



# Service Component Architecture Web Service Binding Specification Version 1.1

## Committee Draft 03 / Public Review 01

10 July 2009

### Specification URIs:

#### This Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd03.html>  
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd03.doc>  
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd03.pdf> (Authoritative)

#### Previous Version:

<http://www.oasis-open.org/committees/download.php/31235/sca-binding-ws-1.1-spec-cd02.doc>  
<http://www.oasis-open.org/committees/download.php/31236/sca-binding-ws-1.1-spec-cd02.pdf>  
(Authoritative)

#### Latest Version:

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

### 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:

- OASIS Committee Draft 03, "Service Component Architecture Assembly Model Specification Version 1.1", March 2009  
<http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd03.pdf>
- OASIS Committee Draft 02, "SCA Policy Framework Version 1.1", February 2009  
<http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd02.pdf>

### 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 .....	8
1.4	Naming Conventions .....	8
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 .....	14
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 Protocol with WS-MakeConnection .....	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.	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

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**] could be represented with a policy set.

## 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

<b>Prefix</b>	<b>Namespace</b>	<b>Notes</b>
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]** OASIS Committee Draft 03, "Service Component Architecture Assembly Model  
36 Specification Version 1.1", March 2009  
37 <http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd03.pdf>
- 38
- 39 **[SCA-Policy]** OASIS Committee Draft 02, "SCA Policy Framework Specification Version 1.1",  
40 February 2009  
41 <http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd02.pdf>
- 42 **[SCA-JCAA]** OASIS Committee Draft 03, "SCA Java Common Annotations and APIs  
43 Specification Version 1.1", May 2009  
44 <http://docs.oasis-open.org/opencsa/sca-j/sca-javacaa-1.1-spec-cd03.pdf>
- 45 **[WSDL11]** E. Christensen et al, *Web Service Description Language (WSDL) 1.1*,  
46 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, W3C Note, March 15 2001.
- 47 **[WSDL20]** Chinnici et al, *Web Service Description Language (WSDL) Version 2.0 Part 1:  
48 Core Language*, <http://www.w3.org/TR/2007/REC-wsdl20-20070626/>, W3C  
49 Recommendation, June 26 2007.
- 50 **[WSI-Profiles]** "Basic Profile Version 1.1" <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>,  
51 "Attachments Profile Version 1.0" <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>,  
52 "Simple SOAP Binding Profile Version 1.0" <http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html>,  
53 "Basic Security Profile Version 1.0" <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>
- 54
- 55 **[JAX-WS]** "JSR 224: Java™ API for XML-Based Web Services (JAX-WS) 2.0"  
56 <http://jcp.org/en/jsr/detail?id=224>
- 57
- 58 **[SOAP11]** Box et al, "Simple Object Access Protocol (SOAP) 1.1"  
59 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>, W3C Note May 2000
- 60
- 61 **[SOAP]** Gudgin et al, "SOAP Version 1.2 Part 1: Messaging Framework"  
62 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>, W3C  
63 Recommendation June 2003; Box et al, "Simple Object Access Protocol (SOAP)  
64 1.1" <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>, W3C Note May 2000
- 65 **[SOAP12Adjuncts]** Gudgin et al, "SOAP Version 1.2 Part 2: Adjuncts (Second Edition)"  
66 <http://www.w3.org/TR/soap12-part2/>, W3C Recommendation April 2007
- 67 **[WS-Addr]** Gudgin et al, "Web Services Addressing 1.0 – Core"  
68 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>, W3C  
69 Recommendation May 2006
- 70 **[W11-SOAP12]** Angelov et al, "WSDL 1.1 Binding Extension for SOAP 1.2"  
71 <http://www.w3.org/Submission/wsdl11soap12/>, W3C Member Submission April  
72 2006
- 73 **[WS-Addr-SOAP]** Gudgin et al, "Web Services Addressing 1.0 – SOAP Binding"  
74 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>, W3C  
75 Recommendation May 2006

76	<b>[WS-MC]</b>	OASIS Standard "Web Services Make Connection (WS-MakeConnection) Version 1.1", February 2009
77		<a href="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-spec-os.doc">http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-spec-os.doc</a>
78		
79	<b>[WS-Policy]</b>	Vedamuthu et al, "Web Services Policy 1.5 – Framework"
80		<a href="http://www.w3.org/TR/2007/REC-ws-policy-20070904">http://www.w3.org/TR/2007/REC-ws-policy-20070904</a> , W3C Recommendation
81		September 2007
82	<b>[WS-PA]</b>	Vedamuthu et al, "Web Services Policy 1.5 – Attachment"
83		<a href="http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904">http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904</a> , W3C
84		Recommendation September 2007

### 85 1.3 Non-Normative References

86	<b>[WSI-AP]</b>	"Attachments Profile Version 1.0" <a href="http://www.w3.org/Profiles/AttachmentsProfile-1.0.html">http://www.w3.org/Profiles/AttachmentsProfile-1.0.html</a>
87		
88	<b>[MTOM]</b>	Gudgin et al, "SOAP Message Transmission Optimization Mechanism"
89		<a href="http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/">http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/</a> , W3C
90		Recommendation January 2005
91	<b>[WS-Security]</b>	Oasis Standard "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)" February 2006
92		<a href="http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf">http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf</a>
93		

