



Service Component Architecture Web Service Binding Specification Version 1.1

Committee Draft 02 Revision 4 + issue 2 rev 5

17th June, 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
Anish Karmarkar, Oracle
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/200903>

Abstract:

The SCA Web Service binding specified in this document applies to the services and references of an SCA composite. 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® 2005, 2009. 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 name "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
1.4	Naming Conventions	7
2	Web Service Binding Schema	9
2.1	Compatibility of SCA Service Interfaces and WSDL portTypes	11
2.2	Endpoint URI resolution	12
2.3	Interface mapping	12
2.4	Production of WSDL description for an SCA service	12
2.5	Additional binding configuration data	13
2.6	Web Service Binding and SOAP Intermediaries	13
2.7	Support for WSDL extensibility	13
2.8	Intents listed in the bindingType	13
2.9	Intents and binding configuration	14
3	Web Service Binding Examples	15
3.1	Example Using WSDL documents	15
3.2	Examples Without a WSDL Document	16
4	Transport Binding	18
4.1	Intents	18
4.2	Default Transport Binding Rules	18
4.2.1	WS-I Basic Profile Alignment	18
4.2.2	Default Transport Binding Rules	18
5	Implementing SCA Callbacks using Web Services	20
5.1	SCA Web Services Callback Protocol	20
5.2	SCA Web Services Callback with WS-MakeConnection Protocol	21
5.3	Policy Assertion for SCA Web Services Callback Protocol	21
5.3.1	Assertion Model	21
5.3.2	Normative Outline	22
5.3.3	Assertion Attachment	22
5.3.4	Assertion Example	22
5.3.5	Security Considerations	23
6	Conformance	24
6.1	SCA WS Binding XML Document	24
6.2	SCA Runtime	24
A.	Web Services XML Binding Schema: sca-binding-webservice.xsd	25
B.	SCA Web Services Callback Protocol Policy Assertion XML Schema: sca-binding-webservice-callback.xsd	26
C.	Conformance Items	27
D.	Appendix - WSDL Generation	31
E.	SCA Web Services Callback Protocol Message Examples	32
E.1	Message Examples Using WS-MakeConnection	34
F.	Acknowledgements	36

G.	Revision History.....	38
1	Introduction.....	5
1.1	Terminology.....	5
1.2	Normative References.....	6
1.3	Non-Normative References.....	6
1.4	Naming Conventions.....	6
2	Web Service Binding Schema.....	8
2.1	Compatibility of SCA Service Interfaces and WSDL portTypes.....	10
2.2	Endpoint URI resolution.....	11
2.3	Interface mapping.....	11
2.4	Production of WSDL description for an SCA service.....	11
2.5	Additional binding configuration data.....	12
2.6	Web Service Binding and SOAP Intermediaries.....	12
2.7	Support for WSDL extensibility.....	12
2.8	Intents listed in the bindingType.....	12
2.9	Intents and binding configuration.....	13
3	Web Service Binding Examples.....	14
3.1	Example Using WSDL documents.....	14
3.2	Examples Without a WSDL Document.....	15
4	Transport Binding.....	17
4.1	Intents.....	17
4.2	Default Transport Binding Rules.....	17
4.2.1	WS-I Basic Profile Alignment.....	17
4.2.2	Default Transport Binding Rules.....	17
5	Conformance.....	19
5.1	SCA WS Binding XML Document.....	19
5.2	SCA Runtime.....	19
A.	Web Services XML Binding Schema: sca-binding-webservice.xsd.....	20
B.	Conformance Items.....	21
C.	Appendix - WSDL Generation.....	24
D.	Acknowledgements.....	25
E.	Non-Normative Text.....	26
F.	Revision History.....	27

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. This specification only defines a binding using WSDL 1.1.

The Web Service binding can point to an existing WSDL [**WSDL11**] 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**]{**WSI-Profiles**}

Formatted: Font color: Auto

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
wsdli	"http://www.w3.org/ns/wsdli-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 [W11-SOAP12]
sca	"http://docs.oasis-open.org/ns/opencsa/sca/200903"	Defined by the SCA specifications

31

32 1.2 Normative References

- 33 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
34 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 35 **[SCA-Assembly]** <http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.pdf>
- 36 **[SCA-Policy]** <http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec.pdf>
- 37 **[SCA-JCAA]** <http://docs.oasis-open.org/opencsa/sca-j/sca-javacaa-1.1-spec.pdf>
- 38 **[WSDL11]** E. Christensen et al, *Web Service Description Language (WSDL) 1.1*,
39 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, W3C Note, March 15 2001.
- 40 **[WSDL20]** Chinnici et al, *Web Service Description Language (WSDL) Version 2.0 Part 1: Core Language*, <http://www.w3.org/TR/2007/REC-wsdl20-20070626/>, W3C Recommendation, June 26 2007.
- 42 **[WSI-Profiles]** <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>
43 <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>
44 <http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html>
45 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>
- 46 **[JAX-WS]** <http://jcp.org/en/jsr/detail?id=224>
- 47 **[SOAP11]** <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 48 **[SOAP]** <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
49 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 50 **[SOAP12Adjuncts]** SOAP Version 1.2 Part 2: Adjuncts (Second Edition)
51 <http://www.w3.org/TR/soap12-part2/>
- 52 **[WS-Addr]** <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
- 54 **[W11-SOAP12]** <http://www.w3.org/Submission/wsdl11soap12/>
- 55 **[WS-Addr-SOAP]** <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>
- 56 **[WS-MC]** <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-spec-os.html>
- 57 **[WS-Policy]** <http://www.w3.org/TR/2007/REC-ws-policy-20070904>
- 58 **[WS-PA]** <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904>
- 59
- 60
- 61

62 1.3 Non-Normative References

- 63 **[WSI-AP]** <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>
- 64 **[MTOM]** <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>
- 65 **[WS-Security]** <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- 66

67 1.4 Naming Conventions

68 This specification follows some naming conventions for artifacts defined by the
69 specification. In addition to the conventions defined by section 1.3 of the Assembly
70 **[SCA-Assembly]** specification, this specification adds three additional conventions:

- 71
72
73
74
75
76
77
78
79
80
81
82
83
1. Where the names of elements and attributes consist partially or wholly of acronyms, the letters of the acronyms use the same case. When the acronym appears at the start of the name of an element or an attribute, or after a period, it is in lower case. If it appears elsewhere in the name of an element or an attribute, it is in upper case. For example, an attribute might be named "uri" or "jndiURL".
 2. Where the names of types consist partially or wholly of acronyms, the letters of the acronyms are in all upper case. For example, an XML Schema type might be named "JCABinding" or "MessageID".
 3. Values, including local parts of QName values, follow the rules for names of elements and attributes as stated above, with the exception that the letters of acronyms are in all upper case. For example, a value might be "JMSDefault" or "namespaceURI".

2 Web Service Binding Schema

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

```
<binding.ws name="xs:NCName"?
  requires="list of xs:QName"?
  policySets="list of xs:QName"?
  uri="xs:anyURI"?
  wsdlElement="xs:anyURI"?
  wsdl:wsdlLocation="list of xs:anyURI pairs"?
  ...>
  <wireFormat/>?
  <operationSelector/>?
  <endpointReference>...</endpointReference>*
  ...
</binding.ws>
```

- **/binding.ws/@name** - as defined in the SCA Assembly Specification [SCA-Assembly].
- **/binding.ws/@requires** - as defined in the SCA Assembly Specification [SCA-Assembly].
- **/binding.ws/@policySets** - as defined in the SCA Assembly Specification [SCA-Assembly].
- **/binding.ws/@uri** - the resolution algorithm of Section 2.2 below describes how this attribute is interpreted. For an SCA reference, the @uri attribute MUST be an absolute value. [BWS20001]
- **/binding.ws/@wsdlElement** - when present this attribute specifies the URI of a WSDL element. The value of the @wsdlElement attribute MUST identify an element in an existing WSDL 1.1 document. [BWS20002] The URI can have the following forms:

- Service:

<WSDL-namespace-URI>#wsdl.service(<service-name>)

If the binding is for an SCA service, the wsdlElement attribute MUST NOT specify the wsdl.service form of URI. If the binding is for an SCA service, the wsdlElement attribute MUST NOT specify the wsdl.service form of URI. [BWS20003]

If the binding is for an SCA reference, the set of available ports for the reference consists of the ports in the WSDL service that have portTypes which are compatible supersets of the SCA reference as defined in the SCA Assembly Model specification [SCA-Assembly] and satisfy all the policy constraints of the binding.

