Coordinating Celtix Applications with ActiveBPEL

Table of Contents

Overview

In many situations, developers need to integrate several Web service applications into a larger application, for example, using a credit checking service, an inventory service, and a shipping service to process a sales order. While it might be possible to write an application that accepts customer input, invokes the various services, and then returns shipping and billing information to the customer, an alternative approach would be to define the process using business process execution language (BPEL) and leave the details of persistence, compensation, coordination and exception handling to a business process engine.

This application note discusses how to use the ActiveBPEL business process engine to coordinate information flow between multiple Celtix Web service applications. This note also demonstrates interoperability between Celtix and the ActiveBPEL engine, which uses Axis as its underlying Web services toolkit.

Required Software

All required software is free and readily available for download.

– JDK v1.5.0.06 (JDK v5.0 update 6) and above, available at http://java.sun.com/j2se/1.5.0/download.jsp. Celtix requires JDK v1.5 (JDK v5.0), which is also acceptable to the other products.

– Apache Ant v1.6.5 and above, available at http://ant.apache.org/.

– Apache Tomcat v5.5 and above, available at http://tomcat.apache.org/download-55.cgi.

– ActiveBPEL v2.0.0 and above, available at http://www.activebpel.org/.

– Celtix milestone 4 and above binary distribution, available at http://forge.objectweb.org/projects/celtix/.

Software Installation

Software should be installed in the same order as listed within Required Software.

JDK

Once the product has been installed, you must set the JAVA_HOME environment variable to the JDK installation directory and add the JAVA_HOME bin subdirectory to the PATH. These settings can be applied at the system level or within each command window. The second approach, which is useful if you have other versions of the JDK installed on the same computer, will be illustrated in this note.

In developing this note, the JDK v1.5.0.06 was installed into the directory C:\jdk1.5.0 and the JAVA_HOME environment variable set to C:\jdk1.5.0. The directory C:\jdk1.5.0\bin was added to the PATH.

Apache Ant

Apache Ant is used to build and run Celtix applications. If desired, you can use the Celtix utility **`wsdl2java`**, and the JDK executables **`javac`** and **`java`**, directly. If this is the approach followed, you do not require an Apache Ant installation. However, it is recommended that you install Apache Ant, and this note will only describe how to build and run the Celtix components through Ant.

Once the product has been installed, you must add the Ant bin subdirectory to the PATH. This setting can be applied at the system level or within each command window.

In developing this note, Ant was installed into the directory C:\Ant\apache-ant-1.6.5. The directory C:\Ant\apache-ant-1.6.5\bin was added to the PATH.

Apache Tomcat

Once the product has been installed, you should write a script that properly sets the environment before starting the servlet container.

In developing this note, Tomcat was installed into the directory C:\Tomcat5.5. A script file, with the following content, was added to the subdirectory C:\Tomcat5.5\bin.

```
set JAVA_HOME=C:\jdk1.5.0
set PATH=%JAVA_HOME%\bin;%PATH%
set CLASSPATH=
tomcat5.exe
```

Use this script, rather than the tomcat5.exe executable directly, to start the servlet container under a suitable environment.

The installation directory, in this note C:\Tomcat5.5, is referred to as CATALINA_HOME. You will need a knowledge of the Tomcat installation directory in order to install the ActiveBPEL process engine.

ActiveBPEL

Installation instructions are available at http://www.activebpel.org/docs/install.html. In short:

- Extract the product archive into a temporary directory, which creates the subdirectory activebpel-2.0.0.
- Within the activebpel-2.0.0 subdirectory, run the install script, passing the value of CATALINA_HOME as an argument. For example,

```
install C:\Tomcat5.5
```

The installation process will copy a number of JAR files to the subdirectory CATALINA_HOME\shared\lib. The installation also creates the subdirectory CATALINA_HOME/bpr and copies several files to this location.

Celtix

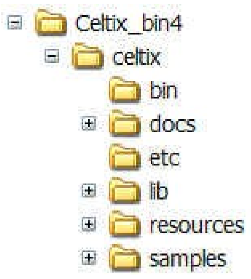
Celtix is available as either a binary or source distribution. If you download the source distribution, you will need to use Apache Maven v2.0.2 or above to build the binary files. This note will only discuss use of the Celtix binary distribution.

The Celtix binary distribution is downloaded as a single JAR file. Save this file to a local directory and then extract into an installation directory as described below.

- Save the Celtix binary distribution archive into a local directory.
- Create an installation directory, for example, C:\Celtix_bin4 for the Celtix milestone 4 binary release.
- Open a command window and confirm that the JAVA_HOME\bin directory is on the PATH. If necessary, modify the PATH.
- Move to the installation directory and use the `java` executable to extract the archive. For example,

```
java -jar <path_to_Celtix_binary_distribution_archive_file>
```

The extraction process will create the following directory hierarchy.



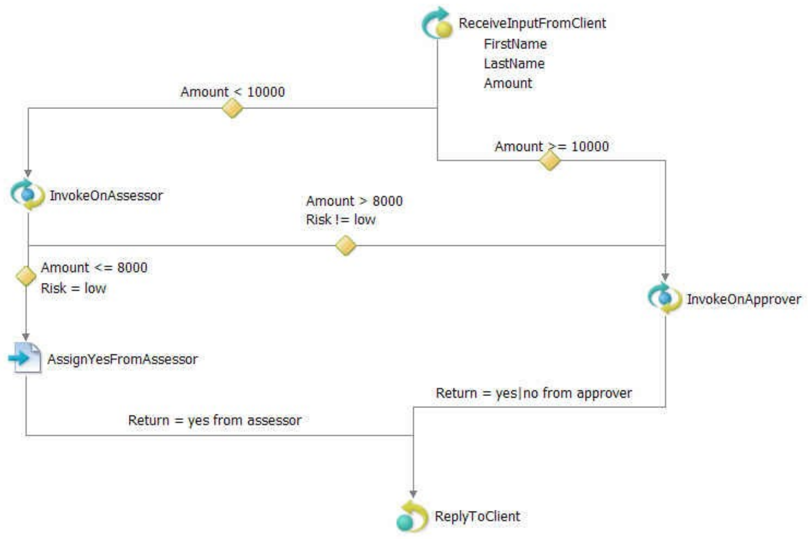
```
Celtix_bin4
  celtix
    bin
    docs
    etc
    lib
    resources
    samples
```