### 94 1.4 Naming Conventions

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

- 98 1. Where the names of elements and attributes consist partially or wholly of  
99 acronyms, the letters of the acronyms use the same case. When the acronym  
100 appears at the start of the name of an element or an attribute, or after a period,  
101 it is in lower case. If it appears elsewhere in the name of an element or an  
102 attribute, it is in upper case. For example, an attribute might be named "uri" or  
103 "jndiURL".
- 104 2. Where the names of types consist partially or wholly of acronyms, the letters of  
105 the acronyms are in all upper case. For example, an XML Schema type might be  
106 named "JCABinding" or "MessageID".
- 107 3. Values, including local parts of QName values, follow the rules for names of  
108 elements and attributes as stated above, with the exception that the letters of  
109 acronyms are in all upper case. For example, a value might be "JMSDefault" or  
110 "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. [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. [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. [BWS20005] 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). [BWS20006]

- 157
- Port:
 

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

159 If the binding is for an SCA service, the portType associated with the specified  
 160 WSDL port MUST be compatible with the SCA service interface as defined in  
 161 section 2.1, and the port MUST satisfy all the policy constraints of the  
 162 binding. [BWS20007] The SCA runtime MUST expose an endpoint for the specified  
 163 WSDL port, or raise an error if it does not support the WSDL port. [BWS20008] If  
 164 the binding is for an SCA reference, the portType associated with the specified  
 165 WSDL port MUST be a compatible superset of the SCA reference interface as  
 166 defined in the SCA Assembly Model specification [SCA-Assembly], and the port  
 167 MUST satisfy all the policy constraints of the binding. [BWS20009] The SCA  
 168 runtime MUST use the specified WSDL port for invocations made using the SCA  
 169 reference, or raise an error if it does not support the WSDL port. [BWS20010]
  - Binding:
 

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

171

172 If the binding is for an SCA service, the portType associated with the specified  
 173 WSDL binding MUST be compatible with the SCA service interface as defined in  
 174 section 2.1, and the WSDL binding MUST satisfy all the policy constraints of the  
 175 binding. [BWS20011] The SCA runtime MUST expose an endpoint for the  
 176 specified WSDL binding, or raise an error if it does not support the WSDL binding.  
 177 [BWS20012]

178 If the binding is for an SCA reference, the portType associated with the specified  
 179 WSDL binding MUST be a compatible superset of the SCA reference interface as  
 180 defined in the SCA Assembly Model specification [SCA-Assembly], and the  
 181 WSDL binding MUST satisfy all the policy constraints of the binding.  
 182 [BWS20013] The SCA runtime MUST use the specified WSDL binding for  
 183 invocations made using the SCA reference, or raise an error if it does not support  
 184 the WSDL binding. [BWS20014]

185 When the *wsdl.binding* form of *wsdlElement* is used, the endpoint address URI  
 186 for an SCA reference MUST be specified by either the *@uri* attribute on the  
 187 binding or a WS-Addressing *EndpointReference* element, except where the SCA  
 188 Assembly Model specification [SCA-Assembly] states that the *@uri* attribute can  
 189 be omitted. [BWS20015]
  - ***/binding.ws/@wsdli:wsdlLocation*** – when present this attribute specifies the  
 190 location(s) of the WSDL document(s) associated with specific namespace(s).  
 191  
 192 The *@wsdli:wsdlLocation* attribute can be used in the event that the `<WSDL-  
 193 namespace-URI>` value in the *@wsdlElement* attribute is not dereferencable, or when  
 194 the intended WSDL document is to be found at a different location than the one  
 195 pointed to by the `<WSDL-namespace-URI>`. The semantics of this attribute are  
 196 specified in Section 7.1 of WSDL 2.0 [WSDL20].  
 197  
 198 If the *@wsdli:wsdlLocation* attribute is used the *@wsdlElement* attribute MUST also  
 199 be specified. [BWS20017]  
 200  
 201 The value of the *@wsdli:wsdlLocation* attribute MUST identify an existing WSDL 1.1  
 202 document. [BWS20018]
  - ***/binding.ws/wireFormat*** – as defined in the SCA Assembly Specification [SCA-  
 201 Assembly]. This specification does not define any new wireFormat elements.  
 202

- 203 • */binding.ws/operationSelector* – as defined in the SCA Assembly Specification  
204 **[SCA-Assembly]**. This specification does not define any new operationSelector  
205 elements.
- 206 • */binding.ws/endpointReference* – when present this element provides the WS-  
207 Addressing **[WS-Addr]** EndpointReference that specifies the endpoint for the service  
208 or reference.

