JBoss Overlord CDL 1.0-M1

Samples Guide

by Gary Brown and Jeff Yu

Overview

The Overlord CDL distribution contains two main types of functionality:

1. the ability to validate executing services against a choreography description (an example of runtime governance).
2. the ability to build an ESB using 'conversation aware' actions which can be checked for conformance against a choreography description (an example of design time governance).

This document will describe the samples available to demonstrate each aspect of the functionality.

Further information about configuring the runtime validation of services against a choreography can be found in the **UserGuide**. Information regarding the conversation aware ESB actions, and how to use them in conjunction with conformance checking against a choreography description, can also be found in the **UserGuide**.

Note

Before attempting to install and run these examples, you must follow the instructions in the **"Installation" Chapter** of the **Getting Started Guide** regarding installing Overlord CDL into a JBossAS environment, and importing the samples into the Eclipse environment.

CDL Validator

2.1. Trailblazer Example

This example can be found in the `trailblazer` folder, which contains an enhanced version of the trailblazer example found in the JBossESB distribution. See the TrailBlazer Guide in the JBossESB distribution (`$JBossESB/docs/samples/TBGuide.pdf`) for more information about the example. The main changes are the introduction of a File Based Bank, and modifications to the message structures to enable a consistent conversation id to be carried with the messages.

Note

The choreography description for the Trailblazer example can be found in the *trailblazer-models* project in the Eclipse environment. If the project has not yet been imported, then please refer to the instructions in the *Getting Started Guide*.

You can open the choreography for the trailblazer (trailblazer.cdm) and also a scenario representing a valid transaction associated with the choreography (LoanRequest.scn). In the choreography description editor, view the "Choreography Flows" tab to see the structure of the process.

To simulate the scenario against the choreography, to ensure that the choreography correctly caters for the valid business scenario, the user should press the green 'play' button in the toolbar, associated with the Scenario Editor.

1. Update the `$JBossAS/server/default/deploy/jbossesb.sar/jbossesb-properties.xml` file, in the section entitled "transports" and specify all of the SMTP mail server settings for your environment.

2. Update the `trailblazer/trailblazer.properties`

   Update the file.bank.monitored.directory and file.output.directory properties. These are folders used by the File Based Bank, and are set to `/tmp/input` and `/tmp/output` by default.

3. Update the `trailblazer/esb/conf/jboss-esb.xml`

   There is a *fs-provider* block, update the directory attribute value to be the same as the file.output.directory value in `trailblazer.properties` file.

4. Start the JBossAS server

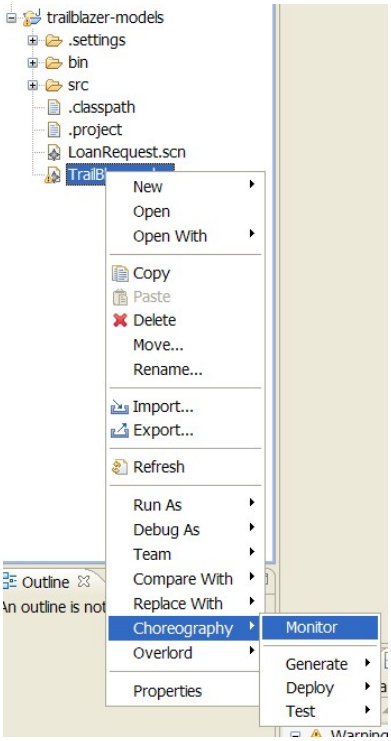
5. From the `trailblazer` folder, execute the command to start the ESB: **ant deploy**

   this should deploy the ESB and WAR files to your JBoss AS `server/default`.

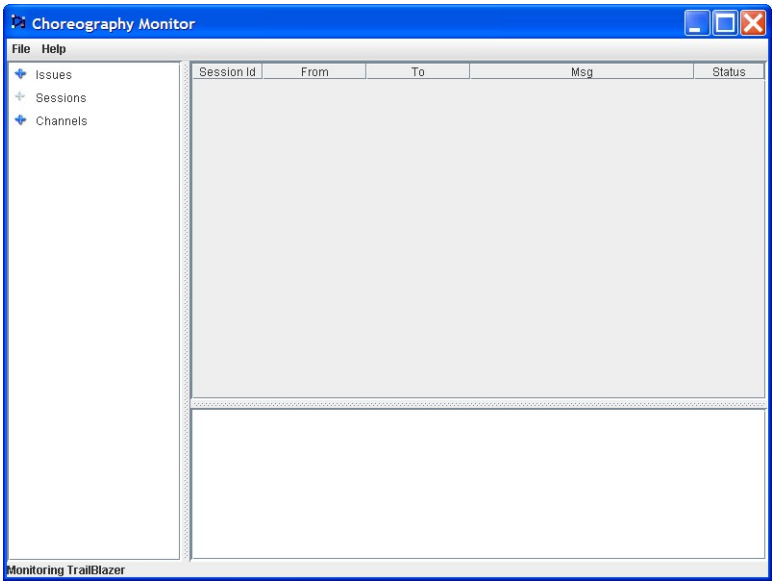
6. From the `trailblazer/banks` folder, execute the command to start the JMS Bank service: **ant runJMSBank**.