After installing Celtix, you will find it useful to write a script that insures that the environment is properly configured. You should then run this script in each command window before trying to work with the Celtix utilities or to compile and run a Celtix application. The following is a typical script.

```
set JAVA_HOME=C:\jdk1.5.0
set CELTIX_HOME=C:\Celtix_bin4\celtix
set ANT_HOME=C:\Ant\apache-ant-1.6.5
set PATH=%ANT_HOME%\bin;%CELTIX_HOME%\bin;%JAVA_HOME%\bin;%PATH%
set CLASSPATH=.;.\build\classes
```

The BPEL Process Diagram

Although the ActiveBPEL product does not include the functionality needed to write a BPEL file, any BPEL file that adheres to the specification can be deployed into this engine. A BPEL file may be written in any text editor or XML editor. It is helpful, however, to have a diagrammatic representation of a BPEL process. The following diagram, created with Active Endpoints ActiveWebflow™ Professional, illustrates the process developed for this note.



The ReceiveInputFromClient and ReplyToClient activities represent the request/response operation exposed by the BPEL process to clients. Later in this note, when you review the WSDL file that represents the process, you will see how these activities correspond to the `process` operation of the bpelService port type. The Invoke activities – InvokeOnAssessor and InvokeOnApprover – represent the two Web services – AssessorService and ApproverService – that are integrated into a common application by this BPEL process.

According to the business logic, if a client submits a request with an Amount less than $10000, the request is sent to the AssessorService, whereas requests with Amounts greater than or equal to $10000 are sent to the ApproverService. The AssessorService, however, assigns Amount requests that are greater than $8000 to a high risk category and forwards them to the ApproverService; requests less than or equal to $8000 are accepted by the AssessorService, which returns an appropriate message to the client. The ApproverService accepts all Amount requests less than $15000, returning an appropriate message to the client. Requests that exceed $15000 are rejected and a rejection message is returned to the client.

When you run this application, you will be able to use the BPEL Admin functionality that is part of the ActiveBPEL engine to follow the execution of each request through the process.

The BPEL Process Description

The BPEL process is described in an XML file with a `.bpel` extension. Additional information required by the BPEL process, specifically the partner link type definitions, is also added to the WSDL file used by the process engine. Copies of these files are included in the Appendix.

The ActiveBPEL Process Deployment Descriptor

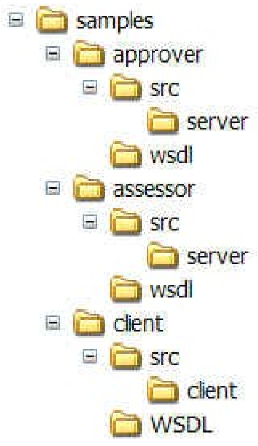
The syntax for this XML file is described at http://www.activebpel.org/docs/file_formats.html. A deployment descriptor file suitable for use in this application is included in the Appendix.

The ActiveBPEL Deployment Archive

This file is a JAR file with a `.bpr` extension. It includes the WSDL file describing the BPEL process, the process deployment descriptor, and the `.bpel` file that describes the BPEL process. The structure of this archive, and instructions for using the jar utility to assemble the file, are described at http://www.activebpel.org/docs/deploy.html.

The Celtix Client Application and Web Services

The two Celtix Web service applications are nearly identical, differing only in their port type, binding, and service definitions. There is also a Celtix client application that will send the initial request to the BPEL process. When developing these applications, it is easiest to model your directory hierarchy after the approach used in the Celtix samples. Under the subdirectory samples within your Celtix installation, create the following directory hierarchy.



The two Web service applications are an ApproverService and an AssessorService. For each service application you will place a WSDL file into the WSDL subdirectory, source code files for the server mainline and servant into the src\server subdirectory, and a build.xml file into the approver and assessor directories. For the client application, you need a WSDL file, a source code file for the client mainline, and a build.xml file. In all three applications, Ant will generate code from the WSDL file and compile the source code files into Java `.class` files. The content of each of these files is given in the Appendix. You can copy the required content into a text file and save in the appropriate directory.

Once the Celtix Web services have been written, you use their WSDL files in developing the BPEL process description. As part of this process, you define a new port type that represents the BPEL process to an external client application. Finally, you write a client application that invokes on the port type representing the BPEL process. The files needed to implement the client application are included in the Appendix.

The ApproverService

The WSDL file for this service defines the approverPT port type, which implements the operation approve. The processing logic in this operation is extremely simple. If the Amount parameter is greater than $15000, the message “no from approver” is returned by the operation; otherwise the message “yes from approver” is returned by the operation. If the request is processed by the Approver Web service, the BPEL process uses the ReplyToClient activity to return the message to the client application.

The AssessorService

The WSDL file for this service defines the assessPT port type, which implements the operation assess. The processing logic in this operation is also quite simple. If the Amount parameter is greater than $8000, the message “high,” representing the risk associated with the request, is returned by the operation; otherwise the message “low” is returned by the operation. If the risk is low, the AssignYesFromAssessor activity provides the message “yes from assessor” to the ReplyToClient activity, which returns the message to the client application. If the risk is high, the BPEL process forwards the request to the InvokeOnApprover activity.

The Client Application

The WSDL file for this application defines the bpelService port type, which implements the operation process. The client application invokes this operation, which starts an instance of the BPEL process. Note that the URL for this service's port directs the request to the servlet container running the BPEL engine.

Developing the Complete Application

Developing the application is straight-forward, although it does involve considerable attention to managing the content of the WSDL files used by the different components. The following listing summarizes the steps taken to develop this application.

