Messaging for a Dynamic World

Mobeon Internal

Approved: K Selin	No: 3/BTS-MAS0001 Uen

Copyright Mobeon AB All rights reserved	Author: Kenneth Selin Title: BTS – Execution Engine	Version: B Date: 2008-08-05	1/4

BTS – Execution Engine

Content

History

Version	Date	Adjustments
A	2006-10-05	First revision (ERMKESE)
B	2008-08-05	Updated info that EE tests work best under IntelliJ (ekensel). Minor changes.

1 Introduction

Due to historical reasons, the automated basic tests have been best maintained when run from IntelliJ, and this is the recommended way to run the basic tests.

This document describes basic test information for Execution Engine. It documents the test tools needed, how to set up the environment for various tests, and the procedure for testing. It also specifies the test cases for basic test.

After reading this document, you will know how to set up, execute and document a basic test for this product.

Note: if you choose to renumber the test cases in this document, you will need to renumber also the test cases in the BTR accordingly.

2 Test Tools

2.1 JUnit

The test cases for EE are based on JUnit, which is a simple framework to write repeatable tests. See ref. [1].

2.2 JMock

JMock is used for some test cases for stubbing various java interfaces involved in some of the tests. See ref. [2].

3 Test Environment

In order to run the basic tests in UNIX, the following environment variables must be set:

- JAVA_HOME pointing to a Java 1.5 installation
- COBERTURA_HOME pointing to a Cobertura installation

This paragraph is intentionally left blank.

4 Test Execution

4.1 Test location

The automated test cases can be found here: /vobs/ipms/mas/execution_engine/test

4.2 All tests

The easiest way to run all tests is to use ant, as follows from the execution engine directory in UNIX:

ant runtest

You can also run this ant target from IntelliJ.

4.3 Class tests

EE has tests verifying the behavior of individual classes. What each test checks is individual per tested class. If you want to find out exactly what the tests do, read the test cases (many have a small javadoc explaining the test case).

You could run the tests individually from IntelliJ by locating the test class, rightclicking on it and select "Run".

4.4 Application tests

EE has tests running applications written in CCXML/VXML/ECMA and verifying the behaviour. If you want to find out exactly what the tests do, read the test cases (many have a small javadoc explaining the test case). These tests are located below the runapp directory.

EE has tests verifying the behavior of individual classes. What each test checks is individual per tested class. If you want to find out exactly what the tests do, read the test cases (many have a small javadoc explaining the test case).

You could run the tests individually from IntelliJ by locating the test class, rightclicking on it and select "Run".

If you want to debug an application test case, use these settings to the virtual machine (otherwise your test may be killed by the test framework):

```
-Dcom.mobeon.junit.runapp.test=VXMLGotoTag1 -
Dcom.mobeon.junit.runapp.timeout=0 -
Dcom.mobeon.junit.runapp.callmanagerwaittime=3600000
```

This example would test the method test VXMLGotoTag1.

4.5 Coverage

Test code coverage by executing:

```
ant runcoverage
```

5 Test Cases

This paragraph is intentionally left blank.

6 Reporting Test Results

This paragraph is intentionally left blank.

7 References

[1] JUnit
www.junit.org

[2] JMock
http://www.jmock.org/

8 Terminology

Term	Explanation