



Service Component Architecture Web Service Binding Specification Version 1.1

Committee Draft 02 Issue 25 Resolution + Issue 2 Proposal v2

16th February, 2009

Specification URIs:

This Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-ws-1.1-spec-cd02.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-ws-1.1-spec-cd02.doc>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-ws-1.1-spec-cd02.pdf> (Authoritative)

Previous Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd01.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd01.doc>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd01.pdf> (Authoritative)

Latest Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-ws-1.1-spec.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-ws-1.1-spec.doc>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-ws-1.1-spec.pdf> (Authoritative)

Latest Approved Version:

Technical Committee:

OASIS Service Component Architecture / Bindings (SCA-Bindings) TC

Chair(s):

Simon Holdsworth, IBM

Editor(s):

Simon Holdsworth, IBM
Khanderao Kand, Oracle
Anish Karmarkar, Oracle
Sanjay Patil, SAP
Piotr Przybylski, IBM

Related work:

This specification replaces or supersedes:

- Service Component Architecture Web Service Binding Specification Version 1.00, March 21 2007

This specification is related to:

- Service Component Architecture Assembly Model Specification Version 1.1

- Service Component Architecture Policy Framework Specification Version 1.1

Declared XML Namespace(s):

<http://docs.oasis-open.org/ns/opencsa/sca/200712>

Abstract:

The SCA Web Service binding specified in this document applies to the services and references of an SCA composites. It defines the manner in which a service can be made available as a web service, and in which a reference can invoke a web service.

This binding is a WSDL-based binding; that means it either references an existing WSDL binding or allows one to specify enough information to generate one. When an existing WSDL binding is not referenced, rules defined in this document allow one to generate a WSDL binding.

Status:

This document was last revised or approved by the OASIS Service Component Architecture / Bindings (SCA-Bindings) TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/sca-bindings/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/sca-bindings/ipr.php>).

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/sca-bindings/>.

Notices

Copyright © OASIS® 2006, 2008. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

Table of Contents

1	Introduction	6
1.1	Terminology	6
1.2	Normative References	7
1.3	Non-Normative References	7
2	Web Service Binding Schema	8
2.1	Endpoint URI resolution	9
2.2	Interface mapping	9
2.3	Production of WSDL description for an SCA service	10
2.4	Additional binding configuration data	10
2.5	Web Service Binding and SOAP Intermediaries	10
2.6	Support for WSDL extensibility	10
2.7	Intents listed in the bindingType	10
2.8	Intents and binding configuration	11
3	Web Service Binding Examples	12
3.1	Example Using WSDL documents	12
3.2	Examples Without a WSDL Document	12
3.3	Example PolicySet Providing The Conversation Intent	13
4	Transport Binding	15
4.1	Intents	15
4.2	Default Transport Binding Rules	15
4.2.1	WS-I Basic Profile Alignment	15
4.2.2	Default Transport Binding Rules	15
5	SCA Web Services Callback Protocol	17
5.1	Composability with WS-MakeConnection	18
5.2	Policy Assertion	18
5.2.1	Assertion Model	18
5.2.2	Normative Outline	19
5.2.3	Assertion Attachment	19
5.2.4	Assertion Example	19
5.2.5	Security Considerations	20
6	Conformance	21
6.1	SCA WS Binding XML Document	21
6.2	SCA Runtime	21
A.	Web Services Binding XML Schema: sca-binding-webservice.xsd	22
B.	SCA Web Services Callback Protocol Policy Assertion XML Schema: sca-binding-webservice-callback.xsd	23
C.	Appendix - WSDL Generation	24
D.	SCA Web Services Callback Protocol Message Examples	25
D.1	Message Examples Using WS-MakeConnection	27
E.	Acknowledgements	29
F.	Non-Normative Text	30
G.	Revision History	31
1	Introduction	5
1.1	Terminology	5

1.2 Normative References	6
1.3 Non Normative References.....	6
2 Web Service Binding Schema	7
2.1 Endpoint URI resolution.....	8
2.2 Interface mapping	8
2.3 Production of WSDL description for an SCA service	9
2.4 Additional binding configuration data	9
2.5 Web Service Binding and SOAP Intermediaries	9
2.6 Support for WSDL extensibility	9
2.7 Intents listed in the bindingType.....	9
2.8 Intents and binding configuration	10
3 Web Service Binding Examples	11
3.1 Example Using WSDL documents	11
3.2 Examples Without a WSDL Document.....	11
3.3 Example PolicySet Providing The Conversation Intent.....	12
4 Transport Binding	14
4.1 Intents.....	14
4.2 Default Transport Binding Rules	14
4.2.1 WS-I Basic Profile Alignment.....	14
4.2.2 Default Transport Binding Rules	14
5 Conformance	16
5.1 SCA WS Binding XML Document.....	16
5.2 SCA Runtime	16
A. Web Services Binding XML Schema: sca-binding-webservice.xsd.....	17
B. Appendix WSDL Generation	18
C. Acknowledgements	19
D. Non Normative Text	20
E. Revision History	21

1 Introduction

The SCA Web Service binding specified in this document applies to the services and references of composites and components [SCA-Assembly]. It defines the manner in which a service can be made available as a web service, and in which a reference can invoke a web service.

This binding is a WSDL-based binding; that means it either references an existing WSDL binding or can be configured to specify enough information to generate one. When an existing WSDL binding is not referenced, rules defined in this document allow one to generate a WSDL binding.

The Web Service binding can point to an existing WSDL [WSDL] document, separately authored, that specifies the details of the WSDL binding to be used to provide or invoke the web service. In this case the SCA web services binding allows anything that is valid in a WSDL binding, including rpc-encoded style and binding extensions. It is the responsibility of the SCA system provider to ensure support for all options specified in the WSDL binding. Interoperation of such services is not guaranteed.