- Write the ApproverService application.
- Write the AssessorService application.
- Combine these two Web service WSDL files and use the combined file in developing the BPEL process.
  - Edit the combined WSDL file, adding the port type that represents the BPEL process and the required BPEL extension content (partner link type entries).
- Write the `.bpel` file.
- Write the process deployment descriptor file.
- Create the deployment archive.
- Edit the combined WSDL file, removing the port type, binding and service definitions related to the Celtix Web services. This leaves only the <types>, <message> and BPEL process port type definitions.
  - Add a <binding> and <service> definition for the port type representing the BPEL process.
- Write a client application against the service representing the BPEL process.
- Start the Web service applications.
- Deploy the application archive into the ActiveBPEL engine.
- Run the client application.
- Review process execution through the ActiveBPEL Admin functionality.

Write the ApprovalService Application

The Appendix includes copies of the server mainline, servant, build.xml, and WSDL files that comprise this application. To recreate the application, copy these files into the proper subdirectories under the samples/approver directory and then compile by issuing `ant build` from within a command window opened to the samples/approver directory. Depending on how the host computer's environment is set up, you may need to run the environment script described in the section of this note dealing with Celtix installation (see page 3).

Write the AssessorService Application

The Appendix includes copies of the server mainline, servant, build.xml, and WSDL files that comprise this application. To recreate the application, copy these files into the proper subdirectories under the samples/approver directory and then compile by issuing `ant build` from within a command window opened to the samples/assessor directory. Depending on how the host computer's environment is set up, you may need to run the environment script described in the section of this note dealing with Celtix installation (see page 3).

The <types> and <message> entries in the WSDL file for this service are identical to ApprovalService WSDL file. The <portType>, <binding>, and <service>/<port> entries differ.

Combine the Two Web Service WSDL Files

Open both WSDL files in separate instances of a text editor. Copy the <portType>, <binding>, and <service>/<port> content from one of the files into the second file. Be certain to save the modified file to a new location. Since the modified WSDL file now includes the <portType>, <binding>, and <service>/<port> definitions for both services, it provides all the information needed to develop the `.bpel` file.

Prior to writing the `.bpel` file, a port type definition representing the BPEL process, is added to the WSDL file.

```
  <wsdl:portType name="bpelService">
    <wsdl:operation name="process">
      <wsdl:input message="tns:requestMessage"/>
      <wsdl:output message="tns:responseMessage"/>
    </wsdl:operation>
  </wsdl:portType>
```

Also, <partnerLinkType> entries, and two namespace declarations must be added the WSDL file.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="bpel_test.wsdl" ...
    xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
>
  ...

<plnk:partnerLinkType name="processLinkType">
   <plnk:role name="processor">
      <plnk:portType name="tns:bpelService"/>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="assessorLinkType">
   <plnk:role name="assessor">
      <plnk:portType name="tns:assessorPT"/>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="approverLinkType">
   <plnk:role name="approver">
      <plnk:portType name="tns:approverPT"/>
   </plnk:role>
</plnk:partnerLinkType>

</wsdl:definitions>
```

The `.bpel` file (included in the Appendix) may now be written.

Write the Process Deployment Descriptor File

The content of this file is specific to the BPEL engine. See http://www.activebpel.org/docs/file_formats.html for the specifics of writing this file for the ActiveBPEL engine. The Appendix includes a file suitable for use with the ActiveBPEL engine.

Create the Deployment Archive

This process is also specific to the BPEL engine. See http://www.activebpel.org/docs/deploy.html for the specifics of preparing the ActiveBPEL archive.

Edit the Combined WSDL File

So far the application includes three WSDL files: one each for the ApprovalService and ApprovalService, and a combined file used by the BPEL process. The combined file includes three port type definitions: approverPT, assessorPT, and bpelService. There are also <binding> and <service>/<port> definitions for the approverPT and assessorPT.

The client application does not require any knowledge about the ApproverService and AssessorService or the partner link type declarations. Therefore, the port type, binding, and service definitions related to these services, as well as the <partnerLinkType> entries, may be removed from the WSDL file. Only the <types>, <message>, and bpelService port type definitions remain. Be certain to save this edited file to a new location.

Before the client application can be written, a <binding> and <service>/<port> must be defined for the bpelService port type. This can be done in a text editor or any WSDL editor. The only challenge in editing this file is to determine what the value of the port's name attribute and <address> element should be. These entries target the client's request to the container hosting the BPEL engine. The value assigned to the name attribute is derived from the service name. In this example, the service is named ProcessService, so the port should be named ProcessServicePort. Since the ActiveBPEL engine is hosted in a Tomcat servlet container, the value of the URL will be:

```
http://<hostname>:tomcat_port/active-bpel/services/ProcessService
```

Use this fourth WSDL file to write the Celtix client application.

Write a Client Application Against the Service Representing the BPEL Process

The Appendix includes copies of the client mainline, build.xml, and WSDL files that comprise this application. To recreate the application, copy these files into the proper subdirectories under the samples/client directory and then compile by issuing **`ant build`** from within a command window opened to the samples/client directory. Depending on how the host computer's environment is set up, you may need to run the environment script described in the section of this note dealing with Celtix installation (see page 3).

Start the Web Service Applications

To run the application, open command windows to the samples/approver and samples/assessor directories. Issue the command **`ant server`** within each command window. Depending on how the host computer's environment is set up, you may need to run the environment script described in the section of this note dealing with Celtix installation (see page 3).

Note that the applications will automatically terminate after 5 minutes, so you need to confirm that the application is still running just before you run the client application. If necessary, restart the Web service applications.

Deploy the Application Archive into the ActiveBPEL Engine

This simply involves copying the deployment archive file into the bpr subdirectory under the Tomcat installation directory. The bpr subdirectory was created during the installation of the ActiveBPEL engine components, as described in the subsection ActiveBPEL under Software Installation.

Once deployed, start Tomcat using the script described in the subsection Apache Tomcat under Software Installation.

Run the Client Application

Open a command window to the samples/client directory. Run the client by issuing **`ant client`**. Depending on how the host computer's environment is set up, you may need to run the environment script described in the section of this note dealing with Celtix installation (see page 3).

In a text editor, open the build.xml file and change the value assigned to the parameter.

