SOA Development Workshop
Lab Instructions

Table of Contents

Lab1 Installation

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

1) Create a folder soa-workshop. Copy soa-cd from CD / flash drive to a soa-workshop. Create a folder in soa-workshop named software.

Install SOA-P with Riftsaw

2) Install SOA-P 5.0. Open soa-cd and unzip soa-5.0.0.GA.zip into soa-workshop/software. Open soa-workshop/software/jboss-soa-p.5.0.0/jboss-as/server/default/conf/props/soa-users.properties and uncomment #admin=admin.

3) Upgrade JBossWS. Unzip jbossws-native-bin-dist.zip
Copy ant.properties.examples to ant.properties. Modify the target container location for jboss510.home in ant.properties. Run ant deploy-jboss510.

4) Deploy Riftsaw. Unzip riftsaw-2.0-0.zip. Copy, re-name, and edit deployment.properties. Run ant deploy (defaults to HSQL, otherwise -Ddatabase="db-name").

5) Start server. Open command prompt and cd to jboss-soa-p-5.0.0.GA/jboss-as/bin and type ./run.sh -c default. Point browser to http://localhost:8080/ and then http://localhost:8080/bpel-console.

Install Eclipse / JBossTools

6) Inside software create a folder named named workspace.

7) Install Eclipse 3.5 SR1 (for Windows, Linux, or Mac) into software (note - Mac must be 32-bit Cocoa). Start Eclipse. Select soa-workshop/software/workspace for the workspace.

8) Use Eclipse Preferences to add Java 1.6 JRE (if not there by default) as the default JRE.

9) Add JBossTools - Add JBoss Tools 3.1 from the archive site provided in soa-cd. Re-start Eclipse.

10) Use Eclipse Preferences to add create a new server of type JBoss Enterprise Application Platform 5.0.

11) Open the Servers view: Window -> Open View -> Other -> Servers. In the Server View, right click New Server, and select JBoss Enterprise Middleware, JBoss Enterprise Application Platform 5.0. Name is SOA-P Server.

12) In the Servers view (lower left), double click on SOA-P server to open its editor. Click on Open launch configuration and check that it is using the default configuration. Also check that the deployment is to server/default/deploy, and check the deploy compressed archives.

13) Check the the BPEL Editor has been added by using File -> New Other -> BPEL. If you don't see a "BPEL 2.0" category in the new wizard, then exit Eclipse and restart it.

Install Savara Tools

14) Unzip savara-tools-eclipse-1.0-xxx into software. Use Update site (Help -> Software Updates -> Add Site and use the local site contained in savara-tools-eclipse-1.0-M1/eclipse.

15) Restart JBDS. Open the Choreography perspective. Try to create a new Choreography model using the File menu. If this is not available, exit Eclipse and restart it.

Install Repository

16) Copy drools-guvnor.war into server/default/deploy. Start server (if not already started), logon to Guvnor, don't install samples.

17) Use the Import / Export capability to import service_repository.xml.

Lab2 Scenarios

Goal: Create a a scenario diagram for the successful policy quote.

Business Requirements:

The business process as expressed by the business analyst is as follows:

The Driver initiates the process by sending in a request for a policy quote to the InsuranceAgency's new PolicyQuoteService. This request contains the type of policy (AUTO) and year of the vehicle along with the driver's name, social security number (SSN), their age, and their drivers license number (DLNumber). The InsuranceAgency uses the ssn and dlNumber to retrieve the driver's number of tickets and accidents from the DMV (via an existing DrivingRecordService). If the number of tickets is greater than 4, then the InsuranceAgency sends the Driver a response stating they are not qualified. If the number of tickets is less than or equal to 4, then the InsuranceAgency sends a request to the CreditAgency's CreditCheckService to obtain a credit score. The InsuranceAgency then calculates a quote based on the number of tickets, age of the driver, credit score, and number of accidents, persists the quote for possible future reference, and responds back to the Driver with the quote.

Note that the DMV's DrivingRecordService and the CreditAgency's CreditCheckService already exist with published WSDLs. Also note that the DrivingRecordService getDrivingRecord operation is one way. The DMV requires the client of this service host a callback service to send the DrivingRecordResponse.

From this description we can deduce a number of tasks that need to be performed as part of a new PolicyQuote process. These tasks include:

- receivePolicyQuote
- sendDrivingRecordRequest
- receiveDrivingRecordResponse
- sendCreditCheckRequest
- calculatePolicyQuote
- createPolicyQuote

Also note that the sample-data folder contains messages that conform to specific XSD's. While we could start with simple message examples, and then refine the examples once we have completed information modeling, and then go back and modify the choreography to use the new messages, we will eliminate the need for this iteration in order to save time.

Choreography modeling is an iterative process. We know there are some external services we need to interact with, but have not yet identified the service boundaries for our calculatePolicyQuote and createPolicyQuote tasks. So we will ignore these last two tasks for the initial Choreography model.

Create Project

1) Create a new project in Eclipse (File -> New -> General -> Project). Name the project it policyquote-models. Copy in the samples-data folder from labs. They contain sample messages for PolicyQuoteRequest, PolicyQuoteResponse, PolicyQuoteInelligible, DrivingRecordRequest, DrivingRecordResponse, CreditCheckRequest, and CreditCheckResponse, as well as a few other samples we will use later.

Create Roles, Relationships, and Base Types

2) Create a new Choreography in the policyquote-models project. (File -> New -> Other -> Choreography -> Choreography Description). Name it PolicyQuote.cdm. Provide Author and Description.

3) In the Roles and Relationships editor, use the Role icon to add five roles: Driver, PolicyQuoteProcessService, DrivingRecordService, CreditCheckService, and DrivingRecordCallbackService. Name the associated Behaviors: Driver, PolicyQuoteProvider, DrivingRecordProvider, CreditScoreProvider, and DrivingRecordReceipt.

4) Use the Relationship tool to draw relationships from Driver to PolicyQuoteProcessService (name it DriverPolicyQuote), PolicyQuoteProcessService to DrivingRecordService (PolicyQuoteDrivingRecord), PolicyQuoteProcessService to CreditCheckService(PolicyQuoteCreditScore), and DrivingRecordService to DrivingRecordCallbackService (DrivingRecordDrivingRecordCallback).

5) Rename the ChannelTypes in the BaseTypes view to remove the Service and Type at the end of the name. E.g., PolicyQuoteProcessChannel instead of PolicyQuoteProcessServiceChannelType. This is just to make the name shorter.

6) At this stage, there should be errors associated with the choreography: "Reference 'Reference Token' must be specified". Double-click one of the errors, this will navigate to the ChannelType that has reported the validation problem. Select the Properties view to see the property that has the error. This problem can be resolved by creating a token defining the type of the endpoint reference. By pressing the New, next to the Reference Token property, it will create a default token (e.g. named Token0).

7) Next navigate to the Base Types tab, and expand the nodes under the Tokens node. Currently only a single child node will be defined, associated with the newly created channel type's reference token. When the new token has been selected, its properties will be displayed in the Properties view. First set the name property to "URIToken", which will result in the description field also being filled in. Second, press the new button to create an InformationType entity for the token. This will be named after the token: URITokenType.

8) The final step in resolving this issue is to define the XSD element or type associated with the newly created information type. Expand the Information Types node, and selecting the URITokenType and specifying the element or type property, such as: xsd:anyURI. Once the URIToken has been fully configured, then simply double-click on each of the other errors in turn, to set the Reference Token

property to URIToken by selecting it from the combo box.

9) Next in the Base Types editor examine the Participant Types. Delete the DrivingRecordCallbackService ParticipantType, and add the DrivingRecordCallbackService under the PolicyQuoteProcessService Role Types

10) Define the Information Types for each type of message: PolicyQuote, PolicyQuoteReply, PolicyQuoteFault, DrivingRecordRequest, DrivingRecordResponse, CreditCheckRequest, and CreditCheckResponse. Do this by dragging the Information Type entry on the palette onto the Information Types node, and renaming it. Specify the Element property using the root element from the corresponding sample messages (e.g., creditCheckRequest from sample-data/CreditCheckRequest.xml). This step is important!

11) Next, since the messages in sample-data folder specify the use of namespaces, use the Base Type editor to add Namespaces for pol, driv, and cred. Use the references in the message to supply the URI and SchemaLocation.