The SCA Web Service binding also provides attributes that can be used to provide the details of a WSDL SOAP binding. This allows a WSDL document to be synthesized in the case that one does not already exist. In this case only WS-I compliant mapping is supported.

The SCA Web Service binding can be further customized through the use of SCA Policy Sets. For example, a requirement to conform to a WS-I profile [WSI-Profiles] could be represented with a policy set.

[This specification also defines a protocol for implementing callbacks using the WS-Addressing Message Addressing Properties \[WS-Addr\].](#)

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [[RFC2119]].

This specification uses predefined namespace prefixes throughout; they are given in the following list. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 1-1 Prefixes and Namespaces used in this specification

Prefix	Namespace	Notes
xs	"http://www.w3.org/2001/XMLSchema"	Defined by XML Schema 1.0 specification
wsa	"http://www.w3.org/2005/08/addressing"	Defined by WS-Addressing 1.0
wsp	"http://www.w3.org/ns/ws-policy"	Defined by WS-Policy 1.5
wsrmp	"http://docs.oasis-open.org/ws-rx/wsrmp/200702"	Defined by WS-ReliableMessaging Policy 1.2
soap11	"http://schemas.xmlsoap.org/soap/envelope/"	Defined by SOAP 1.1
soap12	"http://www.w3.org/2005/08/addressing"	Defined by SOAP 1.2
wsdl	"http://www.w3.org/ns/wsdl-instance"	Defined by WSDL 2.0
wsoap11	"http://schemas.xmlsoap.org/wsdl/soap/"	Defined by WSDL 1.1 [WSDL11]
wsoap12	"http://schemas.xmlsoap.org/wsdl/soap12/"	Defined by [WSDL11-SOAP12]
sca	"http://docs.oasis-open.org/ns/opencsa/sca/200712"	Defined by the SCA specifications

Comment [ask1]: Page: 5
Mentioning portType here does not seem appropriate.

Comment [ask2]: Page: 5
Dave would like to get rid of this.

28 1.2 Normative References

- 29 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
30 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 31 [SCA-Assembly] <http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.pdf>
- 32 [SCA-Policy] <http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec.pdf>
- 33 [SCA-JCAA] <http://docs.oasis-open.org/opencsa/sca-j/sca-javacaa-1.1-spec.pdf>
- 34 [WSDL11] E. Christensen et al, *Web Service Description Language (WSDL) 1.1*,
35 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, W3C Note, March 15 2001.
- 36 [WSDL] E. Christensen et al, *Web Service Description Language (WSDL) 1.1*,
37 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, W3C Note, March 15 2001.
- 38 R. Chinnici et al, *Web Service Description Language (WSDL) Version 2.0 Part 1: Core*
39 *Language*, <http://www.w3.org/TR/2007/REC-wsdl20-20070626/>, W3C
40 Recommendation, June 26 2007.
- 41 [WSI-Profiles] <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>
42 <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>
43 <http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html>
44 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>
- 45 [JAX-WS] <http://jcp.org/en/jsr/detail?id=224>
- 46 [SOAP] <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
47 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 48 [SOAP12Adjuncts] SOAP Version 1.2 Part 2: Adjuncts (Second Edition)
49 <http://www.w3.org/TR/soap12-part2/>
- 50 [WS-Addr] <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
- 51 [WS-Addr-SOAP] <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>
- 52 [WSDL11-SOAP12] <http://www.w3.org/Submission/wsdl11soap12/>
- 53 [WS-MC] <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-spec-os.html>
- 54 [WS-Policy] <http://www.w3.org/TR/2007/REC-ws-policy-20070904>
- 55 [WS-PA] <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904>
- 56
- 57

58 1.3 Non-Normative References

- 59 [WSI-AP] <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>
- 60 [MTOM] <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>
- 61 [WS-RM] <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cd-01.html>
- 62 [WS-Security] <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- 63

64 2 Web Service Binding Schema

65 The Web Service binding element is defined by the following pseudo-schema.

```
66 <binding.ws name="xs:NCName"?  
67     requires="list of xs:QName"?  
68     policySets="list of xs:QName"?  
69     uri="xs:anyURI"?  
70     wsdlElement="xs:anyURI"?  
71     wsdl:wsdlLocation="list of xs:anyURI pairs"?  
72     ...>  
73 <endpointReference>...</endpointReference>*<br>74 ...<br>75 </binding.ws>
```