```
<target name="client" description="run demo client">
        <property name="param" value="7000"/>
        <celtixrun classname="client.Client"
                   param1="${basedir}/wsdl/bpel_process.wsdl"
                   param2="${op}" param3="${param}"/>
    </target>
```

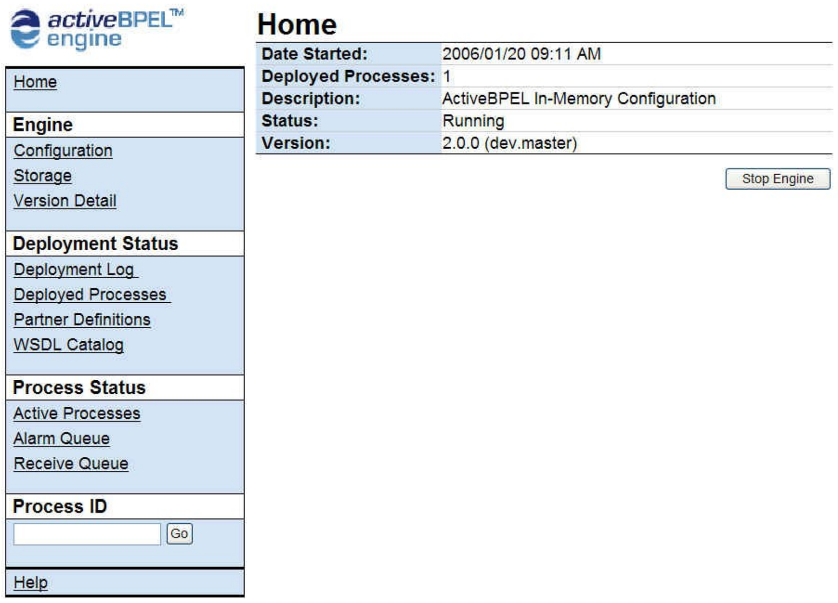
Save the file and rerun the client. Test several values: below $8000, between $8000 and $15000, and above $15000. Observe the output in the command windows running the Celtix Web services.

Review Process Execution Through the ActiveBPEL Admin Functionality

Open a Web browser and enter the URL:

```
http://<hostname>:tomcat_port/BpelAdmin
```

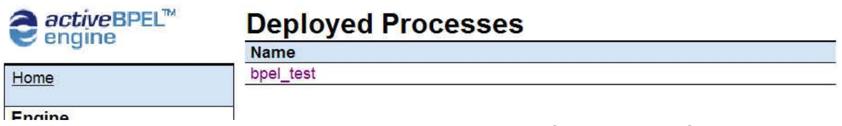
This opens the BPEL Admin home page.



Home

Date Started:	2006/01/20 09:11 AM
Deployed Processes:	1
Description:	ActiveBPEL In-Memory Configuration
Status:	Running
Version:	2.0.0 (dev.master)

Under the Deployment Status heading, click on the Deployed Processes link, which opens a page listing all deployed processes.

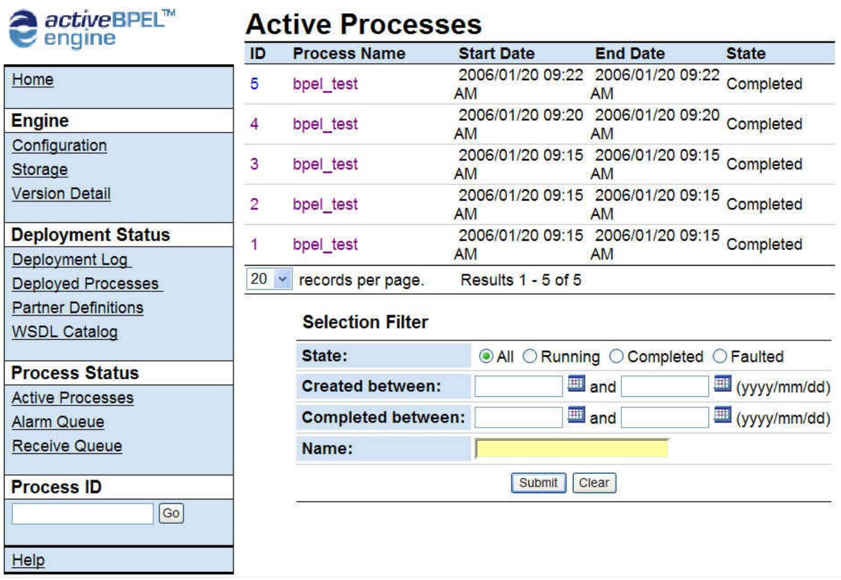


Deployed Processes

Name
bpel_test

Click on the link to the process, and you can review the content of the BPEL file and deployment descriptor for this process.

If you click on the Active Processes link under the Process Status heading, you will be presented with a page through which you can review what happened each time this process ran.



Active Processes

ID	Process Name	Start Date	End Date	State
5	bpel_test	2006/01/20 09:22 AM	2006/01/20 09:22 AM	Completed
4	bpel_test	2006/01/20 09:20 AM	2006/01/20 09:20 AM	Completed
3	bpel_test	2006/01/20 09:15 AM	2006/01/20 09:15 AM	Completed
2	bpel_test	2006/01/20 09:15 AM	2006/01/20 09:15 AM	Completed
1	bpel_test	2006/01/20 09:15 AM	2006/01/20 09:15 AM	Completed

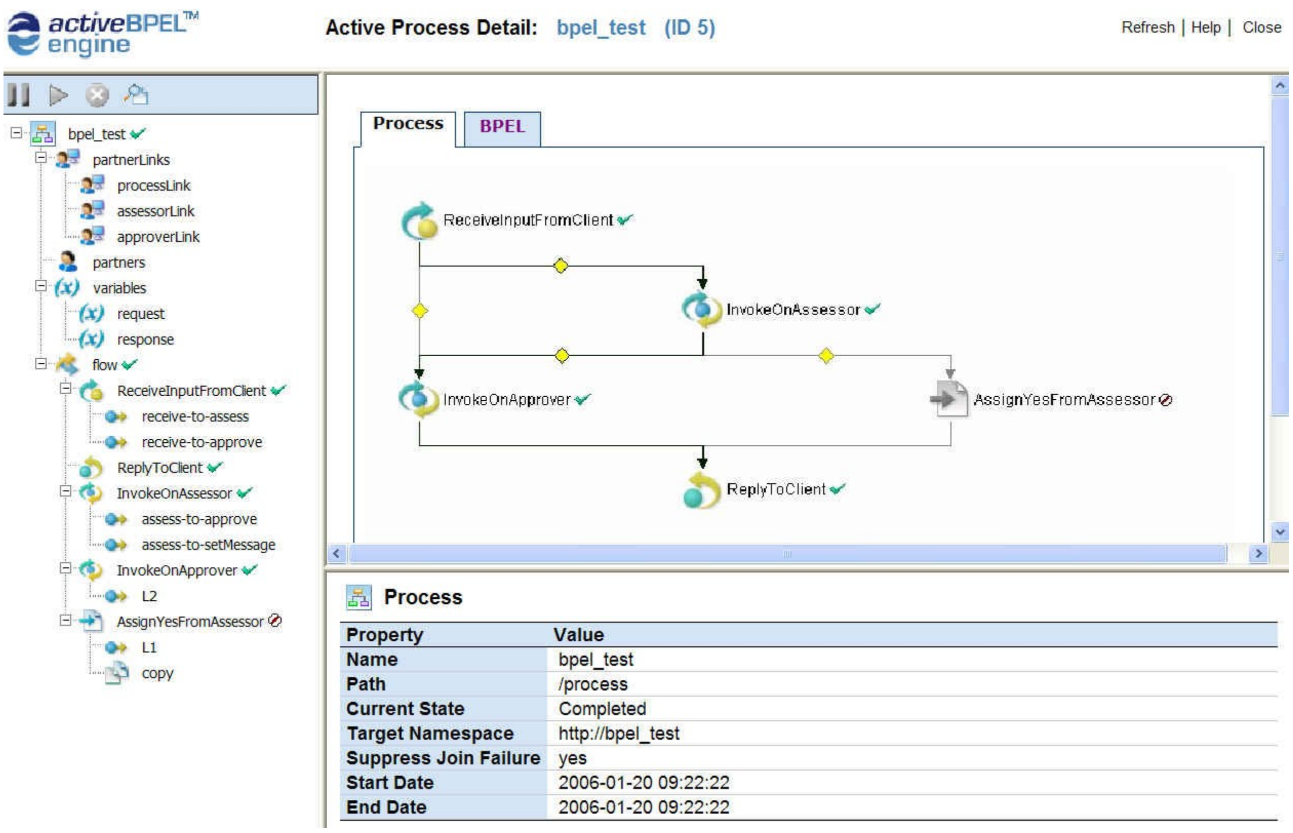
20 records per page. Results 1 - 5 of 5

Selection Filter

State:	[✓ All] Running Completed Faulted
Created between:	and (yyyy/mm/dd)
Completed between:	and (yyyy/mm/dd)
Name:	

Submit Clear

Click on the link to a process instance, and you are presented with a diagrammatic representation of the process flow.



Process

Property	Value
Name	bpel_test
Path	/process
Current State	Completed
Target Namespace	http://bpel_test
Suppress Join Failure	yes
Start Date	2006-01-20 09:22:22
End Date	2006-01-20 09:22:22

Notice how activities that where not executed (in the above screen shot, the AssignYesFromAssessor) are “grayed” out while the process flow is highlighted in color and bold linking arrows. If you click on any of the entries listed along the left-hand side of the page, the tabular data below the process diagram will summarize the instance specific details associated with that item. For example, selecting one of the items under the variables entry allows you to review the XML content of the corresponding message.

Appendix

To recreate this application, copy the following content into the application directories.

ApproverService Application

WSDL File

This file should be copied into the subdirectory samples/approver/wsdl. Save as **`bpel_test.wsdl`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--WSDL file template-->
<!--Created by IONA Artix Designer-->
<definitions name="bpel_test.wsdl"
    targetNamespace="http://org.objectweb/celtix/bpel_test"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://org.objectweb/celtix/bpel_test"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <types>
        <schema attributeFormDefault="unqualified"
                elementFormDefault="unqualified"
            targetNamespace="http://org.objectweb/celtix/bpel_test"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <complexType name="customer_request">
                <sequence maxOccurs="1" minOccurs="1">
                    <element maxOccurs="1" minOccurs="1" name="firstName"
                             type="string"/>
                    <element maxOccurs="1" minOccurs="1" name="lastName"
                             type="string"/>
                    <element maxOccurs="1" minOccurs="1" name="amount"
                             type="int"/>
                </sequence>
            </complexType>
        </schema>
    </types>
    <message name="requestMessage">
        <part name="in" type="tns:customer_request"/>
    </message>
    <message name="responseMessage">
        <part name="out" type="xsd:string"/>
    </message>
    <portType name="approverPT">
        <operation name="approve">
            <input message="tns:requestMessage" name="approveRequest"/>
            <output message="tns:responseMessage" name="approveResponse"/>
        </operation>
    </portType>
    <binding name="approverPTSOAPBinding" type="tns:approverPT">
        <soap:binding style="rpc"
                   transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="approve">
            <soap:operation soapAction="" style="rpc"/>
            <input name="approveRequest">
                <soap:body use="literal"/>
            </input>
            <output name="approveResponse">
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="ApproverService">
        <port binding="tns:approverPTSOAPBinding" name="SoapPort">
            <soap:address
                location=
                "http://localhost:9000/celtix/services/ApproverWebService"/>
        </port>
    </service>
</definitions>
```

build.xml File

This file should be copied into the subdirectory samples/approver. Save as **`build.xml`**.

```
<?xml version="1.0"?>
<project name="approver service" default="build" basedir=".">

    <import file="../common_build.xml"/>

    <target name="client" description="run demo client">
        <property name="param" value=""/>
        <celtixrun classname="client.Client"
                   param1="${basedir}/wsdl/bpel_test.wsdl"
                   param2="${op}" param3="${param}"/>
    </target>

    <target name="server" description="run demo server">
        <celtixrun classname="server.Server"
                   param1="${basedir}/wsdl/bpel_test.wsdl"/>
    </target>

    <target name="generate.code">
        <echo level="info" message="Generating code using wsdl2java..."/>
        <wsdl2java file="bpel_test.wsdl"/>
    </target>
</project>
```