If the wsdl.service form of wsdlElement is used on an SCA reference binding, the set of available ports for the reference MUST contain at least one port. If the wsdl.service form of wsdlElement is used on an SCA reference binding, the set of available ports for the reference MUST contain at least one port. [BWS20004]

The set of available ports represents a single SCA reference binding with respect to the multiplicity of that SCA reference. If the wsdl.service form of wsdlElement is used on an SCA reference binding, the SCA runtime MUST raise an error if there are no available ports that it supports. If the wsdl.service form of wsdlElement is used on an SCA reference binding, the SCA runtime MUST raise

131 an error if there are no available ports that it supports. [BWS20005] When an
132 invocation is made using an SCA reference binding with the *wsdl.service* form of
133 *wsdlElement*, the SCA runtime MUST use exactly one port from the set of
134 available ports for the reference (with port selection on a per-invocation basis
135 permitted). When an invocation is made using an SCA reference binding with the
136 *wsdl.service* form of *wsdlElement*, the SCA runtime MUST use exactly one port
137 from the set of available ports for the reference (with port selection on a per-
138 invocation basis permitted). [BWS20006]

139 • Port:

140 <WSDL-namespace-URI>#wsdl.port(<service-name>/<port-name>)

141 If the binding is for an SCA service, the portType associated with the specified
142 WSDL port MUST be compatible with the SCA service interface as defined in
143 section 2.1, and the port MUST satisfy all the policy constraints of the
144 binding. [BWS20007] The SCA runtime MUST expose an endpoint for the specified
145 WSDL port, or raise an error if it does not support the WSDL port. [BWS20008] If
146 the binding is for an SCA reference, the portType associated with the specified
147 WSDL port MUST be a compatible superset of the SCA reference interface as
148 defined in the SCA Assembly Model specification [SCA-Assembly], and the port
149 MUST satisfy all the policy constraints of the binding. If the binding is for an SCA
150 reference, the portType associated with the specified WSDL port MUST be a
151 compatible superset of the SCA reference interface as defined in the SCA
152 Assembly Model specification [SCA-Assembly], and the port MUST satisfy all the
153 policy constraints of the binding. [BWS20009] The SCA runtime MUST use the
154 specified WSDL port for invocations made using the SCA reference, or raise an
155 error if it does not support the WSDL port. [BWS20010]

156 • Binding:

157 <WSDL-namespace-URI>#wsdl.binding(<binding-name>)

158 If the binding is for an SCA service, the portType associated with the specified
159 WSDL binding MUST be compatible with the SCA service interface as defined in
160 section 2.1, and the WSDL binding MUST satisfy all the policy constraints of the
161 binding. [BWS20011] The SCA runtime MUST expose an endpoint for the
162 specified WSDL binding, or raise an error if it does not support the WSDL binding.
163 [BWS20012]

164 If the binding is for an SCA reference, the portType associated with the specified
165 WSDL binding MUST be a compatible superset of the SCA reference interface as
166 defined in the SCA Assembly Model specification [SCA-Assembly], and the
167 WSDL binding MUST satisfy all the policy constraints of the binding. If the binding
168 is for an SCA reference, the portType associated with the specified WSDL binding
169 MUST be a compatible superset of the SCA reference interface as defined in the
170 SCA Assembly Model specification [SCA-Assembly], and the WSDL binding
171 MUST satisfy all the policy constraints of the binding. [BWS20013] The SCA
172 runtime MUST use the specified WSDL binding for invocations made using the
173 SCA reference, or raise an error if it does not support the WSDL binding.
174 [BWS20014]

175 When the *wsdl.binding* form of *wsdlElement* is used, the endpoint address URI
176 for an SCA reference MUST be specified by either the *@uri* attribute on the
177 binding or a WS-Addressing *EndpointReference* element, except where the SCA
178 Assembly Model specification [SCA-Assembly] states that the *@uri* attribute can
179 be omitted. When the *wsdl.binding* form of *wsdlElement* is used, the endpoint
180 address URI for an SCA reference MUST be specified by either the *@uri* attribute

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

181 on the binding or a WS-Addressing *EndpointReference* element, except where the
182 SCA Assembly Model specification [SCA-Assembly] states that the @uri
183 attribute can be omitted. [BWS20015]

- **/binding.ws/@wsdl:wsdlLocation** – when present this attribute specifies the location(s) of the WSDL document(s) associated with specific namespace(s).

186 The @wsdl:wsdlLocation attribute MAY be specified by the binding in the event that
187 the <WSDL-namespace-URI> in the 'endpoint' attribute is not dereferencable, or
188 when the intended WSDL document is to be found at a different location than the
189 one pointed to by the <WSDL-namespace-URI>. The @wsdl:wsdlLocation attribute
190 MAY be specified by the binding in the event that the <WSDL-namespace-URI> in
191 the 'endpoint' attribute is not dereferencable, or when the intended WSDL document
192 is to be found at a different location than the one pointed to by the <WSDL-
193 namespace-URI>. [BWS20016]

194 If the @wsdl:wsdlLocation attribute is used the @wsdlElement attribute MUST also
195 be specified. If the @wsdl:wsdlLocation attribute is used the @wsdlElement attribute
196 MUST also be specified. [BWS20017] The semantics of this attribute are specified in
197 Section 7.1 of WSDL 2.0 [WSDL20]. The value of the @wsdl:wsdlLocation attribute
198 MUST identify an existing WSDL 1.1 document. [BWS20018]

- **/binding.ws/wireFormat** – as defined in the SCA Assembly Specification [SCA-Assembly]. This specification does not define any new wireFormat elements.
- **/binding.ws/operationSelector** – as defined in the SCA Assembly Specification [SCA-Assembly]. This specification does not define any new operationSelector elements.
- **/binding.ws/endpointReference** – when present this element provides the WS-Addressing [WS-Addr] EndpointReference that specifies the endpoint for the service or reference.
- **/binding.ws/@{any}** - this is an extensibility mechanism to allow extensibility via attributes.
- **/binding.ws/any** – this is an extensibility mechanism to allow extensibility via elements.

211 A binding.ws element MUST NOT contain more than one of any of the following: the
212 @uri attribute; the @wsdlElement attribute referring to a WSDL port or to a WSDL
213 service; the endpointReference element. [BWS20019]

214 The endpoint address URI for an SCA service or the callback element of an SCA
215 reference is determined as specified in section 2.2. For the callback element of an SCA
216 service, the binding MUST NOT specify an endpoint address URI or a WS-Addressing
217 EndpointReference. For the callback element of an SCA service, the binding MUST NOT
218 specify an endpoint address URI or a WS-Addressing EndpointReference. [BWS20020]

219 The SCA runtime MUST support all the attributes of the <binding.ws> element, namely
220 @name, @uri, @requires, @policySets, @wsdlElement, and
221 @wsdl:wsdlLocation. [BWS20021]

222 The SCA runtime SHOULD support the element <endpointReference>. [BWS20022] If an
223 SCA runtime does not support the element <endpointReference>, then it MUST reject
224 an SCA WS Binding XML document (as defined in Section 5.1) that contains the
225 element. If an SCA runtime does not support the element <endpointReference>, then it
226 MUST reject an SCA WS Binding XML document (as defined in Section 5.1) that contains
227 the element. [BWS20023]

228 The <binding.ws> element MUST conform to the XML schema defined in sca-binding-
229 webservice.xsd. [BWS20024]

230 2.1 Compatibility of SCA Service Interfaces and WSDL portTypes

231 A WSDL portType is compatible with an SCA service interface if and only if all of the
232 following conditions are satisfied:

- 233 1. The SCA service interface is remotable.
- 234 2. The operations on the portType are the same as the operations on the SCA
235 service interface, with the same operation name, same input types (taking order
236 as significant), same output types (taking order as significant), and same
237 fault/exception types. If the SCA service interface is not a WSDL portType, it is
238 mapped to a WSDL portType for the purposes of this comparison. The mapping
239 is defined in the relevant SCA specification for the interface type. If the interface
240 cannot be mapped to WSDL, the SCA service interface is not compatible with the
241 WSDL portType.
- 242 3. WSDL 1.1 message parts can point either to an XML Schema element declaration
243 or to an XML Schema type declaration. When determining compatibility between
244 two WSDL operations, a message part that points to an XML Schema element is
245 considered to be incompatible with a message part that points to an XML Schema
246 type.
- 247 4. If either the portType or the SCA service interface declares an SCA callback
248 interface, then both the portType and the SCA service interface declare callback
249 interfaces and these callback interfaces are compatible according to points 1
250 through 3 above.