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

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

216 The SCA runtime MUST support all the attributes of the <binding.ws> element, namely  
217 @name, @uri, @requires, @policySets, @wsdlElement, and  
218 @wsdli:wsdlLocation. **[BWS20021]**

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

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

## 225 2.1 Compatibility of SCA Service Interfaces and WSDL portTypes

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

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

## 246 2.2 Endpoint URI resolution

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

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

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

- 261 A. The value of the @uri attribute
- 262 B. The value of the wsa:Address element of the endpointReference element
- 263 C. The value of the address element of the WSDL port referenced by the  
264 @wsdlElement attribute
- 265 D. The value of the address element of one of the set of available WSDL ports as  
266 specified under the definition of the @wsdlElement attribute when it references a  
267 WSDL service element

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

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

## 271 2.3 Interface mapping

272 When *binding.ws* is used on a service or reference with an interface that is not defined  
273 by *interface.wsdl*, the SCA runtime MUST derive a WSDL portType for the service or  
274 reference from the interface using the rules defined for that SCA interface type.  
275 [BWS20027]

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

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

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

283

## 284 2.4 Production of WSDL description for an SCA service

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

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

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

294

## 295 2.5 Additional binding configuration data

296 SCA runtime implementations can provide additional metadata that is associated with a  
297 web service binding. This is done by providing extension points in the schema; refer to  
298 Appendix A: Web Services XML Binding Schema for the locations of these extension  
299 points.

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

303

## 304 2.6 Web Service Binding and SOAP Intermediaries

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

307

## 308 2.7 Support for WSDL extensibility

309 When a binding.ws element uses the @wsdlElement attribute, the details of the binding  
310 are specified by the WSDL element referenced by the value of the attribute. Per the  
311 WSDL specification, WSDL allows for extensibility via elements as well as attributes, and  
312 it specifies rules for processing such elements. This specification does not constrain the  
313 use of such extensibility in WSDL and relies on the rules specified in the WSDL  
314 specification for processing such extended elements.

315 An SCA runtime MUST support the WSDL extensions defined in the namespace  
316 associated with the prefix "sca" (as defined in section 1.1). [BWS20032]

317 The SCA runtime MUST support the WSDL 1.1 binding extension for SOAP 1.1 over HTTP  
318 [WSDL11], as identified by the WSDL element wsoap11:binding that has the  
319 @transport attribute with a value of "http://schemas.xmlsoap.org/soap/http".  
320 [BWS20033]

321 The SCA runtime SHOULD support the WSDL 1.1 binding extension for SOAP 1.2 over  
322 HTTP [W11-SOAP12], as identified by the WSDL element wsoap12:binding that has  
323 the @transport attribute with a value of "http://schemas.xmlsoap.org/soap/http".  
324 [BWS20034]

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

## 329 2.8 Intents listed in the bindingType

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

## 332 2.9 Intents and binding configuration

333 This binding mandates support for SOAP 1.1 and encourages SOAP 1.2 support. The  
334 <bindingType> element associated with this binding MUST include the SOAP.1\_1 intent  
335 in its @mayProvides or @alwaysProvides attributes. [BWS20035] The <bindingType>  
336 element associated with this binding SHOULD include the SOAP.1\_2 intent in its  
337 @mayProvides attribute. [BWS20036] For more details on the <bindingType> element  
338 see [SCA-Policy].

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

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

---

## 343 3 Web Service Binding Examples

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

347

### 348 3.1 Example Using WSDL documents

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

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

354 The first reference's binding is defined by the specified WSDL service in the WSDL  
355 document at the given location. The reference can use any of the WSDL service's ports  
356 to invoke the target service. The second reference's binding is defined by the specified  
357 WSDL binding. The specific endpoint URI to be invoked is provided via the *@uri*  
358 attribute.

359

```
360 <?xml version="1.0" encoding="ASCII"?>
361 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
362   name="MyValueComposite">
363   <service name="MyValueService">
364     <interface.java interface="services.myvalue.MyValueService"/>
365     <binding.ws wsdlElement="http://www.example.org/MyValueService#
366 wsdli.binding(MyValueService/MyValueServiceSOAP)"/>
367     ...
368   </service>
369   ...
370   ...
371   ...
372   ...
373   <reference name="StockQuoteReference1">
374     <interface.java interface="services.stockquote.StockQuoteService"/>
375     <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
376 wsdli.service(StockQuoteService)"
377     wsdlLocation="http://www.example.org/StockQuoteService
378 http://www.example.org/StockQuoteService.wsdl"/>
379   </reference>
380   ...
381   <reference name="StockQuoteReference2">
382     <interface.java interface="services.stockquote.StockQuoteService"/>
383     <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
384 wsdli.binding(StockQuoteBinding)"
385     wsdlLocation="http://www.example.org/StockQuoteService
386 http://www.example.org/StockQuoteService.wsdl"
387     uri="http://www.example.org/StockQuoteService5"/>
388   </reference>
389 </composite>
```

## 390 3.2 Examples Without a WSDL Document

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

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

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

403

```
404 <?xml version="1.0" encoding="ASCII"?>
405 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
406           name="MyValueComposite">
407
408     <service name="MyValueService">
409       <interface.java interface="services.myvalue.MyValueService"/>
410       <binding.ws/>
411       ...
412     </service>
413
414     ...
415
416     <reference name="StockQuoteService">
417       <interface.java interface="services.stockquote.StockQuoteService"/>
418       <binding.ws uri="http://www.example.org/StockQuoteService"/>
419     </reference>
420 </composite>
```

