BPMIntegrationGuide

From jBPM

The target audience for this guide are developers who wish to provide an implementation of the API and verify its correctness through the CTS.

Contents

[hide]

1 Integrating the API
2 Integrating the CTS
3 Excluding CTS Test
4 Implementing the API
5 Partial Compliance
6 Full Compliance
7 Eclipse STP Compliance
8 Release Cycle

Integrating the API

The API is a Maven artefact available from the JBoss Maven Repository

You can add dependency to the API like this

```
<dependency>
  <groupId>org.jboss.bpm</groupId>
  <artifactId>jboss-bpm-api</artifactId>
  <version>1.0.0.Alpha1</version>
  <scope>compile</scope>
</dependency>
```

Integrating the CTS

The CTS is also a Maven artefact available from the JBoss Maven Repository

You can add dependency to the CTS like this

```
<dependency>
  <groupId>org.jboss.bpm</groupId>
  <artifactId>jboss-bpm-testsuite</artifactId>
  <version>1.0.0.Alpha1</version>
  <scope>provided</scope>
  <type>zip</type>
</dependency>
```

Before Maven can execute the CTS, it must be unpacked

```
        <plugin>
          <artifactId>maven-dependency-plugin</artifactId>
          <executions>
            <execution>
              <phase>generate-sources</phase>
              <goals>
                <goal>unpack</goal>
              </goals>
              <configuration>
                <artifactItems>
                  <artifactItem>
                    <groupId>org.jboss.bpm</groupId>
                    <artifactId>jboss-bpm-testsuite</artifactId>
                    <type>zip</type>
                  </artifactItem>
                </artifactItems>
                <outputDirectory>src/test</outputDirectory>
              </configuration>
            </execution>
          </executions>
        </plugin>
```

Excluding CTS Test

Initially, you probably don't want to run the entire CTS. You can exclude CTS Test with a Maven Surefire configuration like this

```
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <configuration>
            <excludes>
              <exclude>org/jboss/bpm/**/*DescriptorTest.java</exclude>
              <exclude>org/jboss/bpm/**/*MarshallerTest.java</exclude>
              <exclude>org/jboss/bpm/**/*STPTest.java</exclude>
              <exclude>org/jboss/bpm/cts/activity/**</exclude>
              <exclude>org/jboss/bpm/cts/endevent/**</exclude>
              <exclude>org/jboss/bpm/cts/engine/**</exclude>
              <exclude>org/jboss/bpm/cts/executioncontext/**</exclude>
              <exclude>org/jboss/bpm/cts/gateway/**</exclude>
              <exclude>org/jboss/bpm/cts/process/**</exclude>
              <exclude>org/jboss/bpm/cts/processmanager/**</exclude>
              <exclude>org/jboss/bpm/cts/signalmanager/**</exclude>
              <exclude>org/jboss/bpm/cts/startevent/**</exclude>
              <exclude>org/jboss/bpm/cts/task/**</exclude>
              <exclude>org/jboss/bpm/pattern/control/exclusivechoice/**</exclude>
              <exclude>org/jboss/bpm/pattern/control/multichoice/**</exclude>
              <exclude>org/jboss/bpm/pattern/control/parallelsplit/**</exclude>
              <exclude>org/jboss/bpm/pattern/control/simplemerge/**</exclude>
              <exclude>org/jboss/bpm/pattern/control/synchronization/**</exclude>
              <exclude>org/jboss/bpm/pattern/data/**</exclude>
            </excludes>
          </configuration>
        </plugin>
```

Note, the above excludes everything except the most trivial CTS test, which is

```
  Running org.jboss.bpm.pattern.control.sequence.SequenceTest
  Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.455 sec
```

Implementing the API

Actually not all interfaces from the API must be implemented at once. To get an idea what the CTS is actually using you can run

```
[tdiesler@tdvaio java]$ find . -name "*.java" | xargs grep -h  "import org.jboss.bpm" | sort -u
import org.jboss.bpm.client.MessageManager;
import org.jboss.bpm.client.ProcessEngine;
import org.jboss.bpm.client.ProcessManager;
import org.jboss.bpm.client.SignalListener;
import org.jboss.bpm.client.SignalManager;
import org.jboss.bpm.EngineShutdownException;
import org.jboss.bpm.InvalidProcessException;
import org.jboss.bpm.model.Assignment;
import org.jboss.bpm.model.Assignment.AssignTime;
import org.jboss.bpm.model.EventBuilder;
import org.jboss.bpm.model.EventDetail;
import org.jboss.bpm.model.EventDetail.EventDetailType;
import org.jboss.bpm.model.Expression;
import org.jboss.bpm.model.Expression.ExpressionLanguage;
import org.jboss.bpm.model.Gateway;
import org.jboss.bpm.model.GatewayBuilder;
import org.jboss.bpm.model.Gateway.GatewayType;
import org.jboss.bpm.model.Message;
import org.jboss.bpm.model.MessageBuilder;
import org.jboss.bpm.model.MessageBuilderFactory;
import org.jboss.bpm.model.Process;
import org.jboss.bpm.model.ProcessBuilder;
import org.jboss.bpm.model.ProcessBuilderFactory;
import org.jboss.bpm.model.Signal;
import org.jboss.bpm.model.Signal.SignalType;
import org.jboss.bpm.model.TaskBuilder;
import org.jboss.bpm.model.Task.TaskType;
import org.jboss.bpm.runtime.BasicAttachments;
import org.jboss.bpm.runtime.BasicExecutionContext;
import org.jboss.bpm.runtime.ExecutionContext;
import org.jboss.bpm.runtime.ExecutionHandler;
import org.jboss.bpm.runtime.Token;
import org.jboss.bpm.test.DefaultEngineTestCase;
```

Some of the above are classes or enums and don't have to be implemented either.

