Service Component Architecture JMS Binding Specification Version 1.1

Committee Specification Draft 06

14 July 2011

Specification URIs:
This version:
> http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csd06.pdf
> (Authoritative)
> http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csd06.html
> http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csd06.doc

Previous version:
> http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd03.pdf
> (Authoritative)
> http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd03.html
> http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd03.doc

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

Technical Committee:
> OASIS Service Component Architecture / Bindings (SCA-Bindings) TC

Chair:
> Simon Holdsworth (simon_holdsworth@uk.ibm.com), IBM

Editors:
> Simon Holdsworth (simon_holdsworth@uk.ibm.com), IBM
> Anish Karmarkar (Anish.Karmarkar@oracle.com), Oracle

Related work:
> This specification replaces or supersedes:

- Service Component Architecture JMS Binding Specification Version 1.00

> This specification is related to:

- Service Component Architecture Assembly Model Specification Version 1.1
- SCA Policy Framework Version 1.1

Declared XML namespace:
> http://docs.oasis-open.org/ns/opencsa/sca/200912

Abstract:
> This document specifies the means by which SCA composites and components, as defined in the SCA Assembly Specification **[SCA-Assembly]**, connect to and access services using a messaging protocol. The connectivity is based on the Java Messaging Service **[JMS]** and is provided by a binding.jms element which applies to the references and services of an SCA component or composite.

The JMS binding provides JMS-specific details of the connection to the required JMS resources. It supports the use of Queue and Topic type destinations.

The binding is especially well suited for use by services and references of composites that are directly deployed, as opposed to composites that are used as implementations of higher-level components. Services and references of deployed composites become system-level services and references, which are intended to be used by non-SCA clients.

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" 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).

Citation format:

When referencing this specification the following citation format should be used:

[SCA-JMSBinding]

Service Component Architecture JMS Binding Specification Version 1.1. 14 July 2011. OASIS Committee Specification Draft 06. http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csd06.html.

Notices

Copyright © OASIS Open 2011. All Rights Reserved.

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

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

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

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

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

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

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

The names "OASIS", "SCA" and "Service Component Architecture" are trademarks 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

This document specifies the means by which SCA composites and components, as defined in the SCA Assembly Specification **[SCA-Assembly]**, connect to and access services using a messaging protocol. The connectivity is based on the Java Messaging Service **[JMS]** and is provided by a binding.jms element which applies to the references and services of an SCA component or composite.

The JMS binding provides JMS-specific details of the connection to the required JMS resources. It supports the use of Queue and Topic type destinations.

The binding is especially well suited for use by services and references of composites that are directly deployed, as opposed to composites that are used as implementations of higher-level components. Services and references of deployed composites become system-level services and references, which are intended to be used by non-SCA clients.

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 RFC Keywords [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.

Prefix	Namespace	Notes
xs	"http://www.w3.org/2001/XMLSchema"	Defined by XML Schema 1.0 specification
sca	"http://docs.oasis-open.org/ns/opencsa/sca/200912"	Defined by the SCA specifications

Table 1-1: Prefixes and Namespaces used in this specification

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.

[JMS] Java™ Message Service Specification v1.1 http://www.oracle.com/technetwork/java/jms/index.html

[JNDI] Java™ Naming and Directory Interface http://www.oracle.com/technetwork/java/jndi/index.html

[WSDL] 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.

R. 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.

[JCA15] J2EE Connector Architecture Specification Version 1.5 http://java.sun.com/j2ee/connector/

[IETFJMS] M. Phillips, P. Adams, D. Rokicki, E. Johnson, *URI Scheme for Java™ Message Service 1.0*, http://www.ietf.org/rfc/rfc6167.txt, IETF RFC 6167, April 2011.

[SCA-Assembly] OASIS Committee Specification Draft 07, *Service Component Architecture Assembly Model Specification Version 1.1*, January 2011 http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-csd07.pdf

40	**[SCA-Policy]**	OASIS Committee Draft 04, *SCA Policy Framework Specification Version 1.1*,
41		September 2010
42		http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd04.pdf

1.3 Non-Normative References

44	**[SCA-JMSTest]**	OASIS Committee Specification Draft 01, *SCA JMS Binding v1.1 TestCases*
45		*Version 1.0*, November 2010,
46		http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-testcases-
47		1.0-csd01.pdf

1.4 Naming Conventions

49 The naming conventions used by artefacts defined in this specification are:

50 • The naming conventions defined by section 1.3 of the SCA Assembly Specification [SCA-Assembly].

51 • Where the names of elements and attributes consist partially or wholly of acronyms, the letters of the
52 acronyms use the same case. When the acronym appears at the start of the name of an element or
53 an attribute, or after a period, it is in lower case. If it appears elsewhere in the name of an element or
54 an attribute, it is in upper case. For example, an attribute might be named "uri" or "jndiURL".

55 • Where the names of types consist partially or wholly of acronyms, the letters of the acronyms are in
56 all upper case. For example, an XML Schema type might be named "JCABinding" or "MessageID".

57 • Values, including local parts of QName values, follow the rules for names of elements and attributes
58 as stated above, with the exception that the letters of acronyms are in all upper case. For example, a
59 value might be "JMSDefault" or "namespaceURI".

1.5 Testcases

61 SCA JMS Binding TestCases Version 1.1 **[SCA-JMSTest]** defines test cases for this specification. The
62 TestCases represent a series of tests that SCA runtimes are expected to pass in order to claim
63 conformance to the requirements of this specification.

2 Messaging Bindings

Messaging bindings form a category of SCA bindings that represent the interaction of SCA composites with messaging providers. It is felt that documenting, and following this pattern is beneficial for implementers of messaging bindings, although it is not strictly necessary.

This pattern is embodied in the JMS binding, described later.

Messaging bindings utilize operation selector and wire format elements to provide the mapping from the native messaging format to an invocation on the target component. A default operation selection and data binding behavior is specified.

In addition, each operation in the interface associated with the service or reference can have properties specified, that influence the way native messages are processed depending on the operation being invoked.

3 JMS Binding Schema

75

76 The JMS binding element is defined by the pseudo-schema in Snippet 3-1.

```
<binding.jms correlationScheme="QName"?
             initialContextFactory="xs:anyURI"?
             jndiURL="xs:anyURI"?
             name="NCName"?
             requires="list of QName"?
             policySets="list of QName"?
             uri="xs:anyURI"? >
    <destination jndiName="xs:anyURI"? type="queue or topic"?
            create="always or never or ifNotExist"?>
        <property name="NMTOKEN" type="string"?>*
    </destination>?
    <connectionFactory jndiName="xs:anyURI"?
                    create="always or never or ifNotExist"?>
        <property name="NMTOKEN" type="string"?>*
    </connectionFactory>?
    <activationSpec jndiName="xs:anyURI"?
                create="always or never or ifNotExist"?>
        <property name="NMTOKEN" type="string"?>*
    </activationSpec>?

    <response>
        <destination jndiName="xs:anyURI"? type="queue or topic"?
                create="always or never or ifNotExist"?>
            <property name="NMTOKEN" type="string"?>*
        </destination>?
        <connectionFactory jndiName="xs:anyURI"?
                        create="always or never or ifNotExist"?>
            <property name="NMTOKEN" type="string"?>*
        </connectionFactory>?
        <activationSpec jndiName="xs:anyURI"?
                    create="always or never or ifNotExist"?>
            <property name="NMTOKEN" type="string"?>*
        </activationSpec>?
        <wireFormat/>?
    </response>?

    <resourceAdapter name="NMTOKEN">?
        <property name="NMTOKEN" type="string"?>*
    </resourceAdapter>?

    <headers type="string"?
            deliveryMode="persistent or nonpersistent"?
            timeToLive="long"?
            priority="0 .. 9"?>
        <property name="NMTOKEN" type="boolean or byte or .. or String"?>*
    </headers>?

    <messageSelection selector="string"?>
        <property name="NMTOKEN" type="string"?>*
    </messageSelection>?

    <operationProperties name="string" selectedOperation="string"?>
        <property name="NMTOKEN" type="string"?>*
        <headers type="string"?
                deliveryMode="persistent or nonpersistent"?
                timeToLive="long"?
                priority="0 .. 9"?>
```

```
134              <property name="NMTOKEN" type="boolean or byte or .. or String"?>*
135          </headers>?
136      </operationProperties>*
137
138      <wireFormat ... />?
139      <operationSelector ... />?
140  </binding.jms>
```

Snippet 3-1: binding.jms Pseudo-Schema

142 The binding can be used in one of two ways, either identifying existing JMS [JMS] resources using JNDI
143 **[JNDI]** names, or providing the required information to enable the JMS resources to be created.