Create Scenario

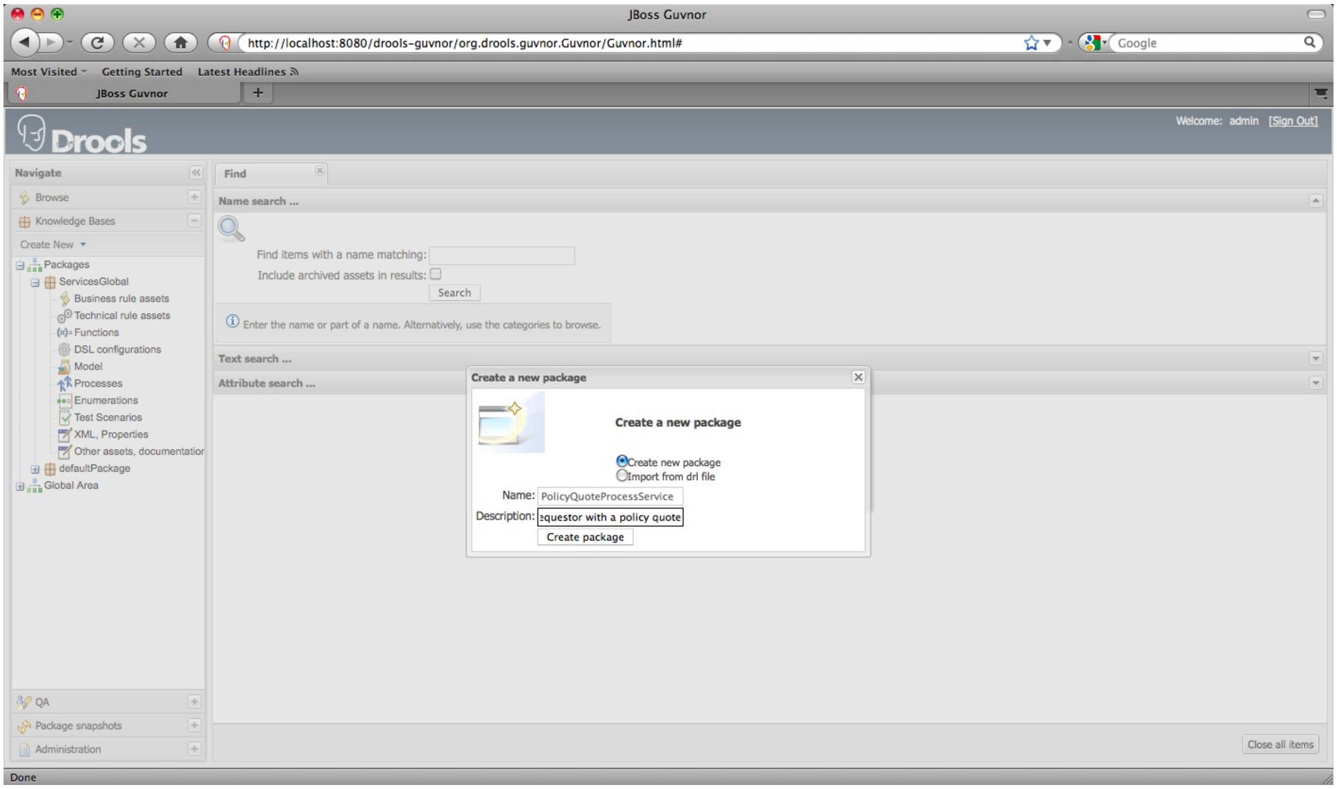
12) Next create a new scenarios (File -> New -> Other -> Choreography -> Scenario). Name it SuccessfulPolicyQuote.scn. In the properties editor, also give it the name "PolicyQuoteSuccess" and Description of "Scenario of a successful policy quote". Under the general properties Choreography Description URL enter the value PolicyQuote.cdm.

13) Use the Participant icon to add four Participants from left to right. Use the Type property drop down and select the appropriate participant: Driver, PolicyQuoteService, DrivingRecordService, CreditCheckService.

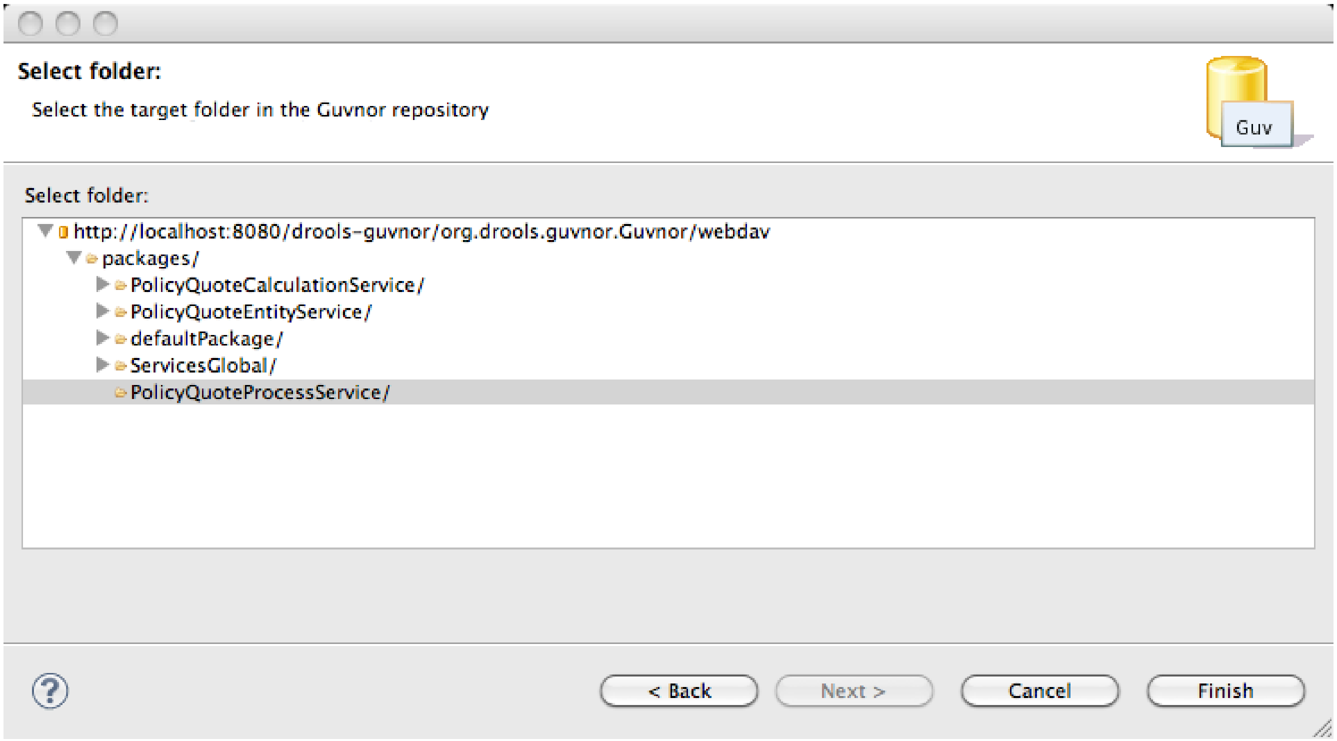
14) Use the MessageLink icon to link Driver to PolicyQuoteProcessService. Highlight the MessageLink line, and select the Message property (policyQuote) from the drop down. Add the Value URL of sample-data/PolicyQuoteRequest.xml. Repeat this step for PolicyQuoteProcessService to DrivingRecordService: (drivingRecordRequest), DrivingRecordService back to PolicyQuoteProcessService: (drivingRecordResponse), PolicyQuoteProcessService to CreditCheckService: (checkCreditRequest), CreditCheckService back to PolicyQuoteProcessService: (creditCheckResponse), and finally PolicyQuoteProcessService back to Driver: (policyQuoteReply).

Add to Repository

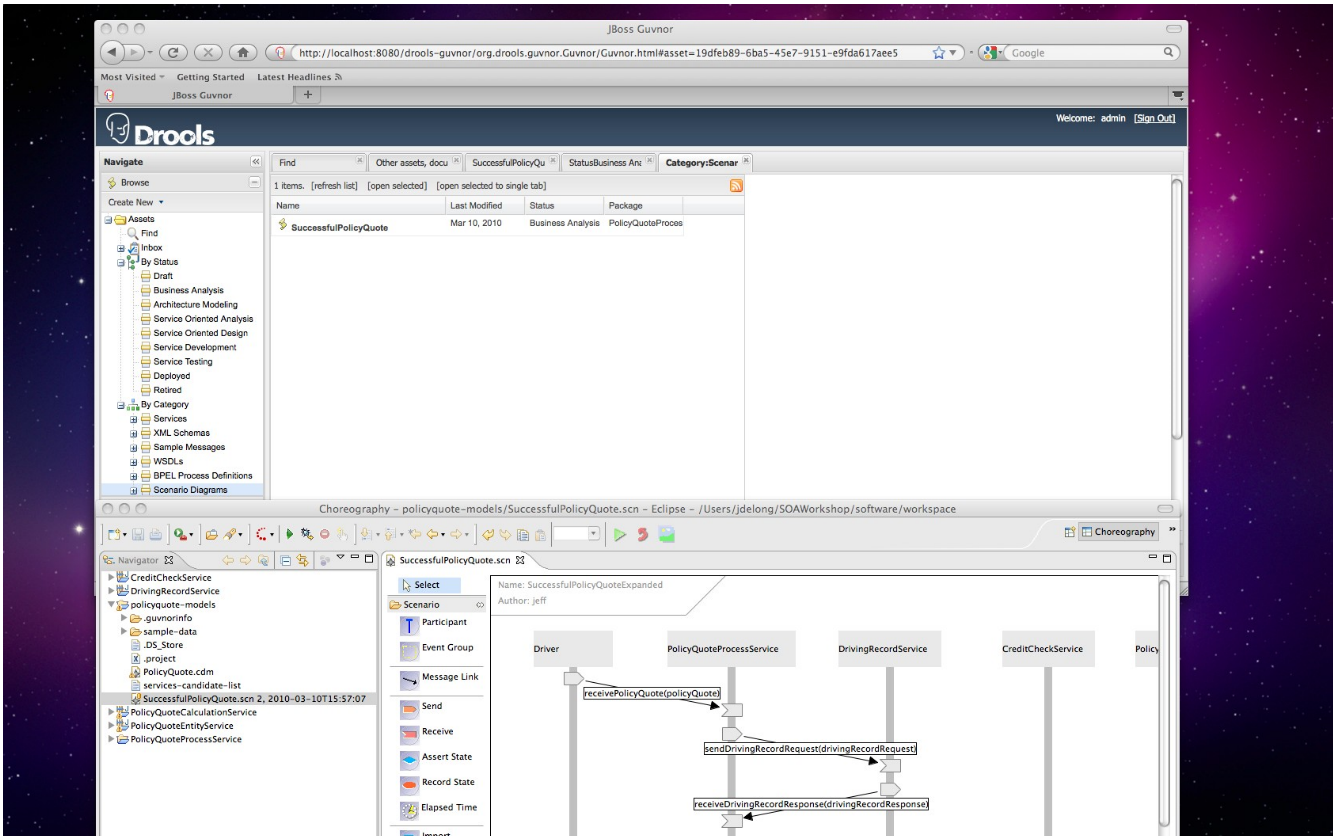
15) Open Guvnor, and create a Knowledge Package and name it PolicyQuoteProcessService. Create a new Category under Services / Orchestrated Service called PolicyQuoteProcessService.



