **Properties** panel, give the **Exclusive Or** gateway a **Name** of **Gateway**.



Figure 2-16. Exclusive Or

4. With the **Exclusive Or** selected, **drag** a second **Exclusive Or** gateway to the right, and **drop** it on the canvas. Use some extra space between the two gateways. (See figure 2.17 below for an illustration)
5. On the **Properties** panel, give the **Exclusive Or** gateway a **Name** of **Join Gateway**.
6. The **End Event** is the circle icon in the shortcut pallet that doesn't have a double line around it. With the second **Exclusive Or** gateway selected, **select** the **End Event**.
7. **Select** the first **Exclusive Or** gateway again. This time **drag** the **Task** icon from the shortcut pallet and **drop** it onto the canvas under the sequence line that connects the two gateways.
8. With the **Task** node selected, go to the **Properties (Task)** panel on the right.
9. **Select Business Rule** for the **TaskType** and name it **Calculation**
10. Find the **RuleFlow Group** property and set the **value** to **calculation**
11. **Select** sequence flow between the first **Exclusive Or** gateway and the **Calculation Task**
12. On the **Properties (Sequence Flow)** panel on the right **select** **drools** for the **Condition Expression Language** and enter **not Rejection()** for the **Condition Expression**. Enter **Non Rejection** for the **Name** property and use a value of 1 for the **Priority** property.

Note: When using the XOR type, the lowest priority route will be evaluated first.

13. On the sequence flow between the two gateways **select drools** again for the **Condition Expression Language** and **Rejection()** for the **Condition Expression**. Enter **Rejection** for the **Name** property and use a value of 2 for the **Priority** property.

2.10.6 Additional Business Rule Tasks

1. **Select** the **Calculation** task. **Click** on another **Task** icon from the shortcut pallet.
2. With the **Task** node selected, go to the **Properties (Task)** panel on the right.
3. Add the following properties to the new task node:

| Property | Value |
|----------------|---------------|
| Name | Surcharge |
| RuleFlow Group | surcharge |
| Task Type | Business Rule |

Figure 2-17. Surcharge Task Properties

4. **Select** the **Surcharge** task. **Select** another **Task** icon from the shortcut pallet.
5. With the **Task** node selected, go to the **Properties (Task)** panel on the right.
6. Add the following properties to the new task node:

| Property | Value |
|----------------|---------------|
| Name | Discount |
| RuleFlow Group | discount |
| Task Type | Business Rule |

Figure 2-18. Discount Task Properties

7. **Select** the **Discount** task. **Select** another **Task** icon from the shortcut pallet.
8. With the **Task** node selected, go to the **Properties (Task)** panel on the right.
9. Add the following properties to the new task node:

| Property | Value |
|----------------|---------------|
| Name | Total |
| RuleFlow Group | total |
| Task Type | Business Rule |

Figure 2-19. Discount Task Properties

10. **Select** the **Total** Task on the diagram; drag the sequence line from the shortcut pallet to the second **Exclusive Or** gateway. Wait until the red highlights turn green prior to dropping the line.
11. Your diagram should resemble the following:

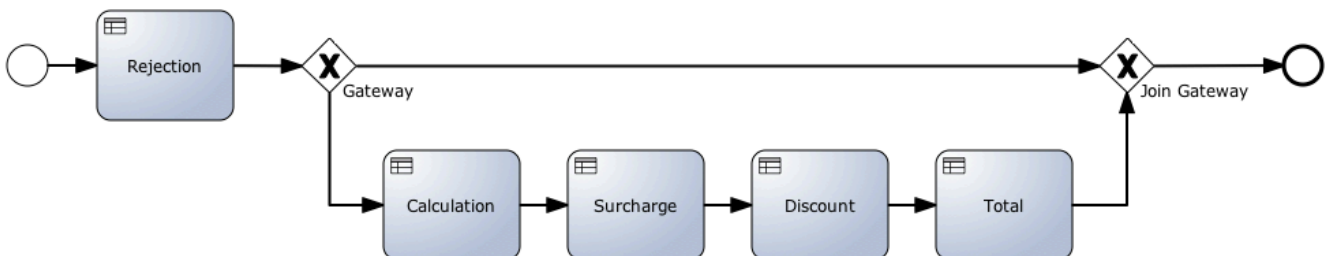


Figure 2-20. Calculate Policy Quote Process Diagram
2.10.7 Save the process image

1. Select the PNG button at the bottom of the pallet
2. Close the popup dialog box
3. Save your progress with **File** → **Save Changes** → **Check in**.

2.10.8 Add a rule to trigger the process

1. In Guvnor, **Select Knowledge Bases** on the left hand pallet.
2. **Select Create New** → **New Rule**. You will get a dialog box.
3. Name the new rule **StartRuleflow**
4. **Select org.acme.insurance.pricing** for the **package** and **Insurance/Policy/Pricing** for the **initial category**.
5. For the **Type (format) of Rule**, **select Business Rule (Guided Editor)**
6. **Click Ok**
7. **Select Show Options**.
8. **Click** the green Plus sign on the right to add an option.
9. On the dialog select an **Attribute of salience**
10. Add a value of 100 for the **salience** option.
11. **Click** the second green plus sign on the right to add a new **THEN** condition.
12. At the bottom of the table, **select Add free form drl**.
13. In the free form box, add the following text:

```
kcontext.getKnowledgeRuntime().startProcess("org.acme.insurance.pricing.policyquotecal
culationprocess");
```

14. Select **Source** → **Validate** to ensure the rule is valid.
15. Select **Source** → **View source** to view the drl syntax.
16. Save your progress with **File** → **Save Changes** → **Check in**.

2.10.9 Add ruleflow groups

In the BRMS, open every rule in the package, and add the **ruleflow-group** attribute using the tables as a guide. For business rules, this is an attribute added in the options as described above for salience. For the web decision table, options are exposed once the “Decision table” is expanded:

| Business Rule Name | Ruleflow-Group Assignment |
|-----------------------|---------------------------|
| AccidentSurcharge | surcharge |
| NewerVehicleSurcharge | surcharge |
| NoviceDriverSurcharge | surcharge |
| CreditScoreDiscount | discount |
| toomanyaccidents | rejection |
| toomanytickets | rejection |
| toold | rejection |
| tooyoung | rejection |

Figure 2-21. Business Rule to Ruleflow-Group mapping

For technical rules, options are typed in the lines below the rule name as follows: ruleflow-group "total"

| Technical Rule Name | Ruleflow-Group Assignment |
|-----------------------|---------------------------|
| PriceMultipleVehicles | total |
| riskyadults | calculation |
| riskyyouths | calculation |
| safeadults | calculation |
| saferyouths | calculation |

Figure 2-22. Technical Rule to Ruleflow-Group mapping

1. The business rules created in the BRMS in lab4 put in ruleflow-group **rejection**
2. The technical rules created in lab2 put in ruleflow-group **calculation**
3. The business rules added in lab7-dsl put in ruleflow-group **surcharge**
4. The decision table rules added in lab4b-web-dec-tables put in ruleflow-group **discount**
5. And the technical rule added in lab8 put in ruleflow-group **total**
6. Use the tables above to check your mappings

At this point, open the org.acme.insurance.pricing package from the left panel, navigate to the Edit tab and proceed to "Validate configuration" before "Build package" to verify your changes.

