Service Component Architecture Web Service Binding Specification Version 1.1

Committee Draft 02 Revision 5~~4~~ + issue 2 rev 6~~5~~

22nd~~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:

Notices

Table of Contents

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]**~~ could be represented with a policy set.

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/wsdl-instance"	Defined by WSDL 2.0

wsoap11	"http://schemas.xmlsoap.org/wsdl/soap/"	Defined by WSDL 1.1 [WSDL11]
wsoap12	"http://schemas.xmlsoap.org/wsdl/soap12/"	Defined by [W11-SOAP12]
sca	"http://docs.oasis-open.org/ns/opencsa/sca/200903"	Defined by the SCA specifications

1.2 Normative References

[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.
[SCA-Assembly]	http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.pdf
[SCA-Policy]	http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec.pdf
[SCA-JCAA]	http://docs.oasis-open.org/opencsa/sca-j/sca-javacaa-1.1-spec.pdf
[WSDL11]	E. Christensen et al, *Web Service Description Language (WSDL) 1.1*, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, W3C Note, March 15 2001.
[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.
[WSI-Profiles]	http://www.ws-i.org/Profiles/BasicProfile-1.1.html
	http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html
	http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html
	http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html
[JAX-WS]	http://jcp.org/en/jsr/detail?id=224
[SOAP11]	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
[SOAP]	http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
[SOAP12Adjuncts]	SOAP Version 1.2 Part 2: Adjuncts (Second Edition)
	http://www.w3.org/TR/soap12-part2/
[WS-Addr]	http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/
[W11-SOAP12]	http://www.w3.org/Submission/wsdl11soap12/
[WS-Addr-SOAP]	http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/
[WS-MC]	http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-spec-os.html
[WS-Policy]	http://www.w3.org/TR/2007/REC-ws-policy-20070904
[WS-PA]	http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904

1.3 Non-Normative References

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

1.4 Naming Conventions

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

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"?
            wsdli: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~~

~~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).~~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]

- Port:

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

  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.[BWS20007] The SCA runtime MUST expose an endpoint for the specified WSDL port, or raise an error if it does not support the WSDL port. [BWS20008] 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.~~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.~~ [BWS20009] 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. [BWS20010]

- Binding:

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

  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. [BWS20011] The SCA runtime MUST expose an endpoint for the specified WSDL binding, or raise an error if it does not support the WSDL binding. [BWS20012]

  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.~~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.~~ [BWS20013] 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. [BWS20014]

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

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

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

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

  The *@wsdli:wsdlLocation* attribute MAY be specified by the binding in the event that the <WSDL-namespace-URI> 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 <WSDL-namespace-URI>.~~The *@wsdli:wsdlLocation* attribute MAY be specified by the binding in the event that the <WSDL-namespace-URI> 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 <WSDL-namespace-URI>.~~ [BWS20016]

  If the *@wsdli:wsdlLocation* attribute is used the *@wsdlElement* attribute MUST also be specified.~~If the *@wsdli:wsdlLocation* attribute is used the *@wsdlElement* attribute MUST also be specified.~~ [BWS20017] The semantics of this attribute are specified in Section 7.1 of WSDL 2.0 **[WSDL20]**. The value of the @wsdli:wsdlLocation attribute 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.

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

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

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

The SCA runtime SHOULD support the element <endpointReference>. [BWS20022] If an SCA runtime does not support the element <endpointReference>, then it MUST reject an SCA WS Binding XML document (as defined in Section 5.1) that contains the element.~~If an SCA runtime does not support the element <endpointReference>, then it MUST reject an SCA WS Binding XML document (as defined in Section 6.1) that contains the element.~~ [BWS20023]

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

2.1 Compatibility of SCA Service Interfaces and WSDL portTypes

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

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

2.2 Endpoint URI resolution

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

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

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

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

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

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

2.3 Interface mapping

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

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

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

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

2.4 Production of WSDL description for an SCA service

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.~~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.~~ [BWS20029]

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.~~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.~~ [BWS20030] This can include out of band mechanisms, for example publication to a UDDI registry.