251 2.2 Endpoint URI resolution

252 This specification does not mandate any particular way to determine the URI for a web
253 services binding on an SCA service. An absolute URI can be indicated by the @uri
254 attribute, by the URI in a wsa:Address element within an endpointReference element, or
255 by the URI indicated in a WSDL port via a @wsdlElement attribute. Implementations
256 can use the specified URI as the service endpoint URI or they can use a different URI
257 which might include portions of the specified URI. For example, the service endpoint
258 URI might be produced by modifying any or all of the host name, the port number, and
259 a portion of the path.

260 Note that if no absolute URI is indicated by any of these elements, implementations can
261 use the structural URI for the binding as a portion of the URI for the eventual deployed
262 endpoint. In addition, the @uri attribute value could be relative; implementations are
263 encouraged to combine this value with the structural URI for the service in determining
264 a deployed URI.

265 The target address for a reference binding is defined as one of the following:

- 266 A. The value of the @uri attribute
- 267 B. The value of the wsa:Address element of the endpointReference element
- 268 C. The value of the address element of the WSDL port referenced by the
269 @wsdlElement attribute
- 270 D. The value of the address element of one of the set of available WSDL ports as
271 specified under the definition of the @wsdlElement attribute when it references a
272 WSDL service element

273 If there is no target address for a reference binding, the SCA runtime MUST raise an
274 error. [BWS20025]

275 For a reference binding, the SCA runtime MUST use the target address. For a reference
276 binding, the SCA runtime MUST use the target address. [BWS20026]

277 2.3 Interface mapping

278 When *binding.ws* is used on a service or reference with an interface that is not defined
279 by *interface.wsdl*, the SCA runtime MUST derive a WSDL portType for the service or
280 reference from the interface using the rules defined for that SCA interface type. When
281 *binding.ws* is used on a service or reference with an interface that is not defined by
282 *interface.wsdl*, the SCA runtime MUST derive a WSDL portType for the service or
283 reference from the interface using the rules defined for that SCA interface type.
284 [BWS20027]

285 An SCA runtime MUST raise an error if the interface on a service or reference element
286 with a *binding.ws* element does not map to a WSDL portType. An SCA runtime MUST
287 raise an error if the interface on a service or reference element with a *binding.ws*
288 element does not map to a WSDL portType. [BWS20028]

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

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

294

295 2.4 Production of WSDL description for an SCA service

296 Any service hosted by an SCA runtime with one or more web service bindings with HTTP
297 endpoints SHOULD return a WSDL description of the service in response to an HTTP GET
298 request with the "?wsdl" suffix to that HTTP endpoint. Any service hosted by an SCA
299 runtime with one or more web service bindings with HTTP endpoints SHOULD return a
300 WSDL description of the service in response to an HTTP GET request with the "?wsdl"
301 suffix to that HTTP endpoint. [BWS20029]

302 If none of the web service bindings for an SCA service have HTTP endpoints, then the
303 SCA runtime SHOULD provide some other means of obtaining the WSDL description of
304 the service. If none of the web service bindings for an SCA service have HTTP endpoints,
305 then the SCA runtime SHOULD provide some other means of obtaining the WSDL
306 description of the service. [BWS20030] This can include out of band mechanisms, for
307 example publication to a UDDI registry.

308 Refer to section 4 for a detailed definition of the rules that are used for generating the
309 WSDL description of an SCA service with one or more web service bindings.

310

311 2.5 Additional binding configuration data

312 SCA runtime implementations MAY provide additional metadata that is associated with a
313 web service binding. SCA runtime implementations MAY provide additional metadata that
314 is associated with a web service binding. [BWS20031]

315 This can be used for example to enable JAX-WS **[JAX-WS]** handlers to be executed as
316 part of the target component dispatch. The specification of such metadata is SCA
317 runtime-specific and is outside of the scope of this document.

318

319 2.6 Web Service Binding and SOAP Intermediaries

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

322

323 2.7 Support for WSDL extensibility

324 When a binding.ws element uses the @wsdlElement attribute, the details of the binding
325 are specified by the WSDL element referenced by the value of the attribute. Per the
326 WSDL specification, WSDL allows for extensibility via elements as well as attributes, and
327 it specifies rules for processing such elements. This specification does not constrain the
328 use of such extensibility in WSDL and relies on the rules specified in the WSDL
329 specification for processing such extended elements.

330 **An SCA runtime MUST support the WSDL extensions defined in the namespace**
331 **associated with the prefix "sca" (as defined in section 1.1). An SCA runtime MUST**
332 **support the WSDL extensions defined in the namespace associated with the prefix "sca"**
333 **(as defined in section 1.1); [BWS20032]**

334 **The SCA runtime MUST support the WSDL 1.1 binding extension for SOAP 1.1 over HTTP**
335 **[WSDL11], as identified by the WSDL element wsoap11:binding that has the**
336 **@transport attribute with a value of "http://schemas.xmlsoap.org/soap/http". The SCA**
337 **runtime MUST support the WSDL 1.1 binding extension for SOAP 1.1 over HTTP**
338 **[WSDL11], as identified by the WSDL element wsoap11:binding that has the**
339 **@transport attribute with a value of "http://schemas.xmlsoap.org/soap/http";**
340 **[BWS20033]**

Formatted: Highlight

341 **The SCA runtime SHOULD support the WSDL 1.1 binding extension for SOAP 1.2 over**
342 **HTTP [W11-SOAP12], as identified by the WSDL element wsoap12:binding that has**
343 **the @transport attribute with a value of "http://schemas.xmlsoap.org/soap/http". The**
344 **SCA runtime SHOULD support the WSDL 1.1 binding extension for SOAP 1.2 over HTTP**
345 **[W11-SOAP12], as identified by the WSDL element wsoap12:binding that has the**
346 **@transport attribute with a value of "http://schemas.xmlsoap.org/soap/http";**
347 **[BWS20034]**

Formatted: Highlight

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

352 2.8 Intents listed in the bindingType

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

355 2.9 Intents and binding configuration

356 This binding mandates support for SOAP 1.1 and encourages SOAP 1.2 support. **The**
357 **<bindingType> element associated with this binding MUST include the SOAP_1_1 intent**
358 **in its @mayProvides or @alwaysProvides attributes. [BWS20035] The <bindingType>**

359 element associated with this binding SHOULD include the SOAP.1_2 intent in its
360 @mayProvides attribute. [BWS20036] For more details on the <bindingType> element
361 see [SCA-Policy].

362 | The SCA runtime MUST raise an error if a web service binding is configured with a policy
363 | intent(s) that conflicts with the binding instance's configuration. The SCA runtime MUST
364 | raise an error if a web service binding is configured with a policy intent(s) that conflicts
365 | with the binding instance's configuration. [BWS20037]

366 For example, it is an error to use the SOAP policy intent in combination with a WSDL
367 binding that does not use SOAP.