16) Open JBDS and open the Guvnor Repository perspective. Add a new Guvnor Repository location. Now open policyquote-models. Right click on the SuccessfulPolicyQuote.scn and choose Guvnor -> Add. Choose the PolicyQuoteProcessService as the folder to save to.

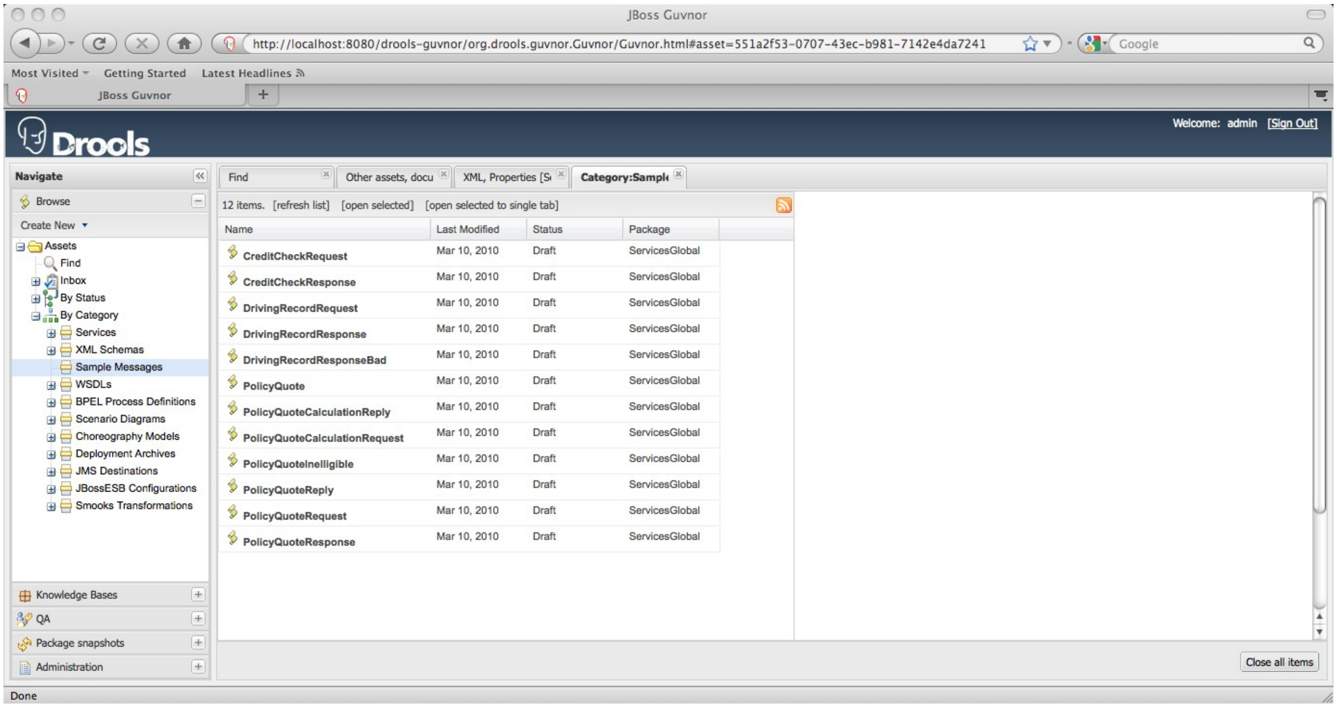


17) Go into Guvnor -> Knowledge Bases -> PolicyQuoteProcessService -> Other assets and open SuccessfulPolicyQuote. Show More Info, and Add Categories Services/Orchestrated Services/PolicyQuoteProcessService and Scenario Diagrams. Next, Change Status to BusinessAnalysis. Save Changes. Now highlight Browse -> Assets and look under Status Business Analysis and Category Scenario Diagrams. You should see the SuccessfulPolicyQuote.



18) Back in JBDS, use the Guvnor → Add feature to add the sample messages to the ServicesGlobal package. (You can actually Shift click the entire set of sample messages and right click Guvnor - > Add to add them all at once.

19) Once they have been added, open them up and add the Category Sample Messages.



20) Next add the PolicyQuote.cdm file into the PolicyQuoteProcessService in Guvnor.

21) Open the PolicyQuote.cdm in Guvnor, you will find it in the Other assets folder under the PolicyQuoteProcessService, and add the categories Services/Orchestrated Services/ PolicyQuoteProcessService and Choreography Model. Change the Status to Business Analysis and Save and close.

Lab3 Choreography

Goal: Create a Choreography model based on the SuccessfulPolicyQuote and UnsuccessfulPolicyQuote scenarios and test the choreography against those scenarios (see solutions directory).

Create Choreography Flow

1) Open the policyquote-models/PolicyQuote.cdm. Select the Choreography Flows editor.

2) Add an Interaction named PolicyQuoteRequest (choose the Interaction entity from the palette and drag it onto the navy blue thick line) at the top of the blue line. Enter the Operation as receivePolicyQuote (i.e., the name of the task). For Channel, select the PolicyQuoteProcessChannel. This will create a channel variable in the current choreography (called PolicyQuoteProcessChannel), and also set the Relationship property to the value relevant for the channel's target role, in this case the Relationship is DriverPolicyQuote.

3) Highlight the Exchange (enclosed within). Name it PolicyQuoteRequest as well, and select a Message Type of PolicyQuote. Leave its Action as Request.

4) Add an Interaction named DrivingRecordRequest immediately below the PolicyQuoteRequest. Enter the Operation as sendDrivingRecordRequest. For Channel select the DrivingRecordChannel. The Relationship is PolicyQuoteDrivingRecord.

5) Highlight the Exchange (enclosed within). Name it DrivingRecordRequest as well, and select a Message Type of DrivingRecordRequest. Leave its Action as Request.

6) Add an Interaction named DrivingRecordResponse below the DrivingRecordRequest. Enter the Operation as receiveDrivingRecordResponse. For Channel, use the DrivingRecordCallbackChannel. The Relationship is DrivingRecordDrivingRecordCallback.

7) Use the Variable icon from the palette and drag in into the State box below the choreography diagram. Name is DrivingRecordResponseVar.

8) Highlight the Exchange (enclosed within). Name it DrivingRecordResponse as well, and select a Message Type of DrivingRecordResponse. Leave its Action as Request (since this is part of two one-way requests). Select the DrivingRecordResponseVar variable as the ReceiveVariable in the exchange properties.

9) Add a Choice below the DrivingRecordResponse. Enter the Description "NumberOfTickets"

10) Add a Conditional below the Choice. Name it "LessThanOrEqualTo4. Enter for the Expression: cdl:getVariable('DrivingRecordResponseVar', ' ' , '//driv:numberOfTickets/text()', 'DrivingRecordCallbackService') <= 4. Referencing the Role as the last parameter makes it so that the

evaluation of the variable occurs only at that role.

11) Add a Sequence beneath the Conditional.

12) Within the Sequence, add an Interaction named CreditCheck. Enter the Operation as sendCreditCheckRequest. For Channel, select the CreditCheckChannel. The Relationship is PolicyQuoteCreditScoring.

13) Highlight the Exchange (enclosed within). Name it CreditCheckRequest, and select a Message Type of CreditCheckRequest. Leave its Action as Request.

14) Use the Exchange Icon to add a second exchange. Name it CreditCheckResponse, and select a Message Type of CreditCheckResponse. Change its Action to Response. Note - we could have done this with two separate interactions, but we chose to use a single Interaction with two exchanges for the request response MEP (as opposed to above where there were two oneway interactions).

15) Add an Interaction named PolicyQuoteResponse below CreditCheck. Enter the Operation as receivePolicyQuote (same as the request). For Channel, add the PolicyQuoteChannel (same as the request). The Relationship is DriverPolicyQuote.

16) Highlight the Exchange (enclosed within). Name it PolicyQuoteResponse as well, and select a Message Type of PolicyQuoteReply. Change its Action to Response.

17) Add another Conditional below the Choice. Name it "GreaterThan4. Enter for the Expression: cdl:getVariable('DrivingRecordResponseVar', ' ' , '//driv:numberOfTickets/text()', 'DrivingRecordCallbackService') > 4

18) Add a Sequence below the Conditional, and add an Interaction named PolicyQuoteInelligible within the sequence. Enter the Operation as receivePolicyQuote. For Channel, add the PolicyQuoteServiceChannelType. The Relationship is DriverPolicyQuote.

19) Highlight the Exchange (enclosed within). Name it PolicyQuoteInelligible as well, and select a Message Type of PolicyQuoteFault. Change its Action to Response. Use the Advanced properties to add a Fault Name of PolicyQuoteInelligible. At this point there should be no errors in the PolicyQuote.cdm file.

Create Identities

20) The first step in defining identity within a choreography is to create the identity token. Select the Base Types tab, and drag a Token from the palette onto the Tokens node. This will create a new token with name Token1. Select the Properties view and change the name to be IDSSN (since this Choreography will use the ssn of the driver to identify correlated messages).

21) The Type field will currently be empty. Press the New button. This will create a new information type called IDSSNType, which can be selected and then a suitable type (xsd:string) defined.

