

JBoss BRMS

Car Insurance Demo

(Part of the Red Hat 6 Products Demo Suite)

RH6PD team

JBoss Solution Architects

Eric D. Schabell

JBoss Technology Evangelist (Integration & BPM)

Table of Contents

1	Introduction.....	3
2	Setup and Configuration.....	3
2.1	Installation.....	3
2.2	Importing Project Repository into BRM.....	4
3	Running the Demo.....	5
3.1	The data model.....	5
	The Customer (Driver).....	5
	The Vehicle (Car).....	6
	The Policy (The Result of the Insurance Application Process).....	6
3.2	The business process	7
	Starting the business process.....	7
	Step 1 - Script - Setting default environment variables.....	8
	Step 2 – Human Task - Get Driver Information	9
	Step 3 – Script - Set environment variables for entered driver information.....	10
	Step 4 – Rule Execution - Run the rules for the driver risk calculation.....	10
	Step 5 – Human Task - Get Car Information.....	12
	Step 6 – Human Task - Set environment variables for entered car information	12
	Step 7 – Rule Execution - Run the rules for the car risk.....	13
	Step 8 – Script – Update Rules	14
	Step 9 – Rule Execution – Calculate Price.....	15
	Step 10 – Script - Set Price.....	16
	Step 11- Human Task – Review Policy.....	16
	Step 12 – Rule Execution – Retract Objects.....	17
4	The Demo Script.....	18
4.1	Web Browser Locations.....	18
4.2	Navigate to the BRMS console http://localhost/jboss-brms	18
4.3	Navigate to the JBPM-CONSOLE.....	20
4.4	Navigate to the web application	21
	Click the START BRMS Demo button.....	22
	Get Driver Form	23
	Get Car Form	24
	Review Policy Form	25
	Retract Objects from Memory.....	26

1 Introduction

This document describes the first of the Six Products Demos from JBoss Middleware. In this guide we will describe a business process for a simple car insurance quote. Essentially we will demonstrate how easy it is to build an IT platform for selling car insurance. Although a simple representation of the actual insurance industry process it will cover all aspects of Enterprise architecture required to get to a fully functioning quote engine in place.

The car insurance quote is implemented with JBoss Business Rules Management System (BRMS). This version of BRMS also includes a powerful embeddable Business Process Management system which is used to control the business process.

2 Setup and Configuration

2.1 Installation

You first need to extract the clone the project (<https://github.com/eschabell/brms-car-insurance-demo>) to a location of your choosing. Once extracted, you will have the following folder structure:

- \brms-car-insurance-demo
 - \installs – Initially empty, but will contain the BRMS platform downloads.
 - \support – Additional supporting files used by the demo.
 - \target – Will be created by running init.sh. Contains the fully configured BRMS runtime server.
 - init.sh – Script to install and configure the run time server environment.
 - Quick Start Guide.odt – This document

Next, download JBoss Enterprise BRMS Platform from the Red Hat Customer Portal (<https://access.redhat.com/jbossnetwork>).

Download BRMS Platform:

1. Under *JBoss Enterprise Platforms*, select the *BRMS Platform* product.
2. Select version *5.3.1* in the *Version* field.
3. Download *JBoss BRMS 5.3.1 Deployable for EAP 6*
(Please note that this is the deployable distribution, not the standalone one.)

Now copy `brms-p-5.3.1.GA-deployable-ee6.zip`, to the projects *installs* folder. Ensure that this file is executable by running:

```
$ chmod +x <path-to-project>/installs/brms-p-5.3.1.GA-deployable-ee6.zip
```

Download EAP 6 Platform:

1. Under *JBoss Enterprise Platforms*, select the *Application Platform* product.
2. Select version *6.0.1* in the *Version* field.

3. Download *Application Platform 6.0.1*

Now copy `jboss-eap-6.0.1.zip`, to the `brms-car-insurance-demo`'s *installs* folder. Ensure that this file is executable by running:

```
$ chmod +x <path-to-project>/installs/jboss-eap-6.0.1.GA.zip
```

Lastly, from the `brms-coolstore-demo` folder, run the *init.sh* script:

```
$ ./init.sh
```

The folder is a ready to run EAP 6 server with BRMS 5.3.1 with the following modifications have been made:

- The *admin*, *krisv*, *mary*, *john*, *erics* account is enabled (password is admin) in the *brms-roles.properties* and *brms-users.properties* file in `brms-car-insurance-demo/target/jboss-eap-6.0/jboss-as/standalone/configuration`
- The security settings are modified by copying *components.xml* to :- `brms-car-insurance-demo/target/jboss-eap-6.0/standalone/deployments`
- Adjusted server security domain to include *brms*, copied new *standalone.xml* to `brms-car-insurance-demo/target/jboss-eap-6.0/jboss-as/standalone/configuration`
- The domain model jar *Model.jar* added to the console at `brms-car-insurance-demo/target/jboss-eap-6.0/deployments/jbpm-gwt-console-server.war/WEB-INF/lib`
- Human Task users are enabled by copying a new *jbpm-human-task-war-web.xml* file to `brms-car-insurance-demo/target/jboss-eap-6.0/jboss-as/standalone/deployments/jbpm-human-task.war/WEB-INF/web.xml`
- Both *business-central-server* and *jbpm-human-task* server needs a netty dependency, added a *MANIFEST.MF* to their respective *WEB-INF/classes/META-INF* directories
- The various testing dependencies will be unzipped and copied into `brms-car-insurance-demo/support/lib/*`
- Deploy RH6PD JavaScript demo web application – *rh6pd-webJsInterface.war* into the deployment directory. *The webJsInterface.war file has been unzipped and copied into the deployments directory.*
- Added patched process designer, *designer.war*.

2.2 Importing Project Repository into BRM

In this section, you will import all of the JBoss Enterprise BRMS artifacts into the Business Rules Manager.

1. Open up your Web browser of choice and navigate to <http://localhost:8080/jboss-brms/>
2. Use the default credentials of *admin/admin*

3. Upon logging in, you will see the following prompt:

**This looks like a brand new repository.
Would you like to install a sample repository?**

Important: Please be sure to select *No thanks*.

4. Select the *Administration* section on the left hand side
5. From the *Administration* list select *Import Export*. This will open the *Import Export* window.
6. Now select *Browse...* and navigate to *brms-car-insurance-demo/support* folder and select the *repository_export.zip* file.
7. Lastly, select the *Import* button. Select *OK* to confirm that you want to import the artifacts.
8. Build the package by selecting from the left navigation *Knowledge Bases* → *Packages* → *org* → *jbpm* → *evaluation* → *carinsurance* → *Edit tab* → *Build Package* button.

3 Running the Demo

3.1 The data model

A distinct and carefully designed data model can make your business and subsequent IT systems very much easier to manage. Data modelling is often referred to as the year two problem when systems are integrated together; there is a sudden realisation that the data on every system is formatted differently.

A well-designed and managed data model, which is implemented up front can save time, money and therefore reduce the costs of running your business.

For our six products demonstration we will design a very simple data model, make links between the component parts and deploy into a central business repository.

- **Note** The RH6PD car insurance demo includes a data model, essentially a set of Java class files that represent the objects as facts. The Model must be present in the BRMS repository and the class path of the application server.

The Customer (Driver)

This is the entity which represents our customer, for the car insurance quote our customer is defined by a simple object called “Driver”.

Driver looks like the following: -

Driver

Description	Type
Name	String
Address	Object
Age	Integer
Social Security Number	String
Driving Licence Number	String
Credit Score	Integer

The Vehicle (Car)

This is the entity which represents the vehicle being insured, for our car insurance quote the vehicle is defined by a simple object called “Car”.