368 3 Web Service Binding Examples

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

372

373 3.1 Example Using WSDL documents

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

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

379 The first reference's binding is defined by the specified WSDL service in the WSDL
380 document at the given location. The reference can use any of the WSDL service's ports
381 to invoke the target service. The second reference's binding is defined by the specified
382 WSDL binding. The specific endpoint URI to be invoked is provided via the *@uri*
383 attribute.

384

```
385 <?xml version="1.0" encoding="ASCII"?>
386 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
387           name="MyValueComposite">
388   <service name="MyValueService">
389     <interface.java interface="services.myvalue.MyValueService"/>
390     <binding.ws wsdlElement="http://www.example.org/MyValueService#
391
392 wsdl.binding(MyValueService/MyValueServiceSOAP)"/>
393     ...
394   </service>
395   ...
396   ...
397
398   <reference name="StockQuoteReference1">
399     <interface.java interface="services.stockquote.StockQuoteService"/>
400     <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
401                 wsdl.service(StockQuoteService) "
402     wsdl:wsdlLocation="http://www.example.org/StockQuoteService
403                       http://www.example.org/StockQuoteService.wsdl"/>
404   </reference>
405
406   <reference name="StockQuoteReference2">
407     <interface.java interface="services.stockquote.StockQuoteService"/>
408     <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
409                 wsdl.binding(StockQuoteBinding) "
410     wsdl:wsdlLocation="http://www.example.org/StockQuoteService
411                       http://www.example.org/StockQuoteService.wsdl"
412     uri="http://www.example.org/StockQuoteService5"/>
413   </reference>
414 </composite>
```

415 3.2 Examples Without a WSDL Document

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

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

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

428

```
429 <?xml version="1.0" encoding="ASCII"?>
430 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
431           name="MyValueComposite">
432
433   <service name="MyValueService">
434     <interface.java interface="services.myvalue.MyValueService"/>
435     <binding.ws/>
436     ...
437   </service>
438
439   ...
440
441   <reference name="StockQuoteService">
442     <interface.java interface="services.stockquote.StockQuoteService"/>
443     <binding.ws uri="http://www.example.org/StockQuoteService"/>
444   </reference>
445 </composite>
```

446

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

452

```
453 <?xml version="1.0" encoding="ASCII"?>
454 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
455           name="MyValueComposite">
456
457   <service name="MyValueService">
458     <interface.java interface="services.myvalue.MyValueService"/>
459     <binding.ws name="MyValueServiceSOAP11" requires="SOAP.1_1"/>
460     <binding.ws requires="SOAP.1_2"/>
461     ...
462   </service>
463
464   ...
465
466   <reference name="StockQuoteService">
467     <interface.java interface="services.stockquote.StockQuoteService"/>
468     <binding.ws uri="http://www.example.org/StockQuoteService"
469               requires="SOAP.1_2"/>
470   </reference>
```

471

```
</composite>
```

472

473 4 Transport Binding

474 The binding.ws element provides numerous ways to specify exactly how messages ought
475 to be transmitted from or to the reference or service. Those ways include references to
476 WSDL binding elements from the @wsdlElement attribute, policy intents, and even
477 vendor extensions within the binding.ws element. This section describes the defaults to
478 be used if the specific transport details are not otherwise specified.

479 4.1 Intents

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

- 482 • SOAP
483 **When the SOAP intent is required, the SCA runtime MUST transmit and receive**
484 **messages using SOAP. One or more SOAP versions can be used. When the SOAP**
485 **intent is required, the SCA runtime MUST transmit and receive messages using**
486 **SOAP. One or more SOAP versions can be used. [BWS40001]**
- 487 • SOAP.1_1
488 **When the SOAP.1_1 intent is required, the SCA runtime MUST transmit and receive**
489 **messages using only SOAP 1.1. When the SOAP.1_1 intent is required, the SCA**
490 **runtime MUST transmit and receive messages using only SOAP 1.1. [BWS40002]**
- 491 • SOAP.1_2
492 **When the SOAP.1_2 intent is required, the SCA runtime MUST transmit and receive**
493 **messages using only SOAP 1.2. When the SOAP.1_2 intent is required, the SCA**
494 **runtime MUST transmit and receive messages using only SOAP 1.2. [BWS40003]**

495 4.2 Default Transport Binding Rules

496 4.2.1 WS-I Basic Profile Alignment

497 To align to WS-I Basic Profile, the resulting WSDL port needs to be all document-literal,
498 or all rpc-literal binding (per WS-I Basic Profile 1.1 R2705 **[WSI-Profiles]**). This means,
499 for any given portType, for all messages referenced by all operations in that portType,
500 either

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

504 **For an SCA service or reference element, the portType from the service's or reference's**
505 **interface or derived from that interface MUST follow either the rpc-literal pattern or the**
506 **document-literal pattern. For an SCA service or reference element, the portType from the**
507 **service's or reference's interface or derived from that interface MUST follow either the**
508 **rpc-literal pattern or the document-literal pattern. [BWS40004]**

509 The rest of this section assumes the short-hand reference of a "rpc-literal" or
510 "document-literal" pattern, depending on which of the two bullet points above it
511 matches.

512 4.2.2 Default Transport Binding Rules

513 The following defines the **default transport binding rules** for the Web Service binding:

- 514
- HTTP-based transfer protocol;
- 515
- SOAP 1.1 binding;
- 516
- "literal" format as described in section 3.5 of **[WSDL11]**;
- 517
- Either the document literal or rpc literal pattern, depending on the service or
- 518
- reference interface as described in section 4.2.1;
- 519
- For document literal pattern, each message uses "document" style, as per
- 520
- section 3.5 of **[WSDL11]**;
- 521
- For rpc-literal pattern, each message uses "rpc" style, as per section 3.5
- 522
- of **[WSDL11]** and the child elements of the SOAP Body element are
- 523
- namespace qualified with a non-empty namespace name;
- 524
- For SOAP 1.1 messages, the SOAPAction HTTP header described in section 6.1.1
- 525
- of **[SOAP11]** represents the empty string, in quotes ("");
- 526
- For SOAP 1.2 messages, the SOAP Action feature described in section 6.5 of
- 527
- [SOAP12Adjuncts]** does not appear;
- 528
- All WSDL message parts are carried in the SOAP body.

529 ~~In the event that the transport details are not otherwise determined, an SCA runtime~~
530 ~~MUST enable the default transport binding rules. In the event that the transport details~~
531 ~~are not otherwise determined, an SCA runtime MUST enable the default transport~~
532 ~~binding rules. [BWS40005]~~

533 ~~When using the default transport binding rules, the SCA runtime MAY provide additional~~
534 ~~WSDL bindings, unless policy is applied that explicitly restricts this. When using the~~
535 ~~default transport binding rules, the SCA runtime MAY provide additional WSDL bindings,~~
536 ~~unless policy is applied that explicitly restricts this. [BWS40006]~~

537 ~~When using the default transport binding rules with the rpc-literal pattern, the SCA~~
538 ~~runtime SHOULD use the structural URI associated with the binding as the namespace of~~
539 ~~the child elements of the SOAP body element. When using the default transport binding~~
540 ~~rules with the rpc-literal pattern, the SCA runtime SHOULD use the structural URI~~
541 ~~associated with the binding as the namespace of the child elements of the SOAP body~~
542 ~~element. [BWS40007]~~

543 **5 Implementing SCA Callbacks using Web Services**

544 **5.1 SCA Web Services Callback Protocol**

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

548 To implement the SCA Web Services Callback Protocol, an SCA binding follows the
549 following rules.

550 1. Every request message that invokes the forward interface MUST contain a
551 Callback EPR. [BWS50002] If the request message contains the `wsa:From` SOAP
552 header block then the `wsa:From` header block specifies the Callback EPR. If the
553 `wsa:From` header block is not present then the `wsa:ReplyTo` header block
554 specifies the Callback EPR.