144 The `binding.jms` element has the attributes:

145 • **/binding.jms** – This is the JMS binding element. The element is extensible so that JMS binding
146 implementers can add additional JMS provider-specific attributes and elements although such
147 extensions are not guaranteed to be portable across runtimes.

148 • **/binding.jms/@uri** – as defined in the SCA Assembly Specification [SCA-Assembly]. This attribute
149 identifies the destination, connection factory or activation spec, and other properties to be used to
150 send/receive the JMS message. There is an implicit `@create="never"` for the resources referred to
151 in the `@uri` attribute. Message header properties and the message selector set via the `@uri` attribute
152 take precedence over those specified in binding elements as defined in section 3.2.

153 The value of the `@uri` attribute MUST have the format defined by the IETF URI Scheme for Java™
154 Message Service 1.0 [IETFJMS] [BJM30001].

155 Snippet 3-2 illustrates the structure of the URI and the set of property names that have specific
156 semantics:

```
157  jms:jndi:<jms-dest>?
158  jndiURL=<jndi-url> &
159  jndiInitialContextFactory=<jndi-initial-context-factory> &
160  jndiConnectionFactoryName=<Connection-Factory-Name> &
161  deliveryMode=<Delivery-Mode> &
162  timeToLive=<Time-To-Live> &
163  priority=<Priority> &
164  selector=<Message-Selector> &
165  <param-name>=<param-value> & …
```

Snippet 3-2: JMS URI Structure

167 When the `@uri` attribute is specified, the SCA runtime MUST raise an error if the referenced
168 resources do not already exist [BJM30002].

169 When the `@uri` attribute is specified, the `destination` element MUST NOT be present
170 [BJM30034].

171 • **/binding.jms/@name** - as defined in the SCA Assembly Specification [SCA-Assembly].

172 • **/binding.jms/@requires** - as defined in the SCA Assembly Specification [SCA-Assembly].

173 • **/binding.jms/@policySets** - as defined in the SCA Assembly Specification [SCA-Assembly].

174 • **/binding.jms/@correlationScheme** – identifies the correlation scheme used when sending reply or
175 callback messages, default value is "sca:messageID". Three specific behaviours are provided.
176 "sca:messageID" indicates that response messages can be correlated with their requests by
177 looking for the request's messageID header value in the response's correlationID header;
178 "sca:correlationID" indicates that response messages can be correlated with their requests by
179 looking for the request's correlationID header value in the response's correlationID header;
180 "sca:none" indicates that the response's correlationID header is not to be used for this purpose and
181 some other means is used for the correlation.

182 If the value of the `@correlationScheme` attribute is "sca:messageID" the SCA runtime MUST set
183 the correlation ID of replies to the message ID of the corresponding request [BJM30003].

184 If the value of the `@correlationScheme` attribute is "`sca:correlationID`" the SCA runtime
185 MUST set the correlation ID of replies to the correlation ID of the corresponding request [BJM30004].

186 If the value of the `@correlationScheme` attribute is "`sca:correlationID`" the SCA runtime
187 MUST set a non-null correlation ID value in requests that it sends [BJM30007].

188 If the value of the `@correlationScheme` attribute is "`sca:none`" the SCA runtime MUST NOT set
189 the correlation ID in responses that it sends[BJM30005].

190 SCA runtimes supporting other correlation schemes can allow additional values for the
191 `@correlationScheme` attribute.

192 • ***/binding.jms/@initialContextFactory*** – the name of the JNDI initial context factory.

193 • ***/binding.jms/@jndiURL*** – the URL for the JNDI provider.

194 • ***/binding.jms/destination*** – identifies the destination that is to be used to process requests by this
195 binding.

196 • ***/binding.jms/destination/@type*** - the type of the request destination. Valid values are "`queue`" and
197 "`topic`". The default value is "`queue`".

198 Whatever the value of the `destination/@type` attribute, the SCA runtime MUST ensure a single
199 response is delivered for request/response operations [BJM30010].

200 • ***binding.jms/destination/@jndiName*** – the JNDI name of the JMS Destination that the binding uses
201 to send or receive messages. The behaviour of this attribute is determined by the value of the
202 `@create` attribute as follows:

203 – If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec`
204 element is "`always`" and the `@jndiName` attribute is present and the resource cannot be created
205 at the location specified by the `@jndiName` attribute then the SCA runtime MUST raise an error
206 [BJM30011].

207 If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec`
208 element is "`always`" and the `@jndiName` attribute is not present and the resource cannot be
209 created, then the SCA runtime MUST raise an error [BJM30037].

210 If the `@jndiName` attribute is omitted this specification places no restriction on the JNDI location
211 of the created resource.

212 – If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec`
213 element is "`ifNotExist`" then the `@jndiName` attribute MUST specify the location of the
214 possibly existing resource [BJM30012].

215 If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec`
216 element is "`ifNotExist`" and the resource does not exist at the location identified by the
217 `@jndiName` attribute and cannot be created there then the SCA runtime MUST raise an error
218 [BJM30013].

219 If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec`
220 element is "`ifNotExist`" and the `@jndiName` attribute refers to an existing resource that is not
221 a JMS Destination of the approprate type, a JMS connection factory or a JMS activation spec
222 respectively then the SCA runtime MUST raise an error [BJM30014].

223 – If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec`
224 element is "`never`" and the `@jndiName` attribute is not specified, or the resource is not present
225 at the location identified by the `@jndiName` attribute, or the location refers to a resource of an
226 incorrect type then the SCA runtime MUST raise an error [BJM30015].

227 • ***/binding.jms/destination/@create*** – indicates whether the destination should be created when the
228 containing composite is deployed. Valid values are "`always`", "`never`" and "`ifNotExist`".
229 "`always`" indicates that new resources are created for use by this binding; "`never`" indicates that
230 existing resources are used and none created; "`ifNotExist`" indicates that if the resources

231   already exist those are used, otherwise new ones are created. Refer to the
232   `destination/@jndiName` attribute for a detailed definition of each case. The default value is
233   "`ifNotExist`".

234 • ***/binding.jms/destination/property*** – defines properties to be used to create the destination, if
235   required.

236 • ***/binding.jms/connectionFactory*** – identifies the connection factory that the binding uses to process
237   request messages. The attributes of this element follow the rules defined for the `destination`
238   element.

239   A `binding.jms` element MUST NOT include both a `connectionFactory` element and an
240   `activationSpec` element [BJM30017].

241   When the `connectionFactory` element is present as a child of the `binding.jms` element, then
242   the destination MUST be defined either by the `destination` element child of the `binding.jms`
243   element or the `@uri` attribute of the `binding.jms` element [BJM30018].

244 • ***/binding.jms/activationSpec*** – identifies the activation spec that the binding uses to connect to a
245   JMS destination to process request messages. The attributes of this element follow the rules defined
246   for the `destination` element.

247   If the `activationSpec` element is present as a child of the `binding.jms` element and the
248   destination is also specified via a `destination` element child of the `binding.jms` element or the
249   `@uri` attribute of the `binding.jms` element then it MUST refer to the same JMS destination as the
250   `activationSpec` [BJM30019].

251   The `activationSpec` element MUST NOT be present when the binding is being used for an SCA
252   reference [BJM30020].

253 • ***/binding.jms/response*** – defines the resources used for handling response messages (receiving
254   responses for a reference, and sending responses from a service).

255 • ***/binding.jms/response/destination*** – identifies the destination that is to be used to process
256   responses by this binding. Attributes follow the rules defined for the parent's `destination` element.
257   For a service, this destination is used to send responses to messages that have a null value for the
258   `JMSReplyTo` destination. For a reference, this destination is used to receive reply messages

259 • ***/binding.jms/response/connectionFactory*** – identifies the connection factory that the binding uses
260   to process response messages. The attributes of this element follow those defined for the
261   `destination` element.

262   A `response` element MUST NOT include both a `connectionFactory` element and an
263   `activationSpec` element [BJM30021].

264 • ***/binding.jms/response/activationSpec*** – identifies the activation spec that the binding uses to
265   connect to a JMS destination to process response messages. The attributes of this element follow
266   those defined for the `destination` element.

267   If a `response/destination` and `response/activationSpec` element are both specified they
268   MUST refer to the same JMS destination [BJM30022].

269   The `response/activationSpec` element MUST NOT be present when the binding is being used
270   for an SCA service [BJM30023].

271 • ***/binding.jms/response/wireFormat*** – identifies the wire format used by responses sent or received
272   by this binding. This value overrides the `wireFormat` specifed at the binding level. Wire formats for
273   this binding are described in Section 4.