22) Now that we have defined the identity token, we need to link it to each of the message types that have been exchanged in the choreography. This is achieved by creating a TokenLocator for each message type and token pair, to provide the XPath expression that will locate the token's value within the message content for the specific message type, and to associate the identity token with the channel types. This association is required to indicate that messages that are exchange over a particular channel type must provide a locator for the tokens associated with that channel type. To associate an IDSSN token with a channel type, expand the Channel Types node in the Base Types tab.

23) Then drag the Identity element from the palette onto the PolicyQuoteProcessChannel. Then select the Properties view. You will see a field called Tokens. Press the button at the end of the field to display a selection list with the list of available tokens. Tick the checkbox next to the IDSSN token and press the Ok button.

24) Once the identity has been defined for the channel type, and the choreography has been saved, you will notice that errors are reported to indicate which interactions have messages being sent on the channel type, but without identity information to enable the messages to be correlated to a particular choreography session.

25) For each of the message type and token pairs reported in the errors, create a token locator by dragging the Token Locator item from the palette onto the Token Locators node. When each locator has been created, set the name to an appropriate value (e.g. PolicyQuoteIDLocator, PolicyQuoteReplyIDLocator, PolicyQuoteFaultIDLocator) and then set the Token field to IDSSN, Type field to the relevant message type (e.g., PolicyQuote, PolicyQuoteReply, PolicyQuoteFault, ...), and finally the Expression field to for example: "//pol:ssn/text()". (Note - look at the samples messages to determine the proper xpath expression, including the proper namespace prefix).

26) Once the token locators have been created for the PolicyQuote interactions, then select the IDSSN identity entity contained within the PolicyQuoteServiceChannel and copy it to the DrivingRecordChannel (via Copy and Paste). This will create an IDSSN identity element for the DrivingRecordChannel. Once the choreography has been saved, then create the token locators to fix the new set of errors that have been created. I.e. Repeat step 25 for DrivingRecordRequestIDLocator and DrivingRecordResponseIDLocator. In this case use //driv:ssn/text() for the xpath expression.

27) Now create the Token Locators for the CreditCheckServiceChannel and associated messages. Select the IDSSN identity entity contained within the PolicyQuoteServiceChannel and copy it to the CreditCheckChannel (via Copy and Paste). This will create an IDSSN identity element for the CreditCheckChannel. Once the choreography has been saved, then create the token locators to fix the new set of errors that have been created. I.e. Repeat step 25 for CreditCheckRequestIDLocator and CreditCheckResponseIDLocator. In this case use //cred:ssn/text() for the xpath expression.

Test Scenarios

28) Open the SuccessfulPolicyQuote.scn scenario. Examine each MessageLink and reselect the appropriate value from the drop down list; to will now include the operation name. E.g., receivePolicyQuote(PolicyQuote) and save the file.

29) Next, run the simulation by pressing the Green Triangle in the Eclipse Toolbar at the very far right. If there is a red node, click on it and review the log. Otherwise click on one of the green nodes and view the log.

30) Copy in the UnsuccessfulPolicyQuote.scn from the solutions directory and run it as well.

Update Repository

31) Next right click on updated PolicyQuote.cdm file and select Guvnor -> Commit.

32) Open the PolicyQuote.cdm in Guvnor, you will find it in the Other assets folder under the PolicyQuoteProcessService. Change the Status from Business Analysis to Architectural Modeling and Save and close.

33) Next right click on the SuccessfuPolicyQuote.scn and commit to Guvnor.

34) Open SuccessfuPolicyQuote.scn in Guvnor and change the status to Service Oriented Analysis.

34) Next right click on the SuccessfuPolicyQuote.scn and add it to Guvnor.

35) Open UnsuccessfuPolicyQuote.scn in Guvnor and add categories Services/Orchestrated Services/PolicyQuoteProcessService and Scenarios and change the status to Service Oriented Analysis.

Lab4 Service Modeling

Goal: Create a Services Candidate Description for each new service that specifies the name of the new candidate service, what type of service it is (Entity, Utility, Task, or Orchestrated), and its candidate tasks. Extend the Scenario and expand the Choreography model to include two new services.

1) Review the Business Requirements from Lab2. It appears from the description of the requirements that two new services may be needed, one to calculation the policy quote price (PolicyQuoteCalculationService) and the other to persist the quote (and PolicyQuoteEntityService).

2) Create a text document in policyquote-models for each new service called services-candidate-description-*serviceName* that describes what the service does, what type of service it is, and its associated tasks (see solution if needed). Open the PolicyQuoteDataModel and review its contents. Are there any new Entity Service candidates or associated tasks required?

3) Open Guvnor and create a Knowledge Package for the two new service candidates: PolicyQuoteEntityService and PolicyQuoteCalculationService. Add new categories for each under Services/Entity Services and Service /Task Services.

4) Use JBDS Guvnor → Add to add the services-candidate-description-*serviceName* files to their respective packages / folders in Guvnor. Using the Guvnor UI, find the file in the Other assets folder under the PolicyQuoteEntityService, and add the categories Services/Entity Services/ PolicyQuoteEntityService and Documentation. Change the Status to Service Oriented Analysis and Save and close. Repeat for the newly added file in PolicyQuoteCalculationService, except use the categories Services/Task Services/PolicyQuoteCalculationService and Documentation.

Update Roles, Relationships, and Base Types

5) Open PolicyQuote.cdm. In the Roles and Relationships editor, use the Role icon to add two more roles: PolicyQuoteCalculationService and PolicyQuoteEntityService. Name the associated Behaviors: PolicyQuoteCalculator, PolicyQuoteEntityProvider.

6) Use the Relationship tool to draw relationships from PolicyQuoteProcessService to PolicyQuoteCalculationService (name it PolicyQuotePolicyQuoteCalculation) and PolicyQuoteProcessService to PolicyQuoteEntityService (name it PolicyQuotePolicyQuoteEntity).

7) Next in the Base Types editor rename the Participant Types PolicyQuoteCalculationService and PolicyQuoteEntityService.

8) At this stage, there should only be a few errors associated with the choreography: "Reference 'Reference Token' must be specified". Double-click one of the errors, this will navigate to the ChannelType that has reported the validation problem. Rename the ChannelType to remove "Service" and "Type" (just to make the name shorter). Select the Properties view to see the property that has the

error. This problem can be resolved by setting the Reference Token property to URIToken by selecting it from the combo box.

Extend the Successful Scenario

9) Copy the SuccessfulPolicyQuote.scn and name it SuccessfulPolicyQuote.scn. Under the general properties Choreography Description URL enter the value PolicyQuote.cdm.

10) Use the Participant icon to add Participants PolicyQuoteCalculationService and PolicyQuoteEntityService at the right of CreditCheckService. Use the Type property drop down and select the appropriate participant.

11) Use the MessageLink icon to link PolicyQuoteProcessService to PolicyQuoteCalculationService. Highlight the MessageLink line, and select the Message property PolicyQuote from the drop down. Add the Value URL of sample-data/PolicyQuote.xml. Repeat this step for PolicyQuoteCalculationService back to PolicyQuoteProcessService: PolicyQuoteReply, PolicyQuoteProcessService to PolicyQuoteEntityService: PolicyQuote, and PolicyQuoteEntityService back to PolicyQuoteProcessService: PolicyQuoteReply.

Update Choreography Flow

12) Next select the Choreography Flows editor in PolicyQuote.cdm. Add an Interaction named PolicyQuoteCalculation after the CreditCheckResponse (choose the Interaction entity from the palette and drag it onto the navy blue thick line). Enter the Operation as calculatePolicyQuote. For Channel, select the PolicyQuoteCalculationServiceChannel. This will create a channel variable in the current choreography (called PolicyQuoteServiceChannel), and also set the Relationship property to the value relevant for the channel's target role, in this case the Relationship is PolicyQuotePolicyQuoteCalculation.

13) Highlight the Exchange (enclosed within). Name it PolicyQuoteCalculation as well, and select a Message Type of PolicyQuote. Leave its Action as Request.

14) Use the Exchange Icon to add a second exchange. Name it PolicyQuoteCalculationReply, and select a Message Type of PolicyQuoteReply. Change its Action to Response.

15) Add an Interaction named PolicyQuoteEntity. Enter the Operation as createPolicyQuote (same as the request). For Channel, add the PolicyQuoteEntityChannel.

16) Highlight the Exchange (enclosed within). Name it PolicyQuoteEntity as well, and select a Message Type of PolicyQuote.

14) Use the Exchange Icon to add a second exchange. Name it PolicyQuoteEntityReply, and select a Message Type of PolicyQuoteReply. Change its Action to Response.

Create Identities

17) Now create the IDSSN for the PolicyQuoteCalculationChannel and PolicyQuoteEntityChannel.

Run Scenario

18) Next, go to the SuccessfulPolicyQuote.scn and run the simulation by pressing the Green Triangle.

19) Copy in the UnsuccessfulPolicyQuote.scn from the solutions directory and run it as well.

Update Repository

20) Next right click on updated PolicyQuote.cdm file and select Guvnor -> Commit.

21) Open the PolicyQuote.cdm in Guvnor, you will find it in the Other assets folder under the PolicyQuoteProcessService. Change the Status from Architectural Modeling to Service Oriented Analysis and Save and close.

22) Next right click on the SuccessfuPolicyQuote and UnsuccessfulPolicyQuote (separately) and commit them to Guvnor.

23) Open each in Guvnor and change the status of to Service Oriented Analysis.