Car looks like the following:-

Description	Type
Make	String
Model	String
Year	Integer
Registration Number	String

The Policy (The Result of the Insurance Application Process)

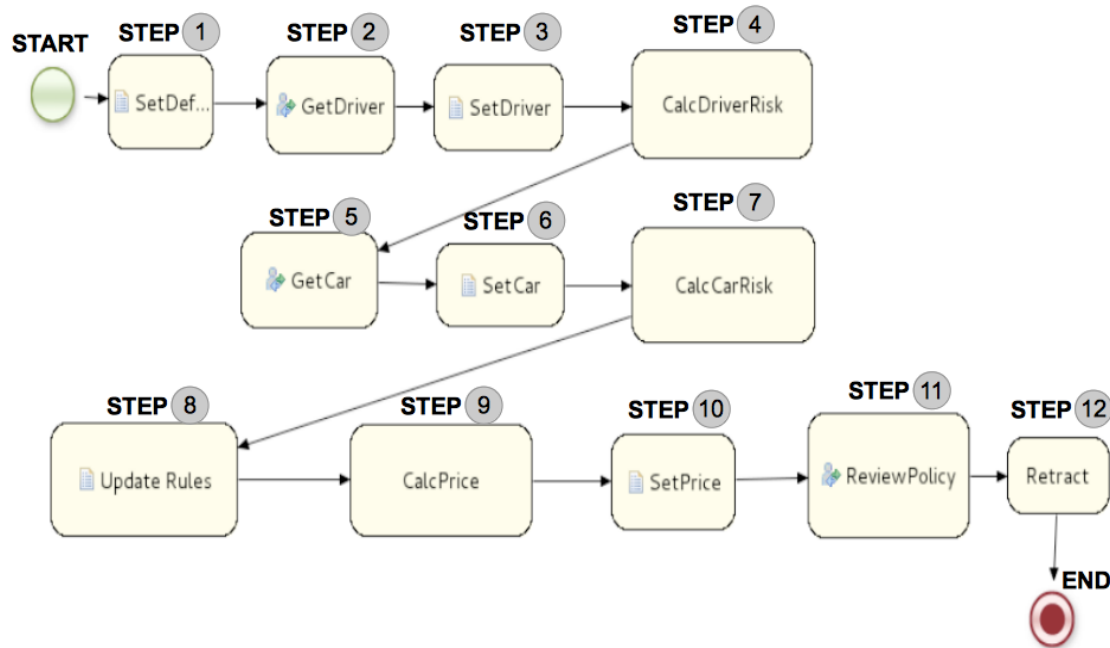
This is the entity which represents the insurance quote itself, this is defined by a simple object called “Policy”.

Policy looks like the following:-

Description	Type
Policy Number	Integer
Policy Date	Date
Named Driver	String
Address	Object Address
Car Registration Number	String
Total Risk	Integer
Driver Risk	Integer
Car Risk	Integer
History Risk	Integer
Price	Currency
Discount	

3.2 The business process

This section describes in some detail the business process and the steps involved in generating an insurance quote. If your just looking for instructions on running the demo, please jump to the next chapter “Demo Script”.



There are three sets of rules which are used to calculate the policy price, the risk of the drive, the risk of the car and finally the pricing rule used to match the driver and car to determine the final price.

There are three human tasks used in the business process. Asking for information about the driver, asking for information on the car and finally the review police human task which we use to review all the details entered and display the final policy price.

Starting the business process

You can start this car insurance business process from the JBPM-CONSOLE simply by selecting the brms-car-insurance-demo process and clicking start. You will be presented with a standard start form generated from the BPMN2.0 code. You will then need to complete the tasks from the tasks menu of the JBPM-CONSOLE.

Preferred....

Or by using the JavaScript application shipped with this demo. Go to <http://localhost:8080/rh6pd> and follow the instructions listed in the Demo Script section of this demo.

Step 1 - Script - Setting default environment variables

```
System.out.println("*** Starting Set Driver ***");

// Set variables
driverRisk="Set Driver: Not Set Yet";
driver = new Driver();
policy = new Policy();

// Set policy object with default values
policy.setDriverRisk(999);
policy.setPrice(999);

// Set driver object up with variable required for firing rules.
driver.setCreditScore(999);
driver.setName(name);
driver.setAge(Integer.parseInt(age));

//Insert into session.
kcontext.setVariable("driver",driver);
kcontext.setVariable("policy",policy);
kcontext.setVariable("name",name);
kcontext.setVariable("address1",address1);
kcontext.setVariable("address2",address2);
kcontext.setVariable("age",age);
kcontext.setVariable("driverRisk",driverRisk);

// Insert into working memory
insert(driver);
insert(policy);

System.out.println("*** Exiting Set Driver ***");</script>
</scriptTask>
<scriptTask id="_17" name="SetDefaultVariables" >
  <script>

// Set Variables
name ="SetCar: Not Set Yet";
address1="SetCar: Not Set Yet";
address2="SetCar: Not Set Yet";
age="SetCar: Not Set Yet";

make      ="SetCar: Not Set Yet";
model     ="SetCar: Not Set Yet";
reg       ="SetCar: Not Set Yet";
year      =1999;

driverRisk="SetCar: Not Set Yet";
carRisk="SetCar: Not Set Yet";

// Driver
kcontext.setVariable("name",name);
kcontext.setVariable("address1",address1);
kcontext.setVariable("address2",address2);
kcontext.setVariable("age",age);

//Car
kcontext.setVariable("make",make);
kcontext.setVariable("model",model);
kcontext.setVariable("reg",reg);
kcontext.setVariable("year",year);

// Risks
kcontext.setVariable("carRisk",carRisk);
kcontext.setVariable("driverRisk",driverRisk)
```


Step 2 – Human Task - Get Driver Information

This is the step in the process where we collecting information about the driver. This task is a human task and is delegated to the human task server which generates a form for the user to complete.

```
<userTask id="_4" name="GetDriver" >
  <ioSpecification>
    <dataInput id="_4_CommentInput" name="Comment" />
    <dataInput id="_4_SkippableInput" name="Skippable" />
    <dataInput id="_4_TaskNameInput" name="TaskName" />
    <dataInput id="_4_GroupIdInput" name="GroupId" />
    <dataInput id="_4_PriorityInput" name="Priority" />
    <dataOutput id="_4_NameOutput" name="Name" />
    <dataOutput id="_4_AgeOutput" name="Age" />
    <dataOutput id="_4_Address2Output" name="Address2" />
    <dataOutput id="_4_Address1Output" name="Address1" />
    <inputSet>
      <dataInputRefs>_4_CommentInput</dataInputRefs>
      <dataInputRefs>_4_SkippableInput</dataInputRefs>
      <dataInputRefs>_4_TaskNameInput</dataInputRefs>
      <dataInputRefs>_4_GroupIdInput</dataInputRefs>
      <dataInputRefs>_4_PriorityInput</dataInputRefs>
    </inputSet>
    <outputSet>
      <dataOutputRefs>_4_NameOutput</dataOutputRefs>
      <dataOutputRefs>_4_AgeOutput</dataOutputRefs>
      <dataOutputRefs>_4_Address2Output</dataOutputRefs>
      <dataOutputRefs>_4_Address1Output</dataOutputRefs>
    </outputSet>
  </ioSpecification>
  ....
</userTask>
```

Step 3 – Script - Set environment variables for entered driver information

This step in the process is used to create the driver and policy objects, set driver variables to default values and the insert the driver and policy setup a number of environment variables to default values.

```
// Set variables
driverRisk="Set Driver: Not Set Yet";
driver = new Driver();
policy = new Policy();

// Set policy object with default values
policy.setDriverRisk(999);
policy.setPrice(999);

// Set driver object up with variable required for firing rules.
driver.setCreditScore(999);
driver.setName(name);
driver.setAge(Integer.parseInt(age));

//Insert into session.
kcontext.setVariable("driver",driver);
kcontext.setVariable("policy",policy);
kcontext.setVariable("name",name);
kcontext.setVariable("address1",address1);
kcontext.setVariable("address2",address2);
kcontext.setVariable("age",age);
kcontext.setVariable("driverRisk",driverRisk);












// Insert into working memory
insert(driver);
insert(policy);
```

Step 4 – Rule Execution - Run the rules for the driver risk calculation.

This step of the process simply calls the rules in the calcDriverRisk decision table. This is done with the following ruleflowgroup statement.

```
<businessRuleTask id="_6" name="CalcDriverRisk" g:ruleFlowGroup="calcDriverRisk" >
```

From the BRMS user interface, expand the rules tab to see the calc Driver risk table.

