Serialization Deserialization of Endpoints and EndpointReferences

A sample scenario

Assuming we have two nodes (Node1, Node2) within an SCA domain. Node1 runs component1 which is wired to component2 on Node2.

- Propagate the endpoint descriptions via the domain registry
  - Serialize the endpoints of component2 at Node2
  - Deserialize the endpoints at Node1
  - Bind the endpoint reference of component1 to the remote endpoints of component2

- Pass the ServiceReference from component1 to component2 via business methods
  - Serialize the endpoint reference (for example, a reference to component3) known to component1 at Node1
  - Deserialize the endpoint reference at Node2
  - Bind the endpoint reference to the context of Node2/Component2
public class Component1Impl {
    @Reference
    ServiceReference<Component3> component3;

    @Reference
    Component2 component2;

    public String op1(...) {
        // Pass a ServiceReference of component3 to component2
        component2.op2(component3, ...);
    }
}

public class Component2Impl {
    public String op2(ServiceReference<Component3> component3Ref, ...) {
        // Call component3 using the service reference
        Component3 component3 = component3Ref.getService();
        component3.op3(...);
    }
}

What information should be serialized?

Endpoint serialization

What information is needed on the consumer side?
- The protocol-specific endpoint address
- The binding structural URI
- The binding type
- The intents and policySets

What information is not needed on the consumer side?
- The component implementation behind the endpoint

I'm not sure if the componentType of the owning component or interface contracts are needed. Interface validation against a remote endpoint can be one only if the interface contract for the endpoint is resolvable in the context of the consuming component. But it can be too restrictive as it breaks the promise of decoupling between the SCA reference and service.

The current code serializes the endpoint as a composite/component/service/binding XML.

<composite name="..." targetNamespace>
    <component name="..." uri="...">
        <implementation.bpel process="..."/> <!-- This should be removed -->
        <service name="..." requires="..." policySets="..."/>
        <interface.wadl interface="..."/> <!-- Is the interface needed? -->
        <binding.ws uri="/..." requires="..." policySets="/..."/>
    </service>
</component>

How to resolve the deserialized endpoint in the context of the consuming component?

We keep the XML document from the Java serialization and load the XML into a composite model to rebuild the endpoint in the context of the consuming component. The current scheme of XML serialization probably contains too much information and some of them cannot be practically resolvable on the client side. For example, the target component implementation or interface contracts.
EndpointReference serialization

What information is needed on the consumer side?

- The binding structural URI
- The binding type
- The intents and policySets
- The resolved target endpoint (including the endpoint address and/or structural URI)

What information is not needed on the consumer side?

- The component implementation behind the endpoint reference

I'm not sure if the componentType of the owning component or interface contracts are needed. Interface validation against a remote endpoint can be one only if the interface contract for the endpoint is resolvable in the context of the consuming component. But it can be too restrictive as it breaks the promise of decoupling between the SCA reference and service.

The current code serializes the endpoint reference as a composite/component/reference/binding XML.

```xml
<composite name="endpointReference" targetNamespace="...">
  <component name="..." uri="...">
    <implementation.java class="..."/>
    <reference name="..." requires="..." policySets="..." target="..."> <!-- reference/@target and binding.ws/@uri are exclusive -->
      <interface.java interface="..." requires="..." policySets="..."/> <!-- Is the interface needed? -->
      <binding.ws uri="..." requires="..." policySets="..."/>
    </reference>
  </component>
</composite>
```

How to resolve the deserialized endpoint reference in the context of the consuming component?

We keep the XML document from the Java serialization and load the XML into a composite model to rebuild the endpoint in the context of the consuming component. The current scheme of XML serialization probably contains too much information and some of them cannot be practically resolvable on the client side. For example, the target component implementation or interface contracts.