XService Workflow Engine Users Manual

This manual uses an example to introduce how to develop deploy and invoke a process.

Sample Process

This process named “example”. It accepts a string as input, and then calls a web service named “toLowerCase” to change the letters in uppercase to lowercase, finally, it returns the result of the service to client. Process is illustrated in figure 1.

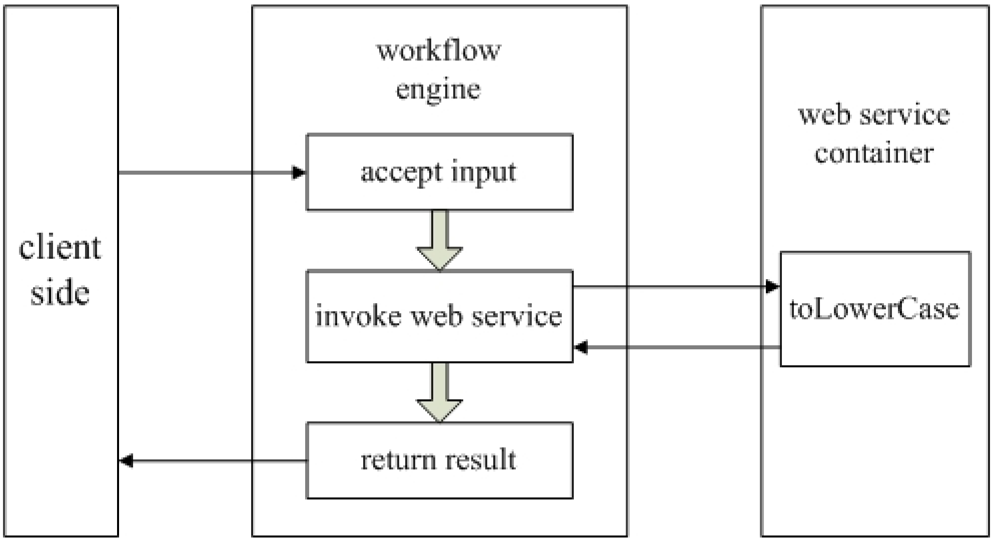


Figure 1. sample process

Web Service

The web service named “toLowerCase” in the example changes the uppercase letter in the input string to lowercase, then outputs the result. Here is the inside logic of the service:

```
public class toLowerCase {
  public String getLowerCase(String str)
  {
    return str.toLowerCase();
  }
}
```

The web service named “toLowerCase” should be deployed in the web service container, as Axis, WebLogic, ACTXService Container and so on. In the directory %Engine_HOME% \example\webservice, the deployment unit named “toLowerCase.wsar” was deployed in the ACT XService Container in this example.

When designing the process, the WSDL file of the “toLowerCase” service is needed. It can be generated by the JAVA2WSDL tool of the web service container automatically or written manually.

Here is the “toLowerCase.wsdl” file.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://act.org" xmlns:tns=http://act.org
      xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:message name="getLowerCaseRequest">
    <wsdl:part name="param0" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getLowerCaseResponse">
    <wsdl:part name="getLowerCaseReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="toLowerCase">
    <wsdl:operation name="getLowerCase" parameterOrder="param0">
      <wsdl:input name="getLowerCaseRequest" message="tns:getLowerCaseRequest"/>
      <wsdl:output name="getLowerCaseResponse" message="tns:getLowerCaseResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="toLowerCaseSOAPBinding" type="tns:toLowerCase">
    <wsdlsoap:binding style="rpc"
              transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getLowerCase">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getLowerCaseRequest">
        <wsdlsoap:body use="encoded"
                  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://act.org"/>
      </wsdl:input>
      <wsdl:output name="getLowerCaseResponse">
        <wsdlsoap:body use="encoded"
                  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://act.org"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="toLowerCaseService">
    <wsdl:port name="toLowerCase" binding="tns:toLowerCaseSOAPBinding">
      <wsdlsoap:address location=" http://localhost:8090/toLowerCaseService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

The string **http://localhost:8090/toLowerCaseService** is the address where the web service deployed, which can be changed according to the actual situation.