	#	Description	ruleflow-group	from	to	risk
	1		calcDriverRisk	17	21	10
	2		calcDriverRisk	22	25	9
	3		calcDriverRisk	26	35	8
	4		calcDriverRisk	36	45	7
	5		calcDriverRisk	46	55	6
	6		calcDriverRisk	56	65	5
	7		calcDriverRisk	66	75	4
	8		calcDriverRisk	76	85	3
	9		calcDriverRisk	86	95	2
	10		calcDriverRisk	96	100	1

Once you have the table displayed you can see the actual rules generated by the decision table by selecting source tab then clicking View Source.

Viewing source for: calcDriverRisk

 Viewing source for: calcDriverRisk

```
1. |//from row number: 1
2. |rule "Row 1 calcDriverRisk"
3. |   ruleflow-group "calcDriverRisk"
4. |   dialect "mvel"
5. |   when
6. |       policy : Policy( this != null )
7. |       driver : Driver( age >= 17 , age <= 21 )
8. |   then
9. |       policy.setDriverRisk( 10 );
10. |end
11. |
12. |//from row number: 2
13. |rule "Row 2 calcDriverRisk"
14. |   ruleflow-group "calcDriverRisk"
15. |   dialect "mvel"
16. |   when
17. |       policy : Policy( this != null )
18. |       driver : Driver( age >= 22 , age <= 25 )
19. |   then
```

3.3

3.4

Step 5 – Human Task - Get Car Information

This is the step in the process where we collecting information about the car. This task is a human task and is delegated to the human task server which generates a form for the user to complete.

```
<userTask id="_13" name="GetCar" >
  <ioSpecification>
    <dataInput id="_13_TaskNameInput" name="TaskName" />
    <dataInput id="_13_GroupIdInput" name="GroupId" />
    <dataOutput id="_13_YearOutput" name="Year" />
    <dataOutput id="_13_ModelOutput" name="Model" />
    <dataOutput id="_13_RegOutput" name="Reg" />
    <dataOutput id="_13_MakeOutput" name="Make" />
    <inputSet>
      <dataInputRefs>_13_TaskNameInput</dataInputRefs>
      <dataInputRefs>_13_GroupIdInput</dataInputRefs>
    </inputSet>
    <outputSet>
      <dataOutputRefs>_13_YearOutput</dataOutputRefs>
      <dataOutputRefs>_13_ModelOutput</dataOutputRefs>
      <dataOutputRefs>_13_RegOutput</dataOutputRefs>
      <dataOutputRefs>_13_MakeOutput</dataOutputRefs>
    </outputSet>
  </ioSpecification>....
</userTask>
```

Step 6 – Human Task - Set environment variables for entered car information

Once the human task for get Car has completed the resulting variables must be mapped back into the environment. The car object is created, variable are copied in from the form and the policy object is updated.

```
System.out.println("create new car");
car = new Car();

// Set policy object with default values
System.out.println("set car risk to 999");
policy.setCarRisk(999);

// Set car object up with variable required for firing rules.
car.setMake(make);
car.setModel(model);
car.setReg(reg);

//Insert into session.
kcontext.setVariable("car",car);
kcontext.setVariable("make",make);
kcontext.setVariable("model",model);
kcontext.setVariable("reg",reg);
kcontext.setVariable("year",year);
kcontext.setVariable("carRisk",carRisk);
kcontext.setVariable("driverRisk",driverRisk);




// Insert into working memory
insert(car);
kcontext.getKnowledgeRuntime().update(kcontext.getKnowledgeRuntime().getFactHandle(policy), policy);
```

Step 7 – Rule Execution - Run the rules for the car risk

This step of the process simply calls the rules in the calcCarRisk decision table. This is done with the following ruleflowgroup statement.


```
<businessRuleTask id="_14" name="CalcCarRisk" g:ruleFlowGroup="calcCarRisk" >
  </businessRuleTask>
```

From the BRMS user interface, expand the rules tab to see the calc Car risk table.

	#	Description	ruleflow-group	make	risk
	1		calcCarRisk	FORD	1
	2		calcCarRisk	BMW	10
	3		calcCarRisk	HONDA	5

Once you have the table displayed you can see the actual rules generated by the decision table by selecting source tab then clicking View Source.

Viewing source for: calcCarRisk

 Viewing source for: calcCarRisk

```
1. |//from row number: 1
2. |rule "Row 1 calcCarRisk"
3. |  ruleflow-group "calcCarRisk"
4. |  dialect "mvel"
5. |  when
6. |    policy : Policy( this != null )
7. |    car : Car( make matches "FORD" )
8. |  then
9. |    policy.setCarRisk( 1 );
10. |end
11. |
12. |//from row number: 2
13. |rule "Row 2 calcCarRisk"
14. |  ruleflow-group "calcCarRisk"
15. |  dialect "mvel"
16. |  when
17. |    policy : Policy( this != null )
18. |    car : Car( make matches "BMW" )
19. |  then
```

Step 8 – Script – Update Rules

Setup the policy and update the rules engine to take into account

A feature of the BRMS rules engine is when the process is loaded for the first time all the rules defined in the car insurance package fire with the intent to calculate the price. Of course at the beginning of the process there is no data loaded into the facts. As we progress through the process we create a driver object which forces the rules to fire once again, this sets the driver risk. We then create the car object which again forces the rules to fire, this time setting the car risk.

At step 8 we have the driver and car risks set in the policy, but the rules engine doesn't have any way that the calcPrice rule now needs to be fired. So we execute the following statements at step 8 to retrieve the policy risk components and then force the policy rules to fire.

```
// Get variables
driverRiskInt  = policy.getDriverRisk();
driverRisk     = new Integer (driverRiskInt).toString();

kcontext.getVariable("policy"));

carRiskInt = policy.getCarRisk();
carRisk    = new Integer (carRiskInt).toString();

// Update working memory
kcontext.getKnowledgeRuntime().update(kcontext.getKnowledgeRuntime().getFactHandle(
policy), policy);
```

Step 9 – Rule Execution – Calculate Price

This step of the process simply calls the rules in the calcPrice decision table. This is done with the following ruleflowgroup statement.

```
<businessRuleTask id="_11" name="CalcPrice" g:ruleFlowGroup="calcPrice" >
  </businessRuleTask>
```

From the BRMS user interface, expand the rules tab to see the calc Price decision table.

	#	Description	ruleflow-group	Driver Risk	Car Risk	price
+	1		calcPrice	1	1	100
+	2		calcPrice	2	1	202
+	3		calcPrice	3	1	303
+	4		calcPrice	4	1	400
+	5		calcPrice	5	1	500
+	6		calcPrice	6	1	600
+	7		calcPrice	7	1	299
+	8		calcPrice	8	1	800
+	9		calcPrice	9	1	1400
+	10		calcPrice	10	1	2300
+	11		calcPrice	1	5	101
+	12		calcPrice	2	5	202
+	13		calcPrice	3	5	303
+	14		calcPrice	4	5	404
+	15		calcPrice	5	5	505

Once you have the table displayed you can see the actual rules generated by the decision table by selecting source tab then clicking View Source.

Viewing source for: calcPrice

 Viewing source for: calcPrice

```
57. | rule
58. |   policy.setPrice( 600 );
59. | end
60. |
61. | //from row number: 7
62. | rule "Row 7 calcPrice"
63. |   ruleflow-group "calcPrice"
64. |   dialect "mvel"
65. |   when
66. |     policy : Policy( driverRisk == 7 , carRisk == 1 )
67. |   then
68. |     policy.setPrice( 299 );
69. | end
70. |
71. | //from row number: 8
72. | rule "Row 8 calcPrice"
73. |   ruleflow-group "calcPrice"
74. |   dialect "mvel"
75. |   when
```

Step 10 – Script - Set Price

A script which sets up environment variables based on the result of calculating the price in the previous steps.

```
// set variables
price="Not Set Yet";
// get variables
price=new Integer(policy.getPrice()).toString();

driverRisk=new Integer(policy.getDriverRisk()).toString();
carRisk=new Integer(policy.getCarRisk()).toString();

// set variables.
kcontext.setVariable("name",name);
kcontext.setVariable("address1",address1);
kcontext.setVariable("address2",address2);
kcontext.setVariable("age",age);
kcontext.setVariable("driverRisk",driverRisk);
kcontext.setVariable("carRisk",carRisk);
kcontext.setVariable("price",price);
kcontext.setVariable("make",make);
kcontext.setVariable("model",model);
```

Step 11- Human Task – Review Policy

This step is used to generate a form which reviews the details of the policy. This includes driver and car information and the final calculated price.