7. From the `trailblazer/banks` folder, execute the command to start the JMS Bank service: **ant runFileBank**.

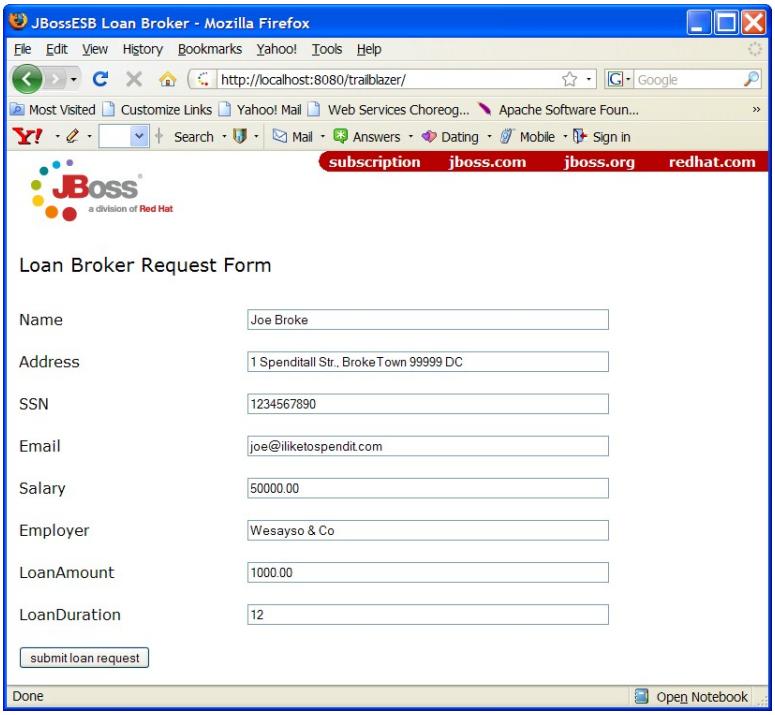
8. In the Eclipse environment, select the popup menu associated with the `trailblazer.cdm` file, and choose the *Choreography->Monitor* menu item.



Wait for the monitor window to start, and indicate that the choreography is being monitored, shown in the status line at the bottom of the window.



9. Start a browser and enter the URL: localhost:8080/trailblazer.



10. Now you can submit quotes, You will see either a loan request rejected (single email) because the score is less than 4, or two emails (one from JMS bank and one from FileBased bank) with valid quotes. When entering subsequent quotes, make sure that the quote reference is updated, so that each session has a unique id.

To demonstrate what occurs when the implementation deviates from the expected behaviour as defined in the choreography description, try the following steps:

1. Run the ant task **ant deploy-error-client** to redeploy the trailblaizer example.
2. Run the commands from step 6 above.

The above steps show how changing the service implementation without updating a choreography can result in behavioural validation errors being detected.

What is changed when we run ant deploy-error-client

Compared to command of **ant deploy**, basically, we have just updated the following code in **$Overlord/samples/trailblazer/client/src/org/jboss/soa/esb/samples/trailblazer/loanbroker/LoanBroker.java** file.

In the following code within the **processLoanRequest** method, we've changed the "4" to "7"

```
        //step 2 - check if score is acceptable
        if (score >= 4) {
```

Issue further loan requests, remembering to change the quote reference each time, until a Credit Check result of between 4 and 6 inclusive occurs, which will result in an out of sequence message being reported (in red) to the Choreography Monitor

Note

It is currently a requirement that the choreography used within the Choreography Monitor is the same as the description used to locally monitor the services (i.e. within the overlord-cdl-validator.esb/models directory).

CDL Conformance

There are two examples to demonstrate the conversation aware ESB actions, and the conformance checking against a choreography. These are `purchasing`, a simple customer/supplier example with two associated Eclipse projects (`purchasing-store` and `purchasing-models`), and `brokerage` which extends the purchasing example through the introduction of a broker that mediates between potentially multiple suppliers to find the best deal, defined within three Eclipse projects (`brokerage-broker`, `brokerage-supplier` and `brokerage-models`).

These examples make use of a common *Credit Agency* service, defined within the `common-creditAgency` Eclipse project, and are executed through the use of client applications defined in the `${OverlordCDL}/samples/client` folder.

Warning

At the moment, the conversation aware ESB runtime doesn't support the hot-deploy. That means if you update the business pojo class, such as **$creditAgency/src/main/com/acme/services/creditAgency/CreditAgencyPurchase.java** file, You need to re-deploy it, and then **restart the server** to cause it to take effect. This issue has been tracked under https://jira.jboss.org/jira/browse/SOAG-72. Will be fixed in the next release.

3.1. Purchasing Example

The purchasing example describes the interactions between a Buyer, Store and Credit Agency. The flow for this example would be:

- Buyer send a 'buy' request to Store
- Store send a 'credit check' request to the Credit Agency.
- If the Credit Agency returns a successful message, then the Store will send a 'BuyConfirmed' to user.
- If the Credit Agency returns a failed message, then the Store will send a 'BuyFailed' to user.