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

427

```
428 <?xml version="1.0" encoding="ASCII"?>
429 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
430           name="MyValueComposite">
431
432     <service name="MyValueService">
433       <interface.java interface="services.myvalue.MyValueService"/>
434       <binding.ws name="MyValueServiceSOAP11" requires="SOAP.1_1"/>
435       <binding.ws requires="SOAP.1_2"/>
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                 requires="SOAP.1_2"/>
445     </reference>
```

446

```
</composite>
```

447

---

## 448 4 Transport Binding

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

### 454 4.1 Intents

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

- 457 • SOAP  
458 When the SOAP intent is required, the SCA runtime MUST transmit and receive  
459 messages using SOAP. One or more SOAP versions can be used. [BWS40001]
- 460 • SOAP.1\_1  
461 When the SOAP.1\_1 intent is required, the SCA runtime MUST transmit and receive  
462 messages using only SOAP 1.1. [BWS40002]
- 463 • SOAP.1\_2  
464 When the SOAP.1\_2 intent is required, the SCA runtime MUST transmit and receive  
465 messages using only SOAP 1.2. [BWS40003]

### 466 4.2 Default Transport Binding Rules

#### 467 4.2.1 WS-I Basic Profile Alignment

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

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

475 For an SCA service or reference element, the portType from the service's or reference's  
476 interface or derived from that interface MUST follow either the rpc-literal pattern or the  
477 document-literal pattern. [BWS40004]

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

#### 481 4.2.2 Default Transport Binding Rules

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

- 483 • HTTP-based transfer protocol;
- 484 • SOAP 1.1 binding;
- 485 • "literal" format as described in section 3.5 of [WSDL11];

- 486
- 487
- Either the document literal or rpc literal pattern, depending on the service or reference interface as described in section 4.2.1;
    - For document literal pattern, each message uses "document" style, as per section 3.5 of **[WSDL11]**;
    - For rpc-literal pattern, each message uses "rpc" style, as per section 3.5 of **[WSDL11]** and the child elements of the SOAP Body element are namespace qualified with a non-empty namespace name;
  - For SOAP 1.1 messages, the SOAPAction HTTP header described in section 6.1.1 of **[SOAP11]** represents the empty string, in quotes ("");
  - 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.

498 In the event that the transport details are not otherwise determined, an SCA runtime  
499 MUST enable the default transport binding rules. **[BWS40005]**

500 When using the default transport binding rules, the SCA runtime MAY provide additional  
501 WSDL bindings, unless policy is applied that explicitly restricts this. **[BWS40006]**

502 When using the default transport binding rules with the rpc-literal pattern, the SCA  
503 runtime SHOULD use the structural URI associated with the binding as the namespace of  
504 the child elements of the SOAP body element. **[BWS40007]**

---

## 505 5 Implementing SCA Callbacks using Web Services

### 506 5.1 SCA Web Services Callback Protocol

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

510 To implement the SCA Web Services Callback Protocol, an SCA binding follows the  
511 following rules.

512 1. Every request message that invokes the forward interface MUST contain a  
513 Callback EPR. [BWS50002] If the request message contains the `wsa:From` SOAP  
514 header block then the `wsa:From` header block specifies the Callback EPR. If the  
515 `wsa:From` header block is not present then the `wsa:ReplyTo` header block  
516 specifies the Callback EPR.

517 If the Callback EPR's `[address]` value is  
518 "`http://www.w3.org/2005/08/addressing/anonymous`" or  
519 "`http://www.w3.org/2005/08/addressing/none`" then the SCA runtime MUST  
520 generate the Invalid Addressing Header fault as specified in Section 6.4.1 of  
521 [WS-Addr-SOAP]. [BWS50004] Such a fault can include additional  
522 `[Subsubcode]` `wsa:OnlyNonAnonymousAddressSupported`.

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

527 3. When the service implementation invokes the callback interface, it MUST use the  
528 Callback EPR from a request message that invoked the forward interface.  
529 [BWS50005] Once the Callback EPR is selected, the SCA runtime MUST follow the  
530 rules defined in Section 3.3 of [WS-Addr] to invoke operations on the callback  
531 interface. [BWS50006]

532 When the service invokes the callback interface, if the request message from which the  
533 Callback EPR was obtained contained the `wsa:MessageID` SOAP header block, the SCA  
534 runtime MUST include a `wsa:RelatesTo` SOAP header block in the callback message.  
535 [BWS50007] The `wsa:RelatesTo` SOAP header block MUST have the relationship type  
536 value of "`http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback`" and  
537 the related message id MUST be the `wsa:MessageID` of the message from which the  
538 Callback EPR was obtained. [BWS50008]

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

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

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

## 552 5.2 SCA Web Services Callback Protocol with WS-MakeConnection

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

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

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

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

570 When the service implementation invokes the callback interface, it uses the Callback EPR  
571 from a request message that invoked the forward interface, and the callback request  
572 message is sent as the response to a `wsmc:MakeConnection` message that contains the  
573 `wsmc:Address` value that matches the MakeConnection Anonymous URI in the Callback  
574 EPR.