2.10.10 Create test scenarios

1. Create a Test Scenario for the entire package of rules
2. Call it **PolicyQuotePackageTest**
3. Use the following to build the test scenario:

| Given | Expected |
|--|---|
| Driver d age = 30 numberOfAccidents = 2 numberOfTickets = 1 creditScore=650 | Policy p1 price = 400 Policy p2 price = 300 Policy pM price = 700 p1.priceDiscount=25 p2.priceDiscount=25 Note: use small green icon to set attribute expectations |
| Policy p1 vehicleYear = 2006 policyType = AUTO driver = d (choose "bound variable") price = 0 priceDiscount=0 | |
| Policy p2 vehicleYear = 2003 policyType = AUTO driver = d (choose "bound variable") price = 0 priceDiscount=0 | |
| Policy pM vehicleYear =0 policyType = MASTER driver = d (choose "bound variable") price = 0 priceDiscount=0 | |

Table 2-23. Package Test

4. Save and run the test scenario

2.11 Lab 11 – Sub-Process

Goal Create the Policy Quote Process with Calculate Quote Sub-process (which uses the Rule Flow from Lab 10) in the BRMS Web Designer. Use a Script Task, before the Calculate Quote sub process to create the driver and policy facts from the input data.

The goal is to learn how to start using the BPMN2 Web Designer to create a business process model.

2.11.1 Lab preparation

2.11.1.1 Add new BPMN2 process to Guvnor

1. **Select Knowledge Bases** on the left hand pallet.
2. **Select Create New → New BPMN2 Process.** You will get a dialog box.
3. Name the new process **policyquoteprocess**
4. For the package, **select org.acme.insurance.pricing**
5. **Click Ok**

2.11.1.2 Add process variable declarations

The business process maintains the state of key data points that must be passed to different activities within the process. BRMS uses variable declarations to hold this information. For this lab, a number of data items are provided from previous processes.

Driver name, age, number of accidents, number of tickets and vehicle year are captured from form input. These data items are used to populate the Policy and Driver facts that will be inserted into the BRMS Working Memory for the Knowledge Base to be used by the Rules Engine.

Variable are required by the process to transfer information both to and from process activities. The following is used to add the variable definitions required for this process.

6. **Click** on an empty area of the diagram (not on any element) to make the **Properties** panel display the properties for the diagram.
7. Select the Imports property and add the following:

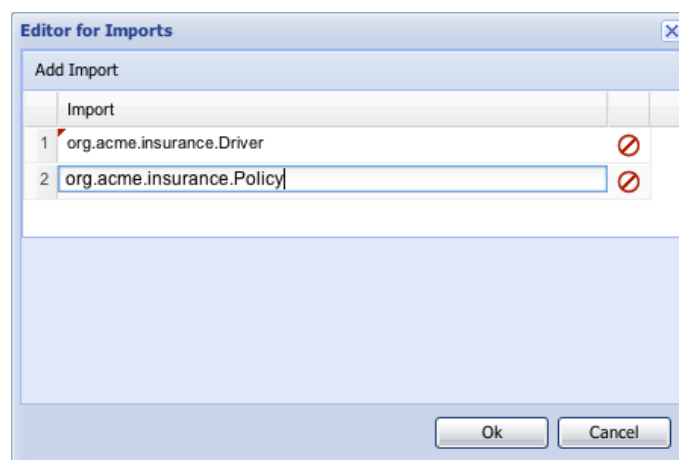


Figure 2-24. Process Imports

8. **Click** the down arrow on the value for the **Variable Definitions** property. This will bring up the **Editor for**

Variable Definitions dialog.

9. Use the **Add Variable** button to add the following:

| Variable Name | Type |
|-------------------|---------|
| driverName | String |
| driver | Driver |
| policy | Policy |
| age | Integer |
| numberOfAccidents | Integer |
| numberOfTickets | Integer |
| vehicleYear | String |
| price | Integer |

Table 2-25. Variable Definitions

2.11.2 Add the start event node and script task

1. Select the ">>" icon in the upper left corner of the drawing pallet to open the **Shape Repository**.
2. Open the **Start Events** pallet and drag a **Start Event** icon onto the drawing pallet.
3. With the start event icon selected, you will see some shortcut items next to the start node:



Figure 2-26. Shortcut Pallet

4. **Click** and **drag** the square with rounded corners icon; drag it to the right of the start event.
5. With the new task node still selected, click on the "<<" icon on the upper right hand corner of the drawing pallet to open the Properties view.
6. Change the **Name** of the Task to **Prepare Data**
7. For the **Task Type** **Select Script**
8. Under the **Task Type**, click the + sign to open the tree of **More Properties**
9. Find the **Script Language** property and set the **value** to **Java**
10. Find the **Script** property; click the value box to open the script dialog.
11. **Copy** and **Paste** the following script:

```
Driver _driver = new Driver();

String ageAsString = (String) kcontext.getVariable("age");
Integer _age = new Integer(ageAsString);
_driver.setAge(_age);

_driver.setDriverName((String)kcontext.getVariable("driverName"));



String accAsString = (String) kcontext.getVariable("numberOfAccidents");
Integer _numberOfAccidents = new Integer(accAsString);
_driver.setNumberOfAccidents(_numberOfAccidents);

String tikAsString = (String) kcontext.getVariable("numberOfTickets");
Integer _numberOfTickets = new Integer(tikAsString);
_driver.setNumberOfTickets(_numberOfTickets);

Policy _policy = new Policy();
_policy.setPolicyType("AUTO");

String vehYearAsString = (String) kcontext.getVariable("vehicleYear");
Integer _vehicleYear = new Integer(vehYearAsString);
_policy.setVehicleYear(_vehicleYear);
_policy.setDriver(_driver);

/* insert objects into working memory */
kcontext.setVariable("driver", _driver);
kcontext.setVariable("policy", _policy);
kcontext.getKnowledgeRuntime().insert(_driver);
kcontext.getKnowledgeRuntime().insert(_policy);
```

12. **Click** the **Ok** button on the dialog to close it.
13. **Select** the **Validate** button  from the toolbar.
14. You will see that it complains about a missing end event. You can optionally add an end event by dragging the thick-lined circle from the shortcut pallet to the right.
15. To remove an unwanted item on the drawing pallet, **select** it and **click** the red  on the toolbar.
16. At this point your diagram should look like the following:

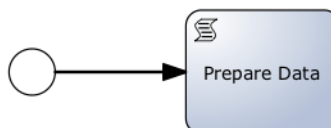


Figure 2-27. Process with Two Elements

2.11.3 Add a reusable sub-process

In BPMN2, a reusable sub-process is used to model a global pre-existing process. Use the following steps to add this activity to the diagram.

