SCA Bindings Issue 2 Proposal v110:

Callback Processing using WS-Addressing 1.0 Capabilities

For the Web services binding, ~~A~~an SCA Runtime MAY implement SCA callback services over the Web services binding using WS-Addressing 1.0 capabilities, as described in this section.

1) Every request message that invokes the forward interface MUST contain a Callback EPR. The Callback EPR MUST be carried in the request message in one of the following ways:

1) The request message contains the `wsa:From` SOAP header block then the `wsa:From` header block specifies the Callback EPR.
2) The `wsa:From` header block is not present then the `wsa:ReplyTo` header block ~~`[reply endpoint]` message addressing property~~ specifies the Callback EPR.

If the Callback EPR's [address] value is "`http://www.w3.org/2005/08/addressing/anonymous`" or "`http://www.w3.org/2005/08/addressing/none`" then the SCA runtime MUST generate the Invalid Addressing Header fault as specified in WS-Addressing 1.0 SOAP Binding, Section 6.4.1 [ref]. Such a fault can include additional [Subsubcode] `wsa:OnlyNonAnonymousAddressSupported`.

2) A request message that invokes the forward interface can contain the `wsa:MessageID` SOAP header block. If there is a need to have the callback request message correlated to an individual forward request message, the wsa:MessageID SOAP header block can be used for this purpose.

3) When the service implementation invokes the callback interface, it MUST use the Callback EPR from a request message that invoked the forward interface, as specified in step 1. Once the Callback EPR is selected, the SCA runtime MUST follow the rules defined in WS-Addressing 1.0 Core Section 3.3 to invoke operations on the callback interface.

When the service invokes the callback interface, if the request message from which the Callback EPR was obtained contained the wsa:MessageID SOAP header block, the SCA runtime MUST include a `wsa:RelatesTo` SOAP header block in the callback message. The `wsa:RelatesTo` SOAP header block MUST have the relationship type value of "`http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback/200812`" and the related message id MUST be the `wsa:MessageID` of the message from which the Callback EPR was obtained.

If the request message from which the Callback EPR was obtained did not contain the wsa:MessageID SOAP header block, the SCA runtime MUST NOT include a wsa:RelatesTo SOAP header block in the callback message with a relationship type value of "http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback/200812".

When a service that offers a bidirectional interface is invoked, depending on the semantics and/or implementation of the service, it is possible that the service might invoke the callback interface before the forward operation ends. In such cases, it is necessary for the invoker of the service to be ready to receive and process callback request(s) from the service, before the forward operation ends.

Callback Processing using WS-Addressing 1.0 Combined with WS-MakeConnection

It is possible that the invoker of a servicea ~~reference~~ that uses a bidirectional interface has a binding that cannot accept connections for callbacks from a service (for example, when it has the `noListener` intent [SCA-Policy]). When this is the case, it is necessary for the binding to support a polling mechanism. An example of a polling mechanism is WS-MakeConnection. [~~(~~WS-MakeConnection].

For the Web services binding, an SCA Runtime MAY implement SCA callback services over the Web services binding using WS-Addressing 1.0 capabilities combined with WS-MakeConnection, as described in this section. When an SCA runtime does implement such a capability, it MUST adhere to the rules described in the previous section (Section x.y.z) and that of WS-MakeConnection, in addition to the rules described in this section.

~~1) Every request message that invokes the forward interface MUST contain a Callback EPR. The Callback EPR MUST be carried in the request message in one of the following ways:~~

~~1)The request message contains the `wsa:From` SOAP header block then the `wsa:From` header block specifies the Callback EPR.~~

~~2)~~1) ~~The `wsa:From` header block is not present then the `[reply endpoint]` message addressing property specifies the Callback EPR.~~

The Callback EPR's [address] value present in the request message that invoked the forward interface MUST follow the form of the MakeConnection Anonymous URI, i.e. "`http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id={unique-String}`".

The unique-String value ~~MUST be~~is a globally unique value such as a UUID, as defined by the WS-MakeConnection specification.

~~2)~~ When the service implementation invokes the callback interface, it ~~MUST~~ uses the Callback EPR from a request message that invoked the forward interface, and the callback request message MUST be sent as the response to a wsmc:MakeConnection

message that contains the wsmc:Address value that matches the MakeConnection Anonymous URI in the Callback EPR.

When a service that offers a bidirectional interface is invoked using WS-MakeConnection Anonymous URI, as the value for the Callback EPR address, depending on the semantics and/or implementation of the service, it is possible that the service might invoke the callback interface before the forward operation ends. In such cases, it is necessary for the invoker of the service to poll for callback request(s) from the service and process them, before the forward operation ends.