274 • ***/binding.jms/headers*** – this element specifies values to be set for standard JMS headers. These
275   values apply to requests from a reference and responses from a service. Section 3.2 defines the
276   priority rules for determining the values for JMS headers and user properties.

277 • ***/binding.jms/headers/@type, @deliveryMode, @timeToLive, @priority*** – specifies the value to
278   use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive or JMSPriority

279	respectively. Valid values for `@deliveryMode` are "`persistent`" and "`nonpersistent`",
280	corresponding to the values defined in the JMS Specification **[JMS]** for the JMSDeliveryMode
281	message header, with "`persistent`" being the default; valid values for `@priority` are "`0`" to
282	"`9`", where "`0`" indicates lowest priority and "`9`" highest priority, with "`4`" being the default; valid
283	values for `@timeToLive` are positive integers, with 0 indicating unlimited time and being the default
284	value.

- 285 • */binding.jms/headers/property* – specifies the value and type for the named JMS user property.

- 286 • */binding.jms/messageSelection* - this element specifies JMS message selection options. This
- 287 element applies to a service receiving messages from the request destination or for a reference
- 288 receiving messages from the callback or reply-to destination.

- 289 • */binding.jms/messageSelection/@selector* - specifies the value to use for the JMS message
- 290 selector. Section 3.3 defines the priority rules for determining the values for the message selector.

- 291 • */binding.jms/resourceAdapter* – specifies name, type and properties of the Resource Adapter Java
- 292 bean. The resource adapter and SCA runtime together define the set of valid properties for
- 293 configuring the resource adapter via the JMS binding.

- 294 The `resourceAdapter` element MUST be present when JMS resources are to be created for a JMS
- 295 provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise
- 296 [BJM30031].

- 297 For JMS providers that do not implement the JCA 1.5 specification [JCA15], information necessary for
- 298 resource creation can be added in provider-specific elements or attributes allowed by the extensibility
- 299 of the `binding.jms` element.

- 300 • */binding.jms/operationProperties* – specifies various properties that are specific to the processing
- 301 of a particular operation.

- 302 • */binding.jms/operationProperties/@name* – The name of the operation in the interface.

- 303 • */binding.jms/operationProperties/@selectedOperation* – The value generated by the
- 304 `operationSelector` that corresponds to the operation in the service or reference interface
- 305 identified by the `operationProperties/@name` attribute. If this attribute is omitted then the value
- 306 defaults to the value of the `operationProperties/@name` attribute.

- 307 The value of the *operationProperties/@selectedOperation* attribute MUST be unique across the
- 308 containing `binding.jms` element [BJM30029].

- 309 • */binding.jms/operationProperties/property* – specifies properties specific to this operation. These
- 310 properties are intended to be used to parameterize the `wireFormat` identified for the binding for a
- 311 particular operation.

- 312 • */binding.jms/operationProperties/headers* – this element specifies values to be set for standard
- 313 JMS headers. These values apply to requests from a reference and responses from a service.
- 314 Section 3.2 defines the priority rules for determining the values for JMS headers and user properties.

- 315 • */binding.jms/operationProperties/headers/@type, @deliveryMode, @timeToLive, @priority* –
- 316 specifies the value to use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive
- 317 or JMSPriority, respectively. Refer to the description of the `binding.jms/headers` element for the
- 318 valid values for these attributes.

- 319 • */binding.jms/operationProperties/headers/property* – specifies the value and type for the named
- 320 JMS user property.

- 321 • */binding.jms/wireFormat* – identifies the wire format used by requests and responses sent or
- 322 received by this binding. Wire formats for this binding are described in Section 4.

- 323 • */binding.jms/operationSelector* – identifies the operation selector used when receiving requests for
- 324 a service. If specified for a reference this provides the default operation selector for callbacks if not
- 325 specified via a callback service element. Operation selectors for this binding are described in Section
- 326 3.2.

327 The `binding.jms` element MUST conform to the XML schema defined in sca-binding-jms-1.1.xsd
328 [BJM30036].

3.1 Extensibility

330 The JMS binding allows further customization of the binding element and its subelements with vendor
331 specific attributes or elements. This is done by providing extension points in the schema; refer to
332 Appendix A, "JMS XML Binding Schema: sca-binding-jms-1.1.xsd" for the locations of these extension
333 points.

3.2 JMS Message Headers and User Properties

335 The JMS binding can be configured to specify that JMS headers are set to specific values in messages
336 sent by the SCA runtime.   The binding provides several places where JMS message headers and user
337 properties can be specified at different levels of granularity.

338 The type of the JMS user property is specified via the property/@type attribute using one of the values
339 define in the JMS specification **[JMS]**: "boolean", "byte", "short", "int", "long", "float", "double", "String" (the
340 default), or "xs:string". "xs:string" and "String" both represent the String user property type, "xs:string" is
341 for backward compatibility only and its use is deprecated.

342 When sending messages for a JMS binding, the SCA runtime MUST set each of the JMSType,
343 JMSDeliveryMode, JMSTimeToLive and JMSPriority headers to values specified in the binding definition
344 in the following priority order:

345   1) the value for the header specified in the `@uri` attribute (highest priority);

346   2) the value for the header specified in the `operationProperties/headers` element matching the
347 operation being invoked;

348   3) the value for the header specified in the `headers` element;

349   4) the default value for the header as specified by the definition of the `binding.jms/headers`
350 element (lowest priority) [BJM30024].

351 When sending messages for a JMS binding, the SCA runtime MUST set each named user property with
352 type and value specified in the binding definition in the following priority order:

353   1) the type and value for the named user property specified in an
354 `operationProperties/headers/property` element matching the name of the operation being
355 invoked (highest priority);

356   2) the type and value for the named user property specified in a `headers/property` element (lowest
357 priority) [BJM30025].

3.3 JMS Message Selection

359 Message selectors can be specified for the JMS binding to receive a specific subset of messages from a
360 given destination, such that only messages that match the selector are delivered to a given JMS binding.
361 This allows more than one JMS binding to share a destination.

362 When receiving messages for a JMS binding, the SCA runtime MUST use a message selector if specified
363 in the binding definition in the following priority order:

364   1) the value for the message selector specified in the `@uri` attribute value's "selector" parameter
365 (highest priority);

366   2) the value for the message selector specified in the `messageSelection/@selector` attribute;

367   3) otherwise no message selector is used (lowest priority) [BJM30026].

4 Operation Selectors and Wire Formats

In general messaging providers deal with message formats and destinations. There is not usually a built-in concept of "operation" that corresponds to that defined in a WSDL [WSDL] portType. Messages have a wire format which corresponds in some way to the schema of an input or output message of an operation in the interface of a service or reference, however additional information is required in order for an SCA runtime to know how to identify the operation and understand the wire format of messages.

The process of identifying the operation to be invoked is *operation selection*; the information that describes the contents of messages is a *wire format*. The `binding` element as described in the SCA Assembly Specification [SCA-Assembly] provides the means to identify specific operation selection via the `operationSelector` element and the wire format of messages received and to be sent using the `wireFormat` element. The `operationSelector` and `wireFormat` elements allow a binding element to specify behaviour defined by the binding specification or custom behaviour provided by an SCA runtime.

When the service with a JMS binding receives a message, the SCA runtime resolves the name of the operation in the service's interface that is to be invoked by using the `operationSelector` and `operationProperties` elements defined for the binding. The *resolved operation name* is defined as follows:

- If the selected operation name generated by the `operationSelector` matches the value of an `operationProperties/@selectedOperation` attribute then the resolved operation name is the value of the `operationProperties/@name` attribute.

- Otherwise the resolved operation name is the selected operation name generated by the `operationSelector`.

When a message is received at an SCA service with JMS binding and the resolved operation name is in the target component's interface, the SCA runtime MUST invoke the target component using the resolved operation name [BJM40010].

When a message is received at an SCA service with JMS binding and the resolved operation name is not in the target component's interface the SCA runtime MUST raise an error [BJM40011].

No standard means is provided for linking the `wireFormat` or `operationSelector` elements with the runtime components that implement their behavior.

The following sections describe the default `operationSelector` and `wireFormat` for a JMS binding.

4.1 Default Operation Selection

The following defines the **default operation selection algorithm** when receiving a request at a service, or a callback at a reference. When using the default operation selection algorithm, the selected operation name is determined as follows:

- If there is only one operation on the service's interface, then that operation is the selected operation name;

- Otherwise, if the JMS user property "`scaOperationName`" is present, then the value of that user property is used as the selected operation name;

- Otherwise, if the message is a JMS text or bytes message containing XML, then the selected operation name is the local name of the root element of the XML payload;

- Otherwise, the selected operation name is "`onMessage`".