575 When a service that offers a bidirectional interface is invoked using WS-MakeConnection  
576 Anonymous URI as the value for the Callback EPR address, depending on the semantics  
577 and/or implementation of the service, it is possible that the service might invoke the  
578 callback interface before the forward operation ends. In such cases, it is necessary for  
579 the binding on the reference-side to start polling for callback request(s) from the  
580 service, before or right after the forward operation request is sent and before a response  
581 is received, and continue polling as long as callback requests are expected. It is possible  
582 that before the response to the forward request is sent a response to one or more  
583 callback requests are required by the service.

## 584 5.3 Policy Assertion for SCA Web Services Callback Protocol

585 WS-Policy Framework **[WS-Policy]** and WS-Policy Attachment **[WS-PA]** collectively  
586 define a framework, model and grammar for expressing the requirements, and general  
587 characteristics of entities in an XML Web services-based system. To enable a Web  
588 service client and a Web service to describe their requirements for implementing SCA  
589 Web Services Callback Protocol, this specification defines a single policy assertion that  
590 leverages the WS-Policy framework.

### 591 5.3.1 Assertion Model

592 The WSCallback policy assertion indicates that the Web service client and the Web  
593 service **MUST** use SCA Web Services Callback Protocol to implement callbacks.  
594 **[BWS50010]** Specifically, the protocol determines the requirements on the forward

595 request message, the EPR used for callbacks and the requirements on the callback  
596 request message.

### 597 5.3.2 Normative Outline

598 The normative outline for the WSCallback assertion is:

```
599 <sca:WSCallback ...>  
600 ...  
601 </sca:WSCallback>
```

602

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

- 604 • **/sca:WSCallback**: A policy assertion that specifies that SCA Web Services  
605 Callback protocol is used when sending messages.

### 606 5.3.3 Assertion Attachment

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

- 609 • Endpoint Policy Subject

610 WS-PolicyAttachment defines a set of WSDL/1.1 policy attachment points for each of the  
611 above Policy Subjects. Since a WSCallback policy assertion specifies a concrete behavior,  
612 it cannot be attached to the abstract WSDL policy attachment points.

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

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

- 619 • wsdl:port
- 620 • wsdl:binding

### 621 5.3.4 Assertion Example

622 The example below shows the use of the WSCallback policy assertion in a WSDL  
623 document.

624

```
625 (01) <wsdl:definitions  
626 (02)   targetNamespace="example.com"  
627 (03)   xmlns:tns="example.com"  
628 (04)   xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"  
629 (05)   xmlns:wsp="http://www.w3.org/ns/ws-policy"  
630 (06)   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"  
631 (07)   xmlns:wssu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
632 wssecurity-utility-1.0.xsd">  
633 (08)  
634 (09) <wsp:UsingPolicy wSDL:required="true" />  
635 (10)  
636 (11) <wsp:Policy wsu:Id="MyPolicy" >  
637 (12)   <sca:WSCallback/>  
638 (13) </wsp:Policy>  
639 (14)
```

```
640 (15) <!-- omitted elements -->
641 (16)
642 (17) <wsdl:binding name="MyBinding" type="tns:MyPortType" >
643 (18)   <wsp:PolicyReference URI="#MyPolicy" />
644 (19)   <!-- omitted elements -->
645 (20) </wsdl:binding>
646 (21)
647 (22)</wsdl:definitions>
```

648

649 Line (09) in the example above indicates that WS-Policy is in use as a required  
650 extension. Lines (11-13) are a policy expression that includes a WSCallback policy  
651 assertion (line 12) to indicate that SCA Web Services Callback protocol is used. Lines  
652 (17-20) are a WSDL binding. Line (18) indicates that the policy in lines (11-13) applies  
653 to this binding, specifically indicating that SCA Web Services Callback protocol is used  
654 over all the messages in the binding.

### 655 5.3.5 Security Considerations

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

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

664

---

## 665 6 Conformance

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

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

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

### 672 6.1 SCA WS Binding XML Document

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

676 An SCA WS Binding XML document **MUST** be a conformant SCA Composite Document or  
677 a SCA ComponentType Document, as defined by the SCA Assembly specification **[SCA-**  
678 **Assembly]**, and **MUST** comply with all statements in Appendix C: Conformance Items  
679 related to elements and attributes in an SCA WS Binding XML document, notably all  
680 "MUST" statements have to be implemented.

### 681 6.2 SCA Runtime

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

- 684 1. The implementation **MUST** comply with all statements in Appendix B:  
685 Conformance Items related to an SCA Runtime, except for those that originate  
686 from Section 5, notably all "MUST" statements have to be implemented.
- 687 2. The implementation **MAY** support the SCA Web Services Callback Protocol. If it  
688 does, it **MUST** comply with all statements in Appendix B: Conformance Items for  
689 the SCA Web Services Callback Protocol.
- 690 3. The implementation **MAY** support the SCA Web Services Callback Protocol in  
691 conjunction with WS-MakeConnection. If it does, it **MUST** comply with all  
692 statements in Appendix B: Conformance Items for the SCA Web Services  
693 Callback Protocol and it **MUST** comply with the requirements of WS-  
694 MakeConnection.
- 695 4. The implementation **MUST** conform to the SCA Assembly Model Specification  
696 Version 1.1 **[SCA-Assembly]**, and to the SCA Policy Framework Version 1.1  
697 **[SCA-Policy]**.
- 698 5. The implementation **MUST** reject a SCA WS Binding XML Document that is not  
699 conformant per Section 6.1.