~~WS-Callback~~Callback Policy Assertions

WS-Policy Framework and WS-Policy Attachment [WS-PolicyAttachment] collectively define a framework, model and grammar for expressing the requirements, and general characteristics of entities in an XML Web services-based system. To enable an Web service client and an Web service to describe their requirements implementing callbacks [ref], this specification defines a single ~~WS-~~Callback policy assertion that leverages the WS-Policy framework.

Assertion Model

The ~~WS-~~Callback policy assertion indicates that the Web service client and the Web service MUST use ~~WS-Callback~~Callback to implement callbacks [ref]. Specifically, the ~~WS-Callback~~Callback protocol determines the requirements on forward request message, the EPR used for callbacks and the requirements on the callback request message.

Normative Outline

The normative outline for the ~~RM-~~Callback assertion is:

```
<sca:WSCallback~~Assertion~~ ... >
  ...
</sca:WSCallback~~Assertion~~>
```

The following describes the content model of the WSCallback~~Assertion~~ element.

/sca~~wsrmp~~:WSCallback~~Assertion~~

A policy assertion that specifies that ~~WS-Callback~~Callback protocol MUST be used when sending messages.

/~~wsrmp~~sca:WSCallback~~Assertion~~/{any}

This is an extensibility mechanism to allow different (extensible) types of information, based on a schema, to be passed.

/~~wsrmp~~sca:WSCallback~~Assertion~~/@{any}

This is an extensibility mechanism to allow different (extensible) types of information, based on a schema, to be passed.

Assertion Attachment

The ~~RM-~~Callback policy assertion is allowed to have the following Policy Subjects [WS-PolicyAttachment]:

- Endpoint Policy Subject

WS-PolicyAttachment defines a set of WSDL/1.1 policy attachment points for each of the above Policy Subjects. Since an ~~WS-Callback~~Callback policy assertion specifies a concrete behavior, it MUST NOT be attached to the abstract WSDL policy attachment points.

The following is the list of WSDL/1.1 elements whose scope contains the Policy Subjects allowed for an ~~WS-Callback~~Callback policy assertion but which MUST NOT have ~~RM~~Callback policy assertions attached:

- wsdl:portType

The following is the list of WSDL/1.1 elements whose scope contains the Policy Subjects allowed for an ~~RM~~ Callback policy assertion and which MAY have ~~WS-Callback~~Callback policy assertions attached:

- wsdl:port
- wsdl:binding

Assertion Example

Table 2 lists an example use of the ~~WS-Callback~~Callback policy assertion.

Table 2: Example policy with ~~WS-Callback~~Callback policy assertion

```
(01)<wsdl:definitions
(02)    targetNamespace="example.com"
(03)    xmlns:tns="example.com"
(04)    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
(05)    xmlns:wsp="http://www.w3.org/ns/ws-policy"
(06)    xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200712"
(07)    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
(08)
(09) <wsp:UsingPolicy wsdl:required="true" />
(10)
(11) <wsp:Policy wsu:Id="MyPolicy" >
(12)   <sca:WSCallback~~Assertion~~/>
(13) </wsp:Policy>
(14)
(15) <!-- omitted elements -->
(16)
(17) <wsdl:binding name="MyBinding" type="tns:MyPortType" >
(18)   <wsp:PolicyReference URI="#MyPolicy" />
(19)   <!-- omitted elements -->
(20) </wsdl:binding>
(21)
(22)</wsdl:definitions>
```

Line (09) in Table 2 indicates that WS-Policy is in use as a required extension.

Lines (11-13) are a policy expression that includes a ~~WS-Callback~~Callback policy assertion (line 12) to indicate that ~~WS-Callback~~Callback must be used.

Lines (17-20) are a WSDL binding. Line (18) indicates that the policy in lines (11-13) applies to this binding, specifically indicating that ~~WS-Callback~~Callback must be used over all the messages in the binding.

Security Considerations

It is strongly RECOMMENDED that policies and assertions be signed to prevent tampering.

It is RECOMMENED that policies SHOULD NOT be accepted unless they are signed and have an associated security token to specify the signer has proper claims for the given policy. That is, a relying party shouldn't rely on a policy unless the policy is signed and presented with sufficient claims to pass the relying parties acceptance criteria.
It should be noted that the mechanisms described in this document could be secured as part of a SOAP message using WS-Security [WS-Security] or embedded within other objects using object-specific security mechanisms.

Schema

A normative copy of the XML Schema [XML-Schema Part1, XML-Schema Part2] description for this specification may be retrieved from the following address:

```
http://docs.oasis-open.org/ns/opencsa/sca/200712/wscallbackp.xsd
```

The following copy is provided for reference.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright(C) OASIS(R) 2005-2009. All Rights Reserved.
     OASIS trademark, IPR and other policies apply.  -->
<xs:schema xmlns:tns="http://docs.oasis-open.org/ws-
rxns/opencsa/wsrmpsca/2007102"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200712"
elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:element name="WSCallbackAssertion">
    <xs:complexType>
      <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Scenarios:

S: service with a bidirectional interface
R: reference connected to service S

The wire binding (in both directions) uses SOAP.