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

2.5 Additional binding configuration data

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

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

2.6 Web Service Binding and SOAP Intermediaries

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

2.7 Support for WSDL extensibility

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

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

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".~~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".~~ [BWS20033]

Formatted: Highlight

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".~~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".~~ [BWS20034]

Formatted: Highlight

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

2.8 Intents listed in the bindingType

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

2.9 Intents and binding configuration

This binding mandates support for SOAP 1.1 and encourages SOAP 1.2 support. The <bindingType> element associated with this binding MUST include the SOAP.1_1 intent in its @mayProvides or @alwaysProvides attributes. [BWS20035] The <bindingType>

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

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.~~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.~~ [BWS20037]

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

3 Web Service Binding Examples

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

3.1 Example Using WSDL documents

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

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

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

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
           name="MyValueComposite">
   <service name="MyValueService">
      <interface.java interface="services.myvalue.MyValueService"/>
      <binding.ws wsdlElement="http://www.example.org/MyValueService#

wsdl.binding(MyValueService/MyValueServiceSOAP)"/>
         ...
   </service>

   ...

   <reference name="StockQuoteReference1">
      <interface.java interface="services.stockquote.StockQuoteService"/>
      <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
                               wsdl.service(StockQuoteService)"
      wsdli:wsdlLocation="http://www.example.org/StockQuoteService
                          http://www.example.org/StockQuoteService.wsdl"/>
   </reference>

   <reference name="StockQuoteReference2">
      <interface.java interface="services.stockquote.StockQuoteService"/>
      <binding.ws wsdlElement="http://www.example.org/StockQuoteService#
                               wsdl.binding(StockQuoteBinding)"
      wsdli:wsdlLocation="http://www.example.org/StockQuoteService
                          http://www.example.org/StockQuoteService.wsdl"
                     uri="http://www.example.org/StockQuoteService5"/>
   </reference>
</composite>
```

3.2 Examples Without a WSDL Document

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

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

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

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
           name="MyValueComposite">

   <service name="MyValueService">
      <interface.java interface="services.myvalue.MyValueService"/>
      <binding.ws/>
      ...
   </service>

   ...

   <reference name="StockQuoteService">
      <interface.java interface="services.stockquote.StockQuoteService"/>
      <binding.ws uri="http://www.example.org/StockQuoteService"/>
   </reference>
</composite>
```

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

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
           name="MyValueComposite">

   <service name="MyValueService">
      <interface.java interface="services.myvalue.MyValueService"/>
      <binding.ws name="MyValueServiceSOAP11" requires="SOAP.1_1"/>
      <binding.ws requires="SOAP.1_2"/>
      ...
   </service>

   ...

   <reference name="StockQuoteService">
      <interface.java interface="services.stockquote.StockQuoteService"/>
      <binding.ws uri="http://www.example.org/StockQuoteService"
                  requires="SOAP.1_2"/>
   </reference>
```