Should should accept the quote to force the process to the final step 12. retract.

```
<userTask id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178" name="ReviewPolicy" >
  <ioSpecification>
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_NameInput" name="Name" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_AgeInput" name="Age" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_ModelInput" name="Model" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_DriverRiskInput" name="DriverRisk" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_Address2Input" name="Address2" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_Address1Input" name="Address1" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_PriceInput" name="Price" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_CarRiskInput" name="CarRisk" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_MakeInput" name="Make" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_CommentInput" name="Comment" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_SkippableInput" name="Skippable" />
    <dataInput id=" _B09AA1BF-19DA-4F57-B074-F36CB7E94178_TaskNameInput" name="TaskName" />
  </ioSpecification>
</userTask>
```



```

<dataInput id="_B09AA1BF-19DA-4F57-B074-F36CB7E94178_GroupIdInput" name="GroupId" />
<dataInput id="_B09AA1BF-19DA-4F57-B074-F36CB7E94178_PriorityInput" name="Priority" />
<inputSet>
...
</userTask>

```

Step 12 – Rule Execution – Retract Objects.

The final step in the process is labelled as retract, here we call a group three of rules,

Rule	Description
RetractDriver	Removes the car object from the working memory of the rules engine.
RetractCar	Removes the car object from the working memory of the rules engine.
RetractPolicy	Removes the policy object from the working memory of the rules engine.

4 The Demo Script

Starting the demonstration

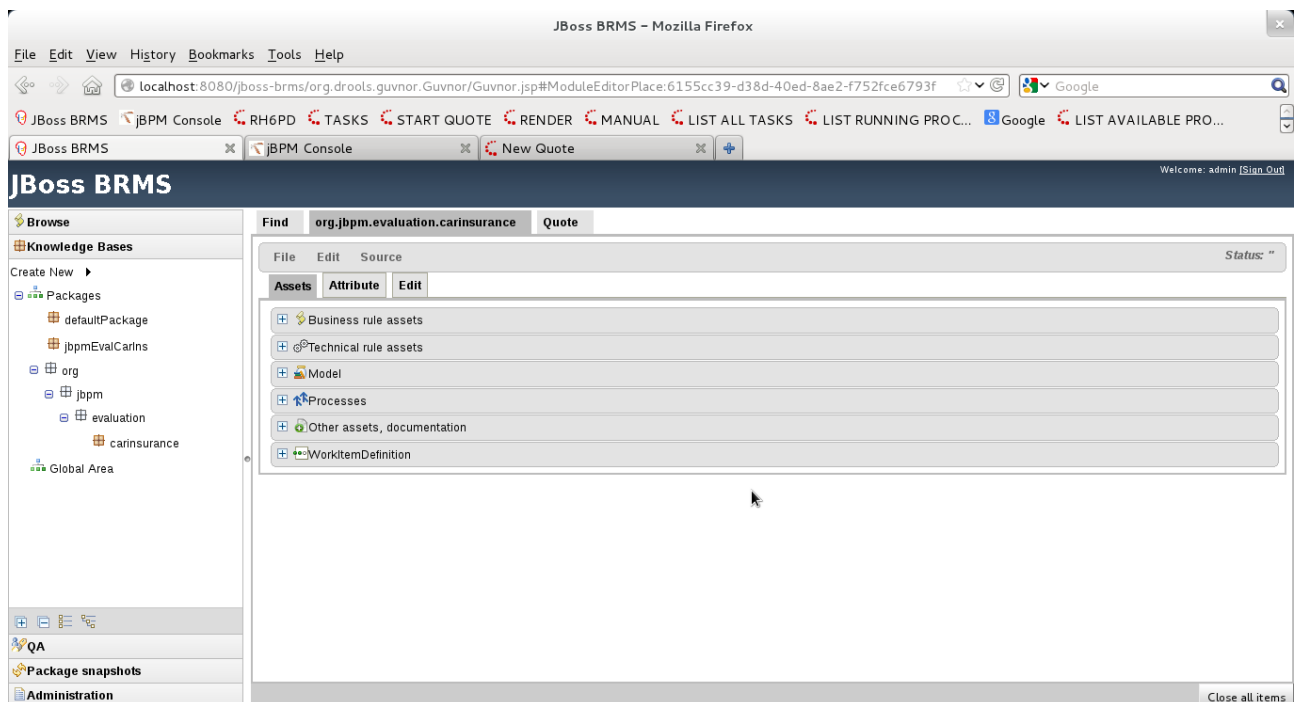
4.1 Web Browser Locations.

Connect to BRMS console	http://localhost:8080/jboss-brms	admin/admin
Connect to JBPM-console	http://localhost:8080/business-central/app.html	admin/admin
Open Home page for webapp	http://localhost:8080/rh6pd	
Start Quote Process	http://localhost:8080/rh6pd/quote.html	
Retrieve Tasks from Process	http://localhost:8080/rh6pd/tasks.html	

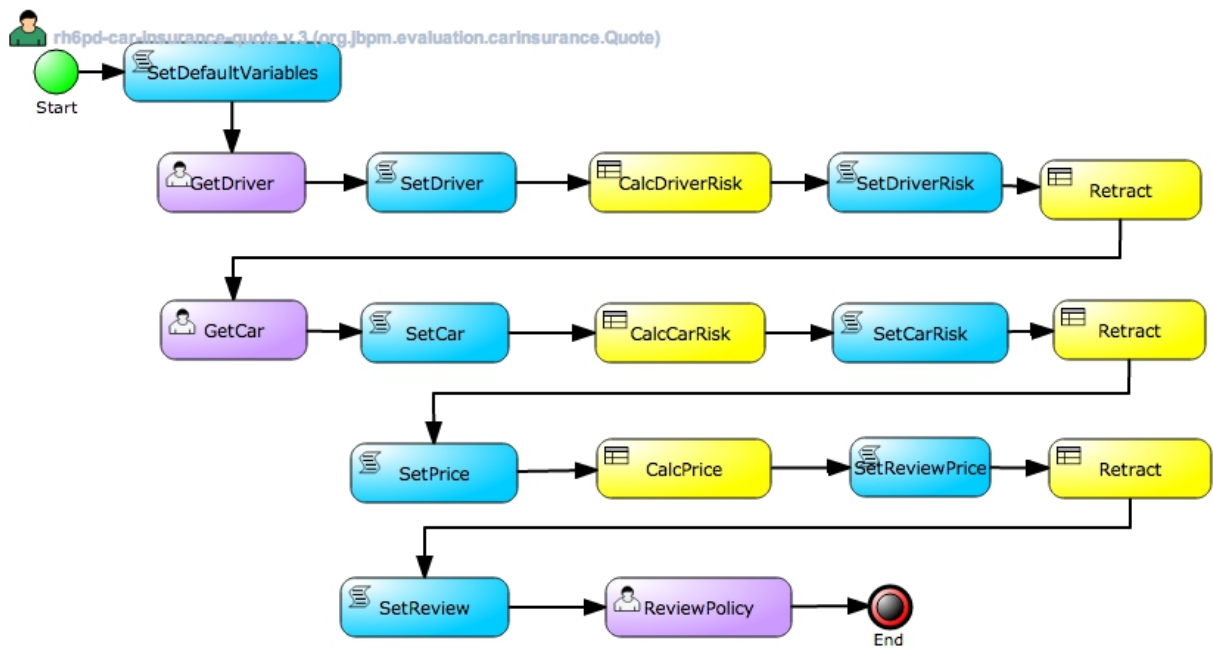
4.2 Navigate to the BRMS console <http://localhost/jboss-brms>

login as admin/admin

navigate to the car insurance package `org.evaluation.jpmp.carinsurance`



open the process show the process in the JBPM web designer



discuss the three human tasks

open rules tab

show calcDriverRisk to determine the risk of a 45 year old driver

	#	Description	ruleflow-group	from	to	risk
	1		calcDriverRisk	17	21	10
	2		calcDriverRisk	22	25	9
	3		calcDriverRisk	26	35	8
	4		calcDriverRisk	36	45	7
	5		calcDriverRisk	46	55	6
	6		calcDriverRisk	56	65	5
	7		calcDriverRisk	66	75	4
	8		calcDriverRisk	76	85	3
	9		calcDriverRisk	86	95	2
	10		calcDriverRisk	96	100	1

show calcCarRisk to determine the risk for a Ford Fiesta

	#	Description	ruleflow-group	make	risk
	1		calcCarRisk	FORD	1
	2		calcCarRisk	BMW	10
	3		calcCarRisk	HONDA	5

show calcPrice to determine the price for a for Driver risk of 7 and a car risk of 1

As you can see we are having a special deal on drivers with a risk of 7 and a car risk of 1, the price indicated shows 299. Watch out for this at the review policy step.