The forward interface consists of only one one-way operation: YouRIt()
The callback interface consists of only one one-way operation: NoYouRIt()

R invokes YouRIt(). Let's call this invocation R1 and it sets the callback address to RC1.
S then calls NoYouRIt() twice: S1 and S2
R invokes YouRIt() again. Let's call this invocation R2 and it sets the callback address to RC1. S then calls NoYouRIt() once: S3
R invokes YouRIt(). Let's call this invocation R3 and it sets the callback address to RC2.
S then calls NoYouRIt() twice: S4 and S5.

Wire messages:

R1:

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:From>
     <wsa:Address>http://example.com/callback</wsa:Address>
     <wsa:ReferenceProperties>
       <myNS:SomeID>1</myNS:SomeID>
     </wsa:ReferenceProperties>
   </wsa:From>
   <wsa:MessageID>urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6</wsa:messageID>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

S1, S2:

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:To>http://example.com/callback</wsa:To>
   <myNS:SomeID>1</myNS:SomeID>
   <wsa:RelatesTo RelationshipType="http://docs.oasis-
open.org/opencsa/sca-bindings/callback">
   urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
   </wsa:RelatesTo>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

R2:

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:From>
     <wsa:Address>http://example.com/callback</wsa:Address>
     <wsa:ReferenceProperties>
       <myNS:SomeID>1</myNS:SomeID>
     </wsa:ReferenceProperties>
```

```
   </wsa:From>
   <wsa:MessageID>urn:uuid:f81d4fae-8dec-11d0-a765-
00a0c91e6bf6</wsa:messageID>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

S3:

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:To>http://example.com/callback</wsa:To>
   <myNS:SomeID>1</myNS:SomeID>
   <wsa:RelatesTo RelationshipType="http://docs.oasis-
open.org/opencsa/sca-bindings/callback">
   urn:uuid:f81d4fae-8dec-11d0-a765-00a0c91e6bf6
   </wsa:RelatesTo>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

R3:

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:From>
     <wsa:Address>http://example.com/callback-
other</wsa:Address>
     <wsa:ReferenceProperties>
       <myNS:SomeID>2</myNS:SomeID>
     </wsa:ReferenceProperties>
   </wsa:From>
   <wsa:MessageID>urn:uuid:f81d4fae-9dec-11d0-a765-
00a0c91e6bf6</wsa:messageID>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

S4, S5:

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:To>http://example.com/callback-other</wsa:To>
   <myNS:SomeID>2</myNS:SomeID>
   <wsa:RelatesTo RelationshipType="http://docs.oasis-
open.org/opencsa/sca-bindings/callback">
   urn:uuid:f81d4fae-9dec-11d0-a765-00a0c91e6bf6
   </wsa:RelatesTo>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

Example of callback using WS-MakeConnection

Consider a one-way operation for a forward interface. As a result of invocation of the forward operation on a service by a reference, a one-way operation on the callback interface is invoked on the reference by the service. The interaction consists of a forward request message from the reference to the service. When using HTTP, the HTTP response would contain an empty entity body. This is followed by a MakeConnection message from the reference to the service. This is a polling message from the reference and establishes a connection. If the callback request is ready when the connection is established, the service sends a request to the reference invoking the callback interface.

1) Forward request message (from the reference to the service)

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:From>
     <wsa:Address>http://docs.oasis-open.org/ws-
rx/wsrm/200702/anonymous?id=650e8400-f29b-11d4-a716-
446655440010

</wsa:Address>
   </wsa:From>
   <wsa:MessageID>urn:uuid:f81d4fae-7dec-11d0-a765-
00a0c91e6bf6</wsa:messageID>
```

```
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

2) MakeConnection polling message (from the reference to the service)

```
<soap:Envelope ...>
 <soap:Header>
    <wsa:Action>http://docs.oasis-open.org/ws-rx/wsmc/200702/MakeConnection</wsa:Action>

    ...
 </soap:Header>
 <soap:Body>
    <wsmc:MakeConnection>
       <wsmc:Address>http://docs.oasis-open.org/ws-rx/wsrm/200702/anonymous?id=650e8400-f29b-11d4-a716-446655440010

</wsmc:Address>
    </wsmc:MakeConnection>
 </soap:Body>
</soap:Envelope>
```

3) Callback request message (from the service to the reference) sent as a “response” to MakeConnection

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:To>http://docs.oasis-open.org/ws-rx/wsrm/200702/anonymous?id=650e8400-f29b-11d4-a716-446655440010
</wsa:To>
   <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-bindings/callback">
   urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
   </wsa:RelatesTo>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```