When a `binding.jms` element specifies the `operationSelector.jmsDefault` element, the SCA runtime MUST use the default operation selection algorithm to determine the selected operation [BJM40008].

412 If no `operationSelector` element is specified then SCA runtimes MUST use
413 `operationSelector.jmsDefault` as the default [BJM40002].

4.2 Default Wire Format

415 The default wire format maps between a `JMSMessage` and the object(s) expected by the component
416 implementation. We encourage component implementers to avoid exposure of JMS [JMS] APIs to
417 component implementations, however in the case of an existing implementation that expects a
418 `JMSMessage`, this provides for simple reuse of that as an SCA component.

419 When using the default wire format, the message body is mapped to the parameters or return value of the
420 target operation as follows:

421 • If there is a single parameter that is a `JMSMessage`, then the `JMSMessage` is passed as is.

422 • Otherwise, if the `JMSMessage` is not a JMS text message or bytes message containing XML it is
423   invalid.

424 • Otherwise if there is a single parameter, or for the return value, the JMS text or bytes XML payload is
425   the XML serialization of that parameter according to the WSDL schema for the message.

426 • Otherwise the multiple parameters are encoded in XML using the document wrapped style, according
427   to the WSDL schema for the message.

428 When a `binding.jms` element specifies the `wireFormat.jmsDefault` element, the SCA runtime
429 MUST use the default wire format [BJM40009].

430 When using the default wire format to send request messages, if there is a single parameter and the
431 interface includes more than one operation, the SCA runtime MUST set the JMS user property
432 "`scaOperationName`" to the name of the operation being invoked [BJM40003].

433 When using the default wire format an SCA runtime MUST be able to receive both JMS text and bytes
434 messages [BJM40005].

435 When using the default wire format an SCA runtime MUST send either a JMS text or a JMS bytes
436 message [BJM40006].

437 If no `wireFormat` element is specified in a JMS binding then SCA runtimes MUST use
438 `wireFormat.jmsDefault` as the default [BJM40004].

439 The default wire format allows a choice of text or bytes format when sending messages; an SCA runtime
440 can restrict this to one or other via additional configuration.

4.2.1 Example of default wire format

442 For the interface definition in Snippet 4-1:

```
<wsdl:definitions name="Coordinates"
targetNamespace="http://tempuri.org/coordinates"
xmlns:tns="http://tempuri.org/coordinates"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <xsd:schema targetNamespace="http://tempuri.org/coordinates">
      <xsd:element name="setCoordinates">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="x" type="xsd:int"/>
            <xsd:element name="y" type="xsd:int"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="setCoordinatesRequestMsg">
```

```
462        <wsdl:part element="tns:setCoordinates" name="setCoordinatesParameters"/>
463      </wsdl:message>
464
465      <wsdl:portType name="Coordinates">
466        <wsdl:operation name="setCoordinates">
467          <wsdl:input message="tns:setCoordinatesRequestMsg"
468                name="setCoordinatesRequest"/>
469        </wsdl:operation>
470      </wsdl:portType>
471    </wsdl:definitions>
```

Snippet 4-1: Example WSDL Interface Definition

When the `setCoordinates` operation is invoked via a reference with a JMS binding that uses the default wire format, the message sent from the JMS binding is a JMS text or bytes message with the content shown in Snippet 4-2:

```
<setCoordinates xmlns="http://tempuri.org/coordinates">
  <x>10</x>
  <y>5</y>
</setCoordinates>
```

Snippet 4-2: JMS Message Content for setCoordinates Operation of Snippet 4-1

5 Policy

The JMS binding provides attributes that control the sending of messages, requests from references and replies from services. These values can be set directly on the binding element for a particular service or reference, or they can be set using policy intents. An example of setting these via intents is shown later.

JMS binding implementations MUST support the JMS intent [BJM50001].

The JMS intent MUST always be included in the `@alwaysProvides` attribute of the JMS `bindingType` [BJM50002]

The following standard intents can also be supported by JMS binding implementations, by inclusion in the `@alwaysProvides` or `@mayProvides` attribute of the JMS `bindingType`:

- atLeastOnce

- atMostOnce

- ordered

The `atLeastOnce`, `atMostOnce` and `ordered` intents are defined in the SCA Policy Specification [SCA-Policy] document in section 8, "Reliability Policy".

This specification does not define a fixed relationship between the reliability intents and the persistence of JMS messages. Deployers/assemblers can configure a nonpersistent delivery mode via the `@deliveryMode` or `@uri` attribute, in order to provide higher performance with a decreased quality of service. However a binding.jms element configured with a nonpersistent delivery mode might not be able to satisfy the `atLeastOnce` policy intent. The SCA Policy Specification [SCA-Policy] requires that an error be raised if the SCA runtime is unable to support the intents on a binding in combination with the specific configuration of that binding.

6 Message Exchange Patterns

502

503 This section describes the message exchange patterns that are possible when using the JMS binding,
504 including one-way, request/response and callbacks. JMS [JMS] has a looser concept of message
505 exchange patterns than WSDL, so this section explains how JMS messages that are sent and received
506 by the SCA runtime relate to the WSDL input/output messages. Each operation in a WSDL interface is
507 either one-way or request/response. Callback interfaces can include both one-way and request/response
508 operations.

6.1 One-way message exchange (no Callbacks)

509

510 A one-way message exchange is one where a request message is sent that does not require or expect a
511 corresponding response message. These are represented in WSDL as an operation with an `input`
512 element and no `output` elements and no `fault` elements. The JMS specification provides the
513 `JMSReplyTo` header as the way for a JMS application to identify the destination on which replies or other
514 messages are to be placed that relate to the one being sent. For one-way requests sent by SCA
515 references with unidirectional interfaces, the `JMSReplyTo` will not usually be set as no reply or other
516 related message is expected.

517 For an SCA service with a JMS binding and unidirectional interface, when a request message is received
518 as part of a one-way MEP, the SCA runtime MUST ignore the `JMSReplyTo` destination header in the
519 JMS message, and not raise an error [BJM60002].

520 The use of one-way exchanges when using a bidirectional interface is described in section 6.4.

6.2 Request/response message exchange (no Callbacks)

521

522 A request/response message exchange is one where a request message is sent and a response
523 message is expected, possibly identified by its correlation identifier. These are represented in WSDL as
524 an operation with an `input` element and an `output` and/or a `fault` element.

525 For an SCA reference with a JMS binding, when a request message is sent as part of a request/response
526 MEP, and the JMS binding has a `response` element with a `destination` defined, then the SCA
527 runtime MUST use that destination for the `JMSReplyTo` header in the JMS message it creates for the
528 request [BJM60004].

529 For an SCA reference with a JMS binding, when a request message is sent as part of a request/response
530 MEP, and the JMS binding does not have a `response` element with a `destination` defined, the SCA
531 runtime MUST provide an appropriate destination on which to receive response messages and use that
532 destination for the `JMSReplyTo` header in the JMS message it creates for the request [BJM60005].

533 For an SCA reference with a JMS binding that does not have a destination specified via the response
534 element, the SCA runtime MUST either receive response messages as defined by the binding's
535 `@correlationScheme` attribute, or use a unique destination for each request/response interaction
536 [BJM60006].

537 For an SCA reference with a JMS binding that has a destination specified via the response element, the
538 SCA runtime MUST receive response messages as defined by the binding's `@correlationScheme`
539 attribute [BJM60003].

540 For an SCA service with a JMS binding, when a response message is sent as part of a request/response
541 MEP where the request message included a non-null `JMSReplyTo` destination, the SCA runtime MUST
542 send the response message to that destination [BJM60007].

543 For an SCA service with a JMS binding, when a response message is sent as part of a request/response
544 MEP where the request message included a null `JMSReplyTo` destination and the JMS binding includes
545 a `response/destination` element the SCA runtime MUST send the response message to that
546 destination [BJM60008].

547 For an SCA service with a JMS binding, when a request message is received as part of a
548 request/response MEP where the request message includes a null `JMSReplyTo` destination and the JMS
549 binding does not include a response/destination then the SCA runtime MUST NOT process the request
550 and MUST raise an error [BJM60009].

551 For an SCA service with a JMS binding, when a response message is sent as part of a request/response
552 MEP the SCA runtime MUST set the correlation identifier in the JMS message that it creates for the
553 response as defined by the JMS binding's `@correlationScheme` attribute [BJM60010].

554 The use of request/response exchanges when using a bidirectional interface is described in section 6.4.

6.3 JMS User Properties

556 This protocol assigns specific behavior to JMS user properties:

557 • `"scaCallbackDestination"` holds a JMS URI that identifies the Destination to which callback
558 messages are sent, in the format defined by the IETF URI Scheme for Java™ Message Service 1.0
559 **[IETFJMS]**.