	#	Description	ruleflow-group	Driver Rsk	Car Risk	price
	1		calcPrice	1	1	100
	2		calcPrice	2	1	202
	3		calcPrice	3	1	303
	4		calcPrice	4	1	400
	5		calcPrice	5	1	500
	6		calcPrice	6	1	600
	7		calcPrice	7	1	299
	8		calcPrice	8	1	800
	9		calcPrice	9	1	1400
	10		calcPrice	10	1	2300
	11		calcPrice	1	5	101
	12		calcPrice	2	5	202
	13		calcPrice	3	5	303
	14		calcPrice	4	5	404
	15		calcPrice	5	5	505

4.3 Navigate to the JBPM-CONSOLE

Login as admin/admin – expand the process tab to reveal the demo process

localhost:8080/business-central/app.html#errai_ToolSet_Processes;Process_Overview.0

Baseball Mail Intranet writing Events JBoss RedHat BPMS BRMS Fuse SOA-P EDS-P J8DS RH Org Chart

Tasks

Processes

Process Overview

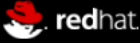
Process Overview

Refresh All Refresh Start Signal Delete Terminate

Process	Instance	State	Start D
rh6pd-car-insurance-quote	3		

4.4 Navigate to the web application

You are now at the webapp demo home page.



Red Hat Six Products Demo

Welcome to the JBoss Middleware six products demonstration. If you are new to JBoss Middleware this suite of simple demos will allow you to get up and running with six of the most popular JBoss products very quickly.

These Demos are aimed at showcasing 6 of Red Hat's JBoss Middleware products i.e

- JBoss Business Process Management (JBPM5)
- JBoss Business Rules Management Platform (BRMS)
- JBoss Enterprise Portal Platform (EPP)
- JBoss Service Orientated Architecture Platform (SOA-P)
- JBoss Operations Network (JON)
- JBoss Enterprise Data Services Platform (EDS)

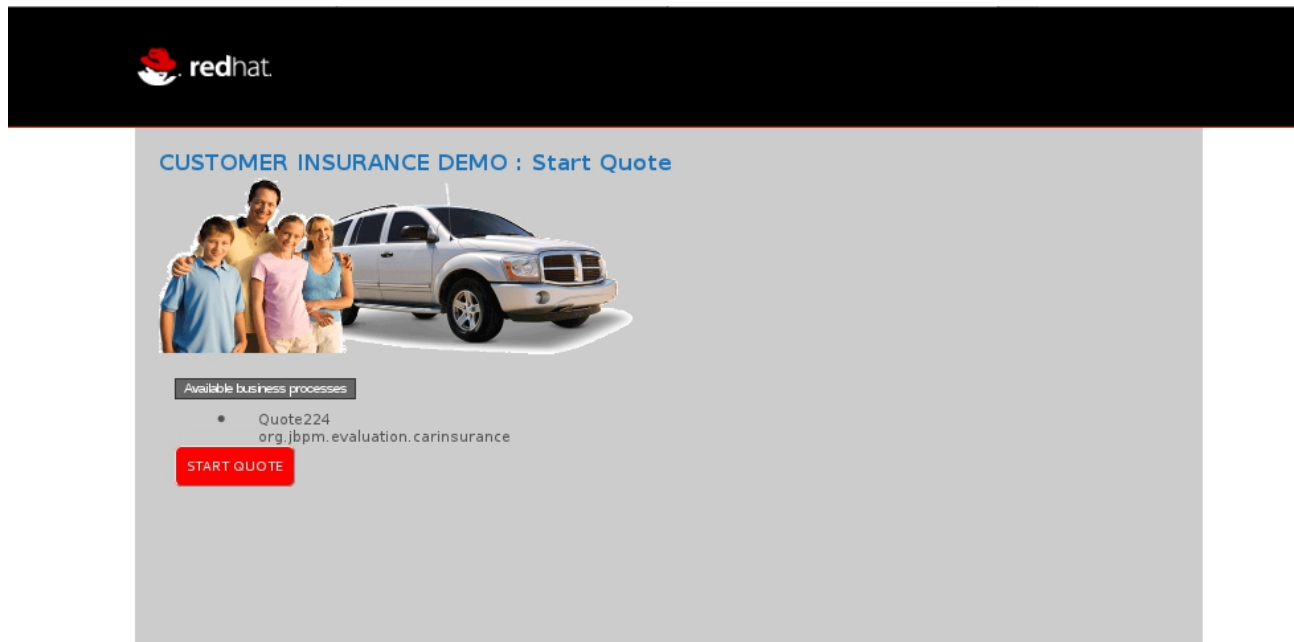
The first of one these demos. A car insurance demo using BRMS and JBPM 5 is available today

[Launch BRMS Demo »](#)

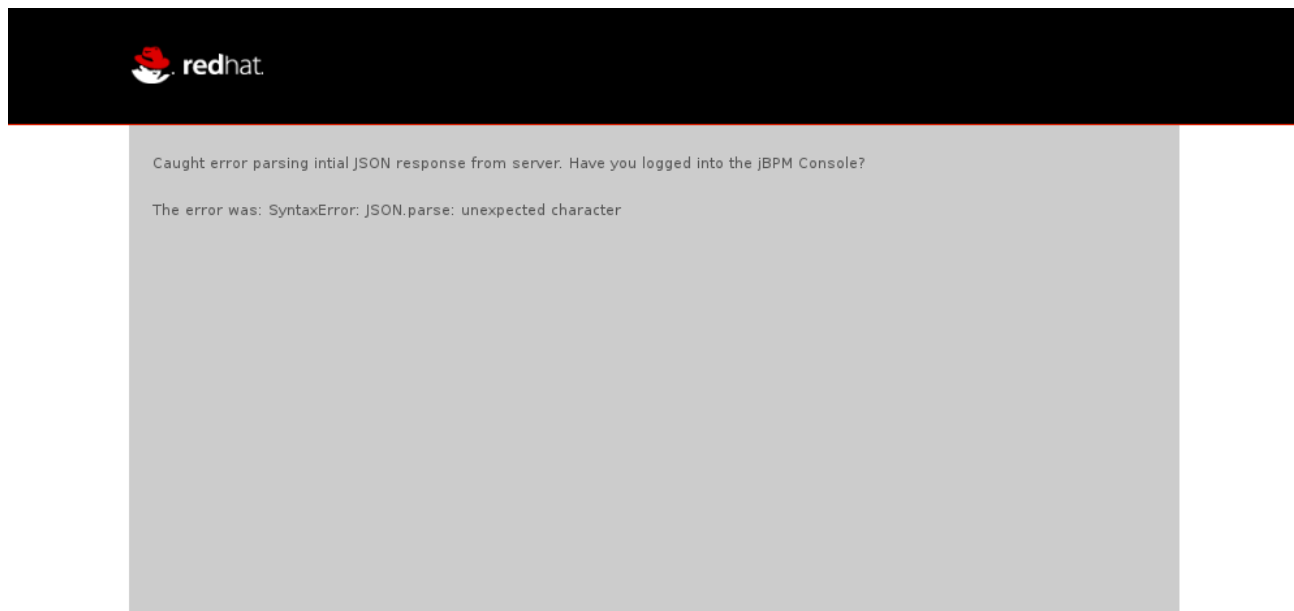
More details on each of the JBoss products can be found at the [JBoss Enterprise Middleware](#) website.

Click the **START BRMS Demo** button

You will be presented with the Quote process start screen.