Server Mainline

This file should be copied into the subdirectory samples/approver/src/server. Save as **`server.java`**.

```
package server;

import javax.xml.ws.Endpoint;

public class Server {

    protected Server() throws Exception {
        System.out.println("Starting Server");

        Object implementor = new ApproverPTImpl();
        String address =
               "http://localhost:9000/celtix/services/ApproverWebService";
        Endpoint.publish(address, implementor);
    }

    public static void main(String args[]) throws Exception {
        new Server();
        System.out.println("Server ready...");

        Thread.sleep(5 * 60 * 1000);
        System.out.println("Server exiting");
        System.exit(0);
    }
}
```

Servant

This file should be copied into the subdirectory samples/approver/src/server. Save as **`ApproverPTImpl.java`**.

```
package server;

import java.util.logging.Logger;
import objectweb.org.celtix.bpel_test.ApproverPT;
import objectweb.org.celtix.bpel_test.ApproverService;
import objectweb.org.celtix.bpel_test.CustomerRequest;

@javax.jws.WebService(name = "ApproverPT", serviceName = "ApproverService",
                      targetNamespace = "http://org.objectweb/celtix/bpel_test",
                      wsdlLocation = "file:./wsdl/bpel_test.wsdl")

public class ApproverPTImpl implements ApproverPT {

    private static final Logger LOG =
        Logger.getLogger(ApproverPTImpl.class.getPackage().getName());


    public String approve(CustomerRequest in) {
        LOG.info("Executing operation assess");

        System.out.println("approve operation invoked");

        String _return = "yes from approver";
      if (in.getAmount() > 15000) {
          _return = "no from approver";
      }

      return _return;

   }
}
```

AssessorService Application

WSDL File

This file should be copied into the subdirectory samples/assessor/wsdl.   Save as **`bpel_test.wsdl`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--WSDL file template-->
<!--Created by IONA Artix Designer-->
<definitions name="bpel_test.wsdl"
    targetNamespace="http://org.objectweb/celtix/bpel_test"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://org.objectweb/celtix/bpel_test"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <types>
        <schema attributeFormDefault="unqualified"
                elementFormDefault="unqualified"
            targetNamespace="http://org.objectweb/celtix/bpel_test"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <complexType name="customer_request">
                <sequence maxOccurs="1" minOccurs="1">
                    <element maxOccurs="1" minOccurs="1" name="firstName"
                             type="string"/>
                    <element maxOccurs="1" minOccurs="1" name="lastName"
                             type="string"/>
                    <element maxOccurs="1" minOccurs="1" name="amount"
                             type="int"/>
                </sequence>
            </complexType>
        </schema>
    </types>
    <message name="requestMessage">
        <part name="in" type="tns:customer_request"/>
    </message>
    <message name="responseMessage">
        <part name="out" type="xsd:string"/>
    </message>
    <portType name="assessorPT">
        <operation name="assess">
```

```
            <input message="tns:requestMessage" name="assessRequest"/>
            <output message="tns:responseMessage" name="assessResponse"/>
        </operation>
    </portType>
    <binding name="assessorPTSOAPBinding" type="tns:assessorPT">
        <soap:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="assess">
            <soap:operation soapAction="" style="rpc"/>
            <input name="assessRequest">
                <soap:body use="literal"/>
            </input>
            <output name="assessResponse">
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="AssessorService">
        <port binding="tns:assessorPTSOAPBinding" name="SoapPort">
            <soap:address
                location=
                "http://localhost:9001/celtix/services/AssessorWebService"/>
        </port>
    </service>
</definitions>
```

build.xml File

This file should be copied into the subdirectory samples/assessor. Save as **build.xml**.

```
<?xml version="1.0"?>
<project name="assessor service" default="build" basedir=".">

    <import file="../common_build.xml"/>

    <target name="client" description="run demo client">
        <property name="param" value=""/>
        <celtixrun classname="client.Client"
                   param1="${basedir}/wsdl/bpel_test.wsdl"
                   param2="${op}" param3="${param}"/>
    </target>

    <target name="server" description="run demo server">
        <celtixrun classname="server.Server"
                   param1="${basedir}/wsdl/bpel_test.wsdl"/>
    </target>

    <target name="generate.code">
        <echo level="info" message="Generating code using wsdl2java..."/>
        <wsdl2java file="bpel_test.wsdl"/>
    </target>
</project>
```

Server Mainline

This file should be copied into the subdirectory samples/assessor/src/server. Save as **server.java**.

```
package server;

import javax.xml.ws.Endpoint;

public class Server {

    protected Server() throws Exception {
        System.out.println("Starting Server");

        Object implementor = new AssessorPTImpl();
        String address =
              "http://localhost:9001/celtix/services/AssessorWebService";
        Endpoint.publish(address, implementor);
    }

    public static void main(String args[]) throws Exception {
        new Server();
        System.out.println("Server ready...");

        Thread.sleep(5 * 60 * 1000);
        System.out.println("Server exiting");
        System.exit(0);
    }
}
```

Servant

This file should be copied into the subdirectory samples/assessor/src/server. Save as **AssessorPTImpl.java**.