6.4 Callbacks

561 Callbacks are SCA's way of representing bidirectional interfaces, where messages are sent in both
562 directions between a client and a service. A callback is the invocation of an operation on a service's
563 callback interface. A callback operation can be one-way or request/response. Messages that correspond
564 to one-way or request/response operations on a bidirectional interface use either the
565 `scaCallbackDestination` user property (for request/response) or the `JMSReplyTo` destination (for
566 one-way) to identify the destination to which messages are to be sent when operations are invoked on the
567 callback interface. The use of `JMSReplyTo` for this purpose is to enable interaction with non-SCA JMS
568 applications, as described below.

569 SCA runtimes MUST follow the behavior described in section 6.4 and its subsections when
570 `binding.jms` is used in both the forward and callback directions [BJM60018].

571 SCA runtimes can use different bindings for forward calls and callbacks, however the behavior and
572 requirements on messages is vendor-specific.

6.4.1 Invocation of operations on a bidirectional interface

574 For an SCA reference with a JMS binding and a bidirectional interface, when a request message is sent
575 as part of a request/response MEP the SCA runtime MUST set the `scaCallbackDestination` user
576 property in the message it creates to a JMS URI string, in the format defined by the IETF URI Scheme for
577 Java™ Message Service 1.0 **[IETFJMS]**, that identifies the destination to which callback messages are to
578 be sent [BJM60011].

579 For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as
580 part of a one-way MEP the SCA runtime MUST set the destination to which callback messages are to be
581 sent as the `JMSReplyTo` destination in the message it creates [BJM60012].

582 For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as
583 part of a request/response MEP, the SCA runtime MUST set the `JMSReplyTo` header in the message it
584 creates as described in section 6.2 [BJM60013].

585 For both one-way and request/response operations, the reference's callback service can be used to
586 identify the destination to which callback messages are to be sent.

587 For an SCA reference with a JMS binding and bidirectional interface, the SCA runtime MUST identify the
588 callback destination from the reference's callback service binding if present, or supply a suitable callback
589 destination if not present [BJM60014].

6.4.2 Invocation of operations on a callback interface

An SCA service with a callback interface can invoke operations on that callback interface by sending messages to the destination identified by the `scaCallbackDestination` user property, the `JMSReplyTo` destination, or the destination identified by the service's callback reference JMS binding.

For an SCA service with a JMS binding, the *callback destination* is identified as follows, in order of priority:

- The `scaCallbackDestination` identified by an earlier request/response operation, if not null;

- the `JMSReplyTo` destination identified by an earlier one-way operation, if not null;

- the request destination of the service's callback reference JMS binding, if specified

For an SCA service with a JMS binding, when a callback request message is sent for either a one-way or request/response MEP, the SCA runtime MUST send the callback request message to the callback destination. [BJM60015].

For an SCA service with a JMS binding, when a callback request message is sent and no callback destination can be identified then the SCA runtime MUST raise an error and throw an exception to the caller of the callback operation [BJM60016].

For an SCA service with a JMS binding, when a callback request message is sent the SCA runtime MUST set the `JMSReplyTo` destination in the callback request message as defined in sections 6.1 or 6.2 as appropriate for the type of the callback operation invoked [BJM60017].

6.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications

When interacting with non-SCA JMS applications, the assembler can choose to model a request/response message exchange using a bidirectional interface with a one-way operation in the forward and callback interfaces. In this case it is likely that the non-SCA JMS application does not support the use of the `scaCallbackDestination` user property. To support this, for one-way messages the `JMSReplyTo` header is used to identify the destination to be used to deliver callback messages, as described in sections 6.4.1 and 6.4.2.

7 Examples

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

7.1 Minimal Binding Example

Snippet 7-1 shows the JMS binding being used with no further attributes or elements. In this case, it is left to the deployer to identify the resources to which the binding is connected.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           name="MyValueComposite">

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

    <reference name="StockQuoteService">
        <interface.java interface="services.stockquote.StockQuoteService"/>
        <binding.jms/>
    </reference>
</composite>
```

Snippet 7-1: Minimal Binding Example

7.2 URI Binding Example

Snippet 7-2 shows the JMS binding using the `@uri` attribute to specify the connection type and its information:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           name="MyValueComposite">

    <service name="MyValueService">
        <interface.java interface="services.myvalue.MyValueService"/>
        <binding.jms uri="jms:MyValueServiceQueue?
                              activationSpecName=MyValueServiceAS&
                              ... "/>
    </service>

    <reference name="StockQuoteService">
        <interface.java interface="services.stockquote.StockQuoteService"/>
        <binding.jms uri="jms:StockQuoteServiceQueue?
                              connectionFactoryName=StockQuoteServiceQCF&
                              deliveryMode=1&
                              ... "/>
    </reference>
</composite>
```

Snippet 7-2: Binding Example with URI Specified

7.3 Binding with Existing Resources Example

Snippet 7-3 shows the JMS binding using existing resources:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
```

```
664            name="MyValueComposite">
665
666        <service name="MyValueService">
667            <interface.java interface="services.myvalue.MyValueService"/>
668            <binding.jms>
669                <destination jndiName="MyValueServiceQ" create="never"/>
670                <activationSpec jndiName="MyValueServiceAS" create="never"/>
671            </binding.jms>
672        </service>
673    </composite>
```

Snippet 7-3: Binding Example Using Existing Resources

7.4 Resource Creation Example

Snippet 7-4 shows the JMS binding providing information to create JMS resources rather than using existing ones:

```
678    <?xml version="1.0" encoding="UTF-8"?>
679    <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
680            name="MyValueComposite">
681
682        <service name="MyValueService">
683            <interface.java interface="services.myvalue.MyValueService"/>
684            <binding.jms>
685                <destination jndiName="MyValueServiceQueue" create="always">
686                    <property name="prop1">XYZ</property>
687                    <property name="destName">MyValueDest</property>
688                </destination>
689                <activationSpec jndiName="MyValueServiceAS" create="always"/>
690                <resourceAdapter jndiName="com.example.JMSRA"/>
691            </binding.jms>
692        </service>
693
694        <reference name="StockQuoteService">
695            <interface.java interface="services.stockquote.StockQuoteService"/>
696            <binding.jms>
697                <destination jndiName="StockQuoteServiceQueue"/>
698                <connectionFactory jndiName="StockQuoteServiceQCF"/>
699                <resourceAdapter name="com.example.JMSRA"/>
700            </binding.jms>
701        </reference>
702    </composite>
```

Snippet 7-4: Binding Example that Creates a Resource

7.5 Request/Response Example

Snippet 7-5 shows the JMS binding using existing resources to support request/response operations. The service uses the `JMSReplyTo` destination to send response messages, and does not specify a response queue:

```
708    <?xml version="1.0" encoding="UTF-8"?>
709    <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
710            name="MyValueComposite">
711
712        <service name="MyValueService">
713            <interface.java interface="services.myvalue.MyValueService"/>
714            <binding.jms correlationScheme="sca:messageID">
715                <destination jndiName="MyValueServiceQ" create="never"/>
716                <activationSpec jndiName="MyValueServiceAS" create="never"/>
717            </binding.jms>
718        </service>
719
720        <reference name="StockQuoteService">
```

```
721          <interface.java interface="services.stockquote.StockQuoteService"/>
722          <binding.jms correlationScheme="sca:messageID">
723              <destination jndiName="StockQuoteServiceQueue"/>
724              <connectionFactory jndiName="StockQuoteServiceQCF"/>
725              <response>
726                  <destination jndiName="MyValueResponseQueue"/>
727                  <activationSpec jndiName="MyValueResponseAS"/>
728              </response>
729          </binding.jms>
730      </reference>
731  </composite>
```

Snippet 7-5: Binding Example with a Response

7.6 Subscription with Selector Example

Snippet 7-6 shows how the JMS binding is used in order to consume messages from existing JMS infrastructure. The JMS binding subscribes using selector:

```
736  <?xml version="1.0" encoding="UTF-8"?>
737  <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
738          name="MyValueComposite">
739      <service name="MyValueService">
740          <interface.java interface="services.myvalue.MyValueService"/>
741          <binding.jms>
742              <destination jndiName="MyValueServiceTopic" create="never"/>
743              <connectionFactory jndiName="StockQuoteServiceTCF"
744                  create="never"/>
745              <messageSelection selector="Price&gt;1000"/>
746          </binding.jms>
747      </service>
748  </composite>
```

Snippet 7-6: Binding Example with a Selector

7.7 Policy Set Example

A policy set defines the manner in which intents map to JMS binding properties. Snippet 7-7 illustrates an example of a policy set that defines values for the @priority attribute using the "priority" intent, and also allows setting of a value for a user JMS property using the "log" intent.

