eXtensible Service Bus (XSB)

XSB DPWS Binding Component over EasyESB

```
== WHAT IS IT? ==
This module allows to execute the XSB DPWS Binding Component with EasyESB.

== REQUIREMENTS ==
Before you compile the project you need to install
#1 Java 6
#2 Maven 3 (http://maven.apache.org/download.html)

== CONFIGURING IT ==
1 download sources:
connectors/clientServer/WebServices/DPWSBindingComponent/easyesbjmeds

2. Add this repository to your settings.xml
<repository>
       <id>choreos-petalsLink</id>
       <name>choreos maven repository</name>
       <url>http://maven.petalslink.com/repo</url>
</repository>

3. Execute mvn install to obtain the binding, or mvn install -Pdistrib
to obtain a easyESB distribution with the DPWS Binding Component.

== RUNNING IT ==
run target/easyesbjmeds/bin/startup.bat to start the easyESB distribution
with the DPWS Binding Component.

== VERIFYING IF IT WORKS ==
You can verify the binding component by running 2 scenarios
-> 2 DPWS applications have a one-way exchange through the XSB Binding Components.
The first application sends a notification that is received by the second application
as a one-way invocation.
-> 2 DPWS applications have a two-way exchange through the XSB Binding Components.
The first application sends a solicit-response that is received by the second
application as a two-way invocation.

Running the first scenario:
1) start the application that receive the one-way invocation:
java -jar Easyesbjmeds-1.0-SNAPSHOT-OneWaySystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the DPWS Binding Component
```

```
4) deploy the corresponding service unit:
src/test/resources/oneWay/UnitOneWay.xml. DPWSDefinitionOneWay.xml
should be in the same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a DPWS Binding Component
6) deploy the service unit of the system that sends the notification:
src/test/resources/notification/UnitNotification.xml.
7) start the application that sends the notification:
java -jar Easyesbjmeds-1.0-SNAPSHOT-NotificationSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification
10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2

Running the second scenario:
1) start the application that receive the two-way invocation:
java -jar Easyesbjmeds-1.0-SNAPSHOT-TwoWaySystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the DPWS Binding Component
4) deploy the corresponding service unit:
src/test/resources/invoke/UnitTwoWay.xml. DPWSDefinitionTwoWay.xml
should be in the same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a DPWS Binding Component
6) deploy the service unit of the system that sends the notification:
src/test/resources/solicit/UnitSolicit.xml.
7) start the application that sends the notification:
java -jar Easyesbjmeds-1.0-SNAPSHOT-SolicitSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification
10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2

== AUTOMATED TESTS ==
The preceding scenarios are implemented with integration tests with JUnit. So
remove skipping test configuration for surefire and then
execute mvn test to launch them in a automated testing.
```

XSB JMS Binding Component over EasyESB

```
== WHAT IS IT? ==
This module allows to execute the XSB JMS Binding Component with EasyESB.

== REQUIREMENTS ==
```

Before you compile the project you need to install
#1 Java 6
#2 Maven 3 (http://maven.apache.org/download.html)

== CONFIGURING IT ==
1 download sources:
connectors/publishSubscribe/JMS/JMSBindingComponent/easyesbjoram/

2. Add this repository to your settings.xml

```
<repository>
       <id>choreos-petalsLink</id>
       <name>choreos maven repository</name>
       <url>http://maven.petalslink.com/repo</url>
</repository>
```

3. execute mvn install to obtain the binding, or mvn install -Pdistrib to obtain a easyESB distribution with the JMS Binding Component.

== RUNNING IT ==
run target/easyesbjmeds/bin/startup.bat to start the easyESB distribution with the JMS Binding Component.

== VERIFYING IF IT WORKS ==
You can verify the binding component by running 2 scenarios
-> 2 JMS applications have a one-way exchange through the XSB Binding Components.
The first application sends a notification that is received by the second application as a one-way invocation.

Running the first scenario:
1) start the application that receive the one-way invocation:
java -jar Easyesbjmeds-1.0-SNAPSHOT-PublishSystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the JMS Binding Component
4) deploy the corresponding service unit:
src/test/resources/publish/UnitPublish.xml. JMSDefinitionPublish.xml
should be in the same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a JMS Binding Component
6) deploy the service unit of the system that sends the notification:
src/test/resources/subscribe/UnitSubscribe.xml.
7) start the application that sends the notification:
java -jar Easyesbjmeds-1.0-SNAPSHOT-SubscribeSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification

```
10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2

== AUTOMATED TESTS ==
The preceding scenarios are implemented with integration tests with JUnit. So
remove skipping test configuration for surefire and then
execute mvn test to launch them in a automated testing.
```

XSB JavaSpaces Binding Component over EasyESB

```
== WHAT IS IT? ==
This module allows to execute the XSB JavaSpaces Binding Component with EasyESB.

== REQUIREMENTS ==
Before you compile the project you need to install
#1 Java 6
#2 Maven 3 (http://maven.apache.org/download.html)
#3 Rio (http://www.rio-project.org/)
     -> download and unzip the rio tarball
         -> set RIO_HOME to the home

== CONFIGURING IT ==
1 download sources:
connectors/tupleSpaces/JavaSpaces/JSBindingComponent/easyesbjini

2. Add this repository to your settings.xml
<repository>
       <id>choreos-petalsLink</id>
       <name>choreos maven repository</name>
       <url>http://maven.petalslink.com/repo</url>
</repository>

3. execute mvn install to obtain the binding, or mvn install -Pdistrib
to obtain a easyESB distribution with the JavaSpaces Binding Component.

== RUNNING IT ==
run target/easyesbjmeds/bin/startup.bat to start the easyESB distribution
with the JavaSpaces Binding Component.

== VERIFYING IF IT WORKS ==
You can verify the binding component by running 2 scenarios
-> 2 JavaSpaces applications have a one-way exchange through the XSB Binding Components.
The first application sends a notification that is received by the second application
```

```
as a one-way invocation.

Running the first scenario:
1) start the application that receive the one-way invocation:
java -jar Easyesbjmeds-1.0-SNAPSHOT-OutSystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the JavaSpaces Binding Component
4) deploy the corresponding service unit:
src/test/resources/out/UnitOut.xml. JMSDefinitionOut.xml
should be in the same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a JavaSpaces Binding Component
6) deploy the service unit of the system that sends the notification:
src/test/resources/register/UnitRegister.xml.
7) start the application that sends the notification:
java -jar Easyesbjmeds-1.0-SNAPSHOT-RegisterSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification
10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2

== AUTOMATED TESTS ==
The preceding scenarios are implemented with integration tests with JUnit. So
remove skipping test configuration for surefire and then
execute mvn test to launch them in a automated testing.
```

Georgios Bouloukakis
georgios.bouloukakis@inria.fr