To check conformance, we need to refer to the model and service implementation projects in the Eclipse environment. The `purchasing-models` project contains the CDL used to perform conformance checking on the `src/main/resources/META-INF/jboss-esb.xml` files within the other projects. A full explanation of the conversation aware ESB actions can be found in the *Conversational Aware ESB* section of the *User Guide* in the `docs` folder.

To provide a simple demonstration of the conformance checking:

1. Double click on `purchasing-store/src/main/resources/META-INF/jboss-esb.xml`
2. Scroll down to the second action, within the first service. This represents a *ReceiveMessageAction* and has a property defining the message type to be received.

3. Edit the 'messageType' property value, e.g. by adding an 'X' to the end of the value.

4. Then save the file. This should result in an error being generated, complaining about a type mismatch.

The information regarding the expected message type is obtained from the choreography description in the `purchasing-models` project. To identify the precise interaction within the choreography that this error relates to, select the context menu associated with the error and choose the Quick Fix menu item. This will display a dialog with a list of fixes, select the *Show referenced description* option and press OK. This will cause the relevant interaction within the choreography description to be displayed.

Another Quick Fix option associated with this error is *Update from Referenced Description*. By selecting this option, you will notice that the message type is changed back to the value without the 'X'.

3.1.1. Running the Example

1. First step is to install the ESB services. (Presumely the JBoss ESB server started already) In a command window,
   - Go to the `$Overlord/samples/purchasing/store` folder and execute **ant deploy**
   - Go to the $Overlord/samples/common/creditAgency folder and execute **ant deploy**
2. Go to the `$Overlord/samples/client` folder and execute **ant runPurchasingClient**, which will send a 'BuyRequest' message to the Store, which will then perform the credit check before returning a response to the client.

To see a different response from the client, change the *isCreditValid* method on the *CreditAgencyPurchase* class to return *false*, within the `common/creditAgency` ESB service implementation, and then re-deploy the Credit Agency service. Then when the client is re-run, a 'BuyFailed' message will be returned.

Tip

You can undeploy the corresponding esb artifact by through command `ant undeploy` in its directory, such as $Overlord/samples/purchasing/store

3.2. Brokerage Example

The brokerage example describes the interactions between a Customer, Broker, Supplier and Credit Agency. The flow for this example would be:

- Customer sends an 'enquiry' request to Broker
- Broker sends the request to one or more Suppliers concurrently
- When all of the quote responses have been received, or a timeout expires, the available information is returned to the Customer

- Customer decides whether to:
  - Cancel the transaction, or
  - Send a 'buy' request to the Broker
- If a 'buy' request is received by the Broker, it will send a 'credit check' request to the Credit Agency
- If the Credit Agency returns a successful message, then the Broker sends a 'buy' request to the Supplier selected by the Customer (in the 'buy' request), followed by a confirmation back to the Customer
- If the Credit Agency returns a failed message, then the Broker will inform the Customer

To check conformance, we need to refer to the model and service implementation projects in the Eclipse environment. The `brokerage-models` project contains the CDL used to perform conformance checking on the `src/main/resources/META-INF/jboss-esb.xml` files within the other brokerage projects. A full explanation of the conversation aware ESB actions can be found in the *Conversational Aware ESB* section of the *User Guide* in the `docs` folder.

To provide a simple demonstration of the conformance checking:

1. Double click on `brokerage-broker/src/main/resources/META-INF/jboss-esb.xml`
2. Scroll down to the second action, within the first service. This represents a *ReceiveMessageAction* and has a property defining the message type to be received.
3. Edit the 'messageType' property value, e.g. by adding an 'X' to the end of the value.
4. Then save the file. This should result in an error being generated, complaining about a type mismatch.

The information regarding the expected message type is obtained from the choreography description in the `brokerage-models` project. To identify the precise interaction within the choreography that this error relates to, select the context menu associated with the error and choose the Quick Fix menu item. This will display a dialog with a list of fixes, select the *Show referenced description* option and press OK. This will cause the relevant interaction within the choreography description to be displayed.

3.2.1. Running the Example

1. First step is to install the ESB services (Presumely the JBoss ESB server started already) In a command window,
   - Go to the `$Overlord/samples/brokerage/supplier` folder and execute **ant deploy**
   - Go to the `$Overlord/samples/brokerage/broker` folder and execute **ant deploy**
   - Go to the `$Overlord/samples/common/creditAgency` folder and execute **ant deploy**
2. Go to the `$Overlord/samples/client` folder and execute **ant runBrokerageClient**, which will initially send an 'enquiry' message to the Broker, which will communicate with the set of Suppliers to

obtain the best quote. The client will then send a 'buy' request, which will result in the Broker performing a credit check before returning a response to the client.

Tip

You can undeploy the corresponding esb artifact by through command `ant undeploy` in its directory, such as $Overlord/samples/brokerage/supplier