```
package server;

import java.util.logging.Logger;
import objectweb.org.celtix.bpel_test.AssessorPT;
import objectweb.org.celtix.bpel_test.AssessorService;
import objectweb.org.celtix.bpel_test.CustomerRequest;

@javax.jws.WebService(name = "AssessorPT", serviceName = "AssessorService",
                      targetNamespace = "http://org.objectweb/celtix/bpel_test",
                      wsdlLocation = "file:./wsdl/bpel_test.wsdl")

public class AssessorPTImpl implements AssessorPT {

    private static final Logger LOG =
        Logger.getLogger(AssessorPTImpl.class.getPackage().getName());


    public String assess(CustomerRequest in) {
        LOG.info("Executing operation assess");

        System.out.println("assess operation invoked");
        String _return = "low";
      if (in.getAmount() > 8000) {
          _return = "high";
      }

      return _return;

   }
}
```

BPELService

WSDL File

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="bpel_test.wsdl"
targetNamespace="http://org.objectweb/celtix/bpel_test"
xmlns:tns="http://org.objectweb/celtix/bpel_test"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <schema attributeFormDefault="unqualified" elementFormDefault="unqualified"
            targetNamespace="http://org.objectweb/celtix/bpel_test"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <complexType name="customer_request">
```

```
              <sequence maxOccurs="1" minOccurs="1">
                  <element maxOccurs="1" minOccurs="1" name="firstName"
                           type="string"/>
                  <element maxOccurs="1" minOccurs="1" name="lastName"
                           type="string"/>
                  <element maxOccurs="1" minOccurs="1" name="amount"
                           type="int"/>
              </sequence>
          </complexType>
      </schema>
</wsdl:types>

<wsdl:message name="responseMessage">
  <wsdl:part name="out" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestMessage">
  <wsdl:part name="in" type="tns:customer_request"/>
</wsdl:message>

<wsdl:portType name="bpelService">
  <wsdl:operation name="process">
    <wsdl:input message="tns:requestMessage"/>
    <wsdl:output message="tns:responseMessage"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="assessorPT">
  <wsdl:operation name="assess">
    <wsdl:input name="assessRequest" message="tns:requestMessage"/>
    <wsdl:output name="assessResponse" message="tns:responseMessage"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="approverPT">
  <wsdl:operation name="approve">
    <wsdl:input name="approveRequest" message="tns:requestMessage"/>
    <wsdl:output name="approveResponse" message="tns:responseMessage"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="approverPTSOAPBinding" type="tns:approverPT">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"
                xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  <wsdl:operation name="approve">
  <soap:operation soapAction="" style="rpc"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    <wsdl:input name="approveRequest">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:input>
    <wsdl:output name="approveResponse">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

<wsdl:binding name="assessorPTSOAPBinding" type="tns:assessorPT">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"
                xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  <wsdl:operation name="assess">
  <soap:operation soapAction="" style="rpc"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    <wsdl:input name="assessRequest">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:input>
    <wsdl:output name="assessResponse">
  <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

```
  <wsdl:service name="ApproverService">
    <wsdl:port name="SoapPort" binding="tns:approverPTSOAPBinding">
    <soap:address
          location="http://localhost:9000/celtix/services/ApproverWebService"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:port>
  </wsdl:service>

  <wsdl:service name="AssessorService">
    <wsdl:port name="SoapPort" binding="tns:assessorPTSOAPBinding">
    <soap:address
          location="http://localhost:9001/celtix/services/AssessorWebService"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:port>
  </wsdl:service>

  <plnk:partnerLinkType name="processLinkType">
   <plnk:role name="processor">
      <plnk:portType name="tns:bpelService"/>
   </plnk:role>
  </plnk:partnerLinkType>
  <plnk:partnerLinkType name="assessorLinkType">
   <plnk:role name="assessor">
      <plnk:portType name="tns:assessorPT"/>
   </plnk:role>
  </plnk:partnerLinkType>
  <plnk:partnerLinkType name="approverLinkType">
   <plnk:role name="approver">
      <plnk:portType name="tns:approverPT"/>
   </plnk:role>
  </plnk:partnerLinkType>

</wsdl:definitions>
```

.bpel File

```
<?xml version="1.0" encoding="UTF-8"?>

<process name="bpel_test" suppressJoinFailure="yes"
targetNamespace="http://bpel_test"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns1="http://org.objectweb/celtix/bpel_test"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <partnerLinks>
      <partnerLink myRole="processor" name="processLink"
                   partnerLinkType="ns1:processLinkType"/>
      <partnerLink name="assessorLink" partnerLinkType="ns1:assessorLinkType"
                   partnerRole="assessor"/>
      <partnerLink name="approverLink" partnerLinkType="ns1:approverLinkType"
                   partnerRole="approver"/>
   </partnerLinks>

   <variables>
      <variable messageType="ns1:requestMessage" name="request"/>
      <variable messageType="ns1:responseMessage" name="response"/>
   </variables>

   <flow>

      <links>
         <link name="receive-to-assess"/>
         <link name="receive-to-approve"/>
         <link name="assess-to-approve"/>
         <link name="assess-to-setMessage"/>
         <link name="L2"/>
         <link name="L1"/>
      </links>
      <receive createInstance="yes" name="ReceiveInputFromClient"
```

```
               operation="process" partnerLink="processLink"
               portType="ns1:bpelService" variable="request">
         <source linkName="receive-to-assess"
                 transitionCondition="bpws:getVariableData
                     ('request', 'in', '/in/amount') &lt; 10000"/>
         <source linkName="receive-to-approve"
                 transitionCondition="bpws:getVariableData
                     ('request', 'in', '/in/amount') &gt;= 10000"/>
      </receive>

      <reply name="ReplyToClient" operation="process" partnerLink="processLink"
                   portType="ns1:bpelService" variable="response">
         <target linkName="L2"/>
         <target linkName="L1"/>
      </reply>

      <invoke inputVariable="request" name="InvokeOnAssessor" operation="assess"
              outputVariable="response" partnerLink="assessorLink"
              portType="ns1:assessorPT">
         <target linkName="receive-to-assess"/>
         <source linkName="assess-to-approve"
                 transitionCondition="bpws:getVariableData
                     ('response', 'out') != 'low'"/>
         <source linkName="assess-to-setMessage"
                 transitionCondition="bpws:getVariableData
                     ('response', 'out') = 'low'"/>
      </invoke>