- 67 • */binding.ws/@name* - as defined in the SCA Assembly Specification [[SCA-Assembly](#)].
- 78 • */binding.ws/@requires* - as defined in the SCA Assembly Specification [[SCA-Assembly](#)].
- 79 • */binding.ws/@policySets* - as defined in the SCA Assembly Specification [[SCA-Assembly](#)].
- 80 • */binding.ws/@uri* - the resolution algorithm of Section 2.1 below describes how this attribute is interpreted.
- 81 • */binding.ws/@wsdlElement* – when present this attribute specifies the URI of a WSDL element. This
82 attribute points to the specified element in an existing WSDL document. The URI can have the following
83 forms:
 - 84 ○ Service:
85 `<WSDL-namespace-URI>#wsdl.service(<service-name>)`
86 In this case, the SCA runtime MUST make all the ports in the WSDL Service that have equivalent
87 portTypes with the SCA service or reference available to the SCA service or reference.
 - 88 ○ Port (WSDL 1.1):
89 `<WSDL-namespace-URI>#wsdl.port(<service-name>/<port-name>)`
90 In this case, the port in the WSDL 1.1 Service identified by the `<binding.ws>` element MUST
91 implement a portType that is equivalent to the one specified for the SCA service or reference. The
92 identified port MUST be made available to the SCA service or reference by the SCA runtime.
 - 93 ○ Endpoint (WSDL 2.0):
94 `<WSDL-namespace-URI>#wsdl.endpoint(<service-name>/<endpoint-name>)`
95 In this case, the endpoint in the WSDL 2.0 Service identified by the `<binding.ws>` element MUST
96 have an equivalent portType with the SCA service or reference. The identified endpoint MUST be
97 made available to the SCA service or reference by the SCA runtime.
 - 98
 - 99 ○ Binding:
100 `<WSDL-namespace-URI>#wsdl.binding(<binding-name>)`
101 In this case, the WSDL binding identified by the `<binding.ws>` element MUST implement a portType
102 that is equivalent to the one specified for the SCA service or reference. The SCA runtime MUST
103 make the service or reference available via the specified WSDL binding. In this case, the endpoint
104 address URI for an SCA reference MUST be specified by either the `@uri` attribute on the binding or a
105 WS-Addressing *EndpointReference* element, except where the SCA Assembly specification states that
106 the `@uri` attribute can be omitted. The endpoint address URI for an SCA service or the callback

107 element of an SCA reference is determined as specified in section 2.1. For the *callback* element of an
108 SCA service, the binding MUST NOT specify an endpoint address URI or a WS-Addressing
109 EndpointReference..

- 110 • */binding.ws/@wsdl:wsdlLocation* – when present this attribute specifies the location(s) of the WSDL
111 document(s) associated with specific namespace(s). This attribute MAY be specified by the binding in the
112 event that the <WSDL-namespace-URI> in the ‘endpoint’ attribute is not dereferencable, or when the
113 intended WSDL document is to be found at a different location than the one pointed to by the <WSDL-
114 namespace-URI>. The use of this attribute indicates that the WSDL binding points to an existing WSDL
115 document. The semantics of this attribute are specified in Section 7.1 of WSDL 2.0 [WSDL].
- 116 • */binding.ws/endpointReference* – when present this element provides the WS-Addressing [WS-Addr]
117 EndpointReference that specifies the endpoint for the service or reference. When this element is present
118 along with the @wsdlElement attribute on the parent element, the @wsdlElement attribute value MUST be
119 of the ‘Binding’ form as specified above, i.e. <WSDL-namespace-URI>#wsdl.binding(<binding-name>).
- 120 • */binding.ws/@{any}* - this is an extensibility mechanism to allow extensibility via attributes.
- 121 • */binding.ws/any* – this is an extensibility mechanism to allow extensibility via elements.

122

123 | The SCA runtime MUST support all the attributes of the <binding.ws> element, namely @name, @uri,
124 @requires, @policySets @wsdlElement, and @wsdl:wsdlLocation.

125 | The SCA runtime SHOULD support the element <endpointReference>. If an SCA runtime does not support the
126 element <endpointReference>, then it MUST reject an SCA WS Binding XML document (as defined in Section
127 6.5.1) that contains the element.

128 | The <binding.ws> element MUST conform to the XML schema defined in sca-binding-webservice.xsd.

129 2.1 Endpoint URI resolution

130 | The rules for resolving the URI at which an SCA service is hosted, or SCA reference targets, when used with
131 binding.ws (in precedence order) are:

- 132 1. The URIs in the endpoint(s) of the referenced WSDL
133 or
134 The URI specified by the *wsa:Address* element of the *endpointReference*,
- 135 2. The explicitly stated URI in the @uri attribute of the *binding.ws* element, which can be relative,
- 136 3. The structural URI as defined by the Assembly specification

137 | An SCA runtime MUST follow rules listed above in determining the URI at which an SCA service is hosted or
138 an SCA reference is targeted.

139 | The URI in the WSDL endpoint or in the *wsa:Address* of an EPR MAY be a relative URI, in which case it is
140 relative to the URI defined in (2) or (3). The *wsa:Address* element MAY be the empty relative URI, in which
141 case it uses the URI defined in (2) or (3) directly. This enables the EPR writer to specify reference parameters,
142 metadata and other EPR contents while letting the deployer choose the URI.

143 | To reference a WSDL document and also specify an EPR, the @wsdlElement attribute MUST refer to a binding
144 element in the WSDL.

145 2.2 Interface mapping

146 | When *binding.ws* is used on a service or reference with an interface that is not defined by *interface.wsdl*, then a
147 WSDL portType for the service or reference is derived from the interface by the rules defined for that SCA
148 interface type. An SCA runtime MUST raise an error if the interface does not map to a WSDL portType.

149 | For example, for *interface.java*, the mapping to a WSDL portType is as defined in the SCA Java Common
150 Annotations and API Specification [SCA-JCAA].

151 *binding.ws* implementations can use appropriate standards, for example WS-I AP 1.0 [WSI-AP] or MTOM
152 [MTOM], to map interface parameters to binary attachments transparently to the target component.

153

154 2.3 Production of WSDL description for an SCA service

155 Any service hosted by an SCA runtime with one or more web service bindings with HTTP endpoints SHOULD
156 return a WSDL description of the service in response to an HTTP GET request with the “?wsdl” suffix to that
157 HTTP endpoint. If none of the web service bindings have HTTP endpoints, then some other means of obtaining
158 the WSDL description of the service SHOULD be provided by the SCA runtime. This can include out of band
159 mechanisms, for example publication to a UDDI registry.

160 Refer to section 4 for a detailed definition of the rules that SHOULD be used for generating the WSDL
161 description of an SCA service with one or more web service bindings.