It is good practice to stub out an implementation completely and throw the NotImplementedException for methods that are not implemented yet. The NotImplementedException conveniently takes a JIRA issue number as one of the parameters. In that way you can easily identify what still needs to be done.

Here is a sample implementation of the EventBuilder

```
public class EventBuilderImpl extends ProcessBuilderImpl implements EventBuilder
{
  public EventBuilderImpl(ProcessImpl proc, FlowObjectImpl flowObject)
  {
    super(proc, flowObject);
  }

  public EventBuilder addEventDetail(EventDetailType detailType)
  {
    throw new NotImplementedException("JBPM-1705", "jBPM3 implementation of the API");
  }

  public EventBuilder addMessageRef(String msgName)
  {
    throw new NotImplementedException("JBPM-1705", "jBPM3 implementation of the API");
  }

  public EventBuilder addSignalRef(SignalType signalType, String signalMessage)
  {
    throw new NotImplementedException("JBPM-1705", "jBPM3 implementation of the API");
  }
}
```

Partial Compliance

An implementation would be partially compliant with a given API version if it passes all of the CTS tests, except for the various descriptor tests

```
          <excludes>
            <exclude>org/jboss/bpm/**/*DescriptorTest.java</exclude>
            <exclude>org/jboss/bpm/**/*MarshallerTest.java</exclude>
            <exclude>org/jboss/bpm/**/*STPTest.java</exclude>
          </excludes>
```

The remaining CTS tests will always create the Process through the ProcessBuilder and not require access to a DialectHandler implementation.

An implementation that requires more excludes than the ones above is not compliant.

Full Compliance

Every CTS test comes in three variants. For example we have

- SequenceTest
- SequenceMarshallerTest
- SequenceDescriptorTest

The SequenceTest is the super test of the other two and uses ProcessBuilder to create the Process. Passing this test would be the first level of compliance.

The SequenceMarshallerTest uses the current dialect to marshall the Process and then recreate the Process from the result. This tests the round trip though the DialectHandler to make sure a given concept can actually be represented in a given dialect.

```
public class SequenceMarshallerTest extends SequenceTest
{
  public Process getProcess() throws IOException
  {
    // Marshall the process to a string
    Process proc = super.getProcess();
    String procXML = marshallProcess(proc);

    // Recreate the process from the marshalled process
    ProcessManager procManager = ProcessManager.locateProcessManager();
    return procManager.createProcess(procXML);
  }
}
```

The SequenceDescriptorTest creates the Process from a dialect descriptor.

```
public class SequenceDescriptorTest extends SequenceTest
{
  public Process getProcess() throws IOException
  {
    URL procURL = getResourceURL("pattern/control/sequence/pattern-control-sequence-" + getDialect() + ".xml");
    ProcessManager pm = ProcessManager.locateProcessManager();
    Process proc = pm.createProcess(procURL);
    return proc;
  }
}
```

When your implementation passes the MarshallerTest it is trivial to pass the DescriptorTest as well. Simple save the marshaller result in a resource file like 'pattern-control-sequence-foo10.xml'

And set some surefire system properties like this

```
      <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <configuration>
          <systemProperties>
            <property>
              <name>jbpm.dialect</name>
              <value>foo10</value>
            </property>
            <property>
              <name>jbpm.dialect.uri</name>
              <value>urn:foo-1.0</value>
            </property>
          </systemProperties>
        </configuration>
      </plugin>
```

In this way the CTS can work with descriptor resources in various dialect formats.

Eclipse STP Compliance

Ideally, a user would not have to manually create descriptor files. Instead a tool should be used. The User Guide shows images produces by the Eclipse STP BPMN Editor . This editor produces a descriptor that is hardly human readable and does also not support many of the concepts that are already tested in the CTS.

In order to still leverage the STP output, your dialect might want to support the notion of 'includes'. Conveniently enough, the API comes with a STP DialectHandler that can create a Process from the STP output. This process could then be included in another Process created from your dialect descriptor. The resulting Process would contain the functional union from both.

For the simple SequenceTest above we would actually not need to augment the STP output.

Release Cycle

A new version of the API+CTS will be released approximately every 8 weeks. Please monitor the API roadmap and changelog for progress.

Retrieved from "http://jbpm.dyndns.org/jbpmwiki/index.php?title=BPMIntegrationGuide "