```
754  <policySet name="JMSPolicy"
755              provides="priority log"
756              appliesTo="binding.jms">
757
758      <intentMap provides="priority" default="medium">
759          <qualifier name="high">
760              <headers priority="9"/>
761          </qualifier>
762          <qualifier name="medium">
763              <headers priority="4"/>
764          </qualifier>
765          <qualifier name="low">
766              <headers priority="0"/>
767          </qualifier>
768      </intentMap>
769
770      <intentMap provides="log">
771          <qualifier>
772              <headers>
773                  <property name="user_example_log">logged</property>
774              </headers>
775          </qualifier>
776      </intentMap>
777  </policySet>
```

778   *Snippet 7-7: Example Policy Set*

779   Given the policy set in Snippet 7-7, the intents can be required on a service or reference as shown in
780   Snippet 7-8:

```
781        <reference name="StockQuoteService" requires="priority.high log">
782            <interface.java interface="services.stockquote.StockQuoteService"/>
783            <binding.jms>
784                <destination name="StockQuoteServiceQueue"/>
785                <connectionFactory name="StockQuoteServiceQCF"/>
786            </binding.jms>
787        </reference>
```

788   *Snippet 7-8: Binding Example with Intents*

8 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 JMS Binding XML Document

b) SCA Runtime

8.1 SCA JMS Binding XML Document

An SCA JMS Binding XML document is an SCA Composite Document or an SCA ComponentType Document, as defined by the SCA Assembly Specification [SCA-Assembly] Section 13.1 that uses the `binding.jms` element.

An SCA JMS Binding XML document MUST be a conformant SCA Composite Document or an SCA ComponentType Document, as defined by the SCA Assembly Specification [SCA-Assembly], and MUST comply with all statements in Appendix B: "Conformance Items" related to elements and attributes in an SCA JMS Binding XML document, notably all "MUST" statements have to be implemented.

8.2 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, notably all "MUST" statements have to be implemented

2. The implementation MUST conform to the SCA Assembly Model Specification Version 1.1 [SCA-Assembly], and to the SCA Policy Framework Version 1.1 [SCA-Policy]

3. The implementation MUST reject an SCA JMS Binding XML Document that is not conformant per Section 8.1

A. JMS XML Binding Schema: sca-binding-jms-1.1.xsd

```
813  <?xml version="1.0" encoding="UTF-8"?>
814  <!-- Copyright(C) OASIS(R) 2005,2010. All Rights Reserved.
815      OASIS trademark, IPR and other policies apply.  -->
816  <schema xmlns="http://www.w3.org/2001/XMLSchema"
817          targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
818          xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912"
819          elementFormDefault="qualified">
820
821     <include schemaLocation="sca-core-1.1-cd05.xsd"/>
822
823     <complexType name="JMSBinding">
824        <complexContent>
825           <extension base="sca:Binding">
826              <sequence>
827                 <element name="destination" type="sca:JMSDestination"
828                         minOccurs="0"/>
829                 <choice minOccurs="0" maxOccurs="1">
830                    <element name="connectionFactory"
831                            type="sca:JMSConnectionFactory"/>
832                    <element name="activationSpec" type="sca:JMSActivationSpec"/>
833                 </choice>
834                 <element name="response" type="sca:JMSResponse" minOccurs="0"/>
835                 <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
836                 <element name="messageSelection" type="sca:JMSMessageSelection"
837                         minOccurs="0"/>
838                 <element name="resourceAdapter" type="sca:JMSResourceAdapter"
839                         minOccurs="0"/>
840                 <element name="operationProperties"
841                         type="sca:JMSOperationProperties"
842                         minOccurs="0" maxOccurs="unbounded"/>
843                 <element ref="sca:extensions" minOccurs="0" maxOccurs="1"/>
844              </sequence>
845              <attribute name="correlationScheme" type="QName"
846                         default="sca:messageID"/>
847              <attribute name="initialContextFactory" type="anyURI"/>
848              <attribute name="jndiURL" type="anyURI"/>
849           </extension>
850        </complexContent>
851     </complexType>
852
853     <simpleType name="JMSCreateResource">
854        <restriction base="string">
855           <enumeration value="always"/>
856           <enumeration value="never"/>
857           <enumeration value="ifNotExist"/>
858        </restriction>
859     </simpleType>
860
861     <complexType name="JMSDestination">
862        <sequence>
863           <element name="property" type="sca:BindingProperty"
864                   minOccurs="0" maxOccurs="unbounded"/>
865        </sequence>
866        <attribute name="jndiName" type="anyURI"/>
867        <attribute name="type" use="optional" default="queue">
868           <simpleType>
869              <restriction base="string">
870                 <enumeration value="queue"/>
871                 <enumeration value="topic"/>
```

```
872                    </restriction>
873                 </simpleType>
874              </attribute>
875              <attribute name="create" type="sca:JMSCreateResource"
876                         use="optional" default="ifNotExist"/>
877          </complexType>
878
879          <complexType name="JMSConnectionFactory">
880              <sequence>
881                 <element name="property" type="sca:BindingProperty"
882                          minOccurs="0" maxOccurs="unbounded"/>
883              </sequence>
884              <attribute name="jndiName" type="anyURI"/>
885              <attribute name="create" type="sca:JMSCreateResource"
886                         use="optional" default="ifNotExist"/>
887          </complexType>
888
889          <complexType name="JMSActivationSpec">
890              <sequence>
891                 <element name="property" type="sca:BindingProperty"
892                          minOccurs="0" maxOccurs="unbounded"/>
893              </sequence>
894              <attribute name="jndiName" type="anyURI"/>
895              <attribute name="create" type="sca:JMSCreateResource"
896                         use="optional" default="ifNotExist"/>
897          </complexType>
898
899          <complexType name="JMSResponse">
900              <sequence>
901                 <element ref="sca:wireFormat" minOccurs="0" maxOccurs="1"/>
902                 <element name="destination" type="sca:JMSDestination" minOccurs="0"/>
903                 <choice minOccurs="0">
904                    <element name="connectionFactory" type="sca:JMSConnectionFactory"/>
905                    <element name="activationSpec" type="sca:JMSActivationSpec"/>
906                 </choice>
907              </sequence>
908          </complexType>
909
910          <complexType name="JMSHeaders">
911              <sequence>
912                 <element name="property" type="sca:JMSUserProperty"
913                          minOccurs="0" maxOccurs="unbounded"/>
914              </sequence>
915              <attribute name="type" type="string"/>
916              <attribute name="deliveryMode" default="persistent">
917                 <simpleType>
918                    <restriction base="string">
919                       <enumeration value="persistent"/>
920                       <enumeration value="nonpersistent"/>
921                    </restriction>
922                 </simpleType>
923              </attribute>
924              <attribute name="timeToLive" type="long" default="0"/>
925              <attribute name="priority" default="4">
926                 <simpleType>
927                    <restriction base="string">
928                       <enumeration value="0"/>
929                       <enumeration value="1"/>
930                       <enumeration value="2"/>
931                       <enumeration value="3"/>
932                       <enumeration value="4"/>
933                       <enumeration value="5"/>
934                       <enumeration value="6"/>
935                       <enumeration value="7"/>
```

```
936              <enumeration value="8"/>
937              <enumeration value="9"/>
938          </restriction>
939        </simpleType>
940      </attribute>
941    </complexType>
942
943    <complexType name="JMSMessageSelection">
944      <sequence>
945        <element name="property" type="sca:BindingProperty"
946                  minOccurs="0" maxOccurs="unbounded"/>
947      </sequence>
948      <attribute name="selector" type="string"/>
949    </complexType>
950
951    <complexType name="JMSResourceAdapter">
952      <sequence>
953        <element name="property" type="sca:BindingProperty"
954                  minOccurs="0" maxOccurs="unbounded"/>
955      </sequence>
956      <attribute name="name" type="string" use="required"/>
957    </complexType>
958
959    <complexType name="JMSOperationProperties">
960      <sequence>
961        <element name="property" type="sca:BindingProperty"
962                  minOccurs="0" maxOccurs="unbounded"/>
963        <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
964      </sequence>
965      <attribute name="name" type="string" use="required"/>
966      <attribute name="selectedOperation" type="string"/>
967    </complexType>
968
969    <complexType name="BindingProperty">
970      <simpleContent>
971        <extension base="string">
972          <attribute name="name" type="NMTOKEN" use="required"/>
973          <attribute name="type" type="string" use="optional"
974                      default="xs:string"/>
975        </extension>
976      </simpleContent>
977    </complexType>
978
979    <simpleType name="JMSUserPropertyType">
980      <restriction base="string">
981        <enumeration value="boolean"/>
982        <enumeration value="byte"/>
983        <enumeration value="short"/>
984        <enumeration value="int"/>
985        <enumeration value="long"/>
986        <enumeration value="float"/>
987        <enumeration value="double"/>
988        <enumeration value="String"/>
989        <enumeration value="xs:string"/>
990      </restriction>
991    </simpleType>
992
993    <complexType name="JMSUserProperty">
994      <simpleContent>
995        <extension base="string">
996          <attribute name="name" type="NMTOKEN" use="required"/>
997          <attribute name="type" type="sca:JMSUserPropertyType"
998                      use="optional" default="String"/>
999        </extension>
```