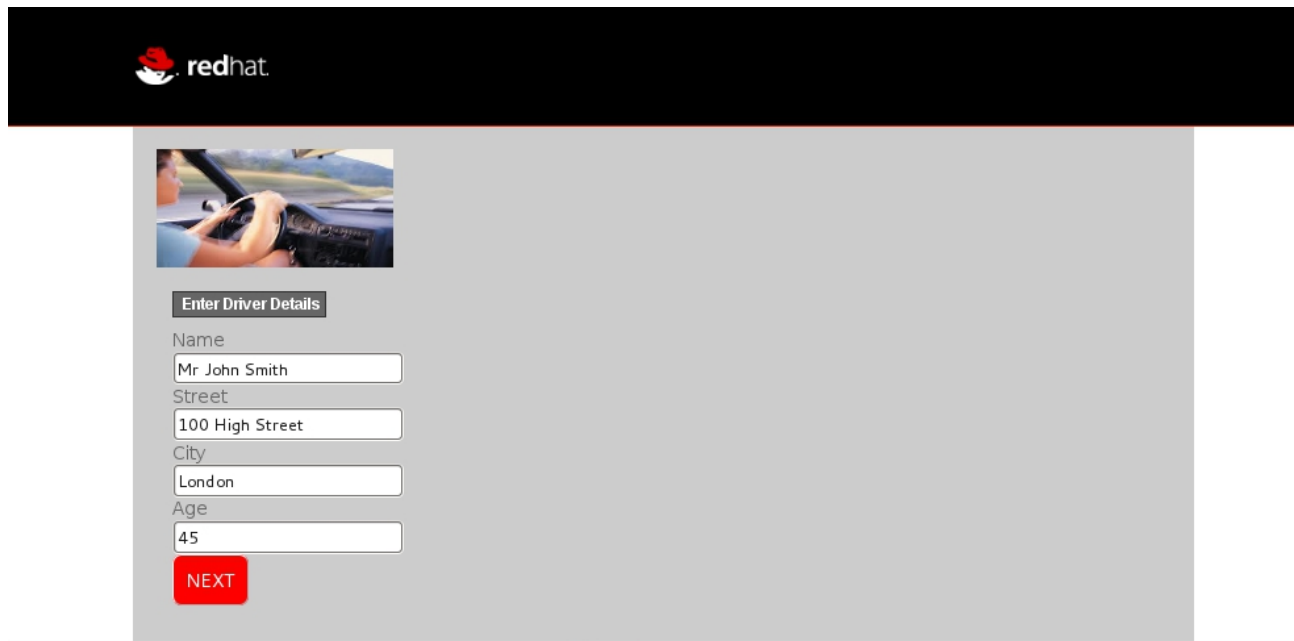


The Quote process start screen displays the current quote process available. If however you get the following message, this means you haven't authenticated with JBPM yet. Please go to the URL for the JBPM-CONSOLE and authenticate there.



Get Driver Form

Clicking start on the Start Quote form will bring up the GetDriver form.

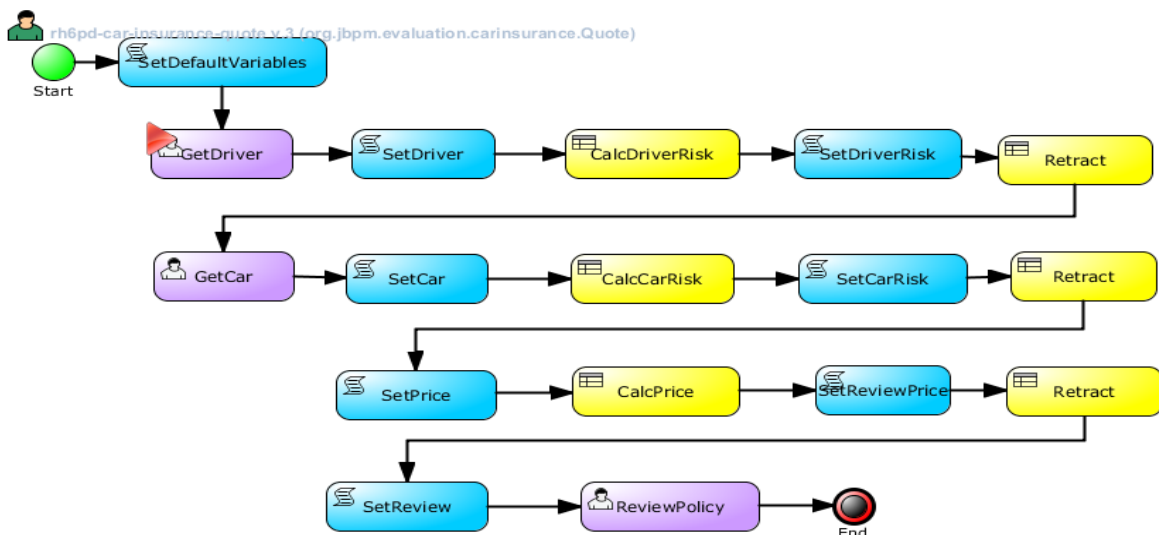


The screenshot shows a web application interface with a black header containing the Red Hat logo. Below the header, there is a small image of a person driving a car. To the right of the image is a form titled "Enter Driver Details". The form contains the following fields:

- Name: Mr John Smith
- Street: 100 High Street
- City: London
- Age: 45

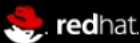
At the bottom of the form is a red button labeled "NEXT".


Once you have the Driver dialog on the screen, this is the point to switch tabs back to the JBPM-CONSOLE and this time selecting the running process and then the diagram box will bring up the process again but this time the image will have a red triangle indicating where you are in the process .



Get Car Form

Returning to the Driver form select default values have been inserted for ease of use. Currently a driver with an age with 45 is selected. Press NEXT to move to the Car form.