700  
701

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

```
702 <?xml version="1.0" encoding="UTF-8"?>
703 <!-- Copyright(C) OASIS 2005, 2009. All Rights Reserved.
704     OASIS trademark, IPR and other policies apply.-->
705
706 <schema xmlns="http://www.w3.org/2001/XMLSchema"
707     targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200903"
708     xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"
709     xmlns:wsdli="http://www.w3.org/ns/wsdli-instance"
710     xmlns:wsa="http://www.w3.org/2005/08/addressing"
711     elementFormDefault="qualified">
712
713     <import namespace="http://www.w3.org/ns/wsdli-instance"
714         schemaLocation="http://www.w3.org/2007/05/wsdli/wsdli20-
715 instance.xsd"/>
716     <import namespace="http://www.w3.org/2005/08/addressing"
717         schemaLocation="http://www.w3.org/2006/03/addressing/ws-
718 addr.xsd"/>
719     <include schemaLocation="sca-core-1.1-cd03.xsd"/>
720
721     <element name="binding.ws" type="sca:WebServiceBinding"
722         substitutionGroup="sca:binding"/>
723     <complexType name="WebServiceBinding">
724         <complexContent>
725             <extension base="sca:Binding">
726                 <sequence>
727                     <element name="endpointReference"
728                         type="wsa:EndpointReferenceType"
729                         minOccurs="0" maxOccurs="unbounded"/>
730                     <any namespace="##other" processContents="lax"
731                         minOccurs="0" maxOccurs="unbounded"/>
732                 </sequence>
733                 <attribute name="wsdlElement" type="anyURI" use="optional"/>
734                 <attribute ref="wsdli:wsdlLocation" use="optional"/>
735             </extension>
736         </complexContent>
737     </complexType>
738
739 </schema>
```

740  
741  
742

---

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

743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761

```
<?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>
```

762  
763

## C. Conformance Items

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

Conformance ID	Description
[BWS20001]	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]	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]	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]	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], and the port MUST satisfy all the policy constraints of the binding.
[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], and the WSDL binding MUST satisfy all the policy constraints of the binding.
[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>wSDLElement</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] states that the <i>@uri</i> attribute can be omitted.
[BWS20017]	If the <i>@wsdl:wsdlLocation</i> attribute is used the <i>@wSDLElement</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>@wSDLElement</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>&lt;binding.ws&gt;</i> element, namely <i>@name</i> , <i>@uri</i> , <i>@requires</i> , <i>@policySets</i> , <i>@wSDLElement</i> , and <i>@wsdl:wsdlLocation</i> .
[BWS20022]	The SCA runtime SHOULD support the element <i>&lt;endpointReference&gt;</i> .
[BWS20023]	If an SCA runtime does not support the element <i>&lt;endpointReference&gt;</i> , then it MUST reject an SCA WS Binding XML document (as defined in Section 5.1) that contains the element.
[BWS20024]	The <i>&lt;binding.ws&gt;</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 WSDL description of the service.
[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], 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], 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]	When the SOAP.1_1 intent is required, the SCA runtime MUST transmit and receive messages using only SOAP 1.1.
[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.
[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 <code>wsa:RelatesTo</code> SOAP header block MUST have the relationship type value of " <code>http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback</code> " and the related message id MUST be the <code>wsa:MessageID</code> 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 <code>wsa:MessageID</code> SOAP header block, the SCA runtime MUST NOT include a <code>wsa:RelatesTo</code> SOAP header block with a relationship type value of " <code>http://docs.oasis-open.org/opencsa/sca-bindings/ws/callback</code> " 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.
[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: <code>wsdl:portType</code>
[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.

766

## D. WSDL Generation

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

- 772 • component name = the value of the @name attribute of the component element  
773 containing the binding.ws element
- 774 • service name = the value of the @name attribute of the service element  
775 containing the binding.ws element
- 776 • binding name = the value of @name attribute of the binding.ws element, or the  
777 default if no @name attribute is present
- 778 • SOAP version = either "SOAP11" or "SOAP12" as appropriate

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

- 780 • wsdl:definitions/@name = <component name> + "." + <service name>
- 781 • wsdl:definitions/@targetNamespace = <structural URI for the service>
- 782 • import each WSDL 1.1 portType, rather than putting them inline
- 783 • wsdl:binding/@name = <binding name> + <SOAP version> + "Binding"
- 784 • wsdl:service/@name = <service name>
- 785 • wsdl:port/@name = <binding name> + <SOAP version> + "Port"

786  
787

## E. SCA Web Services Callback Protocol Message Examples

788 The message examples in this section are for a configuration that consists of a reference  
789 R that is wired to a Service S. S has a bidirectional interface and the binding used in  
790 both directions, forward and callback, is binding.ws configured for SOAP. The forward  
791 interface and the callback interface both contain a single one-way operation.