```
1000          </simpleContent>
1001      </complexType>
1002
1003      <complexType name="JMSDefaultWireFormatType">
1004          <complexContent>
1005              <extension base="sca:WireFormatType"/>
1006          </complexContent>
1007      </complexType>
1008
1009      <complexType name="JMSDefaultOperationSelectorType">
1010          <complexContent>
1011              <extension base="sca:OperationSelectorType"/>
1012          </complexContent>
1013      </complexType>
1014
1015      <element name="binding.jms" type="sca:JMSBinding"
1016              substitutionGroup="sca:binding"/>
1017
1018      <element name="wireFormat.jmsDefault"
1019              type="sca:JMSDefaultWireFormatType"
1020              substitutionGroup="sca:wireFormat"/>
1021
1022      <element name="operationSelector.jmsDefault"
1023              type="sca:JMSDefaultOperationSelectorType"
1024              substitutionGroup="sca:operationSelector"/>
1025  </schema>
```

B. Conformance Items

1027   This section contains a list of conformance items for the SCA JMS Binding specification.

Conformance ID	Description
[BJM30001]	The value of the `@uri` attribute MUST have the format defined by the IETF URI Scheme for Java™ Message Service 1.0 [IETFJMS]
[BJM30002]	When the `@uri` attribute is specified, the SCA runtime MUST raise an error if the referenced resources do not already exist
[BJM30003]	If the value of the `@correlationScheme` attribute is "`sca:messageID`" the SCA runtime MUST set the correlation ID of replies to the message ID of the corresponding request
[BJM30004]	If the value of the `@correlationScheme` attribute is "`sca:correlationID`" the SCA runtime MUST set the correlation ID of replies to the correlation ID of the corresponding request
[BJM30005]	If the value of the `@correlationScheme` attribute is "`sca:none`" the SCA runtime MUST NOT set the correlation ID in responses that it sends
[BJM30007]	If the value of the `@correlationScheme` attribute is "`sca:correlationID`" the SCA runtime MUST set a non-null correlation ID value in requests that it sends
[BJM30010]	Whatever the value of the `destination/@type` attribute, the SCA runtime MUST ensure a single response is delivered for request/response operations
[BJM30011]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`always`" and the `@jndiName` attribute is present and the resource cannot be created at the location specified by the `@jndiName` attribute then the SCA runtime MUST raise an error
[BJM30012]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" then the `@jndiName` attribute MUST specify the location of the possibly existing resource
[BJM30013]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" and the resource does not exist at the location identified by the `@jndiName` attribute and cannot be created there then the SCA runtime MUST raise an error
[BJM30014]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" and the `@jndiName` attribute refers to an existing resource that is not a JMS Destination of the approprate type, a JMS connection factory or a JMS activation spec respectively then the SCA runtime MUST raise an error
[BJM30015]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`never`" and the `@jndiName` attribute is not specified, or the resource is not present at the location identified by the `@jndiName` attribute, or the location refers to a resource of an incorrect type then the SCA runtime MUST raise an error

[BJM30017]	A `binding.jms` element MUST NOT include both a `connectionFactory` element and an `activationSpec` element
[BJM30018]	When the `connectionFactory` element is present as a child of the `binding.jms` element, then the destination MUST be defined either by the `destination` element child of the `binding.jms` element or the `@uri` attribute of the `binding.jms` element
[BJM30019]	If the `activationSpec` element is present as a child of the `binding.jms` element and the destination is also specified via a `destination` element child of the `binding.jms` element or the `@uri` attribute of the `binding.jms` element then it MUST refer to the same JMS destination as the `activationSpec`
[BJM30020]	The `activationSpec` element MUST NOT be present when the binding is being used for an SCA reference
[BJM30021]	A `response` element MUST NOT include both a `connectionFactory` element and an `activationSpec` element
[BJM30022]	If a `response/destination` and `response/activationSpec` element are both specified they MUST refer to the same JMS destination
[BJM30023]	The `response/activationSpec` element MUST NOT be present when the binding is being used for an SCA service
[BJM30024]	When sending messages for a JMS binding, the SCA runtime MUST set each of the JMSType, JMSDeliveryMode, JMSTimeToLive and JMSPriority headers to values specified in the binding definition in the following priority order: 1) the value for the header specified in the `@uri` attribute (highest priority); 2) the value for the header specified in the `operationProperties/headers` element matching the operation being invoked; 3) the value for the header specified in the `headers` element; 4) the default value for the header as specified by the definition of the `binding.jms/headers` element (lowest priority)
[BJM30025]	When sending messages for a JMS binding, the SCA runtime MUST set each named user property with type and value specified in the binding definition in the following priority order: 1) the type and value for the named user property specified in an `operationProperties/headers/property` element matching the name of the operation being invoked (highest priority); 2) the type and value for the named user property specified in a `headers/property` element (lowest priority)
[BJM30026]	When receiving messages for a JMS binding, the SCA runtime MUST use a message selector if specified in the binding definition in the following priority order: 1) the value for the message selector specified in the `@uri` attribute value's "selector" parameter (highest priority); 2) the value for the message selector specified in the `messageSelection/@selector` attribute; 3) otherwise no message selector is used (lowest priority)

[BJM30029]	The value of the ***operationProperties/@selectedOperation*** attribute MUST be unique across the containing `binding.jms` element
[BJM30031]	The `resourceAdapter` element MUST be present when JMS resources are to be created for a JMS provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise
[BJM30034]	When the `@uri` attribute is specified, the `destination` element MUST NOT be present
[BJM30036]	The `binding.jms` element MUST conform to the XML schema defined in sca-binding-jms-1.1.xsd
[BJM30037]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "always" and the `@jndiName` attribute is not present and the resource cannot be created, then the SCA runtime MUST raise an error
[BJM40002]	If no `operationSelector` element is specified then SCA runtimes MUST use `operationSelector.jmsDefault` as the default
[BJM40003]	When using the default wire format to send request messages, if there is a single parameter and the interface includes more than one operation, the SCA runtime MUST set the JMS user property "`scaOperationName`" to the name of the operation being invoked
[BJM40004]	If no `wireFormat` element is specified in a JMS binding then SCA runtimes MUST use `wireFormat.jmsDefault` as the default
[BJM40005]	When using the default wire format an SCA runtime MUST be able to receive both JMS text and bytes messages
[BJM40006]	When using the default wire format an SCA runtime MUST send either a JMS text or a JMS bytes message
[BJM40008]	When a `binding.jms` element specifies the `operationSelector.jmsDefault` element, the SCA runtime MUST use the default operation selection algorithm to determine the selected operation
[BJM40009]	When a `binding.jms` element specifies the `wireFormat.jmsDefault` element, the SCA runtime MUST use the default wire format
[BJM40010]	When a message is received at an SCA service with JMS binding and the resolved operation name is in the target component's interface, the SCA runtime MUST invoke the target component using the resolved operation name
[BJM40011]	When a message is received at an SCA service with JMS binding and the resolved operation name is not in the target component's interface the SCA runtime MUST raise an error
[BJM50001]	JMS binding implementations MUST support the JMS intent
[BJM50002]	The JMS intent MUST always be included in the `@alwaysProvides` attribute of the JMS `bindingType`
[BJM60002]	For an SCA service with a JMS binding and unidirectional interface, when a request message is received as part of a one-way MEP, the SCA runtime MUST ignore the `JMSReplyTo` destination header in the JMS message, and

	not raise an error