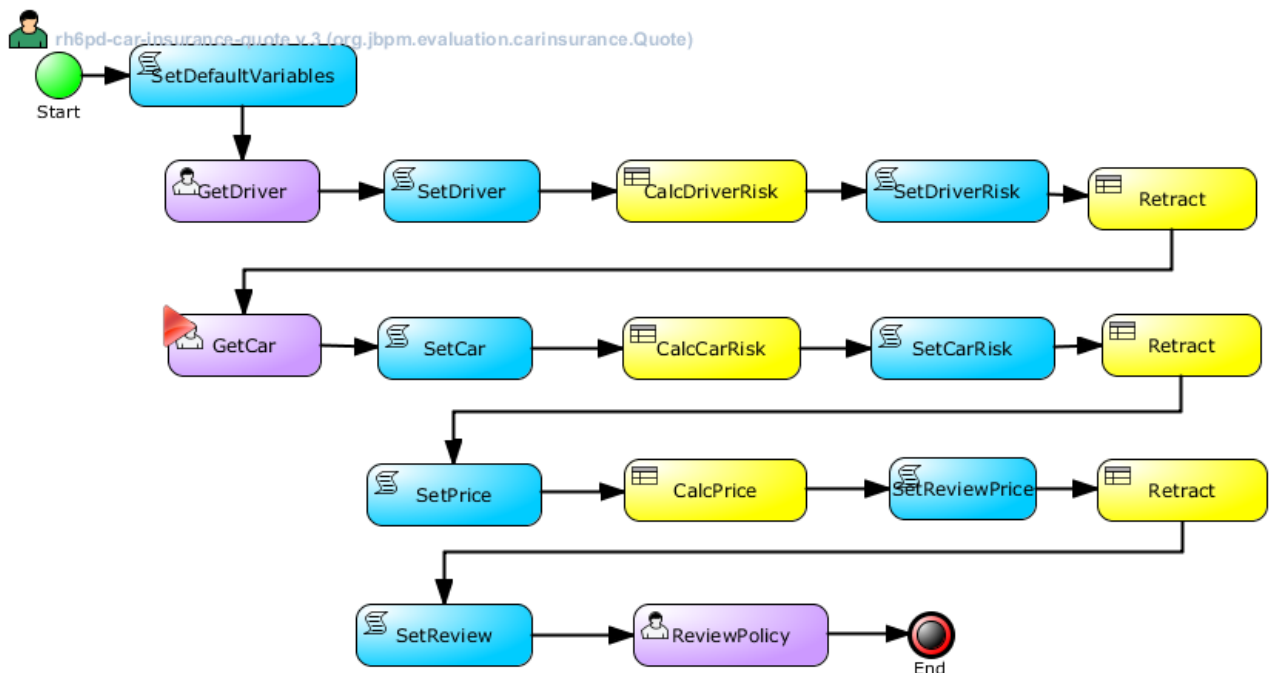
Enter Car Details

Make

Model

Registration Mark

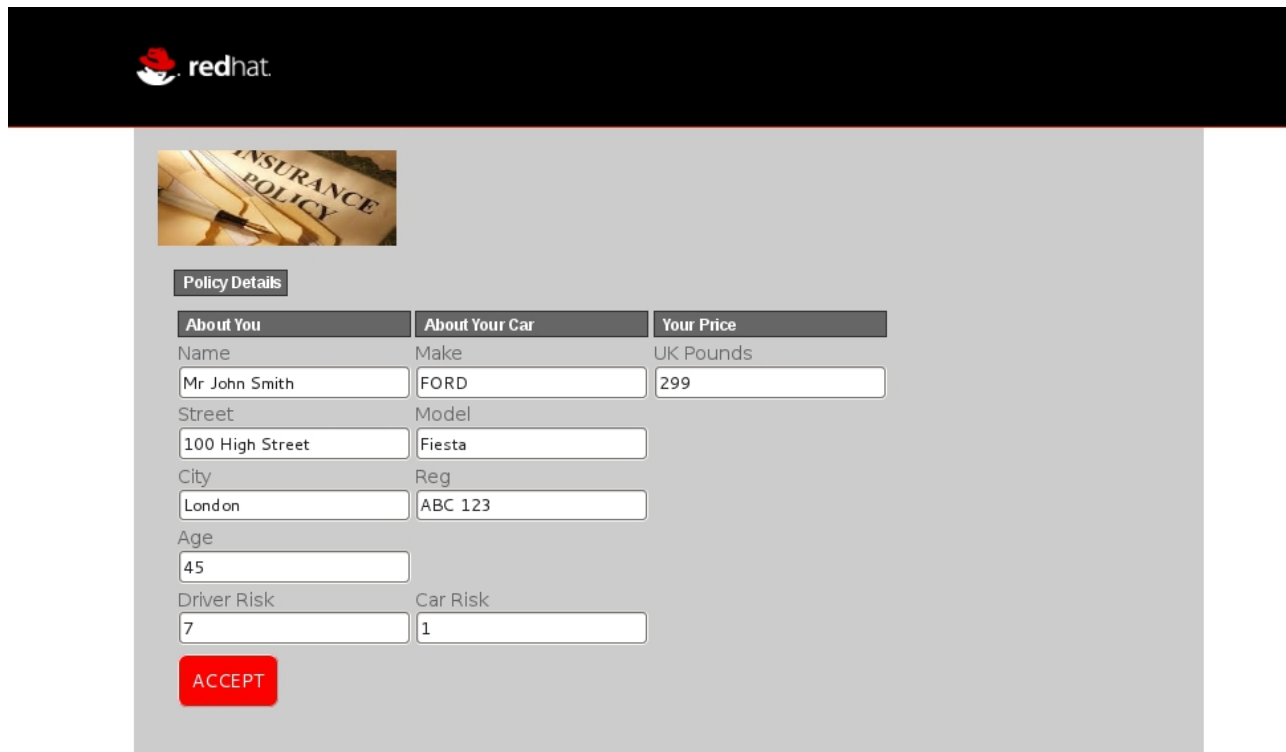
Now from the JBPM-CONSOLE you can now see the process has moved on to the getCar step.



Select a FORD Fiesta from the drop down list and click NEXT.

Review Policy Form

The Final form displays the quote. This is the review policy step in the process.

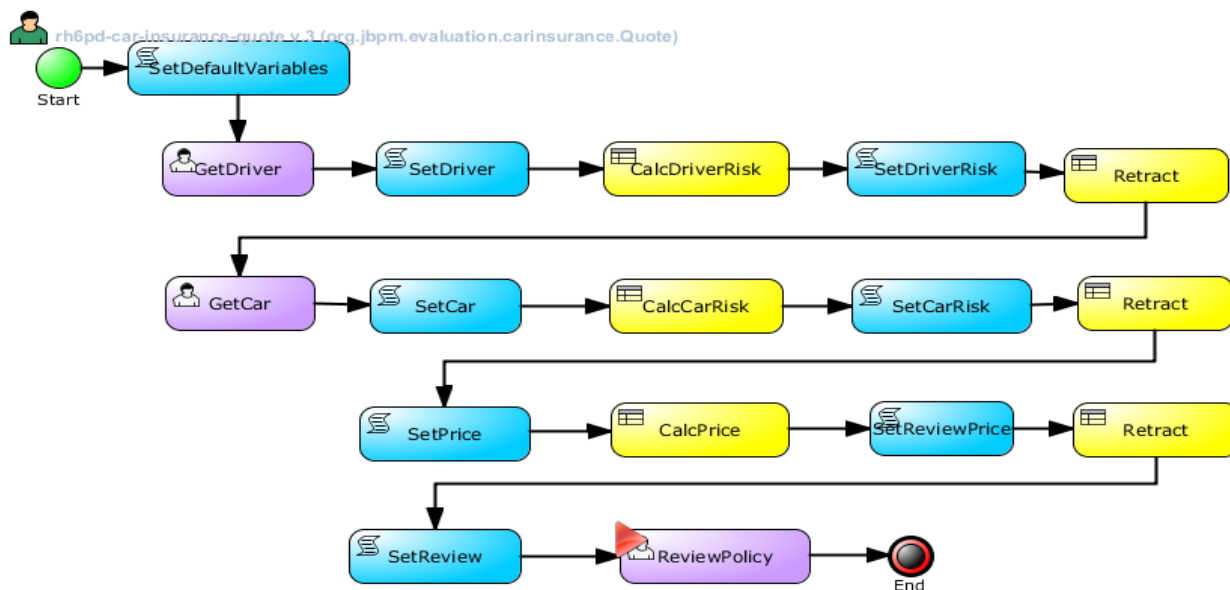


The screenshot shows a web form titled "INSURANCE POLICY" with a "Policy Details" section. The form is divided into three columns: "About You", "About Your Car", and "Your Price".

About You	About Your Car	Your Price
Name Mr John Smith	Make FORD	UK Pounds 299
Street 100 High Street	Model Fiesta	
City London	Reg ABC 123	
Age 45		
Driver Risk 7	Car Risk 1	

Below the form is a red "ACCEPT" button.

Again you can see the process has advanced to the review policy step.