162

163 2.4 Additional binding configuration data

164 SCA runtime implementations MAY provide additional metadata that is associated with a web service binding,
165 for example to enable JAX-WS [JAX-WS] handlers to be executed as part of the target component dispatch.
166 The specification of such metadata is SCA runtime-specific and is outside of the scope of this document.

167

168 2.5 Web Service Binding and SOAP Intermediaries

169 The Web Service binding does not provide any direct or explicit support for SOAP intermediaries [SOAP].

170

171 2.6 Support for WSDL extensibility

172 When a *binding.ws* element uses the @wsdlElement attribute, the details of the binding are specified by the
173 WSDL element referenced by the value of the attribute. Per the WSDL specification, WSDL allows for
174 extensibility via elements as well as attributes, and it specifies rules for processing such elements. This
175 specification does not constrain the use of such extensibility in WSDL and relies on the rules specified in the
176 WSDL specification for processing such extended elements.

177 This specification requires that an SCA runtime MUST support the WSDL extensions defined in the namespace
178 associated with the prefix "sca" (as defined in section 1.1).

179 The SCA runtime MUST support the WSDL 1.1 binding extension for SOAP 1.1 over HTTP [WSDL11], as
180 identified by the WSDL element wsoap11:binding that has the @transport attribute with a value of
181 "http://schemas.xmlsoap.org/soap/http".

182 The SCA runtime SHOULD support the WSDL 1.1 binding extension for SOAP 1.2 over HTTP [WSDL11-
183 SOAP12], as identified by the WSDL element wsoap12:binding that has the @transport attribute with a value of
184 "http://schemas.xmlsoap.org/soap/http".

185 Because a WSDL document might contain extension elements that cannot be supported by the SCA runtime,
186 when using the @wsdlElement form of *binding.ws* it is not possible to determine whether the binding is
187 supported by the SCA runtime without parsing the referenced WSDL element and its dependent elements.

188 2.7 Intents listed in the bindingType

189 This specification places no requirements on the intents that are listed as either @alwaysProvides or
190 @mayProvides in the bindingType for *binding.ws*.

191 **2.8 Intents and binding configuration**

192 This binding mandates support for SOAP 1.1 and encourages SOAP 1.2 support. The <bindingType> element
193 associated with this binding MUST include the SOAP.1_1 intent in its @mayProvides or @alwaysProvides
194 attributes. If the <bindingType> element associated with this binding does not include the SOAP.1_1 intent in
195 its @alwaysProvides attribute, then the <bindingType> element SHOULD include the SOAP.1_2 intent in its
196 @mayProvides attribute. For more details on the <bindingType> element see [SCA-Policy].

197 The SCA runtime MUST raise an error if the web service binding is configured with a policy intent(s) that
198 conflicts with a binding instance's configuration. For example, it is an error to use the SOAP policy intent in
199 combination with a WSDL binding that does not use SOAP.

200 3 Web Service Binding Examples

201 The following snippets show the sca.composite file for the MyValueComposite file containing the service
202 element for the MyValueService and reference element for the StockQuoteService. Both the service and
203 reference use a Web Service binding.

204

205 3.1 Example Using WSDL documents

206 This example shows a service and reference using the SCA Web Service binding, using existing WSDL
207 documents in both cases. In each case there is a single binding element, whose name defaults to the
208 service/reference name.

209 The service's binding is defined by the WSDL document associated with the given URI. This service conforms
210 to WS-I Basic Profile 1.1.

211 The reference's first binding is defined by the specified WSDL service in the WSDL document at the given
212 location. The reference can use any of the WSDL service's ports/endpoints to invoke the target service. The
213 reference's second binding is defined by the specified WSDL binding. The specific endpoint URI to be invoked
214 is provided via the @uri attribute.

215

```
216 <?xml version="1.0" encoding="ASCII"?>
217 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
218   name="MyValueComposite">
219   <service name="MyValueService">
220     <interface.java interface="services.myvalue.MyValueService"/>
221     <binding.ws wsdlElement="http://www.example.org/MyValueService#
222       wsdl.endpoint(MyValueService/MyValueServiceSOAP)"/>
223     ...
224   </service>
225   ...
226   ...
227   <reference name="StockQuoteReference1">
228     <interface.java interface="services.stockquote.StockQuoteService"/>
229     <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
230       wsdl.service(StockQuoteService)"
231       wsdl:wsdlLocation="http://www.example.org/StockQuoteService
232       http://www.example.org/StockQuoteService.wsdl"/>
233   </reference>
234   ...
235   ...
236   <reference name="StockQuoteReference2">
237     <interface.java interface="services.stockquote.StockQuoteService"/>
238     <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
239       wsdl.binding(StockQuoteBinding)"
240       wsdl:wsdlLocation="http://www.example.org/StockQuoteService
241       http://www.example.org/StockQuoteService.wsdl"
242       uri="http://www.example.org/StockQuoteService5"/>
243   </reference>
244 </composite>
```

245 3.2 Examples Without a WSDL Document

246 The next example shows the simplest form of the binding element without WSDL document, assuming all
247 defaults for portType mapping and SOAP binding synthesis. The service and reference each have a single
248 binding element, whose name defaults to the service/reference name.

249 The service is to be made available at a location determined by the deployment of this component. It will have
250 a single port address and SOAP binding, with a simple WS-I BasicProfile 1.1 compliant binding, and using the
251 default options for mapping the Java interface to a WSDL portType.

252 The reference indicates a service to be invoked which has a SOAP binding and portType that matches the
253 default options for binding synthesis and interface mapping. One particular use of this case would be where the
254 reference is to an SCA service with a web service binding which itself uses all the defaults.