[BJM60003]	For an SCA reference with a JMS binding that has a destination specified via the response element, the SCA runtime MUST receive response messages as defined by the binding's `@correlationScheme` attribute
[BJM60004]	For an SCA reference with a JMS binding, when a request message is sent as part of a request/response MEP, and the JMS binding has a `response` element with a `destination` defined, then the SCA runtime MUST use that destination for the `JMSReplyTo` header in the JMS message it creates for the request
[BJM60005]	For an SCA reference with a JMS binding, when a request message is sent as part of a request/response MEP, and the JMS binding does not have a `response` element with a `destination` defined, the SCA runtime MUST provide an appropriate destination on which to receive response messages and use that destination for the `JMSReplyTo` header in the JMS message it creates for the request
[BJM60006]	For an SCA reference with a JMS binding that does not have a destination specified via the response element, the SCA runtime MUST either receive response messages as defined by the binding's `@correlationScheme` attribute, or use a unique destination for each request/response interaction
[BJM60007]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a non-null `JMSReplyTo` destination, the SCA runtime MUST send the response message to that destination
[BJM60008]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a null `JMSReplyTo` destination and the JMS binding includes a `response/destination` element the SCA runtime MUST send the response message to that destination
[BJM60009]	For an SCA service with a JMS binding, when a request message is received as part of a request/response MEP where the request message includes a null `JMSReplyTo` destination and the JMS binding does not include a response/destination then the SCA runtime MUST NOT process the request and MUST raise an error
[BJM60010]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP the SCA runtime MUST set the correlation identifier in the JMS message that it creates for the response as defined by the JMS binding's `@correlationScheme` attribute
[BJM60011]	For an SCA reference with a JMS binding and a bidirectional interface, when a request message is sent as part of a request/response MEP the SCA runtime MUST set the `scaCallbackDestination` user property in the message it creates to a JMS URI string, in the format defined by the IETF URI Scheme for Java™ Message Service 1.0 **[IETFJMS]**, that identifies the destination to which callback messages are to be sent
[BJM60012]	For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as part of a one-way MEP the SCA runtime MUST set the destination to which callback messages are to be sent as the `JMSReplyTo` destination in the message it creates

[BJM60013]	For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as part of a request/response MEP, the SCA runtime MUST set the `JMSReplyTo` header in the message it creates as described in section 6.2
[BJM60014]	For an SCA reference with a JMS binding and bidirectional interface, the SCA runtime MUST identify the callback destination from the reference's callback service binding if present, or supply a suitable callback destination if not present
[BJM60015]	For an SCA service with a JMS binding, when a callback request message is sent for either a one-way or request/response MEP, the SCA runtime MUST send the callback request message to the callback destination.
[BJM60016]	For an SCA service with a JMS binding, when a callback request message is sent and no callback destination can be identified then the SCA runtime MUST raise an error and throw an exception to the caller of the callback operation
[BJM60017]	For an SCA service with a JMS binding, when a callback request message is sent the SCA runtime MUST set the `JMSReplyTo` destination in the callback request message as defined in sections 6.1 or 6.2 as appropriate for the type of the callback operation invoked
[BJM60018]	SCA runtimes MUST follow the behavior described in section 6.4 and its subsections when `binding.jms` is used in both the forward and callback directions

1028

C. Acknowledgements

1029

1030 The following individuals have participated in the creation of this specification and are gratefully
1031 acknowledged:

1032 **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	TIBCO 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.

D. Revision History

Revision	Date	Editor	Changes Made
1	2007-09-25	Anish Karmarkar	Applied the OASIS template + related changes to the Submission
2	2008-03-12	Simon Holdsworth	Updated text for RFC2119 conformance Updates to resolve following issues: BINDINGS-1 BINDINGS-5 BINDINGS-6 BINDINGS-12 BINDINGS-14 BINDINGS-18 BINDINGS-26 Applied updates discussed at Bindings TC meeting of 27th March
3	2008-06-19	Simon Holdsworth	* Applied most of the editorial changes from Eric Johnson's review
cd01	2008-08-01	Simon Holdsworth	Updates to resolve following issues: BINDINGS-13 (JMS part) BINDINGS-20 (complete) BINDINGS-30 (JMS part) BINDINGS-32 (JMS part) BINDINGS-33 (complete) BINDINGS-34 (complete) BINDINGS-35 (complete) BINDINGS-38 (JMS part)
cd01-rev1	2008-10-16	Simon Holdsworth	Updated text for RFC2119 conformance throughout Updates to resolve following issues: BINDINGS-41 BINDINGS-46 BINDINGS-47
cd01-rev2	2008-12-01	Simon Holdsworth	Added comments identifying those updates that relate to RFC2119 language (issue 52)
cd01-rev3	2008-12-02	Simon Holdsworth	Final RFC2119 language updates BINDINGS-52
cd01-rev4	2009-01-09	Simon Holdsworth	Updates to resolve following issues:

			BINDINGS-7
			BINDINGS-31
			BINDINGS-40
			BINDINGS-42
			BINDINGS-44
			BINDINGS-50
cd02	2009-02-16	Simon Holdsworth	Rename and editorial updates
cd02-rev1	2009-05-22	Simon Holdsworth	Updates to resolve issue BINDINGS-62 (conformance statement numbering)
			Updated assembly namespace to 200903
			Fixed errors in schema
cd02-rev2	2009-05-22	Simon Holdsworth	Updates to resolve following issues:
			BINDINGS-39
			BINDINGS-59
			BINDINGS-65
			BINDINGS-66
			BINDINGS-67
			BINDINGS-68
			BINDINGS-70
			BINDINGS-71
cd02-rev3	2009-06-18	Simon Holdsworth	Editorial concerns addressed
			Added acknowledgements appendix
cd02-rev4	2009-06-19	Simon Holdsworth	Updates to resolve following issues
			BINDINGS-74
			Some editorial updates
			Fixed normative statement missed in application of BINDINGS-67
cd02-rev5	2009-06-24	Simon Holdsworth	Updates to resolve following issues
			BINDINGS-77
			Renamed document to old form
			Removed editorial commentary
			Editorial fixes around external references; changed all links to hyperlinks
cd02-rev6	2009-06-24	Simon Holdsworth	Fixed application of BINDINGS-74
			Fixed broken cross reference
			Changed ASCII to UTF-8 in examples
cd03	2009-06-29	Simon Holdsworth	Updates to resolve following issues
			BINDINGS-80
			BINDINGS-81
cd03-rev1	2010-01-24	Simon Holdsworth	Editorial fix to XML schema name

			Updated to resolve following issues
			BINDINGS-48
			BINDINGS-83
			BINDINGS-85
			BINDINGS-90
			BINDINGS-93
			BINDINGS-94
			BINDINGS-96
			BINDINGS-97
			BINDINGS-98
			BINDINGS-103
			BINDINGS-108
			BINDINGS-109
			BINDINGS-110
cd03-rev2	2010-02-12	Simon Holdsworth	Editorial fixes to cross-references
			Fix cd03-rev1 change to add BINDINGS-110
			Updated to resolve following issues
			BINDINGS-95
			BINDINGS-104
			BINDINGS-105
			BINDINGS-106
cd03-rev3	2010-02-17	Bryan Aupperle	Add captions to all diagrams
cd03-rev4	2010-02-22	Simon Holdsworth	Updated assembly namespace to 200912
			Editorial updates from action items and issues
			BINDINGS-101
			BINDINGS-102
			20091015-3: no change to copyright (currently consistent with all other SCA specs)
			20091015-8: removed non-normative references section
			20091015-9: cleaned up naming conventions section
			20091015-10: cleaned up some phrases that used "may" or "allows"
			20091015-12: no changes made (currently consistent with all other SCA specs)
cd03-rev5	2010-03-18	Simon Holdsworth	Fixed application of issue BINDINGS-108
			Editorial cleanup
			Changed assembly reference to CD05
cd03-rev6	2010-04-16	Simon Holdsworth	Applied resolution to BINDINGS-128
cd04	2010-04-30	Simon Holdsworth	Rename and fix acknowledgements, fixup for publication

cd04-rev1	2010-10-05	Simon Holdsworth	Applied resolutions for issues:
			BINDINGS-134
			BINDINGS-135
			BINDINGS-136
			BINDINGS-138
			BINDINGS-139
			Updated SCA policy spec reference and IETF JMS URI draft reference
cd04-rev2	2010-10-26	Simon Holdsworth	Applied resolutions for issues:
			BINDINGS-141
cd04-rev3	2010-10-29	Simon Holdsworth	Applied resolutions for issues:
			BINDINGS-140
csprd03-rev1	2011-05-22	Simon Holdsworth	Applied resolutions for issues:
			BINDINGS-144
			BINDINGS-149
			BINDINGS-154
			BINDINGS-156
			BINDINGS-157
			BINDINGS-158
			BINDINGS-159
csdprd03-rev2	2011-07-04	Simon Holdsworth	Applied resolutions for issues:
			BINDINGS-169

1035