1. On the **Shape Repository**, open the **Activities** sub-menu to locate the **Reusable Subprocess** icon:

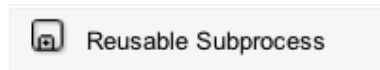


Figure 2-28. Reusable Sub-process icon

2. **Drag** the icon to the right of the **Prepare Data Task**.
3. Change the name to **Calculate Policy Quote**
4. **Select** the **Called Element** property. From the dialog, select the org.acme.insurance.pricing.policyquotecalculatonprocess as illustrated:

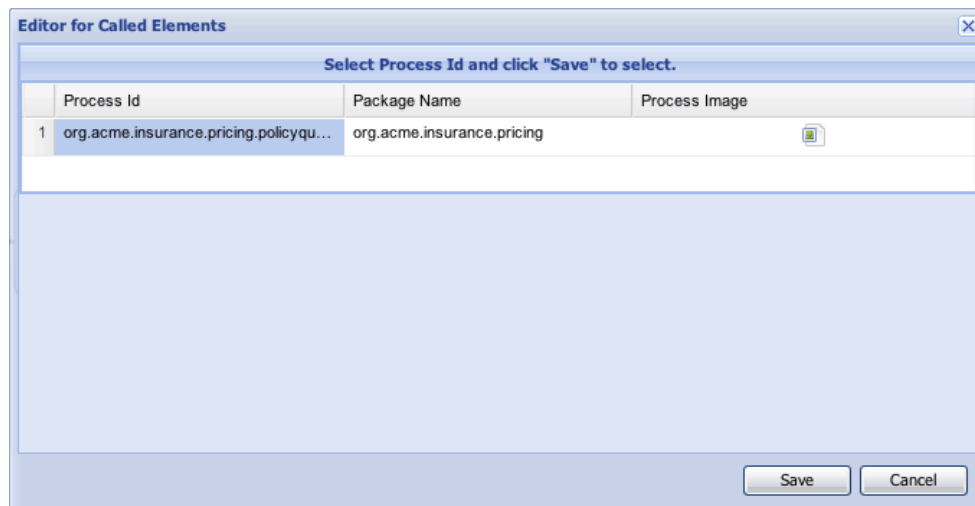


Figure 2-29. Called Element Selection

5. **Select** the **Prepare Data Task**; from the shortcut menu, **drag** a sequence flow arrow from the shortcut pallet to the new sub-process. The display will show red indicators changing to green when you are at the proper place to release the drag action.
6. At this point your diagram should look like the following:

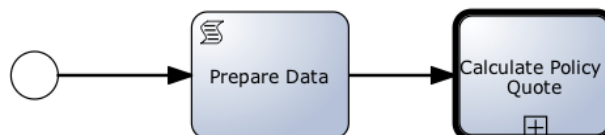


Figure 2-30. Process with Three Elements

Policy quotes coming out the system at this point are strictly based on the rules in the BRMS engine. This is a great start, but sometimes there is a need for special handling of certain cases in order to remain competitive. For this purpose, we need to setup an automatic manual review of any policy with a price quote over \$500. We will set that up in the next lab. Save the incomplete process to make sure your work is not lost.

2.12 Lab 12 – Human Tasks

Goal Use the Web Designer to create a User Task (Review Quote) after Calculate Quote to the Reviewers group. Add an Exclusive Gateway to route to the User Task if the policy price is greater than \$500. Add Users and Reviewer group to the relative properties files. Create a Test Scenario in Guvnor to test the process.

2.12.1 Create a user task

1. **Select** the **Calculate Policy Quote** node on the canvas to bring up the shortcut menu.
2. **Drag** the **Exclusive Or** gateway from the pallet to the right, and **drop** it on the canvas.
3. With the **Exclusive Or** selected, **drag** a second **Exclusive Or** gateway to the right, and **drop** it on the canvas.
4. **Select** the first **Exclusive Or** gateway again. This time **drag** the **Task** icon from the shortcut pallet and **drop** it onto the canvas under the sequence line that connects the two gateways.
5. With the **Task** node selected, go to the **Properties (Task)** panel on the right.
6. **Select User** for the **TaskType**
7. For the **Name** property enter **Review Quote**
8. **Select** the **Review Quote** node on the diagram again; drag the sequence line from the shortcut pallet to the second **Exclusive Or** gateway. Wait until the red highlights turn green prior to dropping the line:

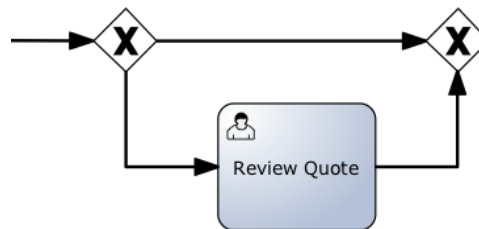


Figure 2-31. Gateways and Human Task

9. Select sequence flow between the first Exclusive Or gateway and the Review Quote Task
10. On the **Properties (Sequence Flow)** panel on the right **select drools** for the **Condition Expression Language** and enter **Policy (price > 500)** for the **Condition Expression**.
11. **Enter Over 500** for the **Name** of the sequence flow
12. On the sequence flow between the two gateways **select drools** again for the **Condition Expression Language** and **Policy (price <= 500)** for the **Condition Expression**.
13. **Enter 500 or less** for the name of the sequence flow
14. With the addition of an **End Event** the diagram should resemble the following:

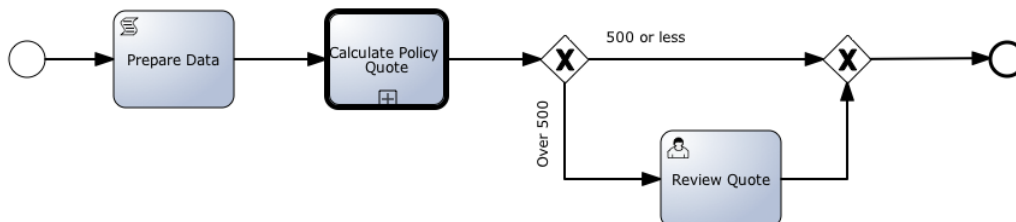


Figure 2-32. Policy Quote Process Diagram

2.12.2 Add Users and Reviewer group

A user must be a member of the reviewer group to perform a quote review. Let's set that up.

1. **Select** the **Review Quote** Node
2. **Expand** the **More Properties** group under the **Properties(Task)**
3. **Select** the **GroupID** property. Enter the name: **reviewer**
4. Near the bottom, set the value of the **Task Name** property to **reviewQuote**
5. Save your progress with **File** → **Save Changes** → **Check in**.

Warning: Spaces in Task Names have been known to cause errors in subsequent processes.