WorkFlow Description

Role Definition

Firstly, the relationship of the roles between the sample process and the web service should be defined in a WSDL file. In this example the file named example_ext.wsdl, as this:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace=http://your.target.namespace/example
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:ns1="http://www.act.org/userdefined" xmlns:ns2="http://act.org"
      xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">
  <import location="example.wsdl" namespace="http://www.act.org/userdefined"/>
  <import location="toLowerCase.wsdl" namespace="http://act.org"/>
  <plnk:partnerLinkType name="recToWorkflow">
    <plnk:role name="recRole">
      <plnk:portType name="ns1:rec"/>
    </plnk:role>
    <plnk:role name="recRole">
      <plnk:portType name="ns1:rec"/>
    </plnk:role>
  </plnk:partnerLinkType>
  <plnk:partnerLinkType name="lowerCaseToWorkflow">
    <plnk:role name="toLowerCaseRole">
      <plnk:portType name="ns2:toLowerCase"/>
    </plnk:role>
    <plnk:role name="recRole">
      <plnk:portType name="ns1:rec"/>
    </plnk:role>
  </plnk:partnerLinkType>
</definitions>
```

Interface Definition

Secondly, the interface of the flow should be defined in a WSDL file which is named example.wsdl in our example, like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="http://www.act.org/userdefined"
        xmlns=http://schemas.xmlsoap.org/wsdl/
        xmlns:toi=http://www.act.org/userdefined
        xmlns:wsdlsoap=http://schemas.xmlsoap.org/wsdl/soap/
        xmlns:xsd=http://www.w3.org/2001/XMLSchema
        xmlns:bpws = "http://schemas.xmlsoap.org/ws/2003/03/business-process/">
    <types>
      <schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
    </types>
    <message name=″recRequest″>
      <part name=″recInput″ type=″xsd:string″/>
    </message>
    <message name=″recResponse″>
     <part name=″recOutput″ type=″xsd:string″/>
    </message>
    <portType name=″rec″>
      <operation name=″rec″>
        <input message=″toi:recRequest″ name=″recInput″/>
        <output message=″toi:recResponse″ name=″recOutput″/>
      </operation>
    </portType>
    <binding name=″rec″ type=″toi:rec″>
      <wsdlsoap:binding required=″true″ style=″rpc″
              transport=″http://schemas.xmlsoap.org/soap/http″/>
      <operation name=″rec″>
        <wsdlsoap:operation soapAction=″″/>
        <input>
          <wsdlsoap:body encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
                use=″encoded″/>
        </input>
        <output>
          <wsdlsoap:body encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
                use=″encoded″/>
        </output>
      </operation>
    </binding>
</definitions>
```

Attribute Definition

Then, some essential attribute of the flow should also be defined in a WSDL file, like example_property.wsdl showed below:

```
<?xml version=″1.0″ encoding=″UTF-8″?>
<definitions targetNamespace=http://org.act/properties
      xmlns=″http://schemas.xmlsoap.org/wsdl/″
      xmlns:bpws=″http://schemas.xmlsoap.org/ws/2003/03/business-process/″
      xmlns:ns1=″http://www.act.org/userdefined″ xmlns:ns2=″http://act.org″>
  <bpws:property name=″lowCaseProp″ type=″xsd:string″/>
  <bpws:propertyAlias messageType=″ns2:getLowerCaseRequest″ part=″param0″
          propertyName=″tns:lowerCaseProp″/>
</definitions>
```