255

```
256 <?xml version="1.0" encoding="ASCII"?>
257 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
258   name="MyValueComposite">
259
260   <service name="MyValueService">
261     <interface.java interface="services.myvalue.MyValueService"/>
262     <binding.ws/>
263     ...
264   </service>
265
266   ...
267
268   <reference name="StockQuoteService">
269     <interface.java interface="services.stockquote.StockQuoteService"/>
270     <binding.ws uri="http://www.example.org/StockQuoteService"/>
271   </reference>
272 </composite>
```

273

274 The next example shows the use of the binding element without a WSDL document, with multiple SOAP
275 bindings with non-default values. The SOAP 1.2 binding name defaults to the service name, the SOAP 1.1
276 binding is given an explicit name. The reference has a web service binding which uses SOAP 1.2, but
277 otherwise uses all the defaults for SOAP binding. The reference binding name defaults to the reference name.

278

```
279 <?xml version="1.0" encoding="ASCII"?>
280 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
281   name="MyValueComposite">
282
283   <service name="MyValueService">
284     <interface.java interface="services.myvalue.MyValueService"/>
285     <binding.ws name="MyValueServiceSOAP11" requires="SOAP.1_1"/>
286     <binding.ws requires="SOAP.1_2"/>
287     ...
288   </service>
289
290   ...
291
292   <reference name="StockQuoteService">
293     <interface.java interface="services.stockquote.StockQuoteService"/>
294     <binding.ws uri="http://www.example.org/StockQuoteService"
295       requires="SOAP.1_2"/>
296   </reference>
297 </composite>
```

298

299 3.3 Example PolicySet Providing The Conversation Intent

300 The following policy set applies to *binding.ws* and provides the conversation intent. The conversation intent is
301 provided by using WS-ReliableMessaging [WS-RM] protocol which has a concept of a Sequence. This
302 Sequence (which appears as a *wsm:Sequence* SOAP header in the message) is used as a correlation
303 mechanism, on the wire, to implement conversational semantics.

```
304 <policySet name="WSRM-Sequence-based-conversation"  
305     provides="sca:conversation"  
306     appliesTo="sca:binding.ws">  
307   <wsp:Policy>  
308     <wsrmp:RMAssertion  
309       xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200608"/>  
310   </wsp:Policy>  
311 </policySet>
```

312

313 4 Transport Binding

314 The binding.ws element provides numerous ways to specify exactly how messages ought to be transmitted from
315 or to the reference or service. Those ways include references to WSDL binding elements from the
316 @wsdlElement attribute, policy intents, and even vendor extensions within the binding.ws element. However,
317 all of those ways to indicate how messages get carried happen to be optional. This section describes the defaults
318 to be used if the specific transport details are not otherwise specified.

319 4.1 Intents

320 So as to narrow the range of choices for how messages are carried, the following policy intents affect the
321 transport binding:

- 322 • SOAP
323 This indicates that messages MUST be transmitted using SOAP. One or more SOAP versions can be used.
- 324 • SOAP.1_1
325 Messages MUST be transmitted using only SOAP 1.1.
- 326 • SOAP.1_2
327 Messages MUST be transmitted using only SOAP 1.2.

328 4.2 Default Transport Binding Rules

329 4.2.1 WS-I Basic Profile Alignment

330 To align to WS-I Basic Profile, the resulting WSDL port needs to be all document-literal, or all rpc-literal
331 binding (R2705). This means, for any given portType, for all messages referenced by all operations in that
332 portType, either

- 333 • that every message part references an XML Schema type (rpc-literal pattern)
- 334 • or that every message references exactly zero or one XML Schema elements (document-literal pattern)

335 For a service element, the portType from the service's interface or derived from the service's interface MUST fit
336 one of these two patterns. The rest of this section assumes the short-hand reference of an "rpc-literal" or
337 "document-literal" pattern, depending on which of the two bullet points above it matches.

338 4.2.2 Default Transport Binding Rules

339 In the event that the transport details are not otherwise determined, an SCA runtime MUST enable the following
340 configuration:

- 341 • HTTP-based transfer protocol
- 342 • Bindings for SOAP 1.1 MUST be provided and additional bindings MAY be provided, unless policy is
343 applied that explicitly restricts this.
- 344 • "literal" format as described in section 3.5 of [WSDL11]
- 345 • For document literal pattern, each message uses "document" style, as per section 3.5 of [WSDL11].
- 346 • For rpc-literal pattern, each message uses "rpc" style, as per section 3.5 of [WSDL11]. In this case, the
347 child elements of the SOAP Body element MUST be namespace qualified with a non-empty
348 namespace name. This namespace SHOULD be the structural URI associated with the binding.
- 349 • For SOAP 1.1 messages, the SOAPAction HTTP header described in section 6.1.1 represents the
350 empty string, in quotes ("").

- 351
- 352
- 353
- For SOAP 1.2 messages, the SOAP Action feature described in section 6.5 of [\[SOAP12Adjuncts\]](#) does not appear.
 - All WSDL message parts are carried in the SOAP body

Formatted: Bullets and Numbering

5 SCA Web Services Callback Protocol

This section defines the SCA Web Services callback protocol that can be used to implement a bidirectional interface in conjunction with the Web Services binding. For examples of wire messages exchanged when using this protocol see Appendix D.

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

To implement this protocol, an SCA binding follows the following rules.

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:

- a. If the request message contains the `wsa:From` SOAP header block then the `wsa:From` header block specifies the Callback EPR.

- b. If the `wsa:From` header block is not present then the `wsa:ReplyTo` header block 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 [WS-Addr-SOAP]. 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"` 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 with a relationship type value of `"http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback"` in the callback message.