2.12.3 Create Data Input Sets and Data Output sets

Data input sets are used to collect data for use by the task; data output sets capture data coming out of the task.

The following sets up the input and output data sets for the **Review Quote** task.

1. **Select** the **Review Quote** task.
2. On the **Properties(Task)** panel **select** the drop down arrow for the value of the **DataInputSet** property.
3. Use the **Add Data Input** button to add the following:

| Name | Type |
|-----------------------|---------|
| taskAge | Integer |
| taskNumberOfAccidents | Integer |
| taskNumberOfTickets | Integer |
| taskVehicleYear | String |
| taskPolicy | Policy |

Table 2-33. Data Input Set

4. The Data Input dialog should resemble the following:

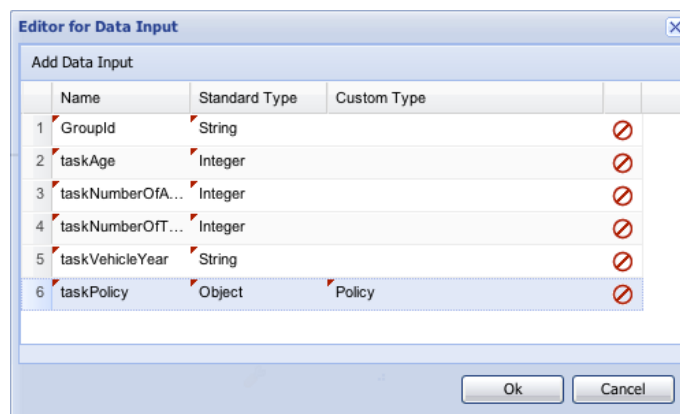


Figure 2-34. Data Input Set Dialog

5. Likewise, use the **DataOutputSet** property to add **taskPrice** as an **Integer** like the following:

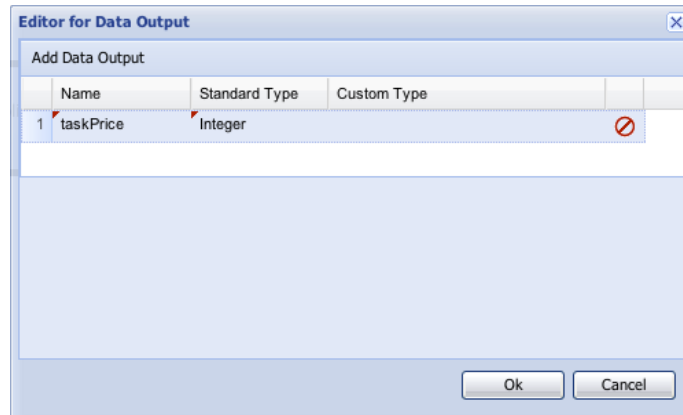


Figure 2-35. Data Output Set Dialog

6. Find the **Assignments** property just above the **DataInputSet** property. Use the **Editor for Data Assignments** dialog to add the assignments illustrated below:

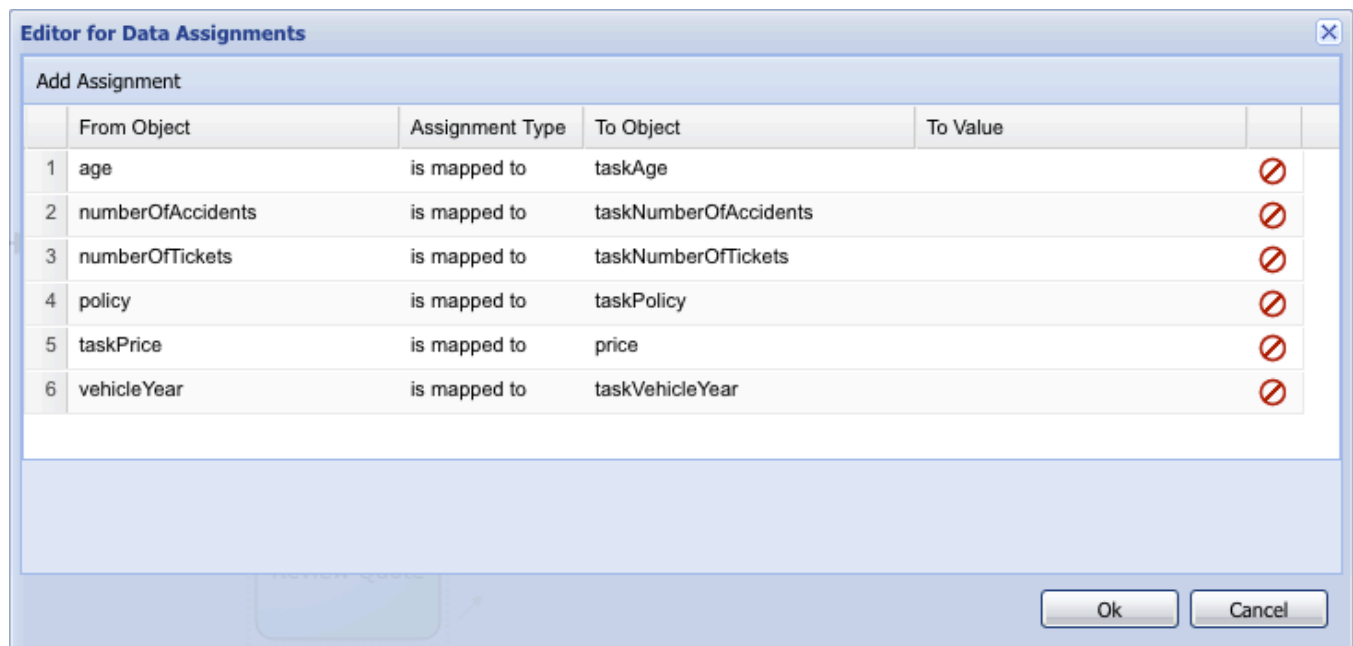


Figure 2-36. Variable Assignments Dialog

7. Save your progress with **File** → **Save Changes** → **Check in**.

2.12.4 Capture the Edited Price

We will use an Exit Script to transfer the new price value from the Review Quote task back to the Policy object.

1. Select the Review Quote task
2. Bring up the dialog for the On Exit Action in the properties panel
3. Select the Add Action button
4. Enter the following script:

```
((Policy)kcontext.getVariable("policy")).setPrice(Integer.valueOf((String)kcontext.getVariable("price")));
```

The price set by the user in the User Task has been mapped back to a process variable called “price”. This script sets the values of this variable on the **policy** variable, also stored in the process.

Save your work on the process by saving and checking it in..

2.12.5 Add Task to Reject Facts

Since the jBPM Console uses a long running session for multiple rule execution cycles, facts must be cleared from the working memory in order to not interfere with later rule executions. The following is used to remove the facts.