Lab5 Service Oriented Design

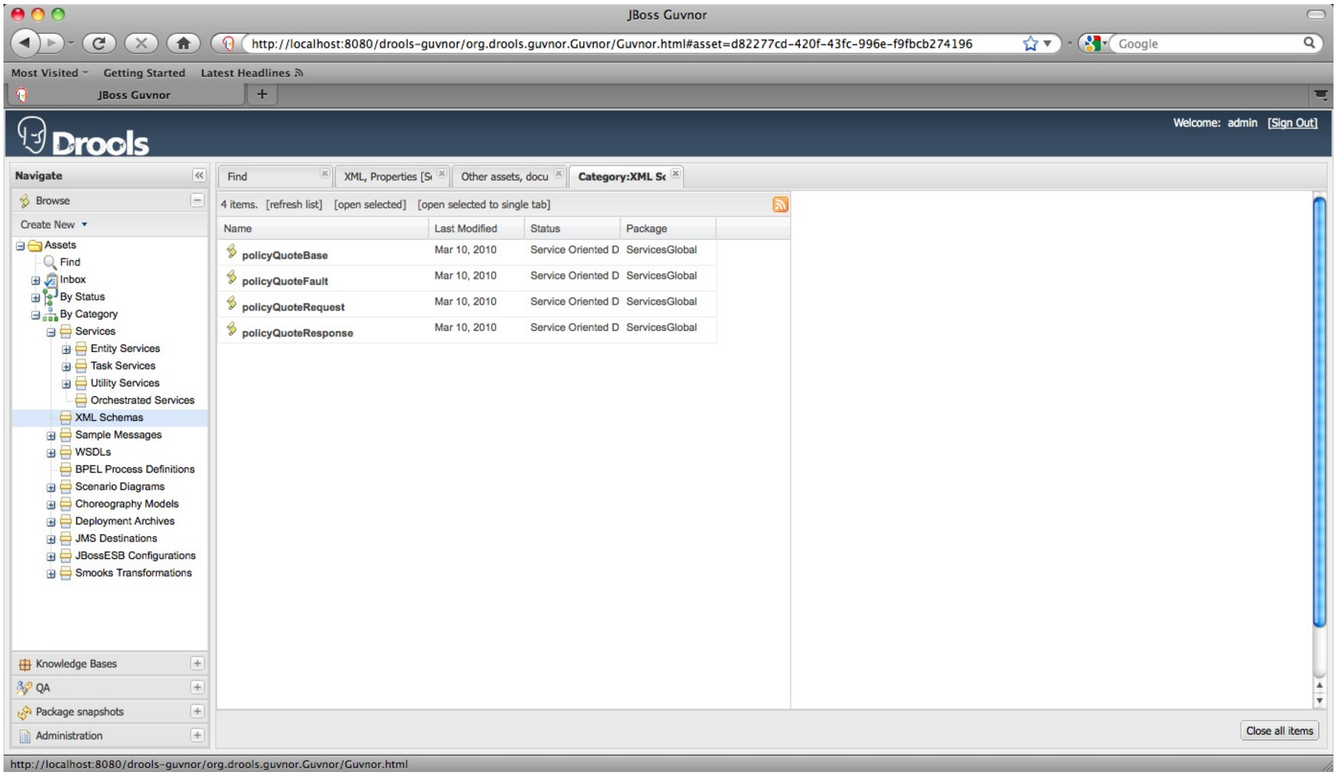
Goal: Associate the message schema to each new service.

Note: As part of our effort to standardize service contracts, we have decided to standardize on the use of XML and an extensible message structure based on a Task element, any common elements (e.g., date) and then a complex type that varies by task. Since we do plan to use XML throughout, we need to define XML schemas for request and response messages from / to clients. While initially we have only identified a single business process (PolicyQuoteProcess), we anticipate that our services may be used in other business processes, and want to promote the use of resuable and composable services. Therefore we will define XML schemas for services that are currently internal to this business process. To the extent possible, we would like to standardize on a "canonical" message structure, as to minimize the number of distinct transformations required. Hence we define a single policyQuote schema (defined in four separate xsd files) that will support several of the services. Typically we would start this schema design as part of the Information Modeling task within Architectural Modeling, and finish it during Service Oriented Design. For the purposes of this lab, these schema files have already been created.

1) Copy the schema folder from labs to the root of policyquote-models in your workspace. Open and examine the policyQuoteBase.xsd, policyQuoteRequest.xsd, policyQuoteResponse.xsd, and policyQuoteFault.xsd. Next examine the various sample messages in sample-data.

2) Add the contents of the entire schema folder to Guvnor in the ServicesGlobal package. Note adding xsd files will cause Eclipse to report errors; these can be ignored.

3) Open in Guvnor, you will find the schema files in the Other assets folder. Add the category XML Schemas. For the PolicyQuoteBase.xsd, PolicyQUoteRequest.xsd, PolicyQuoteResponse.xsd, and PolicyQuoteFault.xsd, also add the categories Services/Orchestrated Services/PolicyQuoteProcessService, Services/Entity Services/PolicyQuoteEntityService, and Services/Task Services/PolicyQuoteCalculationService. Set the Status to Service Oriented Design for each file.


JBoss Guvnor
http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/Guvnor.html#asset=d82277cd-420f-43fc-996e-f9fbcb274196
Most Visited
Getting Started
Latest Headlines
JBoss Guvnor
Drools
Welcome: admin [Sign Out]
Navigate
Browse
Create New
Assets
Find
Inbox
By Status
By Category
Services
Entity Services
Task Services
Utility Services
Orchestrated Services
XML Schemas
Sample Messages
WSDLs
BPEL Process Definitions
Scenario Diagrams
Choreography Models
Deployment Archives
JMS Destinations
JBossESB Configurations
Smooks Transformations
Knowledge Bases
QA
Package snapshots
Administration
Find
XML, Properties [S
Other assets, docu
Category:XML Sc
4 items. [refresh list] [open selected] [open selected to single tab]
Name
Last Modified
Status
Package
policyQuoteBase
policyQuoteFault
policyQuoteRequest
policyQuoteResponse
Mar 10, 2010
Service Oriented D
ServicesGlobal
Close all items
http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/Guvnor.html

Lab6 ESB Services

Goal: Create an ESB Project for the PolicyQuoteEntityService and the PolicyQuoteCalculationService (existing external services have already been created and are in the lab/resources folder).

1) Use JBoss Developer Studio New -> Project -> ESB -> ESB Project. Name it PolicyQuoteEntityService. Choose soa-p 5.0 server as the server.

2) Create a JCA Provider (for the JMS queue). Highlight Providers and click on Add with the following values:

Name	JBossMessaging
ConnectionFactory	XAConnectionFactory
ChannelID	PolicyQuoteEntityChannel

Under Advanced:

Jndi Context Factory	org.jnp.interfaces.NamingContextFactory
Jndi URL	localhost

Expand under the Providers JBossMessaging and select PolicyQuoteEntityChannel. Expand under PolicyQuoteEntityChannel and select Filter. Add values:

Destination Name	queue/PolicyQuoteEntity_ESB
Destination Type	QUEUE (this is the default)
Transacted	true

Leave other values blank.

3) Create a Service. Highlight Services and click on Add. Provide the following values

Name	PolicyQuoteEntityService
Category	EntityServices
Description	Persists Policy Quote data

4) Add a JMS Listener to the Service with values:

Name	PolicyQuoteEntityChannelListener
Channel ID Ref	PolicyQuoteEntityChannel

5) Add an Action to the Service of type System Println. Give it a

Name	PrintBefore
Message	Message when entering PolicyQuoteEntityService.

Print Full	True

6) Use JBoss Developer Studio New -> Project -> ESB -> ESB Project. Name it PolicyQuoteCalculationService. Choose soa-p 5.0 as the server.

7) Create a JCA Provider (for the JMS queue). Highlight Providers and click on Add with the following values:

Name	JBossMessaging
ConnectionFactory:	XAConnectionFactory
ChannelID	PolicyQuoteCalculationChannel

Under Advanced:

Jndi Context Factory	org.jnp.interfaces.NamingContextFactory
Jndi URL	localhost

Expand under the Providers JBossMessaging and select PolicyQuoteCalculationChannel. Expand under PolicyQuoteCalculationChannel and select Filter. Add values:

Destination Name	queue/PolicyQuoteCalculation_ESB
Destination Type	QUEUE (this is the default)
Transacted	true

Leave other values blank.

8) Create a Service. Highlight Services and click on Add. Provide the following values

Name	PolicyQuoteCalculationService
Category	TaskServices
Description	Calculation Policy Quote data

9) Add a JMS Listener to the Service with values:

Name	PolicyQuoteCalculationChannelListener
Channel ID Ref	PolicyQuoteCalculationChannel

10) Add an Action to the Service of type System Println. Give it a

Name	PrintBefore
Message	Message when entering PolicyQuoteCalculationService.
Print Full	True

Add to Repository

11) Next open the PolicyQuoteEntityService in JBDS. Add the jboss-esb.xml into

thePolicyQuoteEntityService in Guvnor.

12) Open Guvnor, you will find the jboss-esb.xml file in the XML, Properties folder of the PolicyQuoteEntityService package. Add the categories Services /Entity Services/PolicyQuoteEntityService and JBossESB Configuration and the Status Service Development.

13) Next open the PolicyQuoteCalculationService in JBDS. Add the jboss-esb.xml and deployment.xml into thePolicyQuoteCalculationService in Guvnor.

14) Open Guvnor, you will find these newly added files in the XML, Properties folder. Add the categories Services /Task Services/PolicyQuoteCalculationService and JBossESB Configuration and the Status Service Development.

Lab7 Transformations

Goal: Transform XML message data to Java for PolicyQuoteEntityService and PolicyQuoteCalculationService

PolicyQuoteEntityService

1) In JBoss Developer Studio open the PolicyQuoteEntityService project. Copy the org folder from lab7-transformations/src to the project src folder. Copy policyQuote.xml and policyQuoteReply.xml from labs/sample-data folder to the project root folder. Copy policyQuote.xsd, policyQuoteReply.xsd, policyQuoteFault.xsd, and types folder from labs/schema folder to the project's esbcontent folder.