555 If the Callback EPR's [address] value is
556 "<http://www.w3.org/2005/08/addressing/anonymous>" or
557 "<http://www.w3.org/2005/08/addressing/none>" then the SCA runtime MUST
558 generate the Invalid Addressing Header fault as specified in Section 6.4.1 of
559 [WS-Addr-SOAP]. [BWS50004] Such a fault can include additional
560 [Subsubcode] `wsa:OnlyNonAnonymousAddressSupported`.

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

565 3. When the service implementation invokes the callback interface, it MUST use the
566 Callback EPR from a request message that invoked the forward interface, as
567 specified in BWS50003. [BWS50005] Once the Callback EPR is selected, the SCA
568 runtime MUST follow the rules defined in Section 3.3 of [WS-Addr] to invoke
569 operations on the callback interface. [BWS50006]

570 When the service invokes the callback interface, if the request message from which the
571 Callback EPR was obtained contained the `wsa:MessageID` SOAP header block, the SCA
572 runtime MUST include a `wsa:RelatesTo` SOAP header block in the callback message.
573 [BWS50007] The `wsa:RelatesTo` SOAP header block MUST have the relationship type
574 value of "<http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback>" and
575 the related message id MUST be the `wsa:MessageID` of the message from which the
576 Callback EPR was obtained. [BWS50008]

577 If the request message from which the Callback EPR was obtained did not contain the
578 `wsa:MessageID` SOAP header block, the SCA runtime MUST NOT include a
579 `wsa:RelatesTo` SOAP header block with a relationship type value of
580 "<http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback>" in the callback
581 message. [BWS50009]

582 When a service that offers a bidirectional interface is invoked, depending on the
583 semantics and/or implementation of the service, it is possible that the service might
584 invoke the callback interface before the forward operation ends. In such cases, it is
585 necessary for the binding on the reference-side to be listening for callback request(s)
586 from the service, before the forward operation request is sent on the wire to the service.

Formatted: Highlight

Formatted

Formatted: Font color: Red

Formatted: Highlight

Formatted: Font color: Red

Formatted: Highlight

Formatted: Highlight

Formatted: Font color: Red

587 and continue listening as long as callback requests are expected. It is possible that
588 before the response to the forward request is sent a response to one or more callback
589 requests are required by the service.

590 **5.2 SCA Web Services Callback with WS-MakeConnection Protocol**

591 It is possible that the invoker of a service that uses a bidirectional interface has a
592 binding that cannot accept connections for callbacks from a service (for example, when
593 it has the `noListener` intent [SCA-Policy]). When this is the case, it is necessary for
594 the binding to support a polling mechanism. An example of a polling mechanism is WS-
595 MakeConnection [WS-MC]. This section describes the use of the SCA Web Services
596 Callback Protocol in conjunction with WS-MakeConnection. For examples of wire
597 messages exchanged when using the SCA Web Services Callback protocol in conjunction
598 with WS-MakeConnection see Appendix E.1.

599 When an SCA runtime implements the SCA Web Services Callback protocol in
600 conjunction with WS-MakeConnection, it has to adhere to the rules described for the
601 SCA Web Services Callback Protocol and also to those of WS-MakeConnection.

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

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

608 When the service implementation invokes the callback interface, it uses the Callback EPR
609 from a request message that invoked the forward interface, and the callback request
610 message is sent as the response to a `wsmc:MakeConnection` message that contains the
611 `wsmc:Address` value that matches the MakeConnection Anonymous URI in the Callback
612 EPR.

613 When a service that offers a bidirectional interface is invoked using WS-MakeConnection
614 Anonymous URI as the value for the Callback EPR address, depending on the semantics
615 and/or implementation of the service, it is possible that the service might invoke the
616 callback interface before the forward operation ends. In such cases, it is necessary for
617 the binding on the reference-side to start polling for callback request(s) from the
618 service, before or right after the forward operation request is sent and before a response
619 is received, and continue polling as long as callback requests are expected. It is possible
620 that before the response to the forward request is sent a response to one or more
621 callback requests are required by the service.

622 **5.3 Policy Assertion for SCA Web Services Callback Protocol**

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

629 **5.3.1 Assertion Model**

630 The WSCallback policy assertion indicates that the Web service client and the Web
631 service MUST use SCA Web Services Callback Protocol to implement callbacks.
632 [BWS50010] Specifically, the protocol determines the requirements on forward request

Formatted: Highlight

Formatted: Font color: Red

633 [message, the EPR used for callbacks and the requirements on the callback request](#)
634 [message.](#)

635 **5.3.2 Normative Outline**

636 The normative outline for the WSCallback assertion is:

```
637 <sca:WSCallback ...>  
638 ....  
639 </sca:WSCallback>
```

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

- 642 • [/sca:WSCallback](#): A policy assertion that specifies that WSCallback protocol is
643 used when sending messages.

644 **5.3.3 Assertion Attachment**

645 The WSCallback policy assertion is allowed to have the following Policy Subjects [**WS-**
646 **PA**]:

- 647 • Endpoint Policy Subject

648 [WS-PolicyAttachment](#) defines a set of WSDL/1.1 policy attachment points for each of the
649 above Policy Subjects. [Since a WSCallback policy assertion specifies a concrete behavior,](#)
650 [it MUST NOT be attached to the abstract WSDL policy attachment points.](#) [BWS50012]

651 [The following is the list of WSDL/1.1 elements whose scope contains the Policy Subjects](#)
652 [allowed for a WSCallback policy assertion but which MUST NOT have WSCallback policy](#)
653 [assertions attached: wsdl:portType](#) [BWS50013].

654 The following is the list of WSDL/1.1 elements whose scope contains the Policy Subjects
655 allowed for a WSCallback policy assertion and which can have WSCallback policy
656 assertions attached:

- 657 • [wsdl:port](#)
- 658 • [wsdl:binding](#)

659 **5.3.4 Assertion Example**

660 The example below shows the use of the WSCallback policy assertion in a WSDL
661 document.

```
662  
663 (01) <wsdl:definitions  
664 (02)   targetNamespace="example.com"  
665 (03)   xmlns:tns="example.com"  
666 (04)   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  
667 (05)   xmlns:wsp="http://www.w3.org/ns/ws-policy"  
668 (06)   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"  
669 (07)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
670 wssecurity-utility-1.0.xsd">  
671 (08)  
672 (09) <wsp:UsingPolicy wsdl:required="true" />  
673 (10)  
674 (11) <wsp:Policy wsu:Id="MyPolicy" >  
675 (12)   <sca:WSCallback/>  
676 (13) </wsp:Policy>  
677 (14)
```

Formatted: Highlight

Formatted: Font color: Red

Formatted: Highlight

Formatted: Highlight

Formatted: Font color: Auto

```
678 (15) <!-- omitted elements -->
679 (16)
680 (17) <wsdl:binding name="MyBinding" type="tns:MyPortType" >
681 (18)   <wsp:PolicyReference URI="#MyPolicy" />
682 (19)   <!-- omitted elements -->
683 (20) </wsdl:binding>
684 (21)
685 (22)</wsdl:definitions>
```

687 Line (09) in the example above indicates that WS-Policy is in use as a required
688 extension. Lines (11-13) are a policy expression that includes a WSCallback policy
689 assertion (line 12) to indicate that SCA Web Services Callback protocol is used. Lines
690 (17-20) are a WSDL binding. Line (18) indicates that the policy in lines (11-13) applies
691 to this binding, specifically indicating that SCA Web Services Callback protocol is used
692 over all the messages in the binding.

693 **5.3.5 Security Considerations**

694 Policies and assertions SHOULD be signed to prevent tampering. [BWS50014] Policies
695 SHOULD NOT be accepted unless they are signed and have an associated security token
696 to specify the signer has proper claims for the given policy. [BWS50015] That is, a
697 relying party shouldn't rely on a policy unless the policy is signed and presented with
698 sufficient claims to pass the relying parties acceptance criteria.

699 Note that the mechanisms described in this document could be secured as part of a
700 SOAP message using WS-Security [WS-Security] or embedded within other objects
701 using object-specific security mechanisms.