2.12.5.1 Create three retractions rules

1. Create three new Business Rules (Recall: **Knowledge Bases** → **Create New** → **New Rule**)
2. For each rule, use **org.acme.insurance.pricing** for the package and Insurance/Policy/Pricing for the initial category.
3. Set the type of each new rule to **Business Rule**
4. Use the rule names from the table below. **Click Ok**
5. Use the following table to create the rules. For the When, scroll down to find the Object type and after choosing it click on the generated sentence to set a variable name on it. For the Then, “Add free form drl”:

| Rule Name | RuleFlow-Group | When | Then |
|------------------|----------------|------------------------|--|
| RetractDriver | retract | driver: Driver() | System.out.println("Retracting Driver " + driver.driverName + " - " + driver.age); retract(driver); |
| RetractPolicy | retract | policy: Policy() | System.out.println("Retracting policy: " + policy.price); retract(policy); |
| RetractRejection | retract | rejection: Rejection() | System.out.println("Retracting rejection: " + rejection); retract(rejection); |

Table 2-37. Retraction Rules

Warning: Be careful to save your progress after each rule is created. (Recall: **Select File** → **Save Changes** → **Check in**)

2.12.5.2 Add a task to execute the rules

6. Select the end event of the process (if you’ve created one) and delete it. The sequence flow leading to it should be automatically removed as well.
7. Drag a new **Task** from the shortcut menu for the 2nd **Exclusive Or** gateway to the right.
8. Name the new task **Retract Facts**
9. Set the **Task type** to **Business Rule**
10. Set the **Ruleflow Group** property to the value **retract**
11. Add a new **End** event after this Business Rule task.
12. Save your progress: **Select File** → **Save Changes** → **Check in**.
13. Your diagram should resemble the following:

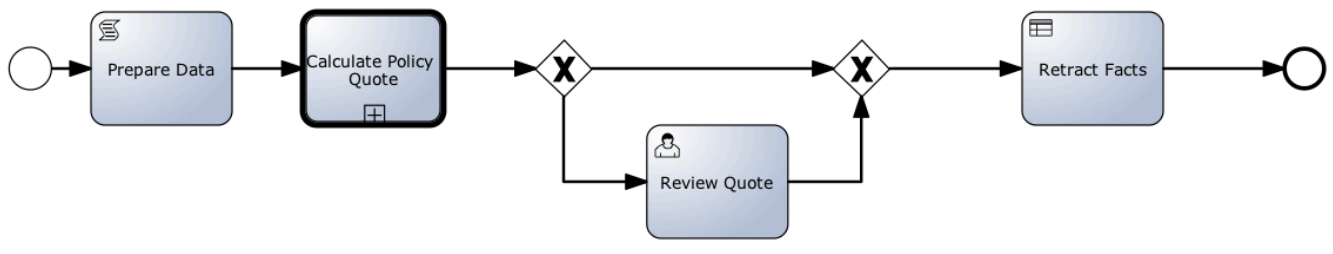


Figure 2-38. Retract Facts Added

2.12.6 Generate a Process diagram image

The task diagram image is required by the jBPM Console to show process execution progress. Do the following to create the diagram:

At the bottom of the screen press the  button

2.12.7 Add Users and Roles to the Configuration

Setting the GroupId task input variable to reviewer means that a user must be in the reviewers group in order to execute the task. Use the following to setup a user:

1. Open `brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/conf/props/brms-users.properties` in a text editor
2. Add a user to the file such as `<your name>=<password>` to the file, using a name and password without spaces.
3. In the `brms-roles.properties`, add `<your name>=user`. This allows you to log into the jbp console.
4. Change directories to the parent directory (`conf`) and create a new file named `roles.properties` with the following content:

```
# These roles are used by the jBPM Console
admin=reviewer
<your name>=reviewer
```

Figure 2-39. roles.properties

5. Open the jBPM console at <http://localhost:8080/jbpm-console>
6. Log in using your username and password
7. You should be able to successfully log into the console
8. Log out of the jbp console. We will use it in the next lab.


2.12.8 Save your progress

1. Return to the **JBoss BRMS** page and navigate to the policy quote process.
2. **Select** the **Validate** button from the toolbar
3. You should get a red **✗** near the **Review Quote** user task.
4. If hover your cursor over the **✗** you will see a context message box stating that it is expecting the user task to have task form assigned. This problem will be resolved in the next Lab.
5. Build the package to ensure your project is sound.
6. Save your work: **Select File** → **Save Changes** → **Check in**.

Lab 13 – Forms

Goal Add a process start form and review quote form using Web Forms Designer, and test the process using the BPM Console.

2.12.9 Add a process start form using Forms Designer

1. Open `policyquoteprocess` in JBoss BRMS Guvnor.
2. Click the “**Generate Task Form template**” button  on the Web Designer’s toolbar
3. The tool will report the build of two forms for you
4. Use the **View Source** links if you wish to examine the generated pages. Close them when done.

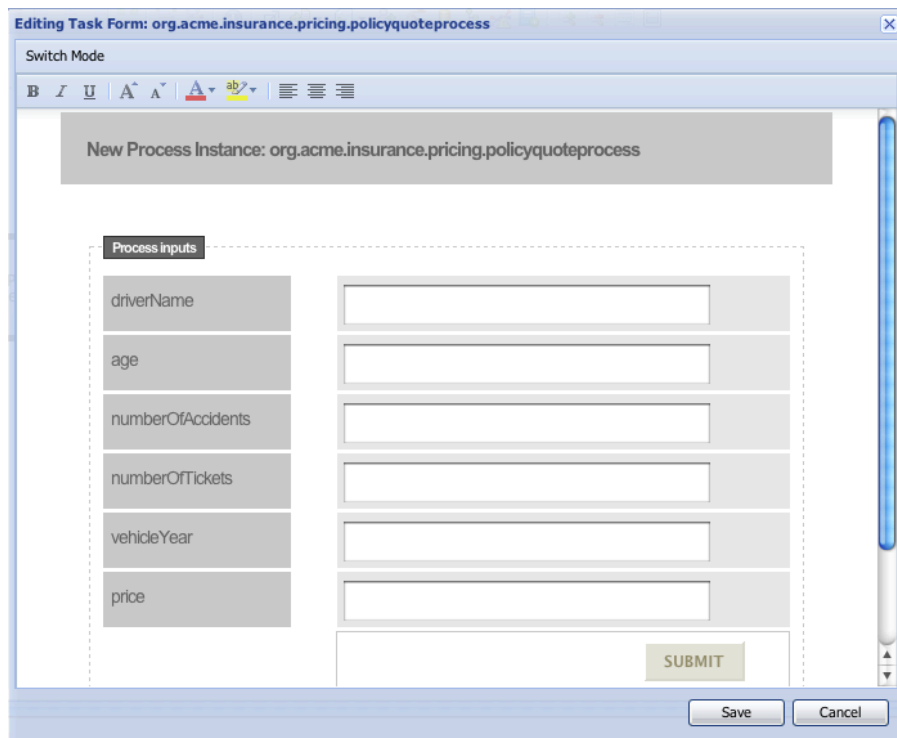
2.12.10 Examine the process Input form

1. Click on the process icon at the top of the diagram to bring up the process instance from editor. It will resemble the following:



Figure 2-40. Process Icon

2. Notice that the generated form includes all of the process variables. This is great, but we need a few adjustments. **Click** on the **Switch Mode** button near the top to get to the source editor.



| Process inputs | |
|-------------------|----------------------|
| driverName | <input type="text"/> |
| age | <input type="text"/> |
| numberOfAccidents | <input type="text"/> |
| numberOfTickets | <input type="text"/> |
| vehicleYear | <input type="text"/> |
| price | <input type="text"/> |

SUBMIT

Save Cancel

Figure 2-41. Process Instance Form Editor

3. Scroll down to the bottom of the source.

4. First, let's comment out the price field since it is not needed as an input to the process

```
<!--
  <label for="name">price</label>
  <div class="div_textbox">
    <input name="price" type="text" class="textbox" id="price" value="" />
  </div>
-->
```

5. Second, change the label names to labels that are more readable. The following is suggested:

| Old label | New label |
|-------------------|---------------------|
| driverName | Driver Name |
| age | Age |
| numberOfAccidents | Number of Accidents |
| numberOfTickets | Number of Tickets |
| vehicleYear | Vehicle Year |

Figure 2-42. Label Changes

6. Use **Switch Mode** again to review your changes
7. **Save** the updated form with the **Save** button at the bottom of the form editor

2.12.11 Check out the Task Form

The Forms generation process also created a default form for the **Review Quote** Task.

1. **Select** the **Review Quote** Task. The form editor can be launched from the second icon at the top of the short-cut menu as illustrated:

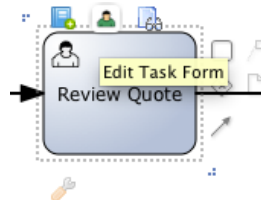


Figure 2-43. Edit Task Form

2. **Review** the form in the editor. You will notice that it has a number of features that you may not want. We will make some improvements here.

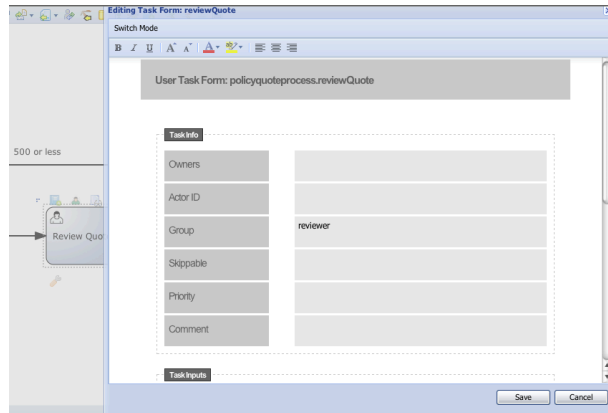


Figure 2-44. Task Form Editor

3. Click the **Switch Mode** button to view the HTML source
4. For starters, let's get the Task Info, legend block out of the way. Find the segment of code illustrated below and place comment markers around it as illustrated below:

```

<!--
<fieldset>
  <legend>Task Info</legend>
  <label for="name">Owners</label>
  <div class="div_checkbox">

  </div>
  <label for="name">Actor ID</label>
  <div class="div_checkbox"></div>
  <label for="name">Group</label>
  <div class="div_checkbox">reviewer</div>
  <label for="name">Skippable</label>
  <div class="div_checkbox"></div>
  <label for="name">Priority</label>
  <div class="div_checkbox"></div>
  <label for="name">Comment</label>
  <div class="div_checkbox"></div>
  <div class="clear"></div>
</fieldset>
-->

```

Figure 2-45. Comment out Task Info

5. The current field labels are the same names as the task variables. For each of the field labels, remove the task prefixes so the labels names are better suited to humans. Also remove the taskPolicy field:

```

<fieldset>
  <legend>Task Inputs</legend>

  <label for="name">Age</label>
  <div class="div_checkbox">
    ${taskAge}
  </div>

  <label for="name">Number Of Accidents</label>
  <div class="div_checkbox">
    ${taskNumberOfAccidents}
  </div>

  <label for="name">Number Of Tickets</label>

```



```

        <div class="div_checkbox">
            ${taskNumberOfTickets}
        </div>

        <label for="name">Vehicle Year</label>
        <div class="div_checkbox">
            ${taskVehicleYear}
        </div>
        <div class="clear"></div>
    </fieldset>

```

Figure 2-46. Fixed field labels

6. In the Task Outputs section of the form, the user needs to either review or change policy price. Change the Task Output section to match the following to present the reviewer with the existing price before prompting them to change it:

```

<legend>Task Outputs</legend>
<form action="complete" method="POST" enctype="multipart/form-data" onsubmit="return
taskFormValidator()">

    <label for="name">Price</label>
    <div class="div_textbox">
        <input name="taskPrice" type="text" class="textbox" id="taskPrice"
value="${taskPolicy.price}" />
    </div>

    <div class="button_div">
        <input name="Submit" type="submit" value="Submit" class="buttons" />

```

Figure 2-47. Enable edits to the policy price

7. Save the new form
8. Save your work: **Select File** → **Save Changes** → **Check in**

2.12.12 Disable the start process placeholder rule

In a previous lab, a rule was used to start processing the `policyquotecalculationalprocess` in a test scenario. This rule is no longer needed since our new process will provide the start event for the sub process. Use the following steps to disable the old rule;

1. With the **pricing** package selected **select** the **Assets** tab and open the **Business rule assets**
2. Find the rule named **StartRuleFlow**; **select** the **open** button on this rule
3. **Select** the **Attributes** tab
4. Under **Metadata**, **click** the **Is Disabled** checkbox
5. Save your progress: **Select File** → **Save Changes** → **Check in**

2.12.13 Build package

1. **Validate** and **Build** the `org.acme.insurance.pricing` package (Recall: **Select Knowledge Bases** → **Packages** → <your package> → **Edit** tab → **Validate configuration** → **Build package**)
2. If the package fails to build then examine the log to resolve the issue
3. Save your work: **Select File** → **Save Changes** → **Check in**

2.12.14 Test the process using the jBPM Console

4. Use the following URL to access the jbp console: <http://localhost:8080/jbpm-console>
5. Log into the JBPM console with your user name
6. On the leftmost pallet **select** **Processes** → **Process Overview**

7. Select the **policyquoteprocess** Process
8. Click the **Start** button
9. Click **Yes** on the **Start new execution** dialog
10. Add the following values to the process input variables

| Field | Value |
|---------------------|--------|
| Driver Name | George |
| Age | 22 |
| Number Of Accidents | 0 |
| Number Of Tickets | 1 |
| Vehicle Year | 2009 |

Table 2-48. Process Test Input Values

11. Click **submit**

2.12.15 View the status of the process

1. **Select** the process instance from the **Instance** table
2. **Click** the **Instance Data** button in the bottom right corner to view the input data.
3. Close the window and **click** the **Diagram** button
4. Note the triangle mark on the **Review Quote** Task of the process diagram
5. Close the process diagram window

2.12.16 Review the quote

1. In the leftmost panel select **Tasks** → **Group Tasks**
2. The **Group Tasks** table will have a row with the Task Name of **reviewQuote**
3. **Select** the row, and **click** the **Claim** button
4. As a reviewer, this particular task has now been assigned to you specifically. **Select** the **Personal Tasks** item on the leftmost panel to see it again.
5. Select the row in the **Personal Tasks** list and **click** the **View** button
6. Make a price change and **click submit**
7. You can view results of the process in your log.