2) Use the File -> New -> Smooks -> Smooks Configuration File and name it PolicyQuoteConfig.smooks. Select the Input Task, right click, and select Add Task and select Java Mapping. Select the Input Task once more, the Input Type as XML, and under Input Data select Add and Browse Workspace -> PolicyQuoteEntityService and select PolicyQuote.xml and hit finish.

3) Next right click in the Selected Task Details to the right of the XML Input Model and select Add -> Java Class, name the BeanId policyQuote and and browse to org.acme.insurance.PolicyQuote for the ClassName. Now drag from the requestDate in the XML Input Model to the requestDate in the policyQuote. Edit the Properties to add the data format (yyyy-MM-dd) Repeat for all the XML elements.

4) Then select the policyQuoteID in the Java object, and right click -> delete (we don't map the policyQuoteID because it will be set by the downstream action).

5) Right click to add another Java class, this time with a beanID of task and the class name org.acme.insurance.Task. Map the task element to taskName. (Note - you may need to open the Source tab and add createOnElement="policyQuote" to each of the the jb:bean elements).

6) Open the jboss-esb.xml and add a SmooksAction (Actions -> Add -> Transformers/Converters -> Smooks Action) with:

Name	CreatePolicyQuoteObject
Smooks Config	PolicyQuoteConfigSmooks.xml
Result Type	JAVA
Set Payload Location	policyPayload

7) Add another SystemPrintln action. Name it PrintAfter with the message "Message after transformation". Set Print Full to true (this will print out the entire message, not just the default body location).

8) Copy deployment.xml from solution/entityMETA-INF to the esb-content/META-INF directory.

9) Right click on the PolicyQuoteConfigSmooks.xml and choose Run As -> Smooks Run Configuration. You should see some printout like:

```
[Java Mapping Results...]
   |--
   |> policyQuote (beanId = "policyQuote")
   |    > policyQuoteID = 0L
   |    > requestDate = "2009-01-01 00:00:00.0 MST"
   |    > driverName = "Bill Smith"
   |    > age = 30I
   |    > ssn = "012345678"
   |    > dlNumber = "987654321"
   |    > numberOfAccidents = 1I
   |    > numberOfTickets = 2I
   |    > creditScore = 500I
   |    > policyType = "AUTO"
   |    > vehicleYear = 2004I
   |    > price = 0I
   |--
   |--
   |> task (beanId = "task")
   |    > taskName = "createPolicyQuote"
   |--
```

PolicyQuoteCalculationService

10) In JBoss Developer Studio open the PolicyQuoteCalculationService project. Copy the org folder from lab7-transformations/src-policydriver to the project src folder. Copy policyQuoteCalculationRequest.xml and policyQuoteCalcualtionResponse.xml from labs/sample-data folder to the project root folder. Copy policyQuote.xsd, policyQuoteReply.xsd, policyQuoteFault.xsd, and types folder from labs/schema folder to the project's esbcontent folder.

11) Use the File -> New -> Smooks -> Smooks Configuration File and name it PolicyQuoteCalculationConfig.smooks. Select the Input Task, right click, and select Add Task and select Java Mapping. Select the Input Task once more, the Input Type as XML, and under Input Data select Add and Browse Workspace -> PolicyQuoteCalculationService and select PolicyQuoteCalculationRequest.xml and hit finish.

12) Next right click in the Selected Task Details to the right of the XML Input Model and select Add -> Java Class, name the BeanId driver and browse to org.acme.insurance.Driver, then another Java Class name the BeanId policy and browse to org.acme.insurance.Policy for the ClassName. Link the driver in policy to driver. Now drag from the requestDate in the XML Input Model to the requestDate in the policy. Edit the Properties to add the data format (yyyy-MM-dd). Repeat for all the XML elements.

13) Right click to add another Java class, this time with a beanID of task and the class name org.acme.insurance.Task. Map the task element to taskName. (Note - you may need to open the Source

tab and add createOnElement="policyQuote" to each of the the jb:bean elements).

14) Open the jboss-esb.xml and add a SmooksAction with:

Name	CreatePolicyAndDriverObjects
Smooks Config	PolicyQuoteCalculationConfigSmooks.xml
Result Type	JAVA
Set Payload Location	policyPayload

14) Add another SystemPrintln action. Name it PrintAfter with the message "Message after transformation". Set Print Full to true (this will print out the entire message, not just the default body location).

15) Copy deployment.xml from solution/calculation/META-INF to the esb-content/META-INF directory.

16) Right click on the PolicyQuoteCalculationConfigSmooks.xml and choose Run As -> Smooks Run Configuration. You should see some printout like:

```
[Java Mapping Results...]
    |--
    |> task (beanId = "task")
    |    > taskName = "calculatePolicyQuote"
    |--
    |--
    |> policy (beanId = "policy")
    |    > requestDate = "2009-01-01 00:00:00.0 MST"
    |    > policyType = "AUTO"
    |    > vehicleYear = 2004I
    |    > price = 0I
    |    > driver (beanId = "driver")
    |        > driverName = "Bill Smith"
    |        > age = 30I
    |        > ssn = "012345678"
    |        > dlNumber = "987654321"
    |        > numberOfAccidents = 1I
    |        > numberOfTickets = 2I
    |        > creditScore = 500I
    |--
```

Lab8 Entity Service

Goal: Add a custom JPA Persistance action to PolicyQuoteEntityService to persist the PolicyQuote data.

1) In JBoss Developer Studio open the PolicyQuoteEntityService project. Copy the src folder from lab8-entity-service/solutions. Copy PolicyQuoteEntity.jar from labs/resources to the root of the project. Then add this jar to the project Java Build Path.

2) Open the jboss-esb.xml and add a Custom Action with class and name:

class	org.acme.insurance.PolicyQuotePersistanceAction
name	JPAAction

Add two properties:

name	entityManagerJNDIName
value	persistence.unit:unitName=#PolicyQuoteEntity
name	manageTransactionsInAction
value	false

3) Highlight Action and add the inXsd, outXsd, and faultXsd under Advanced.

In Xsd	/policyQuoteRequest.xsd
Out Xsd	/policyQuoteResponse.xsd
Fault Xsd	/policyQuoteFault.xsd

4) Open PolicyQuotePersistenceAction and review the code. Change the code to get the message data from the policyPayload, since the previous Smooks action uses this location (i.e., set-payload-location = policyPayload). So the updated line of code should look like:

```
Map policyPayload = (Map) newMessage.getBody().get("policyPayload");
```

5) Add another SystemPrint action. Name it PrintAfterPersistance with the message "PolicyQuote after persistence operations have completed". Set the Print FUll option to true.

6) Next we need to take the results of the persistence action, and transform it back to XML. Copy the PolicyQuoteReplyConfigSmooks.xml from the solutions/esbcontent folder. Then add another SmooksAction with:

Name	CreatePolicyQuoteReply
Smooks Config	PolicyQuoteReplyConfigSmooks.xml
Get Payload Location	policyPayload
Result Type	STRING

7) Add another SystemPrint action. Name it PrintAfterReplyCreated with the message "PolicyQuote after reply message created".

8) Set the MEP at the top of the Actions List to RequestResponse.

9) Next copy labs/resources/PolicyQuoteEntity.jar and policyquote-queue-service.xml to the server/default/deploy directory.

10) Start the server first (if not already running). Most people prefer to run this from a separate command prompt instead of from inside Eclipse.

11) Right click on the SOA-P server in the Server view, and select Add and Remove and then move the PolicyQuoteEntityService to the right (Configure) list. Right click on the PolicQuoteEntityService and select Full Publish.

12) Next use SoapUI to send a SOAP Message to the service and check the response. The WSDL is in jboss-soa-p.5.0.0/jboss-as/server/default/data/wsdl/PolicyQuoteEntityService.esb/EntityServices. First try the createPolicyQuote task, then the getPolicyQuote task. Or, copy the WSDL to the PolicyQuoteEntityService project, and use the Eclipse Web Services Explorer (change to the Java EE perspective and look for it on the tool bar).

13) Open the JBoss JMX console, http://localhost:8080/jmx-console. Scan down to jboss -> database=localDB, service=Hypersonic. On the next page scan down startDatabaseManager. In the GUI, enter SELECT * FROM POLICYQUOTE and hit the Execute button.

Update Repository

14) Commit the updated jboss-esb.xml file into Guvnor.

15) Add the deployment.xml, PolicyQuoteConfigSmooks.xml, PolicyQuoteReplyConfigSmooks.xml, PolicyQuoteEntity.jar and PolicyQuotePersistanceAction.java into the PolicyQuoteEntityService in Guvnor. Add Task.java into ServicesGlobal.

16) Open in Guvnor, you will find these newly added files in the XML, Properties folder of the PolicyQuoteEntityService and ServiceGlobal. Add the categories Services /Entity Services/PolicyQuoteEntityService and either JbossESB Configuration, SmooksTransformation, Deployment Archive, or Java Source, and the Status Service Development. Note that PolicyQuoteEntity.jar is under the Model folder.

Lab9 Task Service

Goal: Create a task service PolicyQuoteCalculationService that uses a set of rules to calculate the quote price.

1) Open the PolicyQuoteCalculationService project. Copy policyquotecalculation.drl file from policyquotesolution/esbContent into the esbcontents directory.

2) Highlight Action and add the inXsd, outXsd, and faultXsd under Advanced.

In Xsd	/policyQuoteRequest.xsd
Out Xsd	/policyQuoteResponse.xsd
Fault Xsd	/policyQuoteFault.xsd

3) Add a BusinessRulesProcessor action after the PrintAfter action:

Name	CalculateQuote
Rule Set	/policyquotecalculation.drl

Add two items to the Object Paths List with values:

body.policyPayload.policy
body.policyPayload.driver

4) Add another SystemPrint action. Name it PrintAfterQuoteCalculation with the message "PolicyQuote price after calculation". Set Print Full to true.