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

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

398 necessary for the binding on the reference-side to be listening for callback request(s)
399 from the service, before the forward operation request is sent on the wire to the service,
400 and continue listening as long as callback requests are expected. It is possible that
401 before the response to the forward request is sent a response to one or more callback
402 requests are required by the service.

403 **5.1 Composability with WS-MakeConnection**

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

409 For the Web services binding, an SCA Runtime MAY implement SCA callback services
410 over the Web services binding using WS-Addressing 1.0 capabilities combined with WS-
411 MakeConnection, as described in this section. When an SCA runtime does implement
412 such a capability, it MUST adhere to the rules described above (Section 5) and that of
413 WS-MakeConnection, in addition to the rules described in this section.

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

418 The unique-String value is a globally unique value such as a UUID, as defined by the
419 WS-MakeConnection specification.

420 When the service implementation invokes the callback interface, it uses the Callback EPR
421 from a request message that invoked the forward interface, and the callback request
422 message MUST be sent as the response to a wsmc:MakeConnection message that
423 contains the wsmc:Address value that matches the MakeConnection Anonymous URI in
424 the Callback EPR.

425 When a service that offers a bidirectional interface is invoked using WS-MakeConnection
426 Anonymous URI, as the value for the Callback EPR address, depending on the semantics
427 and/or implementation of the service, it is possible that the service might invoke the
428 callback interface before the forward operation ends. In such cases, it is necessary for
429 the binding on the reference-side to start polling for callback request(s) from the
430 service, before or right after the forward operation request is sent and before a response
431 is received, and continue polling as long as callback requests are expected. It is possible
432 that before the response to the forward request is sent a response to one or more
433 callback requests are required by the service.

434 **5.2 Policy Assertion**

435 WS-Policy Framework [WS-Policy] and WS-Policy Attachment [WS-PA] collectively define
436 a framework, model and grammar for expressing the requirements, and general
437 characteristics of entities in an XML Web services-based system. To enable a Web
438 service client and a Web service to describe their requirements for implementing SCA
439 Web Services Callback Protocol (see SCA Web Services Callback Protocol), this
440 specification defines a single policy assertion that leverages the WS-Policy framework.

441 **5.2.1 Assertion Model**

442 The WSCallback policy assertion indicates that the Web service client and the Web
443 service MUST use SCA Web Services Callback Protocol to implement callbacks.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

444 Specifically, the protocol determines the requirements on forward request message, the
445 EPR used for callbacks and the requirements on the callback request message.

446 5.2.2 Normative Outline

447 The normative outline for the WSCallback assertion is:

```
448 <sca:WSCallback ...>  
449 ...  
450 </sca:WSCallback>
```

452 The following describes the content model of the WSCallback element.

- 453 • /sca:WSCallback: A policy assertion that specifies that WSCallback protocol
454 MUST be used when sending messages.

455 5.2.3 Assertion Attachment

456 The WSCallback policy assertion is allowed to have the following Policy Subjects [WS-
457 PA]:

- 458 • Endpoint Policy Subject

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

462 The following is the list of WSDL/1.1 elements whose scope contains the Policy Subjects
463 allowed for a WSCallback policy assertion but which MUST NOT have WSCallback policy
464 assertions attached:

- 465 • wsdl:portType

466 The following is the list of WSDL/1.1 elements whose scope contains the Policy Subjects
467 allowed for a WSCallback policy assertion and which MAY have WSCallback policy
468 assertions attached:

- 469 • wsdl:port
- 470 • wsdl:binding

471 5.2.4 Assertion Example

472 Table 2 lists an example use of the WSCallback policy assertion.

474 Table 2: Example policy with WSCallback policy assertion

```
475 (01) <wsdl:definitions  
476 (02)   targetNamespace="example.com"  
477 (03)   xmlns:tns="example.com"  
478 (04)   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  
479 (05)   xmlns:wsp="http://www.w3.org/ns/ws-policy"  
480 (06)   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"  
481 (07)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
482 wssecurity-utility-1.0.xsd">  
483 (08)  
484 (09) <wsp:UsingPolicy wsdl:required="true" />  
485 (10)  
486 (11) <wsp:Policy wsu:Id="MyPolicy" >  
487 (12)   <sca:WSCallback/>
```

Formatted: Bullets and Numbering

Formatted

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513

```
(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 WSCallback policy assertion (line 12) to indicate that SCA Web Services Callback protocol 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 SCA Web Services Callback protocol must be used over all the messages in the binding.

5.2.5 Security Considerations

Policies and assertions SHOULD be signed to prevent tampering. 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.

Formatted: Bullets and Numbering

514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546

5.6 Conformance

The XML schema pointed to by the RDDDL document at the namespace URI, defined by this specification, are considered to be authoritative and take precedence over the XML schema defined in the appendix of this document.

There are two categories of artifacts for which this specification defines conformance:

- a) SCA WS Binding XML Document
- b) SCA Runtime

5.16.1 SCA WS Binding XML Document

An SCA WS Binding XML document is an SCA Composite Document, or an SCA ComponentType Document, as defined by the SCA Assembly specification Section 13.1 [SCA-Assembly], that uses the <binding.ws> element.

An SCA WS Binding XML document MUST be a conformant SCA Composite Document or a SCA ComponentType Document, as defined by the SCA Assembly specification [SCA-ASSEMBLY], and MUST comply with all the applicable requirements specified in this specification.

5.26.2 SCA Runtime

An implementation that claims to conform to the requirements of an SCA Runtime defined in this specification has to meet the following conditions:

1. The implementation MUST comply with all statements in Appendix XXX: Conformance Items related to an SCA Runtime, except for those that originate from Section 5, notably all "MUST" statements have to be implemented.
2. The implementation MAY support the Web Services Callback Protocol. If it does, it MUST comply with all statements in Appendix XXX: Conformance Items related to an SCA Runtime that originate from Section 5.
- ~~2.3.~~ The implementation MUST conform to the SCA Assembly Model Specification Version 1.1 [SCA-Assembly], and to the SCA Policy Framework Version 1.1 [SCA-Policy].
- ~~3.4.~~ The implementation MUST reject a SCA WS Binding XML Document that is not conformant per Section ~~6~~5.1.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Comment [ask3]: Page: 16
Moved to the top of the section and modified to match the Java CAA wordings

Comment [ask4]: Page: 16
Moved to section 5.1 and to section 2

547

A. Web Services Binding XML Schema: sca-binding-webservice.xsd

548