702

- Formatted: Highlight
- Formatted: Font color: Red
- Formatted: Highlight
- Formatted: Font color: Red

703 **56** Conformance

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

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

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

710 **5.16.1 SCA WS Binding XML Document**

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

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

718 **5.26.2 SCA Runtime**

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

- 721 1. The implementation **MUST** comply with all statements in Appendix B:
722 Conformance Items related to an SCA Runtime, except for those that originate
723 from Section 5, notably all "MUST" statements have to be implemented.
- 724 2. The implementation MAY support the SCA Web Services Callback Protocol. If it
725 does, it MUST comply with all statements in Appendix B: Conformance Items for
726 the SCA Web Services Callback Protocol.
- 727 ~~1-3.~~ The implementation MAY support the SCA Web Services Callback in
728 conjunction with the WS-MakeConnection Protocol. If it does, it MUST comply
729 with all statements in Appendix B: Conformance Items for the SCA Web Services
730 Callback and it MUST comply with the requirements of WS-MakeConnection
731 Protocol.
- 732 ~~2-4.~~ The implementation **MUST** conform to the SCA Assembly Model Specification
733 Version 1.1 **[SCA-Assembly]**, and to the SCA Policy Framework Version 1.1
734 **[SCA-Policy]**.
- 735 ~~3-5.~~ The implementation **MUST** reject a SCA WS Binding XML Document that is not
736 conformant per Section ~~6~~**5.1**.

737
738

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

739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (C) OASIS 2005, 2009. All Rights Reserved.
      OASIS trademark, IPR and other policies apply.-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200903"
  xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"
  xmlns:wsdl="http://www.w3.org/ns/wsdl-instance"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  elementFormDefault="qualified">

  <import namespace="http://www.w3.org/ns/wsdl-instance"
    schemaLocation="http://www.w3.org/2007/05/wsdl/wsdl20-
instance.xsd"
  />
  <import namespace="http://www.w3.org/2005/08/addressing"
    schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"
  />
  <include schemaLocation="sca-core-1.1-cd03.xsd"/>

  <element name="binding.ws" type="sca:WebServiceBinding"
    substitutionGroup="sca:binding"/>
  <complexType name="WebServiceBinding">
    <complexContent>
      <extension base="sca:Binding">
        <sequence>
          <element ref="sca:wireFormat"
            minOccurs="0" maxOccurs="1" />
          <element ref="sca:operationSelector"
            minOccurs="0" maxOccurs="1" />
          <element name="endpointReference"
            type="wsa:EndpointReference"
            minOccurs="0" maxOccurs="unbounded"/>
          <any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="wsdlElement" type="anyURI" use="optional"/>
        <attribute ref="wsdl:wsdlLocation" use="optional"/>
        <anyAttribute namespace="##any" processContents="lax"/>
      </extension>
    </complexContent>
  </complexType>
</schema>
```

784
785
786

787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805

806
807

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 2005, 2009. All Rights Reserved.
      OASIS trademark, IPR and other policies apply
-->
<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>
```

- Formatted: Font: Courier New
- Formatted: Font: Courier New
- Formatted: Font: Courier New, English (U.S.)
- Formatted: Font: Courier New

808

B.C. Conformance Items

809

This section contains a list of conformance items for the SCA Web Service Binding specification.

Conformance ID	Description
[BWS20001][BWS20004]	For an SCA reference, the @uri attribute MUST be an absolute value.
[BWS20002]	The value of the @wsdlElement attribute MUST identify an element in an existing WSDL 1.1 document.
[BWS20003]	If the binding is for an SCA service, the wsdlElement attribute MUST NOT specify the wsdl.service form of URI.
[BWS20004]	If the wsdl.service form of wsdlElement is used on an SCA reference binding, the set of available ports for the reference MUST contain at least one port.
[BWS20005][BWS20005]	If the wsdl.service form of wsdlElement is used on an SCA reference binding, the SCA runtime MUST raise an error if there are no available ports that it supports.
[BWS20006]	When an invocation is made using an SCA reference binding with the wsdl.service form of wsdlElement, the SCA runtime MUST use exactly one port from the set of available ports for the reference (with port selection on a per-invocation basis permitted).
[BWS20007][BWS20007]	If the binding is for an SCA service, the portType associated with the specified WSDL port MUST be compatible with the SCA service interface as defined in section 2.1, and the port MUST satisfy all the policy constraints of the binding.
[BWS20008]	The SCA runtime MUST expose an endpoint for the specified WSDL port, or raise an error if it does not support the WSDL port.
[BWS20009][BWS20009]	If the binding is for an SCA reference, the portType associated with the specified WSDL port MUST be a compatible superset of the SCA reference interface as defined in the SCA Assembly Model specification [SCA-Assembly][SCA-Assembly], and the port MUST satisfy all the policy constraints of the binding.
[BWS20010][BWS20010]	The SCA runtime MUST use the specified WSDL port for invocations made using the SCA reference, or raise an error if it does not support the WSDL port.
[BWS20011]	If the binding is for an SCA service, the portType associated with the specified WSDL binding MUST be compatible with the SCA service interface as defined in section 2.1, and the WSDL binding MUST satisfy all the policy constraints of the binding.
[BWS20012]	The SCA runtime MUST expose an endpoint for the specified WSDL binding, or raise an error if it does not support the WSDL binding.
[BWS20013]	If the binding is for an SCA reference, the portType associated with the specified WSDL binding MUST be a compatible superset of the SCA reference interface as defined in the SCA Assembly Model specification [SCA-Assembly][SCA-Assembly], and the WSDL binding MUST satisfy all the policy constraints of the binding.

Formatted: Highlight

Formatted: Highlight

[BWS20014]	The SCA runtime MUST use the specified WSDL binding for invocations made using the SCA reference, or raise an error if it does not support the WSDL binding.
[BWS20015]	When the <i>wSDL.binding</i> form of <i>wSDL.Element</i> is used, the endpoint address URI for an SCA reference MUST be specified by either the <i>@uri</i> attribute on the binding or a WS-Addressing <i>EndpointReference</i> element, except where the SCA Assembly Model specification [SCA-Assembly][SCA-Assembly] states that the <i>@uri</i> attribute can be omitted.
[BWS20016]	The <i>@wsdl:wsdlLocation</i> attribute MAY be specified by the binding in the event that the <i><WSDL-namespace-URI></i> in the 'endpoint' attribute is not dereferencable, or when the intended WSDL document is to be found at a different location than the one pointed to by the <i><WSDL-namespace-URI></i> .
[BWS20017]	If the <i>@wsdl:wsdlLocation</i> attribute is used the <i>@wSDL.Element</i> attribute MUST also be specified.
[BWS20018]	The value of the <i>@wsdl:wsdlLocation</i> attribute MUST identify an existing WSDL 1.1 document.
[BWS20019]	A <i>binding.ws</i> element MUST NOT contain more than one of any of the following: the <i>@uri</i> attribute; the <i>@wSDL.Element</i> attribute referring to a WSDL port or to a WSDL service; the <i>endpointReference</i> element.
[BWS20020]	For the <i>callback</i> element of an SCA service, the binding MUST NOT specify an endpoint address URI or a WS-Addressing <i>EndpointReference</i> .
[BWS20021]	The SCA runtime MUST support all the attributes of the <i><binding.ws></i> element, namely <i>@name</i> , <i>@uri</i> , <i>@requires</i> , <i>@policySets</i> , <i>@wSDL.Element</i> , and <i>@wsdl:wsdlLocation</i> .
[BWS20022]	The SCA runtime SHOULD support the element <i><endpointReference></i> .
[BWS20023]	If an SCA runtime does not support the element <i><endpointReference></i> , then it MUST reject an SCA WS Binding XML document (as defined in Section 5.1) that contains the element.
[BWS20024]	The <i><binding.ws></i> element MUST conform to the XML schema defined in <i>sca-binding-webservice.xsd</i> .
[BWS20025]	If there is no target address for a reference binding, the SCA runtime MUST raise an error.
[BWS20026]	For a reference binding, the SCA runtime MUST use the target address.
[BWS20027]	When <i>binding.ws</i> is used on a service or reference with an interface that is not defined by <i>interface.wSDL</i> , the SCA runtime MUST derive a WSDL <i>portType</i> for the service or reference from the interface using the rules defined for that SCA interface type.
[BWS20028]	An SCA runtime MUST raise an error if the interface on a service or reference element with a <i>binding.ws</i> element does not map to a WSDL <i>portType</i> .
[BWS20029]	Any service hosted by an SCA runtime with one or more web service bindings with HTTP endpoints SHOULD return a WSDL description of the service in response to an HTTP GET request with the "?wsdl!" suffix to that HTTP endpoint.
[BWS20030]	If none of the web service bindings for an SCA service have HTTP endpoints, then the SCA runtime SHOULD provide some other means of obtaining the