5) Next we need to take the results of the rules execution, and transform the POJOs back to XML. Copy the PolicyQuoteCalculatioReplyConfigSmooks.xml from the solutions/esbcontent folder to the project esbContent folder. Open and examine its contents.

7) Add another SmooksAction with:

Name	CreatePolicyQuoteCalculationReply
Smooks Config	PolicyQuoteCalculationReplyConfigSmooks.xml
Get Payload Location	policyPayload
Result Type	STRING

7) Add another SystemPrint action. Name it PrintAfterReplyCreated with the message "PolicyQuote after reply message created".

8) Set the MEP at the top of the Actions List to RequestResponse.

9) Start the server first (if not already running). Most people prefer to run this from a separate command prompt instead of from inside Eclipse.

10) Right click on the SOA-P server in the Server view, and select Add and Remove and then move the PolicyQuoteCalculationService to the right (Configure) list. Right click on the PolicQuoteCalculationService and select Full Publish.

11) Next use SoapUI to send a SOAP Message to the service and check the response. The WSDL is in jboss-soa-p.5.0.0/jboss-as/server/default/data/wsdl/PolicyQuoteCalculationService.esb/TaskServices. Use the calculatePolicyQuote task. Or, copy the WSDL to the PolicyQuoteEntityService project, and use the Eclipse Web Services Explorer (change to the Java EE perspective and look for it on the tool bar).

12) Examine the console and check the message before and after the policy quote calculation. A price should be in the XML reply message.

Update Repository

13) Next add the policyquotecalculation.drl, PolicyQuoteCalculationConfigSmooks.xml and PolicyQuoteCalculationReplyConfigSmooks.xml, Policy.java, and Driver.java into the PolicyQuoteCalculationService in Guvnor.

14) Open Guvnor, you will find these newly added files in the XML, Properties folder. Add the categories Services/Task Services/PolicyQuoteCalculationService and either SmooksTransformation or Java Source and the Status Service Development. The policyquotecalculation.drl will be in the Technical rule assets folder. Open it and add the categories Services/Task Services/PolicyQuoteCalculationService and the Status Service Development.

15) Add the PolicyQuoteCalcualtionService.wsdl and PolicyQuoteEntityService.wsdl found in server/default/data/wsdl/... to their respective packages in Guvnor. Open the packages in Guvnor and assign the appropriate categories and status (e.g., Services/Task Services/PolicyQuoteCalculationService and WSDL and Status Service Development).

16) Add CreditCheckService.wsdl and DrivingRecordService.wsdl from the labs/wsdl folder to Guvnor in the ServicesGlobal package.

Lab10 Orchestrated Service

Goal: Create a BPEL process for the PolicyQuoteProcessService.

Note - if using SAVARA tooling, the skeleton for the process could be generated from the Choreography model. This would save time, however it would not provide the experience of creating the process from scratch.

Note - this lab is very lengthy, so break points have been included so that the lab can alternate with lectures if desired.

1) Open JBDS Designer and create a new BPEL project. File -> New -> Other -> BPEL 2.0 -> BPEL Project. Name it PolicyQuoteProcessService, Switch to the JBossAS or the Choreography perspective. (there is no BPEL perspective, however the either of these perspectives have a good set of views for BPEL process development).

Use the File -> New -> Guvnor -> Resources from Guvnor and add the WSDL files from PolicyQuoteEntityService, PolicyQuoteCalculationService, and ServicesGlobal into the root of this project (4 WSDL files).

2) Create a new BPEL Process. File -> New -> Other -> BPEL 2.0 -> New BPEL Process File. Select an Empty BPEL Process (it will be synchronous, since this will be easier to test with a simple SOAP client, however we already have the wsdl, and don't want the wizard to create the client partnerLink for us). Name it PolicyQuoteProcess. Select the Target Namespace (either choice in the list should be fine).

Create Partner Links

3) Create PartnerLinks using the Partner Links + on the right hand side. Name the first one CreditCheckService. Use the Browse button in the Properties view (Details), then in the Choose Partner Link Type window select From Project and Filter Show Port Types. This will then list the PortTypes from the WSDL files imported in step 1. Highlight the CreditCheckServicePortType (IMPORTANT) and hit Ok. This will create a new PartnerLinkType. Name it CreditCheckServiceLT, and then name the role CreditCheckServiceRole and select the CreditCheckServicePortType. Then select Finish. Next Set Namespace Prefix to spl. Examine the Properties View, Under the Partner Role select the CreditCheckServiceRole radio button. Make certain that the CreditCheckServiceOp appears under the Partner Operations box.

Do the same for the following list of PartnerLinks (the first one was just completed).

Partner Link	PartnerLinkType	PartnerRole	My Role
CreditCheckService	CreditCheckServiceLT	CreditCheckServiceRole	
DrivingRecordService	DrivingRecordServiceLT	DrivingRecordServiceRol e	
PolicyQuoteCalculation Service	PolicyQuoteCalculationS erviceLT	PolicyQuoteCalculationS erviceRole	
DrivingRecordCallback Service	DrivingRecordCallbackS erviceLT		DrivingRecordCallbac kServiceRole
PolicyQuoteEntityServi ce	PolicyQuoteEntityService LT	PolicyQuoteEntityService Role	
PolicyQuoteProcessSer vice	PolicyQuoteProcessServi ceLT		PolicyQuoteServiceRo le

Create Variables

4) Create Variables using the Variables + on the right hand side. Name the first one PolicyQuoteRequest. Use the Browse button in the Properties view, and then Show XSD Types select From Project and then filter on Messages. Select the PolicyQuoteProcessServiceRequest and then use a namespace prefix of orchws. Do the same for the following list of Variables (the first one is already completed).

Variable	Namespace Prefix	MessageType
PolicyQuoteRequest	orchws	PolicyQuoteProcessServiceRequest
PolicyQuoteResponse	orchws	PolicyQuoteProcessServiceResponse
CreditCheckRequest	cas	CreditCheckServiceReq
CreditCheckResponse	cas	CreditCheckServiceRes
DrivingRecordRequest	dmv	DrivingRecordServiceReq
DrivingRecordResponse	drcs	DrivingRecordCallbackServiceReq
PolicyQuoteCalculationRequest	calc	PolicyQuoteCalculationServiceReq
PolicyQuoteCalculationResponse	calc	PolicyQuoteCalculationServiceRes
PolicyQuoteEntityRequest	polent	PolicyQuoteEntityServiceReq
PolicyQuoteEntityResponse	polent	PolicyQuoteEntityServiceRes
PolicyQuoteFault		PolicyQuoteProcessServiceFault

Create the Sequence

5) Rename the sequence main to PolicyQuoteSequence. Use the Palette to add a Receive Activity (click directly on the leftmost of the two parallel gray vertical bars that divides the canvas from the area with PartnerLinks, Variables, Correlation Sets, and Message Exchanges, select the Receive icon and drag it beneath the PolicyQuoteSequence). Name it ReceivePolicyQuoteRequest. In the Properties View select the PolicyQuoteProcessService as the Partner Link, and select PolicyQuoteProcessServiceOp as the Operation. Uncheck the Use WSDL Message Parts Mapping and select the Variable PolicyQuoteRequest. Check Create a new Process instance if one does not already exist.

6) Next add a Reply activity and name it SendPolicyQuoteResponse. In the Properties View select the PolicyQuoteProcessService as the Partner Link, and select PolicyQuoteProcessServiceOp as the Operation. Uncheck the Use WSDL Message Parts Mapping and select the Variable PolicyQuoteResponse.

7) Next drag and drop an Invoke activity from the Palette between the ReceivePolicyQuoteRequest and the SendPolicyQuoteResponse. Name it SendDrivingRecordRequest. In the Properties View select the DrivingRecordService as the Partner Link, and select DrivingRecordServiceOp as the Operation. Uncheck the Use WSDL Message Parts Mapping and select the Variable DrivingRecordRequest.

BREAK

8) Now add an Assign to assign the data from the PolicyQuoteRequest to the SendDrivingRecordRequest. Drag and drop the Assign onto the palette in between the ReceivePolicyQuoteRequest and the SendDrivingRecordRequest. Name it AssignDataForDrivingRecordRequest. Use the New button to create four new copy operations in the Properties view to copy from the PolicyQuoteRequest/in/receivePolicyQuote/policyQuoteInfo elements to the corresponding DrivingRecordRequest elements (copy name, ssn, dlNumber, and age). When requested, set the 'from' namespace prefix to pol and the 'to' namespace prefix to driv. When it asks to create an Initializer, select Yes.

9) Next drag and drop a Receive activity from the Palette below the SendDrivingRecordRequest. Name it ReceiveDrivingRecordResponse. In the Properties View select the DrivingRecordCallbackService as the Partner Link, and select DrivingRecordCallbackServiceOp as the Operation. Uncheck the Use WSDL Message Parts Mapping and select the Variable DrivingRecordResponse.

Add a Correlation Set

10) Create a Correlation Set using the Correlation Sets + on the right hand side. Name it SSNCorrelationSet. On the Properties View select Add to create a new BPEL Property. Name it SSN, hit Browse for the Type, Select Show XSD Types From Import and filter Simple Types (uncheck all other boxes), and select String as the Type. Change the namespace to xs. Then add three PropertyAliases. Select New, then Browse, Show XSD Types From Project, filter on Messages. Select PolicyQuoteProcessServiceRequest, and within that message choose the Part receivePolicyQuote/policyQuoteInfo/ssn. Do the same for DrivingRecordServiceReq and DrivingRecordCallbackServiceReq.