```
549 <?xml version="1.0" encoding="UTF-8"?>
550 <!-- (c) Copyright OASIS 2006, 2008 -->
551 <schema xmlns="http://www.w3.org/2001/XMLSchema"
552   targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200712"
553   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200712"
554   xmlns:wsdl="http://www.w3.org/ns/wsdl-instance"
555   xmlns:wsa="http://www.w3.org/2005/08/addressing"
556   elementFormDefault="qualified">
557
558   <import namespace="http://www.w3.org/ns/wsdl-instance"
559     schemaLocation="http://www.w3.org/2007/05/wsdl/wsdl20-instance.xsd"
560   />
561   <import namespace="http://www.w3.org/2005/08/addressing"
562     schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"
563   />
564   <include schemaLocation="sca-core.xsd"/>
565
566   <element name="binding.ws" type="sca:WebServiceBinding"
567     substitutionGroup="sca:binding"/>
568   <complexType name="WebServiceBinding">
569     <complexContent>
570       <extension base="sca:Binding">
571         <sequence>
572           <element name="endpointReference"
573             type="wsa:EndpointReference"
574             minOccurs="0" maxOccurs="unbounded"/>
575           <any namespace="##other" processContents="lax"
576             minOccurs="0" maxOccurs="unbounded"/>
577         </sequence>
578         <attribute name="wsdlElement" type="anyURI" use="optional"/>
579         <attribute ref="wsdl:wsdlLocation" use="optional"/>
580         <anyAttribute namespace="##any" processContents="lax"/>
581       </extension>
582     </complexContent>
583   </complexType>
584 </schema>
```

586

587
588
589

590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605

606

B. SCA Web Services Callback Protocol Policy Assertion XML Schema: sca-binding-webservice- callback.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) Copyright OASIS 2006, 2008 -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200903"
  elementFormDefault="qualified">

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

607

B.C. Appendix - WSDL Generation

608 Due to the number of factors that determine how a WSDL might be generated, including compatibility with
609 existing WSDL uses, precise details cannot be specified. For example, implementation decisions can affect the
610 way WSDL might be generated. For reference, and consistency, this section suggests non-normative choices for
611 some of the various details involved in generating WSDL. For brevity, the following definitions apply:

- 612 • component name = the value of the @name attribute of the component element containing the
613 binding.ws element
- 614 • service name = the value of the @name attribute of the service element containing the binding.ws
615 element
- 616 • binding name = the value of @name attribute of the binding.ws element, or the default if no @name
617 attribute is present
- 618 • SOAP version = either "SOAP11" or "SOAP12" as appropriate

619 With those definitions in place, here are the suggested choices:

- 620 • wsdl:definitions/@name = <component name> + "." + <service name>
- 621 • wsdl:definitions/@targetNamespace = <structural URI for the service>
- 622 • import each WSDL 1.1 portType, rather than putting them inline
- 623 • wsdl:binding/@name = <binding name> + <SOAP version> + "Binding"
- 624 • wsdl:service/@name = <service name>
- 625 • wsdl:port/@name = <binding name> + <SOAP version> + "Port"

D. SCA Web Services Callback Protocol Message Examples

The message examples in this section are for a configuration that consists of a reference R that is wired to a Service S. S has a bidirectional interface and the binding used in both directions, forward and callback, is binding.ws configured for SOAP. Both the forward interface, as well as the callback interface, each, consists of only one one-way operation.

The following message exchanges take place between R and S:

1. R invokes the forward operation and sets the callback address to RC1. Let's call the message that invokes the forward operation R1. S then calls the callback operation twice. Let's call the callback messages S1 and S2
2. R invokes the forward operation again with the same callback address RC1. Let's call the message that invokes the forward operation R2. S then calls the callback operation once. Let's call the callback message S3.
3. R invokes the forward operation yet another time, but this time uses a different callback address: RC2. Let's call the message that invokes the forward operation R3. S then calls the callback operation twice. Let's call the callback messages S4 and S5.

The messages R1, R2, R3, S1, S2, S3, S4 and S5 are listed below. The namespace prefix 'soap' can be bound to either the SOAP 1.1 or SOAP 1.2 namespace. The 'wsa' prefix is bound to the WS-Addressing 1.0 namespace.

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:

```
667 <soap:Envelope ...>
668 <soap:Header>
669 <wsa:To>http://example.com/callback</wsa:To>
670 <myNS:SomeID>1</myNS:SomeID>
671 <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
672 bindings/ws/callback">urn:uuid:f81d4fae-7dec-11d0-a765-
673 00a0c91e6bf6</wsa:RelatesTo>
674 ...
675 </soap:Header>
676 <soap:Body>
677 ...
678 </soap:Body>
679 </soap:Envelope>
680
```

R2:

Formatted

