

WS-BPEL

Table of contents

1 Engine Review.....	2
2 WS-BPEL Correlation.....	2

I am doing a considerable amount of work with BPEL workflow engines at the moment. Anything I have noticed that may be interest to others is noted here.

1. Engine Review

The following is a quick review of the BPEL engines which I have assessed.

ActiveBPEL	The engine of choice in my opinion. It works very well and has a great community behind it. The forum is very busy with responses from the Admins regularly. This engine is open source and free to use under GPL. I have got everything I required working on this engine unlike the IBM alphaworks engine.
Collaxa BPEL Orchestration Server	Now Oracle BPEL Process Manager.
IBM BPWS4J (alphaworks)	The original BPEL engine from alphaworks is very dated now. The community support behind it is very poor. Also the engine is closed source with no sign of updates anytime soon. There are a number of known issues such as lack of support for multiple port types per service (horrible hacks excluded). I think IBM are concentrating their efforts on their WebSphere Business Integration Server commercial product in this space.
Oracle BPEL Process Manager	This engine is an evolution of the Collaxa engine. Unfortunatly the download is huge (persumably due to persistence software). Also the engine requires Internet Explorer for administration. This is due primarily to poor JavaScript code on the authors part. As a further insult to Linux users the eclipse plugin is in a .exe which even Wine couldn't open. On the plus note the documentation is good. I'd give this engine a miss unless your familiar with the Oracle product suite.
Twister	Recently joined the Apache Software Foundation, this engine is very much a prototype and requires alot more development before being used.

2. WS-BPEL Correlation

In a peer-to-peer BPEL environment it is essential to have message correlation. This is true because Web Services are by their nature stateless. What would occur if a peer A sent an asynchronous message to peer B and peer B replied sometime later asynchronously to A? We would have no guarantee that we are still talking to the original instance on peer A. In essence we might be butting into the wrong conversation on peer A.

Traditional distributed computing technologies have used instance ids as a solution to this issue. In WS-BPEL this interference can be avoided by using correlation. The correlations use key fields of data which may uniquely identify the conversation/instance. A good example of such a key field would be a bank account number. So how do you define your correlations....

- 1. Define a property (name and data type) in your WSDL which will be used by the correlation set.

```
<bpws:property name="correlationData" type="xsd:int"/>
```

- 2. Define a propertyAlias for each piece of correlation data. The property name can be the same for many aliases (refers to step 1 above).

```
<bpws:propertyAlias
messageType="CoreBankingP2P:CallbackType" part="accountNumber"
  property="CoreBankingP2P:correlationData" />
<bpws:propertyAlias
messageType="CoreBankingP2P:ServiceRequestType" part="accountNumber"
  property="CoreBankingP2P:correlationData" />
```

- 3. Define the correlation set in the related BPEL document before any sequence or flows.

```
<correlationSets>
  <correlationSet name="CS1"
properties="CoreBankingP2P:correlationData" />
</correlationSets>
```

- 4. Reference the correlation set from within the BPEL sequence. The engine will create a correlation set instance for each conversation (assuming you have createInstance="yes" set on one of the receives).

```
.....
<receive name="receiveRequest" partnerLink="Client"
portType="CoreBankingP2P:CoreBankingP2PPortType"
```

```

createInstance="yes">      operation="applyForCC" variable="ServiceRequest"
                           <correlations>
                               <correlation initiate="yes"
pattern="out" set="CS1"/>
                           </correlations>
                           </receive>
                           .....
                           <receive name="P2Pcallback"
partnerLink="CoreBankingP2PPLT"
portType="CoreBankingP2P:CoreBankingCallbackP2PPortType"
                           operation="callback" variable="CallbackRequest">
                           <correlations>
                               <correlation set="CS1"/>
                           </correlations>
                           </receive>
                           .....

```

- On receipt of a message the BPEL engine examines the correlation data. If a matching correlation data set conversation/instance can be found it will service the received message, P2Pcallback in the case above.
- NB: The correlation data must be sent to any service that may be interacting with a BPEL process which requires the correlation data. It must also be sent to services that will indirectly call back the BPEL process in need of correlation data. The WSDL of the other P2P services need not be changed to facilitate the correlation set but the WSDL of the BPEL service exposing the service to be consumed must have a provision to receive this correlation data.