11) Next add the correlation set to the various activities. Start with the ReceivePolicyQuoteRequest Activity. Highlight the activity, then go to Properties View and select the Correlations tab. Press the Add button on the right. Since there is only one Correlation Set defined, it will select SSNCorrelationSet. This associates the activity with the correlation set. Click the cell under Initiation and select Yes. Leave the direction as Receive. Because this Receive activity will always instantiate the process, it will always initiate the correlation set, too. Repeat these steps for SendDrivingRecordRequest (Send, No) ReceiveDrivingRecordResponse (Receive, No), and SendPolicyQuoteResponse (Send, No).

BREAK

12) Drag an If activity to the space below the ReceiveDrivingRecordResponse. Highlight the If and in the Properties View Detail select Create a New Condition. Leave the Expression Language Same as Process (XPath 1.0 in BPEL 2.0). In the text box below use Code Completion (Ctrl Space) to add the condition: $DrivingRecordResponse.in/driv:numberOfTickets <= 4. (Instead of copying this expression in; use Ctrl Space, and select first the DrivingRecordResponse, then the . , then Ctrl Space and select in, the add /, then Ctrl Space and select driv:numberOfTickets, then space to show a list of operators,

and select <= and then 4).

13) Drag and Drop the Sequence Container from the Palette onto the Process canvas and drop it into the activity portion of the If. Name the new Sequence OkDriverSequence. Drag the SendPolicyQuoteResponse into the OkDriverSequence.

14) Next drag and drop an Invoke activity from the Palette at the top of the OkDriverSequence. Name it SendCreditCheckRequest. In the Properties View select the CreditCheckService as the Partner Link, and select CreditCheckServiceOp as the Operation. Uncheck the Use WSDL Message Parts Mapping and select the Variable CreditCheckRequest for the input variable and CreditCheckResponse for the output variable.

15) Now add another Assign (call it AssignDataForCreditCheckRequest) before the SendCreditCheckRequest. Create a single copy operations to copy from the PolicyQuoteRequest/receivePolicyQuote/policyQuoteInfo/ssn to the SendCreditCheckRequest. Use the namespace prefix cred. When it asks to create an Initializer, select Yes.

16) Next invoke the PolicyQuoteCalculationService. Drag the Invoke activity from the Palette to the process canvas and add it in between SendCreditCheckRequest and SendPolicyQuoteResponse. Name it SendPolicyQuoteCalculationRequest. In the Properties View select the PolicyQuoteCalculationService as the Partner Link, and select PolicyQuoteCalculationServiceOp as the Operation. Uncheck the Use WSDL Message Parts Mapping and select the Variable PolicyQuoteCalculationRequest for the input variable and PolicyQuoteCalculationResponse for the output variable.

17) Add an Assign activity before SendPolicyQuoteCalculationRequest. Name it AssignDataForCalculateQuote. Create several new copy operations to copy the following. When asked to create an Initializer, select Yes.

From	To
PolicyQuoteRequest/receivePolicyQuote/requestDate	PolicyQuoteCalculationRequest/calculatePolicyQuote/requestDate
PolicyQuoteRequest/receivePolicyQuote/policyQuoteInfo/policyType	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/policyType
PolicyQuoteRequest/receivePolicyQuote/policyQuoteInfo/vehicleYear	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/vehicleYear
PolicyQuoteRequest/receivePolicyQuote/policyQuoteInfo/driverName	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/driverName
PolicyQuoteRequest/receivePolicyQuote/policyQuoteInfo/ssn	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/ssn
PolicyQuoteRequest/receivePolicyQuote/policyQuoteInfo/dlNumber	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/dlNumber
PolicyQuoteRequest/receivePolicyQuote/policyQuoteInfo/age	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/age
DrvingRecordResponse/numberOfAccidents	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/numberOfAccidents
DrvingRecordResponse/numberOfTickets	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/numberOfTickets
CreditCheckResponse/score	PolicyQuoteCalculationRequest/calculatePolicyQuote/policyQuoteInfo/creditScore

Then select Fixed Value on the From, enter a value calculatePolicyQuote, and select PolicyQuoteCalculationRequest/calculatePolicyQuote/task on the To. Then use the Move Up button to move this copy operation to the third copy operation (after the requestDate). Note it is important that the copy operations be performed in the sequence specified.

18) Next invoke the PolicyQuoteEntityService. Drag the Invoke activity from the Palette to the process canvas and add it in between SendPolicyQuoteCalculationRequest and SendPolicyQuoteResponse. Name it SendPolicyQuoteEntityRequest. In the Properties View select the PolicyQuoteEntityService as the Partner Link, and select PolicyQuoteEntityServiceOp as the Operation. Uncheck the Use WSDL Message Parts Mapping and select the Variable PolicyQuoteEntityRequest for the input variable and PolicyQuoteEntityResponse for the output variable.

19) Add an Assign activity before SendPolicyQuoteEntityRequest. Name it AssignDataForPolicyQuoteEntityRequest. Create several new copy operations to copy the following data. When asked to create an Initializer, select Yes.

From	To
PolicyQuoteCalculationResponse/calculatePolicyQuote/requestDate	PolicyQuoteEntityResponse/createPolicyQuote/requestDate
PolicyQuoteCalculationResponse/calculatePolicyQuote/policyQuoteInfo/policyType	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/policyType
PolicyQuoteCalculationResponse/calculatePolicyQuote/policyQuoteInfo/vehicalYear	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/vehicalYear
PolicyQuoteCalculationResponse/calculatePolicyQuote/policyQuoteInfo/driverName	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/driverName
PolicyQuoteCalculationResponse/calculatePolicyQuote/policyQuoteInfo/ssn	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/ssn
PolicyQuoteCalculationResponse/calculatePolicyQuote/policyQuoteInfo/dlNumber	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/dlNumber
PolicyQuoteCalculationResponse/calculatePolicyQuote/policyQuoteInfo/age	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/age
PolicyQuoteCalculationResponse/calculatePolicyQuote/numberOfAccidents	PolicyQuoteEntityRequest/createPolicyQuote/numberOfAccidents
PolicyQuoteCalculationResponse/calculatePolicyQuote/numberOfTickets	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/numberOfTickets
PolicyQuoteCalculationResponse/calculatePolicyQuote/creditScore	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/creditScore
PolicyQuoteCalculationResponse/calculatePolicyQuote/price	PolicyQuoteEntityRequest/createPolicyQuote/policyQuoteInfo/price

Then select Fixed Value on the From, enter a value createPolicyQuote, and select PolicyQuoteEntityRequest/createPolicyQuote/task on the To. Then use the Move Up button to move this copy operation to the third copy operation. Note it is important that the copy operations be performed in the sequence specified.

20) Add an Assign activity before SendPolicyQuoteResponse. Name it AssignDataForResponse. Create several new copy operations to copy:

From	To
PolicyQuoteEntityResponse/createPolicyQuote/requestDate	PolicyQuoteResponse/receivePolicyQuote/requestDate
PolicyQuoteEntityResponse/createPolicyQuote/policyQuoteInfo	PolicyQuoteResponse/receivePolicyQuote/policyQuoteInfo

Then select Fixed Value on the From, enter a value receivePolicyQuote, and select PolicyQuoteResponse/receivePolicyQuote/task on the To.

BREAK

Note - if time permits lab can be expanded to include else branch for handling numberOfTickets > 4 condition.

BREAK

Deployment

21) Create a new ODE Deployment descriptor. File -> New -> Other -> BPEL 2.0 -> Apache ODE Deployment Descriptor. Set the Partner Links for the Inbound interfaces, and then for the Outbound Interfaces.

21) Copy CreditCheckService.esb, DrivingRecordService.esb from resources folder to the server /default/deploy directory. If you have not performed the preceding labs, then also copy policyquote-queue-service.xml, PolicyQuoteCalculationService.esb, PolicyQuoteEntity.jar and PolicyQuoteEntityService.esb to the server /default/deploy directory.

22) Start the server if it is not already started.

23) Deploy the PolicyQuoteProcess to the server using Full Publish option in the Servers view. Note Full Publish creates a new version of the process definition in the server each time you deploy.

24) Use SoapUI to create two SoapUI projects, one to consume the PolicyQuoteProcessService.wsdl and the other DrivingRecordCallbackService.wsdl (two SoapUI projects), and populate the respective request messages using the sample messages. There WSDL files can be found in the server/default/data/wsdl/PolicyQuoteProcessService folder.

25) Send the policyQuoteRequest message, wait about five seconds, the send the drivingRecordResponse.

Update Repository

26) Open the PolicyQuoteProcessService project in JBDS. Add the PolicyQuoteProcessService bpel process definition to Guvnor in the PolicyQuoteProcessService package.

27) Open in Guvnor, you will find it in the Other assets folder (not sure why it did not wind up under Processes). Add the categories Services/Orchestrated Services/PolicyQuoteProcessService and BPEL Process Definition. Change the Status to Service Development and Save and close.

28) Next add the PolicyQuoteProcessArtifacts.wsdl, PolicyQuoteProcessService.wsdl, and DrivingRecordCallbackService.wsdl to the PolicyQuoteProcessService in Guvnor. Open these assets in Guvnor and assign categories Services/Orchestrated Services/PolicyQuoteProcessService and WSDL. Change the Status to Service Development and Save and close. Note adding WSDL files will cause Eclipse to report errors; these can be ignored.

29) AddCreditCheckService.esb and DrivingRecordService.esbl to ServicesGlobal by using the Create New -> File in Guvnor. Add the category DeploymentArchive and the Status Deployed.

30) Add policyquote-destinations.xml from labs/resources to ServicesGlobal by using the Create New -> File in Guvnor. Add the category JMS Destinations and the Status Deployed.

31) Export the repository.