Formatted: Highlight

	WSDL description of the service.
[BWS20031]	SCA runtime implementations MAY provide additional metadata that is associated with a web service binding.
[BWS20032]	An SCA runtime MUST support the WSDL extensions defined in the namespace associated with the prefix "sca" (as defined in section 1.1).
[BWS20033]	The SCA runtime MUST support the WSDL 1.1 binding extension for SOAP 1.1 over HTTP [WSDL11][WSDL11] , as identified by the WSDL element wsoap11:binding that has the @transport attribute with a value of "http://schemas.xmlsoap.org/soap/http".
[BWS20034]	The SCA runtime SHOULD support the WSDL 1.1 binding extension for SOAP 1.2 over HTTP [W11-SOAP12][W11-SOAP12] , as identified by the WSDL element wsoap12:binding that has the @transport attribute with a value of "http://schemas.xmlsoap.org/soap/http".
[BWS20035]	The <bindingType> element associated with this binding MUST include the SOAP.1_1 intent in its @mayProvides or @alwaysProvides attributes.
[BWS20036]	The <bindingType> element associated with this binding SHOULD include the SOAP.1_2 intent in its @mayProvides attribute.
[BWS20037]	The SCA runtime MUST raise an error if a web service binding is configured with a policy intent(s) that conflicts with the binding instance's configuration.
[BWS40001]	When the SOAP intent is required, the SCA runtime MUST transmit and receive messages using SOAP. One or more SOAP versions can be used.
[BWS40002] [BWS40002]	When the SOAP.1_1 intent is required, the SCA runtime MUST transmit and receive messages using only SOAP 1.1.
[BWS40003] [BWS40003]	When the SOAP.1_2 intent is required, the SCA runtime MUST transmit and receive messages using only SOAP 1.2.
[BWS40004]	For an SCA service or reference element, the portType from the service's or reference's interface or derived from that interface MUST follow either the rpc-literal pattern or the document-literal pattern.
[BWS40005]	In the event that the transport details are not otherwise determined, an SCA runtime MUST enable the default transport binding rules.
[BWS40006]	When using the default transport binding rules, the SCA runtime MAY provide additional WSDL bindings, unless policy is applied that explicitly restricts this.
[BWS40007]	When using the default transport binding rules with the rpc-literal pattern, the SCA runtime SHOULD use the structural URI associated with the binding as the namespace of the child elements of the SOAP body element.
[BWS50002]	Every request message that invokes the forward interface MUST contain a Callback EPR.
[BWS50004]	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 Section 6.4.1 of [WS-Addr-SOAP] .
[BWS50005]	When the service implementation invokes the callback interface, it MUST use the Callback EPR from a request message that invoked the forward interface.

Formatted: Highlight

Formatted: Highlight

Formatted: Font color: Red

Formatted: Highlight

Formatted: Font color: Red

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

	as specified in BWS50003.
[BWS50006]	Once the Callback EPR is selected, the SCA runtime MUST follow the rules defined in Section 3.3 of [WS-Addr] to invoke operations on the callback interface.
[BWS50007]	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.
[BWS50008]	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.
[BWS50009]	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.
[BWS50010]	The WSCallback policy assertion indicates that the Web service client and the Web service MUST use SCA Web Services Callback Protocol to implement callbacks.
[BWS50012]	Since a WSCallback policy assertion specifies a concrete behavior, it MUST NOT be attached to the abstract WSDL policy attachment points.
[BWS50013]	The following is the list of WSDL/1.1 elements whose scope contains the Policy Subjects allowed for a WSCallback policy assertion but which MUST NOT have WSCallback policy assertions attached: wsd:portType
[BWS50014]	Policies and assertions SHOULD be signed to prevent tampering.
[BWS50015]	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.

- Formatted: Font color: Red
- Formatted: Highlight
- Formatted: Highlight
- Formatted: Highlight
- Formatted: Font color: Red
- Formatted: Highlight
- Formatted: Highlight
- Formatted: Font color: Red
- Formatted: Highlight
- Formatted: Font color: Red
- Formatted: Highlight
- Formatted: Font color: Red
- Formatted: Highlight

810

C.D. Appendix - WSDL Generation

811

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

812

813

814

815

816

817

- component name = the value of the @name attribute of the component element containing the binding.ws element

818

819

- service name = the value of the @name attribute of the service element containing the binding.ws element

820

821

- binding name = the value of @name attribute of the binding.ws element, or the default if no @name attribute is present

822

- SOAP version = either "SOAP11" or "SOAP12" as appropriate

823

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

824

- wsdl:definitions/@name = <component name> + "." + <service name>

825

- wsdl:definitions/@targetNamespace = <structural URI for the service>

826

- import each WSDL 1.1 portType, rather than putting them inline

827

- wsdl:binding/@name = <binding name> + <SOAP version> + "Binding"

828

- wsdl:service/@name = <service name>

829

- wsdl:port/@name = <binding name> + <SOAP version> + "Port"

830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869

E. 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. The forward interface and the callback interface both contain a single 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 difference 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:

```
870 <soap:Envelope ...>
871 <soap:Header>
872 <wsa:To>http://example.com/callback</wsa:To>
873 <myNS:SomeID>1</myNS:SomeID>
874 <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
875 bindings/ws/callback">urn:uuid:f81d4fae-7dec-11d0-a765-
876 00a0c91e6bf6</wsa:RelatesTo>
877 ...
878 </soap:Header>
879 <soap:Body>
880 ...
881 </soap:Body>
882 </soap:Envelope>
883
```

884

885

R2:

```
886 <soap:Envelope ...>
887 <soap:Header>
888 <wsa:From>
889 <wsa:Address>http://example.com/callback</wsa:Address>
890 <wsa:ReferenceProperties>
891 <myNS:SomeID>1</myNS:SomeID>
892 </wsa:ReferenceProperties>
893 </wsa:From>
894 <wsa:MessageID>urn:uuid:f81d4fae-8dec-11d0-a765-
895 00a0c91e6bf6</wsa:messageID>
896 ...
897 </soap:Header>
898 <soap:Body>
899 ...
900 </soap:Body>
901 </soap:Envelope>
902
```

903

904

S3:

```
905 <soap:Envelope ...>
906 <soap:Header>
907 <wsa:To>http://example.com/callback</wsa:To>
908 <myNS:SomeID>1</myNS:SomeID>
909 <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
910 bindings/ws/callback">
911 urn:uuid:f81d4fae-8dec-11d0-a765-00a0c91e6bf6
912 </wsa:RelatesTo>
913 ...
914 </soap:Header>
915 <soap:Body>
916 ...
917 </soap:Body>
918 </soap:Envelope>
919
```

919

920

R3:

```
921 <soap:Envelope ...>
922 <soap:Header>
923 <wsa:From>
924 <wsa:Address>http://example.com/callback-other</wsa:Address>
925 <wsa:ReferenceProperties>
926 <myNS:SomeID>2</myNS:SomeID>
927 </wsa:ReferenceProperties>
928 </wsa:From>
929 <wsa:MessageID>urn:uuid:f81d4fae-9dec-11d0-a765-
930 00a0c91e6bf6</wsa:messageID>
931 ...
932 </soap:Header>
933 <soap:Body>
934 ...
935 </soap:Body>
936 </soap:Envelope>
```

S4, S5:

```
941 <soap:Envelope ...>
942 <soap:Header>
943 <wsa:To>http://example.com/callback-other</wsa:To>
944 <myNS:SomeID>2</myNS:SomeID>
945 <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
946 bindings/ws/callback">urn:uuid:f81d4fae-9dec-11d0-a765-
947 00a0c91e6bf6</wsa:RelatesTo>
948 ...
949 </soap:Header>
950 <soap:Body>
951 ...
952 </soap:Body>
953 </soap:Envelope>
```

E.1 Message Examples Using WS-MakeConnection

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.