```
</composite>
```

4 Transport Binding

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

4.1 Intents

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

- SOAP
  When the SOAP intent is required, the SCA runtime MUST transmit and receive messages using SOAP. One or more SOAP versions can be used.~~When the SOAP intent is required, the SCA runtime MUST transmit and receive messages using SOAP. One or more SOAP versions can be used.~~ [BWS40001]
- SOAP.1_1
  When the SOAP.1_1 intent is required, the SCA runtime MUST transmit and receive messages using only SOAP 1.1.~~When the SOAP.1_1 intent is required, the SCA runtime MUST transmit and receive messages using only SOAP 1.1.~~ [BWS40002]
- SOAP.1_2
  When the SOAP.1_2 intent is required, the SCA runtime MUST transmit and receive messages using only SOAP 1.2.~~When the SOAP.1_2 intent is required, the SCA runtime MUST transmit and receive messages using only SOAP 1.2.~~ [BWS40003]

4.2 Default Transport Binding Rules

4.2.1 WS-I Basic Profile Alignment

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

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

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.~~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.~~ [BWS40004]

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

4.2.2 Default Transport Binding Rules

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

- HTTP-based transfer protocol;
- SOAP 1.1 binding;
- "literal" format as described in section 3.5 of **[WSDL11]**;
- 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.

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

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

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.~~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.~~ [BWS40007]

5 Implementing SCA Callbacks using Web Services

5.1 SCA Web Services Callback Protocol

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

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

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

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

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

3. When the service implementation invokes the callback interface, it MUST use the Callback EPR from a request message that invoked the forward interface.~~When the service implementation invokes the callback interface, it MUST use the Callback EPR from a request message that invoked the forward interface, as specified in BWS50003.~~ [BWS50005] 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.~~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.~~ [BWS50006]

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.~~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.~~ [BWS50007] 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.~~The `wsa:RelatesTo` SOAP header~~

Formatted: Highlight

Formatted: Highlight

~~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.~~ [BWS50008]

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.~~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.~~ [BWS50009]

When a service that offers a bidirectional interface is invoked, depending on the semantics and/or implementation of the service, it is possible that the service might invoke the callback interface before the forward operation ends. In such cases, it is necessary for the binding on the reference-side to be listening for callback request(s) from the service, before the forward operation request is sent on the wire to the service, and continue listening as long as callback requests are expected. It is possible that before the response to the forward request is sent a response to one or more callback requests are required by the service.

5.2 SCA Web Services Callback Protocol with WS-MakeConnection ~~Protocol~~

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

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

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

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

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

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

5.3 Policy Assertion for SCA Web Services Callback Protocol

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

5.3.1 Assertion Model

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

5.3.2 Normative Outline

The normative outline for the WSCallback assertion is:

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

The following describes the content model of the WSCallback element.

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

5.3.3 Assertion Attachment

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

- Endpoint Policy Subject

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

Formatted: Not Highlight

Formatted: Not Highlight

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: wsdl:portType~~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: wsdl:portType~~ [BWS50013]

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

- wsdl:port
- wsdl:binding

5.3.4 Assertion Example

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

```
(01)<wsdl:definitions
(02)    targetNamespace="example.com"
(03)    xmlns:tns="example.com"
(04)    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
(05)    xmlns:wsp="http://www.w3.org/ns/ws-policy"
(06)    xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"
(07)    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
(08)
(09) <wsp:UsingPolicy wsdl:required="true" />
(10)
(11) <wsp:Policy wsu:Id="MyPolicy" >
(12)   <sca:WSCallback/>
(13) </wsp:Policy>
(14)
(15) <!-- omitted elements -->
(16)
(17) <wsdl:binding name="MyBinding" type="tns:MyPortType" >
(18)   <wsp:PolicyReference URI="#MyPolicy" />
(19)   <!-- omitted elements -->
(20) </wsdl:binding>
(21)
(22)</wsdl:definitions>
```

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

5.3.5 Security Considerations

Policies and assertions SHOULD be signed to prevent tampering. [BWS50014] 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. [BWS50015] That is, a relying party shouldn't rely on a policy unless the policy is signed and presented with sufficient claims to pass the relying parties acceptance criteria.

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

6 Conformance

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

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

a) SCA WS Binding XML Document

b) SCA Runtime

6.1 SCA WS Binding XML Document

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

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

6.2 SCA Runtime

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

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

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

```
<?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:wsdli="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="wsdli:wsdlLocation" use="optional"/>
                <anyAttribute namespace="##any" processContents="lax"/>
            </extension>
        </complexContent>
    </complexType>