2.12.17 Additional test data

With our new review quote form in place, we should be able to test several scenarios. Some suggestion are provided in the following tables:

| Field | Value |
|-------------------|--------|
| driverName | George |
| Age | 22 |
| numberOfAccidents | 0 |
| NumberOfTickets | 1 |
| vehicleYear | 2009 |

Table 2-49. Scenario – Safe Youth

| Field | Value |
|-------------------|-------|
| driverName | Sassy |
| Age | 14 |
| numberOfAccidents | 0 |
| NumberOfTickets | 0 |
| vehicleYear | 2011 |

Table 2-50. Scenario – Too young rejection

| Field | Value |
|-------------------|-------|
| driverName | Larry |
| Age | 33 |
| numberOfAccidents | 1 |
| NumberOfTickets | 0 |
| vehicleUYear | 2008 |

Table 2-51. Scenario – Risky adult

2.13 Lab 14 – Service Tasks

Goal Add a Custom Service Task to the process to perform an audit on all reviewed quotes. Use the Web Designer to add the Task and JBDS to create the task handler. Re-test it using the jBPM Console.

2.13.1 Add the icon for the task

1. In JBoss BRMS Guvnor, **select Knowledge Bases → Create New → Create a file**
2. Enter the name **AuditReview** and use **png** for the extension
3. **Click OK**
4. **Click Browse...**; Browse and upload **myIcon.png** from **labs/lab14-Service-Tasks/icon/**
5. **Select Upload**
6. Save your progress: **Select File → Save Changes → Check in**
7. Your icon asset can now be found under **Other assets, documentation**

2.13.2 Create a new Work Item Definition for the Task

1. **Select Knowledge Bases → Create New → Work Item Definition**
2. Name it **AuditReviewWID**; Set the package; **Click OK**
3. In the editor for the Work Item Definition we will make some changes. First change the name to **AuditReviewTask**
4. Change **MyFirstParam** to **policy**; change its type to **ObjectDataType**
5. Change **MySecondParam** to **driver**; change its type to **ObjectDataType**
6. Remove **MyThirdParam** (and the comma following the **driver** param declaration)
7. Change the **displayName** from **My Task** to **Audit Review Task**
8. For the icon, place your cursor between the two empty double quotes, then **select** the **Select icon to add** drop down list; **select** the **AuditReview** icon from the list.
9. Edit the URL just added as appropriate (e.g., replace **drools-guvnor** with **jboss-brms**)
10. Save your progress: **Select File → Save Changes → Check in**

2.13.3 Use the new Task in the Designer

1. Open the **policyquoteprocess** diagram again (close first, if previously open)
2. **Open** the **Shape Repository → Service Tasks** pallet; Your **Audit Review Task** is now available
3. Drag the **Audit Review Task** from the pallet to the diagram; Connect it to match the following:

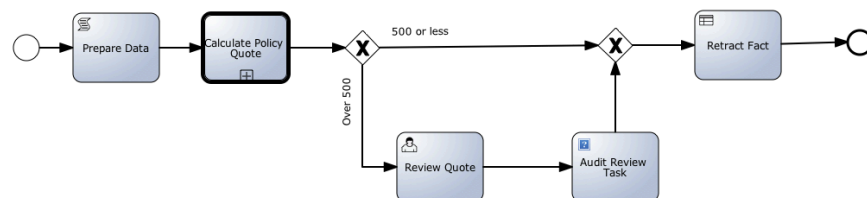


Figure 2-52. Custom Task Addition

2.13.4 Assign input data

1. With the **Audit Review Task** selected, open the properties panel and open the **DataInputSet** dialog.
2. Notice that the data input variables are already set and match the work item definition.

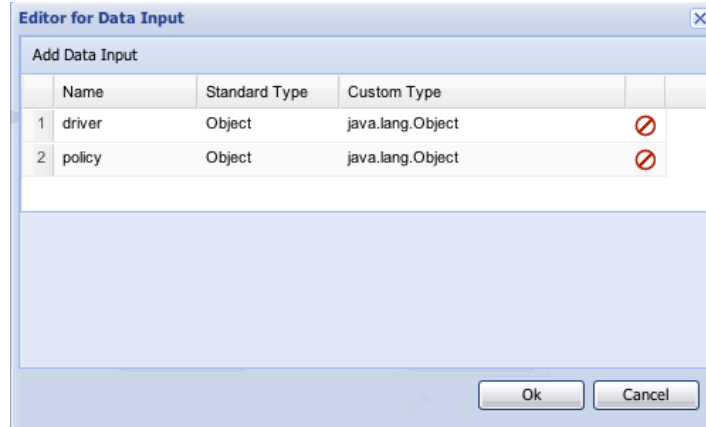


Figure 2-53. DataInputSet for the Service Task

- Open the Assignments property and assign process variables to corresponding data input variables as illustrated below:

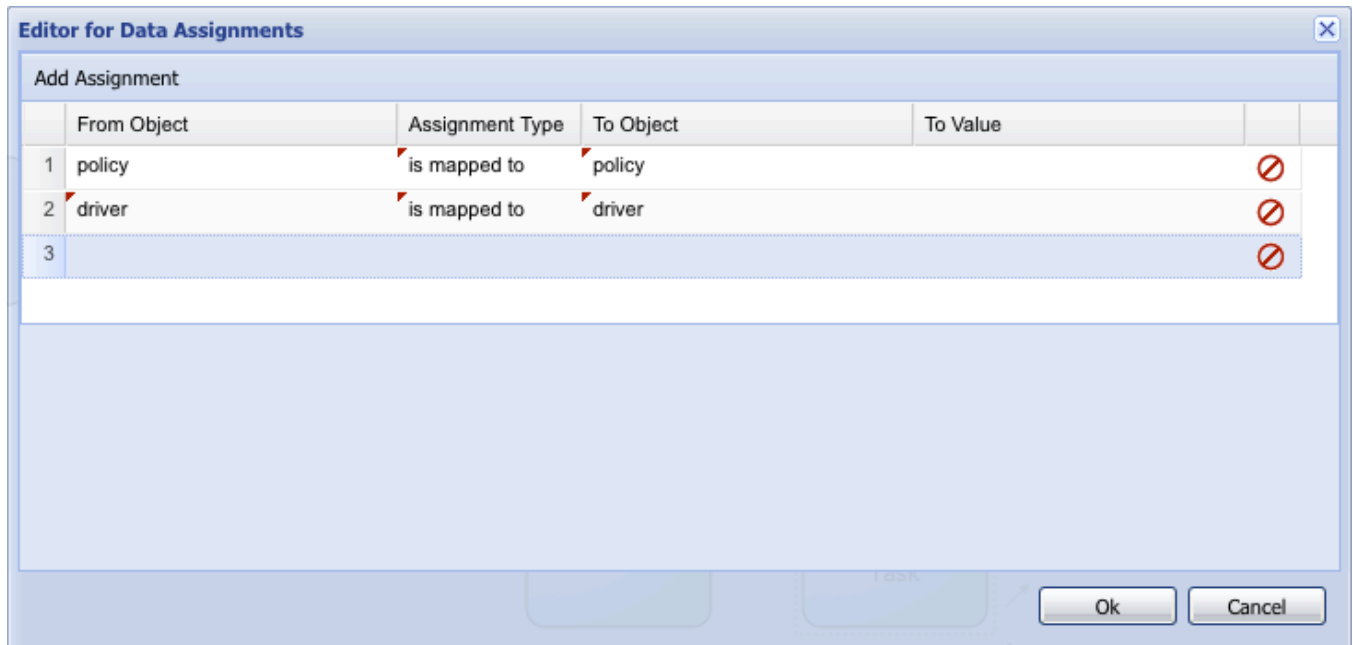


Figure 2-54. Service Task Assignments

- Use the **PNG** button to generate a new diagram image for your process.
- Save your progress: **Select File** → **Save Changes** → **Check in**
- Rebuild the package

2.13.5 Create a Work Item Handler to execute the task

- Stop your JBoss server. We will restart it later after adding a class to the runtime. **NOTE:** The server takes a long time to gracefully shut down when it is being pinged by the jBPM Console. You may find it easier to first close that browser tab.
- If you have not performed Labs 1 thru 9 then simply review the class code below and then skip ahead to 2.13.6 jBPM5 only class.
- Open the **policyquote-rules** project in JBDS. Create a new project if necessary.

4. In the **com.sample** package, create a new java class file named **AuditReviewConsole**
5. Add the following as the class content:

```
package com.sample;

import org.acme.insurance.Driver;
import org.acme.insurance.Policy;
import org.acme.insurance.Rejection;
import org.drools.runtime.process.WorkItem;
import org.drools.runtime.process.WorkItemHandler;
import org.drools.runtime.process.WorkItemManager;

public class AuditReviewConsole implements WorkItemHandler{

    @Override
    public void executeWorkItem(WorkItem workItem, WorkItemManager manager)
    {
        // extract parameters

        Policy policy = (Policy) workItem.getParameter("policy");
        Driver driver = (Driver) workItem.getParameter("driver");

        // Do some work

        System.out.println("Audit Driver: " + driver);
        System.out.println("Audit Policy: " + policy);

        // notify manager that work item has been completed
        manager.completeWorkItem(workItem.getId(), null);

    }

    @Override
    public void abortWorkItem(WorkItem workItem, WorkItemManager manager)
    {
    }

}
```

Figure 2-55. Custom Work Item Handler

6. Save the class

2.13.6 jBPM5 only class

If you did not perform labs 1 thru 9, then you may not have JBDS available to build the above class. In this case follow the instructions below using the **AuditReviewConsole.class** file (i.e., corresponding com package) provided in **labs/lab14-Service-Tasks/solution/service**

2.13.7 Tell the process engine what to do with the AuditReviewTask TaskType

1. Under the default settings, JBDS compiles the class and places the resulting **AuditReviewConsole.class** file in the bin folder for the project. Copy the file and it's package folders to the process engine with a command similar to the following:

```
==> cp -r com brms-workshop/software/brms-standalone-5.3.0/jboss-
as/server/default/deploy/gwt-console-server.war/WEB-INF/classes/
```

2. Within the server conf folder create a META-INF folder

```
==> cd brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/conf
```

```
==> mkdir META-INF
```

3. In the META-INF folder, create a file named `drools.session.conf` with contents

```
drools.workItemHandlers = auditReviewHandler.conf
```

4. Save the file. Still in the same folder create another file named `auditReviewHandler.conf` with the following contents:

```
[  
"AuditReviewTask": new com.sample.AuditReviewConsole()  
]
```

Note: AuditReviewTask MUST match the Name of our custom task

5. Your process engine has been informed. Now start your server to pickup the new configurations

2.13.8 Re-test the process in BPM-console

Test the process and view the audit trail on the console.

APPENDICES

2.14 Appendix A: Troubleshooting – Corrective Actions

| Common Issues | Resolution |
|---|--|
| Jbpm console process list is blank or Build Errors | <ol style="list-style-type: none"> 1. Ensure that the domain jar file (insurancepolicy.jar) has been uploaded into the project. 2. Also copy the domain jar file to the <code>/gwt-console-server.war/WEB-INF/lib</code> directory. 3. |
| Jbpm console process list is blank | <ol style="list-style-type: none"> 1. Verify that both the defaultPackage and your target package have both been built. <p>The jBPM console uses a properties file to hold parameters for locating the location and port for Guvnor. The default values are usually correct for this lab.</p> <ol style="list-style-type: none"> 2. Open <code>brms-workshop/software/brms-standalone-5.3.0/jboss-as/server/default/deploy/gwt-console-server.war/WEB-INF/classes/jbpm.console.properties</code> 3. Observe the connection properties in this file and make changes if necessary. 4. After a new package build (first time build), restart the server. |
| org.drools.WorkItemHandlerNotFoundException: Could not find work item handler for AuditReviewTask | <ol style="list-style-type: none"> 1. Verify that <code>drools.session.conf</code> is in the <code>conf/META-INF</code> folder 2. Verify that the contents of <code>auditReviewHandler.conf</code> is surrounded by “[” and “]” characters |
| Web Designer malfunctions | <p>Add the Web Designer to the server</p> <ol style="list-style-type: none"> 1. With the server not running, remove the <code>designer.war</code> file from the <code>brms-workshop/software/brms-standalone-5.3/jboss-as/server/default/deploy</code> directory. 2. From your <code>brms-cd/WebDesigner</code> folder, copy the <code>designer-x.x-jboss.war</code> file to the <code>brms-workshop/software/brms-standalone-5.3/jboss-as/server/default/deploy</code> directory. 3. Modify the <code>profiles/jbpm.xml</code> in <code>designer-x.x-jboss.war</code> to: <code>subdomain="jboss-brms/org.drools.guvnor.Guvnor/oryxedito"</code> |

Figure 2-56. Troubleshooting corrective action table