792 The following message exchanges take place between R and S:

- 793 1. R invokes the forward operation and sets the callback address to **RC1**. Let's call  
794 the message that invokes the forward operation R1. S then calls the callback  
795 operation twice. Let's call the callback messages S1 and S2
- 796 2. R invokes the forward operation again with the same callback address **RC1**. Let's  
797 call the message that invokes the forward operation R2. S then calls the callback  
798 operation once. Let's call the callback message S3.
- 799 3. R invokes the forward operation yet another time, but this time uses a difference  
800 callback address: **RC2**. Let's call the message that invokes the forward operation  
801 R3. S then calls the callback operation twice. Let's call the callback messages S4  
802 and S5.

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

### 806 **R1:**

```
807 <soap:Envelope ...>  
808   <soap:Header>  
809     <wsa:From>  
810       <wsa:Address>http://example.com/callback</wsa:Address>  
811       <wsa:ReferenceProperties>  
812         <myNS:SomeID>1</myNS:SomeID>  
813       </wsa:ReferenceProperties>  
814     </wsa:From>  
815     <wsa:MessageID>urn:uuid:f81d4fae-7dec-11d0-a765-  
816 00a0c91e6bf6</wsa:messageID>  
817     ...  
818   </soap:Header>  
819   <soap:Body>  
820     ...  
821   </soap:Body>  
822 </soap:Envelope>
```

824

### 825 **S1, S2:**

```
826 <soap:Envelope ...>
827 <soap:Header>
828   <wsa:To>http://example.com/callback</wsa:To>
829   <myNS:SomeID>1</myNS:SomeID>
830   <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
831 bindings/ws/callback">urn:uuid:f81d4fae-7dec-11d0-a765-
832 00a0c91e6bf6</wsa:RelatesTo>
833   ...
834 </soap:Header>
835 <soap:Body>
836   ...
837 </soap:Body>
838 </soap:Envelope>
839
```

840

841 **R2:**

```
842 <soap:Envelope ...>
843 <soap:Header>
844   <wsa:From>
845     <wsa:Address>http://example.com/callback</wsa:Address>
846     <wsa:ReferenceProperties>
847       <myNS:SomeID>1</myNS:SomeID>
848     </wsa:ReferenceProperties>
849   </wsa:From>
850   <wsa:MessageID>urn:uuid:f81d4fae-8dec-11d0-a765-
851 00a0c91e6bf6</wsa:messageID>
852   ...
853 </soap:Header>
854 <soap:Body>
855   ...
856 </soap:Body>
857 </soap:Envelope>
858
```

859

860 **S3:**

```
861 <soap:Envelope ...>
862 <soap:Header>
863   <wsa:To>http://example.com/callback</wsa:To>
864   <myNS:SomeID>1</myNS:SomeID>
865   <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
866 bindings/ws/callback">
867     urn:uuid:f81d4fae-8dec-11d0-a765-00a0c91e6bf6
868   </wsa:RelatesTo>
869   ...
870 </soap:Header>
871 <soap:Body>
872   ...
873 </soap:Body>
874 </soap:Envelope>
875
```

875

876 **R3:**

```
877 <soap:Envelope ...>
878   <soap:Header>
879     <wsa:From>
880       <wsa:Address>http://example.com/callback-other</wsa:Address>
881       <wsa:ReferenceProperties>
882         <myNS:SomeID>2</myNS:SomeID>
883       </wsa:ReferenceProperties>
884     </wsa:From>
885     <wsa:MessageID>urn:uuid:f81d4fae-9dec-11d0-a765-
886 00a0c91e6bf6</wsa:messageID>
887     ...
888   </soap:Header>
889   <soap:Body>
890     ...
891   </soap:Body>
892 </soap:Envelope>
893
894
```

895

#### 896 **S4, S5:**

```
897 <soap:Envelope ...>
898   <soap:Header>
899     <wsa:To>http://example.com/callback-other</wsa:To>
900     <myNS:SomeID>2</myNS:SomeID>
901     <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
902 bindings/ws/callback">urn:uuid:f81d4fae-9dec-11d0-a765-
903 00a0c91e6bf6</wsa:RelatesTo>
904     ...
905   </soap:Header>
906   <soap:Body>
907     ...
908   </soap:Body>
909 </soap:Envelope>
910
```

910

## 911 **E.1 Message Examples Using WS-MakeConnection**

912 In this case the reference R cannot host a listener and uses WS-MakeConnection to poll  
913 for callback requests. The interaction between the two consists of reference R sending a  
914 forward request R4. When using HTTP, the HTTP response to R4 contains an empty  
915 entity body. This is followed by a MakeConnection message from the reference to the  
916 service. This is a polling message from the reference and establishes a connection. If the  
917 callback request is ready when the connection is established, the service sends a  
918 callback request S6 to the reference in the entity body of the HTTP response.