</schema>
```

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>
```

C. Conformance Items

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

Conformance ID	Description
[BWS20001]~~[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]~~[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 *wsdl.binding* form of *wsdlElement* is used , the endpoint address URI for an SCA reference MUST be specified by either the *@uri* attribute on the binding or a WS-Addressing *EndpointReference* element, except where the SCA Assembly Model specification [SCA-Assembly]~~[SCA-Assembly]~~ states that the *@uri* attribute can be omitted.
[BWS20016]	The *@wsdli:wsdlLocation* attribute MAY be specified by the binding in the event that the <WSDL-namespace-URI> 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 <WSDL-namespace-URI>.
[BWS20017]	If the *@wsdli:wsdlLocation* attribute is used the *@wsdlElement* attribute MUST also be specified.
[BWS20018]	The value of the @wsdli:wsdlLocation attribute MUST identify an existing WSDL 1.1 document.
[BWS20019]	A binding.ws element MUST NOT contain more than one of any of the following: the @uri attribute; the @wsdlElement attribute referring to a WSDL port or to a WSDL service; the endpointReference element.
[BWS20020]	For the *callback* element of an SCA service, the binding MUST NOT specify an endpoint address URI or a WS-Addressing EndpointReference.
[BWS20021]	The SCA runtime MUST support all the attributes of the <binding.ws> element, namely @name, @uri, @requires, @policySets, @wsdlElement, and @wsdli:wsdlLocation.
[BWS20022]	The SCA runtime SHOULD support the element <endpointReference>.
[BWS20023]	If an SCA runtime does not support the element <endpointReference>, then it MUST reject an SCA WS Binding XML document (as defined in Section 5.1) that contains the element.
[BWS20024]	The <binding.ws> element MUST conform to the XML schema defined in sca-binding-webservice.xsd.
[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 *binding.ws* is used on a service or reference with an interface that is not defined by *interface.wsdl*, the SCA runtime MUST derive a WSDL portType 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 binding.ws element does not map to a WSDL portType.
[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]**~~**[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: 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]**~~**[WS-Addr]**~~ to invoke operations on the callback interface.
[BWS50007]~~[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]~~[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]~~[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: wsdl: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: Highlight

D. Appendix - WSDL Generation

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:

- component name = the value of the @name attribute of the component element containing the binding.ws element
- service name = the value of the @name attribute of the service element containing the binding.ws element
- binding name = the value of @name attribute of the binding.ws element, or the default if no @name attribute is present
- SOAP version = either "SOAP11" or "SOAP12" as appropriate

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

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

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 S4 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:

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

R2:

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:From>
     <wsa:Address>http://example.com/callback</wsa:Address>
     <wsa:ReferenceProperties>
       <myNS:SomeID>1</myNS:SomeID>
     </wsa:ReferenceProperties>
   </wsa:From>
   <wsa:MessageID>urn:uuid:f81d4fae-8dec-11d0-a765-
00a0c91e6bf6</wsa:messageID>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

S3:

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

R3:

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

<u>S4, S5:</u>

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

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.

<u>R4:</u>

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:From>
     <wsa:Address>http://docs.oasis-open.org/ws-
rx/wsmc/200702/anonymous?id=650e8400-f29b-11d4-a716-446655440010</wsa:Address>
   </wsa:From>
   <wsa:MessageID>urn:uuid:f81d4fae-10dec-11d0-a765-
00a0c91e6bf6</wsa:messageID>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

<u>MakeConnection polling message (from R to S):</u>

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

<u>S6:</u>

```
<soap:Envelope ...>
 <soap:Header>
   <wsa:To>http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id=650e8400-
f29b-11d4-a716-446655440010</wsa:To>
   <wsa:RelatesTo RelationshipType="http://docs.oasis-open.org/opencsa/sca-
bindings/ws/callback">urn:uuid:f81d4fae-10dec-11d0-a765-
00a0c91e6bf6</wsa:RelatesTo>
    ...
 </soap:Header>
 <soap:Body>
   ...
 </soap:Body>
</soap:Envelope>
```

F. Acknowledgements

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

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 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

[optional; should not be included in OASIS Standards]

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

cd02-rev1	2009-06-02	Anish Karmarkar	* 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	* 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	* 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.
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)