      <invoke inputVariable="request" name="InvokeOnApprover" operation="approve"
              outputVariable="response" partnerLink="approverLink"
              portType="ns1:approverPT">
         <target linkName="receive-to-approve"/>
         <target linkName="assess-to-approve"/>
         <source linkName="L2"/>
      </invoke>

      <assign name="AssignYesFromAssessor">
         <target linkName="assess-to-setMessage"/>
         <source linkName="L1"/>
         <copy>
            <from expression="'yes from assessor'"/>
            <to part="out" variable="response"/>
         </copy>
      </assign>

   </flow>
</process>
```

Deployment Descriptor File

```
<?xml version="1.0" encoding="UTF-8"?>
<process location="bpel/BPEL/bpel_test.bpel" name="bpelns:bpel_test"
persistenceType="full" xmlns="http://schemas.active-
endpoints.com/pdd/2005/09/pdd.xsd" xmlns:bpelns="http://bpel_test"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
   <partnerLinks>
      <partnerLink name="approverLink">
         <partnerRole endpointReference="static">
            <wsa:EndpointReference
             xmlns:s="http://org.objectweb/celtix/bpel_test">
               <wsa:Address>
                   http://localhost:9000/celtix/services/ApproverWebService
               </wsa:Address>
               <wsa:ServiceName PortName="SoapPort">
                   s:ApproverService
               </wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
      <partnerLink name="assessorLink">
```

```
         <partnerRole endpointReference="static">
            <wsa:EndpointReference
             xmlns:s="http://org.objectweb/celtix/bpel_test">
               <wsa:Address>
                   http://localhost:9001/celtix/services/AssessorWebService
               </wsa:Address>
               <wsa:ServiceName PortName="SoapPort">
                   s:AssessorService
               </wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
      <partnerLink name="processLink">
         <myRole allowedRoles="" binding="RPC-LIT" service="ProcessService"/>
      </partnerLink>
   </partnerLinks>
   <wsdlReferences>
      <wsdl location="project:/BPEL/WSDL/bpel_test.wsdl"
            namespace="http://org.objectweb/celtix/bpel_test"/>
   </wsdlReferences>
</process>
```

Client Application

WSDL File

This file should be copied into the subdirectory samples/client/wsdl. Save as **`bpel_test.wsdl`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="bpel_process.wsdl"
    targetNamespace="http://org.objectweb/celtix/bpel_test"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://org.objectweb/celtix/bpel_test"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <types>
        <schema attributeFormDefault="unqualified"
                elementFormDefault="unqualified"
            targetNamespace="http://org.objectweb/celtix/bpel_test"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <complexType name="customer_request">
                <sequence maxOccurs="1" minOccurs="1">
                    <element maxOccurs="1" minOccurs="1" name="firstName"
                             type="string"/>
                    <element maxOccurs="1" minOccurs="1" name="lastName"
                             type="string"/>
                    <element maxOccurs="1" minOccurs="1" name="amount"
                             type="int"/>
                </sequence>
            </complexType>
        </schema>
    </types>
    <message name="responseMessage">
        <part name="out" type="xsd:string"/>
    </message>
    <message name="requestMessage">
        <part name="in" type="tns:customer_request"/>
    </message>
    <portType name="bpelService">
        <operation name="process">
            <input message="tns:requestMessage" name="process"/>
            <output message="tns:responseMessage" name="processResponse"/>
        </operation>
    </portType>
    <binding name="bpelServiceSOAPBinding" type="tns:bpelService">
        <soap:binding style="rpc"
```

```
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="process">
            <soap:operation soapAction="" style="rpc"/>
            <input name="process">
                <soap:body use="literal"/>
            </input>
            <output name="processResponse">
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="ProcessService">
        <port binding="tns:bpelServiceSOAPBinding" name="ProcessServicePort">
            <http:address
                location=
                "http://localhost:8080/active-bpel/services/ProcessService"/>
        </port>
    </service>
</definitions>
```

build.xml File

This file should be copied into the subdirectory samples/client.   Save as **`build.xml`**.

```
<?xml version="1.0"?>
<project name="process service" default="build" basedir=".">

    <import file="../common_build.xml"/>

    <target name="client" description="run demo client">
        <property name="param" value="7000"/>
        <celtixrun classname="client.Client"
                   param1="${basedir}/wsdl/bpel_process.wsdl"
                   param2="${op}" param3="${param}"/>
    </target>

    <target name="generate.code">
        <echo level="info" message="Generating code using wsdl2java..."/>
        <wsdl2java file="bpel_process.wsdl"/>
    </target>
</project>
```

Client Mainline

This file should be copied into the subdirectory samples/client/src/client. Save as **`client.java`**.

```
package client;

import java.io.File;
import java.net.URL;

import javax.xml.namespace.QName;
import objectweb.org.celtix.bpel_test.BpelService;
import objectweb.org.celtix.bpel_test.ProcessService;
import objectweb.org.celtix.bpel_test.CustomerRequest;

public final class Client {

    private static final QName SERVICE_NAME =
        new QName("http://org.objectweb/celtix/bpel_test", "ProcessService");


    private Client() {
    }

    public static void main(String[] args) throws Exception {

        if (args.length == 0) {
            System.out.println("please specify wsdl");
            System.exit(1);
        }

        URL wsdlURL;
        File wsdlFile = new File(args[0]);
        if (wsdlFile.exists()) {
            wsdlURL = wsdlFile.toURL();
        } else {
            wsdlURL = new URL(args[0]);
        }

        ProcessService service = new ProcessService(wsdlURL, SERVICE_NAME);
        BpelService bs = (BpelService)service.getProcessServicePort();

        System.out.println("Invoking process...");
        CustomerRequest r = new CustomerRequest();
        r.setFirstName("Harry");
        r.setLastName("Potter");
        r.setAmount(new Integer(args[2]).intValue());
        System.out.println("server responded with: " + bs.process(r));
        System.out.println();

        System.exit(0);
    }
}
```