BPEL Monitor and Tracker Installation

(Logicoy OpenESB 2.3.1)

ABOUT LOGICOY

- LogiCoy incorporated in February 2009.
- Headquarters in Los Angeles, California with 24/7 offices in US, U.K., and India, is a Global Information Technology and Services company providing products and services in integration and middleware across various industries including financial, healthcare, telecommunications, manufacturing, and government.
- We are the original architects, developers and engineering managers and directors for Seebeyond eGate, ICAN, Sun Java CAPS, Mural MDM,Open/GlassFish ESB products from inception to end and know the code inside and out.
- We are the primary committers of the OpenESB codebase in the OpebESB Open source repository and actively working on the enhancents and new versions of OpenESB.
- We have augmented our product development with very seasoned SOA/EAI Technical Leads and Architects for integration solutions with field implementation experience using Sun Java CAPS, GlassFish ESB, eGate and other integration tools in healthcare integration projects.

Contact us at: info@logicoy.com

ABOUT THE AUTHOR

Param is an OpenESB/Java developer in Logicoy software technology. He has contributed in Open ESB 2.3/2.3.1/2.3.1-U2 Logicoy and community releases in components bug/enhancements and in Driver test case run. He has modified and tested various test cases on JMSBC, FTPBC, SMTPBC, BPELSE, POJOSE, etc. He is having experience in OpenESB Application development, working with BPELSE in real time clients projects. He is having good experience in Java EE web application and web services.

TABLE OF CONTENTS

INTRODUCTION ABOUT MESSAGE TRACKING AND MONITORING

During the ESB development and troubleshooting you might want to track and monitor the messages coming in and out from the BPEL SE. This feature of the ESB is helping developers to track messages, each and every small activities performance and its status. Application is developed in such a way that gives full control to user what to track and what not. Also user can redirect the tracking traffic anytime to its own new JNDI data source. This document will help to understand the application and its easy configuration on your existing OpenESB 2.3.1 setup.

What is BPEL Monitoring?

- Monitor all bpel processes deployments.
- Monitor each service units/assemblies.
- Monitor activities in bpel flow.
- Monitor bpel process variables and scopes.
- Monitor bpel instances [Messages].

What is message tracking?

- Track each and every messages coming in-out between any component of ESB.
- Stores message exchange ids, endpoints, bpel instances etc.
- Can track message payload along with messages if configured.
- Track across any component [Not bound to sun-bpel-engine]
- Track messages and store results in any DB sources.
- [Configurable JNDI in casa file]
    - <message-tracking xmlns="http://www.sun.com/jbi/qos/message-tracking" externalize-payload="false" jndi="jdbc/mysqlLocal" store-payload="true" tracking="true"/>

.

Directory Information

You will get this application as a zip achieve. This archieve contains following directories. This section will describe you the details for each directories and their purpose.

- component-installers: Contains installers for required binding components and service engines.
- DB-CreationScripts: Database creation script need to be executed before running monitoring and tracking application.
- Web-App : Web application archive for installing the web application in your glassfish server.
- tracking-jar: Message tracking jar need to be copy in glassfish target.

DEPLOYMENT INSTRUCTION FOR BPEL MONITORING AND TRACKING

You can deploy the monitoring and tracking feature in an existing openesb 2.3.1 installation by simply following below steps.

STEP - 1 - DATABASE CHANGES:

Running SQL Script: [DB-CreationScripts\MYSQL_MONITOR_AND_TRACK.sql]

For BPEL monitoring you need to create required tables in your database. For best performance use Mysql DB.

SQL > source MYSQL_MONITOR_AND_TRACK.sql

Or you can use any tool to run the sql file. This execution will create following required tables

csf_logger_log,

csf_logger_log_mapping,

monitorbpelactivity,

monitorbpelactivityvariable,

monitorbpelinstance,

monitorbpelprocess,

monitorbpelvariable,

monitornmproperty,

monitornmsimpleproperty,

monitorsimplevariable,

monitorvariableattachment,

serviceunit

Make sure all tables are created properly without any errors.

Create JNDI and connection pool for configured DB in glassfish server:

For BPEL monitoring (Not for message tracking) you have to create the two JNDI, **jdbc/bpelseNonXA** and **jdbc/bpelseXA**. These two JNDI configurations need to be created by the connection pool pointing to the DB where we have required monitoring and tracking tables. For this you can use ESB admin interface. As this is basic configuration we are not giving much details for doing this setup. In case of any help please contact logicoy support.

STEP - 2 - TURN ON THE BPEL SE MONITORING

You need to turn on the BPEL monitoring in installed bpelse on required target.

A) From Admin Console

Change the properties for bpelse to enable monitoring in admin console.

WEB Console - > JBI > Components > sun-bpel-engine > Configuration ---> Monitoring - Manage Monitoring

B) From OpenESB IDE

1) Right click on bpelse from your openesb ide glassfish running server.

2) Check the check-box for monitoring

That's it!!! BPELSE is now ready to monitor your processes.

STEP - 3 - DEPLOYMENT INSTRUCTION FOR MESSAGE TRACKING

For Message tracking you need to put the required jar file [messageTracer-1.0-SNAPSHOT.jar] into your glassfish classpath. Simply copy the jar to glassfish lib folder so that it can be shared by all instances and targets.

STEP - 4 - UPDATE THE BINDING COMPONENTS AND SERVICE ASSEMBLY TO USE THE MESSAGE TRACKING

Please get the required BCs and SEs installer to use the message tracking. For upgrading your existing components, you can use asadmin utility or Netbeans IDE in your development box.

Congratulations!!!! Your ESB is now fully configured to work on monitoring environment.

APPLICATION DEVELOPMENT GUIDE:

Once you create the BPEL and Composite application Project for example CompApp1. You need to enable message-tracking and payload storage configuration.

To do this, you can click on qos setting from CASA editor of your ESB IDE and simply check the check-box in front of message-tracking.

NOTE:

1) Message tracking enable will just store the message exchange ids and endpoint information. It will not track the payload coming in and out from BCs and SEs. You can enable it from your qos setting.

2) You can enable message tracking in any one of the qos endpoint. It will enable for entire application as per the application design.

Once you did the configuration from CASA editor, you can see following changes in your casa file. To view this just go to the source code of casa xml file. Changes will looks like follows.

……

<connections>

<connection consumer="endpoint1" provider="endpoint2" state="unchanged">

<message-tracking xmlns="http://www.sun.com/jbi/qos/message-tracking" externalize-payload="false" store-payload="true" tracking="true"/>

</connection>

</connections>

……

MUST: Required Manual Configuration for JNDI

As you are doing the Up gradation of OpenESB2.3.1 Manually for this feature, There is no separate installer made for this enhancement. You need to do this only one manual step to configure data source.

You need to give the JNDI name for composite applications for message-tracking. As N number of application can have N number of JNDI data source where ESB will store the message payload and tracking information.

To do this you can open your application/src/conf/<app>.casa file in any editor and add/update jndi="<JNDI_NAME>". For example

...

<message-tracking xmlns="http://www.sun.com/jbi/qos/message-tracking" externalize-payload="false" jndi="jdbc/mysqlLocal" store-payload="true" tracking="true"/>

...

Now your application CompApp1 is ready to deploy with message-tracking enable.

:

KNOWN ISSUES

- With Database BC the message tracking is not working. So for now you don't need to upgrade the databasebc for this feature.
- You cannot configure the JNDI once the Composite application is deployed in server.
- BPELSE JNDI names are fixed, User cannot change it.