#### 919 **R4:**

```
920 <soap:Envelope ...>
921   <soap:Header>
922     <wsa:From>
923       <wsa:Address>http://docs.oasis-open.org/ws-
924 rx/wsmc/200702/anonymous?id=650e8400-f29b-11d4-a716-446655440010</wsa:Address>
925     </wsa:From>
926     <wsa:MessageID>urn:uuid:f81d4fae-10dec-11d0-a765-
927 00a0c91e6bf6</wsa:messageID>
928     ...
929   </soap:Header>
930   <soap:Body>
931     ...
932   </soap:Body>
933 </soap:Envelope>
```

934

### 935 **MakeConnection polling message (from R to S):**

```
936 <soap:Envelope ...>
937   <soap:Header>
938     <wsa:Action>http://docs.oasis-open.org/ws-
939 rx/wsmc/200702/MakeConnection</wsa:Action>
940     ...
941   </soap:Header>
942   <soap:Body>
943     <wsmc:MakeConnection>
944       <wsmc:Address>http://docs.oasis-open.org/ws-
945 rx/wsmc/200702/anonymous?id=650e8400-f29b-11d4-a716-
946 446655440010</wsmc:Address>
947     </wsmc:MakeConnection>
948   </soap:Body>
949 </soap:Envelope>
```

950

### 951 **S6:**

```
952 <soap:Envelope ...>
953   <soap:Header>
954     <wsa:To>http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id=650e8400-
955 f29b-11d4-a716-446655440010</wsa:To>
956     <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
957 bindings/ws/callback">urn:uuid:f81d4fae-10dec-11d0-a765-
958 00a0c91e6bf6</wsa:RelatesTo>
959     ...
960   </soap:Header>
961   <soap:Body>
962     ...
963   </soap:Body>
964 </soap:Envelope>
```

---

965 **F. Acknowledgements**

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

968 **Participants:**

<b>Participant Name</b>	<b>Affiliation</b>
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 Kavantzas	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.



## G. Revision History

971 [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"> <li>* Partially applied the resolution of issue 14 in the conformance section.</li> <li>* Applied resolution to issue 9.</li> <li>* Applied resolution to issue 15.</li> <li>* Applied resolution to issue 16.</li> <li>* Applied resolution to issue 10.</li> <li>* Applied resolution to issue 8.</li> <li>* Applied resolution to issue 3.</li> </ul>
3	2008-06-12	Simon Holdsworth	<ul style="list-style-type: none"> <li>* Completed application of resolution to issue 10</li> <li>* Applied most of the editorial changes from Eric Johnson's review</li> </ul>
4	2008-08-13	Anish Karmarkar	<ul style="list-style-type: none"> <li>* Applied rest of Eric Johnson's ed review comments.</li> <li>* Applied resolution of issue 13.</li> <li>* Reapplied resolution of issue 15 (it was not applied correctly before)</li> <li>* Applied resolution of issue 19.</li> <li>* Applied resolution of issue 30.</li> <li>* Applied resolution of issue 32.</li> <li>* Applied resolution of issue 36.</li> <li>* Applied resolution of issue 38.</li> </ul>
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"> <li>Applied resolution of issue 11</li> <li>Applied resolution of issue 49</li> <li>Applied action item 20080904-1</li> </ul>
cd02	2009-02-16	Simon Holdsworth	Renamed, applied editorial issues

cd02-rev1	2009-06-02	Anish Karmarkar	<ul style="list-style-type: none"> <li>* Applied resolution of issue 61 by using the document at <a href="http://www.oasis-open.org/apps/org/workgroup/sca-bindings/download.php/32160/sca-binding-ws-1.1-spec-cd02-issue61-rev3.doc">http://www.oasis-open.org/apps/org/workgroup/sca-bindings/download.php/32160/sca-binding-ws-1.1-spec-cd02-issue61-rev3.doc</a> as the base document.</li> <li>* Updated NS URI (Applied action item 20090311-2).</li> <li>* Updated Copyright statement in various places.</li> <li>* Updated schema per <a href="http://lists.oasis-open.org/archives/sca-bindings/200903/msg00057.html">http://lists.oasis-open.org/archives/sca-bindings/200903/msg00057.html</a> (Applied action item 20090312-1).</li> <li>* Applied resolution of issue 23, 25, 43, 54, 55, 64.</li> <li>* Replaced 3 occurrences of 'required' with 'specified'.</li> <li>* Recreated all bookmarks, cross-references, and conformance item table.</li> </ul>
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"> <li>* Removed ‘.’ from 40005, reformatted 40006/40007.</li> <li>* minor ed changes pointed out by SimonN.</li> <li>* minor formatting changes.</li> <li>* modified BWS20018 to remove the first sentence.</li> </ul>
cd02-rev4	2009-06-17	Anish Karmarkar	<ul style="list-style-type: none"> <li>* Not fixed in this rev, but issue 57 resolution was applied in previous rev.</li> <li>* Added list of participants in the Ack section.</li> <li>* Ed changes pointed out by Eric.</li> </ul>
cd02-rev5	2009-06-22	Anish Karmarkar	* Port of the fix made in JMS/JCA binding for issues 74/75. Specifically SCA WS Binding XML document requirements were made less vague (by referring to attributes/elements)
cd02-rev6	2009-06-24	Anish Karmarkar	<ul style="list-style-type: none"> <li>* Applied resolution of issue 76, 79, 82.</li> <li>* Some very minor ed changes.</li> <li>* Reverted the document naming scheme to the old scheme.</li> </ul>
cd02-rev7	2009-07-01	Simon Holdsworth	<ul style="list-style-type: none"> <li>* Applied resolution of issue 2</li> <li>* Fixed application of resolution of issue 76</li> </ul>
cd03	2009-07-01	Simon Holdsworth	Renamed for cd03