Flow Definition

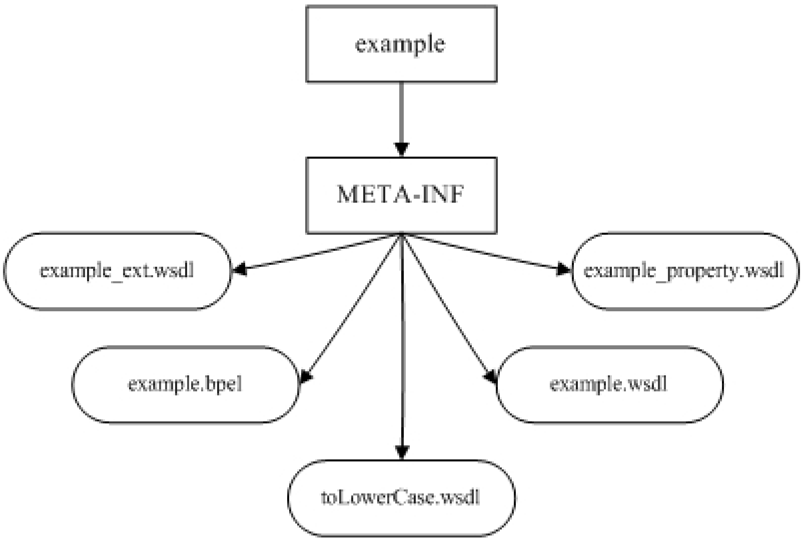
Finally, every part of the flow should be defined in a BPEL file which is named example.bpel archive in this example, as this:

```
<?xml version="1.0" encoding="gb2312"?>
<process name="example" targetNamespace="http://your.target.namespace/example"
          xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
          queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
          expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
          suppressJoinFailure="no"
          enableInstanceCompensation="no"
          abstractProcess="yes"
          xmlns:tns="http://your.target.namespace/process"
          xmlns:props="http://org.act/properties"
          xmlns:ns1="http://act.org"
          xmlns:ns2="http://www.act.org/userdefined">
  <partnerLinks>
    <partnerLink name="recPL" partnerLinkType="tns:recToWorkflow" myRole="recRole"
          partnerRole="recRole"/>
    <partnerLink name="toLowCasePL" partnerLinkType="tns:lowerCaseToWorkflow"
          myRole="recRole" partnerRole="toLowerCaseRole"/>
  </partnerLinks>
  <variables>
    <variable name="recIn" messageType="ns2:recRequest"/>
    <variable name="recOut" messageType="ns2:recResponse"/>
    <variable name="lowerCaseIn" messageType="ns1:getLowerCaseRequest"/>
    <variable name="lowerCaseOut" messageType="ns1:getLowerCaseResponse"/>
  </variables>
  <correlationSets>
    <correlationSet name="id" properties="props:lowerCaseProp"/>
  </correlationSets>
  <sequence name="" joinCondition="OR" suppressJoinFailure="no">
    <receive name="receive1" joinCondition="OR" suppressJoinFailure="no"
        partnerLink="recPL" portType="ns2:rec" operation="rec" variable="recIn"
        createInstance="yes">
    </receive>
    <assign name="assign1" joinCondition="OR" suppressJoinFailure="no">
      <copy>
        <from variable="recIn" part="recInput"/>
        <to variable="lowerCaseIn" part="param0"/>
      </copy>
    </assign>
    <invoke name="invoke toLowCase" joinCondition="OR" suppressJoinFailure="no"
        partnerLink="toLowCasePL" portType="ns1:toLowerCase"
        operation="getLowerCase" inputVariable="lowerCaseIn"
        outputVariable="lowerCaseOut">
      <correlations>
```

```
        <correlation set="id" initiate="yes" pattern="out-in"/>
      </correlations>
    </invoke>
    <assign name="assign2" joinCondition="OR" suppressJoinFailure="no">
      <copy>
        <from variable="lowerCaseOut" part="getLowerCaseReturn"/>
        <to variable="recOut" part="recOutput"/>
      </copy>
    </assign>
    <reply name="" joinCondition="OR" suppressJoinFailure="no"
        partnerLink="recPL" portType="ns2:rec" operation="rec" variable="recOut"/>
  </sequence>
</process>
```

Create Archive of the Process

After description of the process, a deployment unit of the XService Workflow Engine (a JAR with an extension of ".bar") should be created from all the files, here are the processes (in directory %Engine_HOME%\example\process, a completed archive example.bar is provided):
Firstly, all files should be organized into a particular structure illustrated below:



Then, the commands below can be executed to create the achive example.bar.

```
C:\> cd example
C:\ example> jar cf example.bar META-INF
```

Flow Deployment

The example.bar archive should be managed by the XService Workflow Manager or copied to the directory %Engine _HOME%\deploy manually. It will be ready to use after restart the XService Workflow Engine.

Client-Side

Client-Side Development

We developed the client-side program to invoke the sample process using the web service client lib which is provided by Axis (in the directory %Engine_HOME%\example\ client). Here is the code.

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;

public class ExampleClient {
    public static void main(String[] args) {
        try {
            String endpoint = "http://localhost:8091/example";
            Service service = new Service();
            Call call = (Call) service.createCall();
            call.setTargetEndpointAddress( new java.net.URL(endpoint) );
            call.setOperationName(new QName("", "rec"));
            String ret = (String) call.invoke( new Object[] { "ABCD" } );
            System.out.println(ret);
            } catch (Exception e) {
                  System.err.println(e.toString());
              }
           }
}
```

The string http://localhost:8091/example is the address where the web service deployed, which can be changed according to the actual situation.

Client-side invoke

Firstly, lib in the directory %Engine_HOME%\example\client\lib should be added to the environmental variable CLASSPATH, and then commands below can be executed in windows to invoke the sample process.

```
C:\> cd xservice-workflow-engine\example\client
C:\ xservice-workflow-engine\example\client> javac ExampleClient.java
C:\ xservice-workflow-engine\example\client> java ExampleClient
abcd
```