```
683 <soap:Envelope ...>
684 <soap:Header>
685 <wsa:From>
686 <wsa:Address>http://example.com/callback</wsa:Address>
687 <wsa:ReferenceProperties>
688 <myNS:SomeID>1</myNS:SomeID>
689 </wsa:ReferenceProperties>
690 </wsa:From>
691 <wsa:MessageID>urn:uuid:f81d4fae-8dec-11d0-a765-
692 00a0c91e6bf6</wsa:messageID>
693 ...
694 </soap:Header>
695 <soap:Body>
696 ...
697 </soap:Body>
698 </soap:Envelope>
699
```

S3:

Formatted

```
702 <soap:Envelope ...>
703 <soap:Header>
704 <wsa:To>http://example.com/callback</wsa:To>
705 <myNS:SomeID>1</myNS:SomeID>
706 <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
707 bindings/ws/callback">
708 urn:uuid:f81d4fae-8dec-11d0-a765-00a0c91e6bf6
709 </wsa:RelatesTo>
710 ...
711 </soap:Header>
712 <soap:Body>
713 ...
714 </soap:Body>
715 </soap:Envelope>
716
```

R3:

Formatted

718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735

```
<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>
```

736

737

S4, S5:

Formatted

738
739
740
741
742
743
744
745
746
747
748
749
750

```
<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/ws/callback">urn:uuid:f81d4fae-9dec-11d0-a765-00a0c91e6bf6</wsa:RelatesTo>
    ...
  </soap:Header>
  <soap:Body>
    ...
  </soap:Body>
</soap:Envelope>
```

751

752

D.1 Message Examples Using WS-MakeConnection

Formatted: Bullets and Numbering

753
754
755
756
757
758
759

In this case the reference R cannot host a listener and uses WS-MakeConnection to poll for callback requests. The interaction between the two consists of reference R sending a forward request R4. When using HTTP, the HTTP response to R4 contains 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 callback request S6 to the reference in the entity body of the HTTP response.

760

R4:

Formatted

```
761 <soap:Envelope ...>
762 <soap:Header>
763 <wsa:From>
764 <wsa:Address>http://docs.oasis-open.org/ws-
765 rx/wsmc/200702/anonymous?id=650e8400-f29b-11d4-a716-446655440010</wsa:Address>
766 </wsa:From>
767 <wsa:MessageID>urn:uuid:f81d4fae-10dec-11d0-a765-
768 00a0c91e6bf6</wsa:messageID>
769 ...
770 </soap:Header>
771 <soap:Body>
772 ...
773 </soap:Body>
774 </soap:Envelope>
```

775

776 **MakeConnection polling message (from R to S):**

Formatted

```
777 <soap:Envelope ...>
778 <soap:Header>
779 <wsa:Action>http://docs.oasis-open.org/ws-
780 rx/wsmc/200702/MakeConnection</wsa:Action>
781 ...
782 </soap:Header>
783 <soap:Body>
784 <wsmc:MakeConnection>
785 <wsmc:Address>http://docs.oasis-open.org/ws-
786 rx/wsmc/200702/anonymous?id=650e8400-f29b-11d4-a716-
787 446655440010</wsmc:Address>
788 </wsmc:MakeConnection>
789 </soap:Body>
790 </soap:Envelope>
```

791

792 **S6:**

Formatted

```
793 <soap:Envelope ...>
794 <soap:Header>
795 <wsa:To>http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id=650e8400-
796 f29b-11d4-a716-446655440010</wsa:To>
797 <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
798 bindings/ws/callback">urn:uuid:f81d4fae-10dec-11d0-a765-
799 00a0c91e6bf6</wsa:RelatesTo>
800 ...
801 </soap:Header>
802 <soap:Body>
803 ...
804 </soap:Body>
805 </soap:Envelope>
```

806

807

808

Formatted

809

C.E. Acknowledgements

810

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

811

Participants:

812

[Participant Name, Affiliation | Individual Member]

813

[Participant Name, Affiliation | Individual Member]

814

Formatted: Bullets and Numbering

D.F. Non-Normative Text

Formatted: Bullets and Numbering

816

E.G. Revision History

817

[optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
1	2007-09-25	Anish Karmarkar	Applied the OASIS template + related changes to the Submission
2	2008-04-02	Anish Karmarkar	* Partially applied the resolution of issue 14 in the conformance section. * Applied resolution to issue 9. * Applied resolution to issue 15. * Applied resolution to issue 16. * Applied resolution to issue 10. * Applied resolution to issue 8. * Applied resolution to issue 3.
3	2008-06-12	Simon Holdsworth	* Completed application of resolution to issue 10 * Applied most of the editorial changes from Eric Johnson's review
4	2008-08-13	Anish Karmarkar	* Applied rest of Eric Johnson's ed review comments. * Applied resolution of issue 13. * Reapplied resolution of issue 15 (it was not applied correctly before) * Applied resolution of issue 19. * Applied resolution of issue 30. * Applied resolution of issue 32. * Applied resolution of issue 36. * Applied resolution of issue 38.
cd01-rev1	2008-10-16	Simon Holdsworth	Applied resolution of issue 41.
cd01-rev2	2008-10-20	Anish Karmarkar	Added rfc2119 statements.
cd01-rev3	2008-11-19	Anish Karmarkar	Incorporated feedback from Bryan, Eric & Dave
cd01-rev3	2008-12-02	Anish Karmarkar	Removed 'required' word associated with description of pseudo-schema + changed section 2.6 (wsdl extensibility) per the TC decision. Both of these were associated with issue 51 (2119 stmts)
cd01-rev5	2009-02-06	Simon Holdsworth	Applied resolution of issue 11 Applied resolution of issue 49 Applied action item 20080904-1
cd02	2009-02-16	Simon Holdsworth	Renamed, applied editorial issues

818