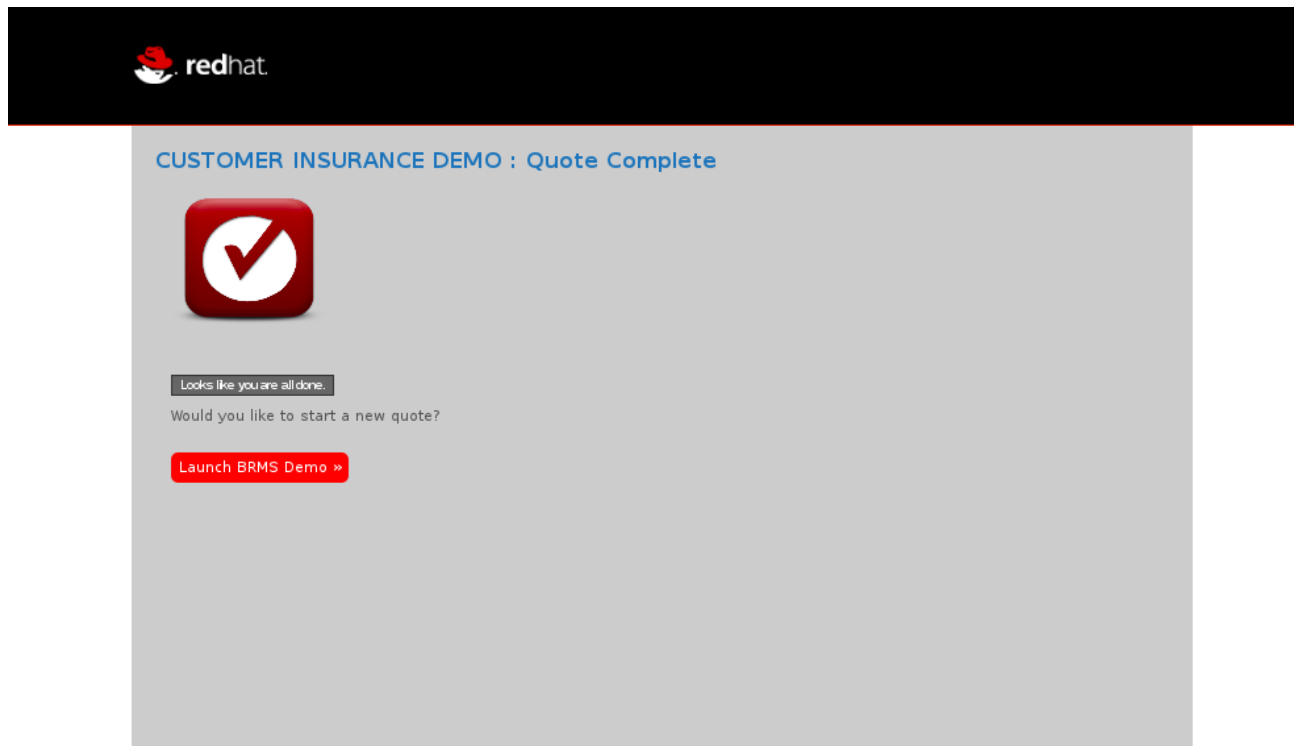


Retract Objects from Memory

From the window you started the BRMS from you should now see the log file detail the three retract statements.

```
11:33:58,524 INFO [RepositoryServlet] PackageVersion: LATEST
11:33:58,524 INFO [RepositoryServlet] PackageIsLatest: true
11:33:58,524 INFO [RepositoryServlet] PackageIsSource: true
11:34:51,146 INFO [STDOUT] Retracting Car
11:34:51,146 INFO [STDOUT] Retracting Policy
11:34:51,146 INFO [STDOUT] Retracting Driver
```

Once accepted the review policy will display the complete form.



Click new_quote to step through the demo again, this time experiment with different driver ages and different car types to see the range of prices. You can edit the prices from the BRMS rules section, in the calcPrice decision table.

You can review the log output to see that all phases of the process have been walked through, it is verbose enough to discuss this if need be.

```
20:29:36,542 INFO [stdout] (Thread-98) Entering SetDriver Node
20:29:36,543 INFO [stdout] (Thread-98) *** Set Driver : starting ***
20:29:36,543 INFO [stdout] (Thread-98) *** Set Driver : insert driver
20:29:36,543 INFO [stdout] (Thread-98) *** Set Driver : insert policy
20:29:36,543 INFO [stdout] (Thread-98) *** Set Driver : name is Mr John Smith
20:29:36,543 INFO [stdout] (Thread-98) *** Set Driver : age is 45
20:29:36,557 INFO [stdout] (Thread-98) *** Set Driver : policy ID is 1 and its price is : 888
```

20:29:36,562 INFO [stdout] (Thread-98) *** Set Driver : exiting ***

20:29:36,562 INFO [stdout] (Thread-98) Leaving SetDriver Node

20:29:36,614 INFO [stdout] (Thread-98) Entering SetDriverRisk Node

20:29:36,614 INFO [stdout] (Thread-98) *** Set Driver Risk : starting ***

20:29:36,614 INFO [stdout] (Thread-98) *** Set Driver Risk : get driver risk from policy is : 7

20:29:36,616 INFO [stdout] (Thread-98) *** Set Driver Risk : set driver risk into environment

20:29:36,616 INFO [stdout] (Thread-98) *** Set Driver Risk : exiting ***

20:29:36,616 INFO [stdout] (Thread-98) Leaving SetDriverRisk Node

20:29:36,621 INFO [stdout] (Thread-98) Retracting Policy

20:29:36,624 INFO [stdout] (Thread-98) Retracting Driver

20:30:36,158 INFO [stdout] (Thread-98) Entering SetCar Node

20:30:36,158 INFO [stdout] (Thread-98) *** Set Car: starting

20:30:36,158 INFO [stdout] (Thread-98) *** Set Car : initial policy ID is 2

20:30:36,159 INFO [stdout] (Thread-98) *** Set Car: set car make to FORD

20:30:36,159 INFO [stdout] (Thread-98) *** Set Car: set car model to Fiesta

20:30:36,159 INFO [stdout] (Thread-98) *** Set Car: set car reg to ABC 123

20:30:36,161 INFO [stdout] (Thread-98) *** Set Car: insert car

20:30:36,163 INFO [stdout] (Thread-98) *** Set Car: policy ID is 2

20:30:36,163 INFO [stdout] (Thread-98) *** Set Car: policy: driver risk is 0

20:30:36,163 INFO [stdout] (Thread-98) *** Set Car: policy: discount is 10

20:30:36,163 INFO [stdout] (Thread-98) *** Set Car: exiting

20:30:36,163 INFO [stdout] (Thread-98) Leaving SetCar Node

20:30:36,169 INFO [stdout] (Thread-98) Entering SetCarRisk Node

20:30:36,169 INFO [stdout] (Thread-98) *** Set Car Risk : starting ***

20:30:36,169 INFO [stdout] (Thread-98) *** Set Car Risk : get car risk from policy is 1

20:30:36,171 INFO [stdout] (Thread-98) *** Set Car Risk : set car risk into environment

20:30:36,171 INFO [stdout] (Thread-98) *** Set Car Risk : exiting ***

20:30:36,171 INFO [stdout] (Thread-98) Leaving SetCarRisk Node

20:30:36,173 INFO [stdout] (Thread-98) Retracting Policy

20:30:36,174 INFO [stdout] (Thread-98) Retracting Car

20:30:36,178 INFO [stdout] (Thread-98) Entering SetPrice Node

20:30:36,178 INFO [stdout] (Thread-98) *** Set Price: starting

20:30:36,178 INFO [stdout] (Thread-98) *** Set Price: VAR car risk is : 1

20:30:36,178 INFO [stdout] (Thread-98) *** Set Price: getVAR car risk is : 1

20:30:36,179 INFO [stdout] (Thread-98) *** Set Price: insert policy

20:30:36,180 INFO [stdout] (Thread-98) *** Set Price : initial policy ID is 1

20:30:36,180 INFO [stdout] (Thread-98) *** Set Price: policy ID is 1

20:30:36,180 INFO [stdout] (Thread-98) *** Set Price: policy: driver risk is 7

20:30:36,181 INFO [stdout] (Thread-98) *** Set Price: policy: discount is 0

20:30:36,181 INFO [stdout] (Thread-98) *** Set Price: exiting

20:30:36,181 INFO [stdout] (Thread-98) Leaving SetPrice Node

20:30:36,186 INFO [stdout] (Thread-98) Entering SetReviewPrice Node

20:30:36,186 INFO [stdout] (Thread-98) *** Set Review Policy : starting ***

20:30:36,186 INFO [stdout] (Thread-98) *** Set Review Policy : get price from policy ***

20:30:36,187 INFO [stdout] (Thread-98) *** Set Driver Risk : set driver risk into environment ***
20:30:36,188 INFO [stdout] (Thread-98) *** Set Review Policy : exiting ***
20:30:36,188 INFO [stdout] (Thread-98) Leaving SetReviewPrice Node

20:30:36,191 INFO [stdout] (Thread-98) Retracting Policy

20:30:36,194 INFO [stdout] (Thread-98) Entering SetReview Node
20:30:36,195 INFO [stdout] (Thread-98) *** Set Review: starting
20:30:36,195 INFO [stdout] (Thread-98) *** Set Review: setting price from policy to 299
20:30:36,195 INFO [stdout] (Thread-98) *** Set Review: setting driver risk from policy to 7
20:30:36,195 INFO [stdout] (Thread-98) *** Set Review: setting car risk from policy to 1
20:30:36,195 INFO [stdout] (Thread-98) *** Set Review: exiting
20:30:36,195 INFO [stdout] (Thread-98) Leaving SetReview Node