R4:

```
964 <soap:Envelope ...>
965 <soap:Header>
966 <wsa:From>
967 <wsa:Address>http://docs.oasis-open.org/ws-
968 rx/wsmc/200702/anonymous?id=650e8400-f29b-11d4-a716-446655440010</wsa:Address>
969 </wsa:From>
970 <wsa:MessageID>urn:uuid:f81d4fae-10dec-11d0-a765-
971 00a0c91e6bf6</wsa:messageID>
972 ...
973 </soap:Header>
974 <soap:Body>
975 ...
976 </soap:Body>
977 </soap:Envelope>
```

978

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

```
980 <soap:Envelope ...>
981 <soap:Header>
982 <wsa:Action>http://docs.oasis-open.org/ws-
983 rx/wsmc/200702/MakeConnection</wsa:Action>
984 ...
985 </soap:Header>
986 <soap:Body>
987 <wsmc:MakeConnection>
988 <wsmc:Address>http://docs.oasis-open.org/ws-
989 rx/wsmc/200702/anonymous?id=650e8400-f29b-11d4-a716-
990 446655440010</wsmc:Address>
991 </wsmc:MakeConnection>
992 </soap:Body>
993 </soap:Envelope>
```

994

995 **S6:**

```
996 <soap:Envelope ...>
997 <soap:Header>
998 <wsa:To>http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id=650e8400-
999 f29b-11d4-a716-446655440010</wsa:To>
1000 <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
1001 bindings/ws/callback">urn:uuid:f81d4fae-10dec-11d0-a765-
1002 00a0c91e6bf6</wsa:RelatesTo>
1003 ...
1004 </soap:Header>
1005 <soap:Body>
1006 ...
1007 </soap:Body>
1008 </soap:Envelope>
```

1009

D.F. Acknowledgements

1010

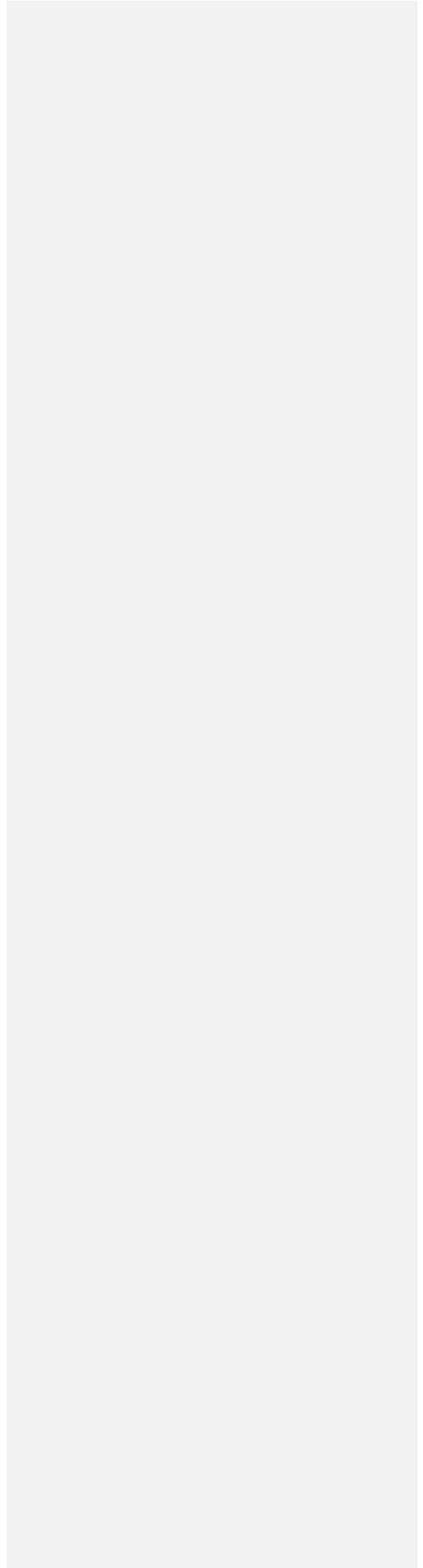
The following individuals have participated in the creation of this specification and are gratefully acknowledged:

1011

1012

Participants:

Participant Name	Affiliation
Bryan Aupperle	IBM
Ron Barack	SAP AG
Michael Beisiegel	IBM
Henning Blohm	SAP AG
David Booz	IBM
Martin Chapman	Oracle Corporation
Jean-Sebastien Delfino	IBM
Laurent Domenech	TIBCO Software Inc.
Jacques Durand	Fujitsu Limited
Mike Edwards	IBM
Billy Feng	Primeton Technologies, Inc.
Nimish Hathalia	TIBCO Software Inc.
Simon Holdsworth	IBM
Eric Johnson	Software Inc.
Uday Joshi	Oracle Corporation
Khanderao Kand	Oracle Corporation
Anish Karmarkar	Oracle Corporation
Nickolaos Kavantzias	Oracle Corporation
Mark Little	Red Hat
Ashok Malhotra	Oracle Corporation
Jim Marino	Individual
Jeff Mischkinsky	Oracle Corporation
Dale Moberg	Axway Software
Simon Nash	Individual
Sanjay Patil	SAP AG
Plamen Pavlov	SAP AG
Peter Peshev	SAP AG
Piotr Przybylski	IBM
Luciano Resende	IBM
Tom Rutt	Fujitsu Limited
Vladimir Savchenko	SAP AG
Scott Vorthmann	TIBCO Software Inc.
Tim Watson	Oracle Corporation
Owen Williams	Avaya, Inc.
Prasad Yendluri	Software AG, Inc.



1014

E.G. Revision History

1015 [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	<ul style="list-style-type: none"> * 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	<ul style="list-style-type: none"> * Completed application of resolution to issue 10 * Applied most of the editorial changes from Eric Johnson's review
4	2008-08-13	Anish Karmarkar	<ul style="list-style-type: none"> * 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	<ul style="list-style-type: none"> 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

cd02-rev1	2009-06-02	Anish Karmarkar	<ul style="list-style-type: none"> * Applied resolution of issue 61 by using the document at http://www.oasis-open.org/apps/org/workgroup/sca-bindings/download.php/32160/sca-binding-ws-1.1-spec-cd02-issue61-rev3.doc as the base document. * Updated NS URI (Applied action item 20090311-2). * Updated Copyright statement in various places. * Updated schema per http://lists.oasis-open.org/archives/sca-bindings/200903/msg00057.html (Applied action item 20090312-1). * Applied resolution of issue 23, 25, 43, 54, 55, 64. * Replaced 3 occurrences of 'required' with 'specified'. * Recreated all bookmarks, cross-references, and conformance item table.
cd02-rev2	2009-06-09	Anish Karmarkar	Ed. fixes. Changed the way the crossrefs/bookmarks for RFC2119 keywords work. Fixed a few references.
cd02-rev3	2009-06-11	Anish Karmarkar	<ul style="list-style-type: none"> * Removed ':' from 40005, reformatted 40006/40007. * minor ed changes pointed out by SimonN. * minor formatting changes. * modified BWS20018 to remove the first sentence.
cd02-rev4	2009-06-17	Anish Karmarkar	<ul style="list-style-type: none"> * Not fixed in this rev, but issue 57 resolution was applied in previous rev. * Added list of participants in the Ack section. * Ed changes pointed out by Eric.

1016