



Service Component Architecture JMS Binding Specification Version 1.1

Committee Draft 02 revision 5

24th June, 2009

Specification URIs:

This Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd02-rev5.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd02-rev5.doc>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd02-rev5.pdf>
(Authoritative)

Previous Version:

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

Latest Version:

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

Latest Approved Version:

Technical Committee:

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

Chair(s):

Simon Holdsworth, IBM

Editor(s):

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

Related work:

This specification replaces or supersedes:

- Service Component Architecture JMS 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:

This document defines the concept and behavior of a messaging binding, and a concrete JMS-based binding that provides that behavior.

The binding specified in this document applies to an SCA composite's services and references. 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.

The messaging binding describes a common pattern of behavior that may be followed by messaging-related bindings, including the JMS binding. In particular it describes the manner in which operations are selected based on message content, and the manner in which messages are mapped into the runtime representation. These are specified in a language-neutral manner.

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.

Status:

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

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

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

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

Notices

Copyright © OASIS® 2006, 2009. All Rights Reserved.

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

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

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

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

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

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

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

The names "OASIS", [insert specific trademarked names and abbreviations here] 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	5
1.1	Terminology	5
1.2	Normative References	5
1.3	Non-Normative References	6
1.4	Naming Conventions	6
2	Messaging Bindings	7
3	JMS Binding Schema	8
3.1	Extensibility	13
4	Operation Selectors and Wire Formats	14
4.1	Default Operation Selection	14
4.2	Default Wire Format	14
4.2.1	Example of default wire format.....	15
5	Policy	17
6	Message Exchange Patterns.....	18
6.1	One-way message exchange (no Callbacks)	18
6.2	Request/response message exchange (no Callbacks)	18
6.3	JMS User Properties.....	19
6.4	Callbacks.....	19
6.4.1	Invocation of operations on a bidirectional interface	19
6.4.2	Invocation of operations on a callback interface	19
6.4.3	Use of JMSReplyTo for callbacks for non-SCA JMS applications	20
7	Examples	21
7.1	Minimal Binding Example.....	21
7.2	URI Binding Example.....	21
7.3	Binding with Existing Resources Example.....	21
7.4	Resource Creation Example	22
7.5	Request/Response Example.....	22
7.6	Use of Predefined Definitions Example	23
7.7	Subscription with Selector Example	23
7.8	Policy Set Example.....	23
8	Conformance	25
8.1	SCA JMS Binding XML Document	25
8.2	SCA Runtime.....	25
A.	JMS XML Binding Schema: sca-binding-jms.xsd	26
B.	Conformance Items	29
C.	Acknowledgements	34
D.	Revision History	35

1 Introduction

This document defines the concept and behavior of a messaging binding, and a concrete [Java Message Service \[JMS\]](#) based binding that provides that behavior. The binding specified in this document applies to an SCA composite's services and references. 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.

The messaging binding describes a common pattern of behavior that may be followed by messaging-related bindings, including the JMS binding. In particular it describes the manner in which operations are selected based on message content, and the manner in which messages are mapped into the runtime representation. These are specified in a language-neutral manner.

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.

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.

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
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.
- [JMS] Java™ Message Service Specification <http://java.sun.com/products/jms/>
- [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/>

32	[IETFJMS]	IETF URI Scheme for Java™ Message Service 1.0
33		http://tools.ietf.org/id/draft-merrick-jms-uri-05.txt ¹
34	[SCA-Assembly]	http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.html
35	[SCA-Policy]	http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec.pdf

36 **1.3 Non-Normative References**

37 **TBD** TBD

38 **1.4 Naming Conventions**

39 This specification follows some naming conventions for artifacts defined by the specification. In addition
40 to the conventions defined by section 1.3 of the [SCA Assembly Specification \[SCA-Assembly\]](#), this
41 specification adds three additional conventions:

- 42 • Where the names of elements and attributes consist partially or wholly of acronyms, the letters of the
43 acronyms use the same case. When the acronym appears at the start of the name of an element or
44 an attribute, or after a period, it is in lower case. If it appears elsewhere in the name of an element or
45 an attribute, it is in upper case. For example, an attribute might be named "uri" or "jndiURL".
- 46 • Where the names of types consist partially or wholly of acronyms, the letters of the acronyms are in
47 all upper case. For example, an XML Schema type might be named "JCABinding" or "MessageID".
- 48 • Values, including local parts of QName values, follow the rules for names of elements and attributes
49 as stated above, with the exception that the letters of acronyms are in all upper case. For example, a
50 value might be "JMSDefault" or "namespaceURI".

¹ Note that this URI scheme is currently in draft. The reference for this specification will be updated when the IETF standard is finalized

51 **2 Messaging Bindings**

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

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

56 Messaging bindings utilize operation selector and wire format elements to provide the mapping from the
57 native messaging format to an invocation on the target component. A default operation selection and
58 data binding behavior is identified, along with any associated properties.

59 In addition, each operation may have specific properties defined, that may influence the way native
60 messages are processed depending on the operation being invoked.

3 JMS Binding Schema

The JMS binding element is defined by the following schema.

```
<binding.jms correlationScheme="QName"?
  initialContextFactory="xs:anyURI"?
  jndiURL="xs:anyURI"?
  requestConnection="QName"?
  responseConnection="QName"?
  operationProperties="QName"?
  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="NMTOKEN"?>*
  </destination?
  <connectionFactory jndiName="xs:anyURI"
    create="always or never or ifNotExist"?
    <property name="NMTOKEN" type="NMTOKEN"?>*
  </connectionFactory?
  <activationSpec jndiName="xs:anyURI"
    create="always or never or ifNotExist"?
    <property name="NMTOKEN" type="NMTOKEN"?>*
  </activationSpec?

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

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

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

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

  <operationProperties name="string" nativeOperation="string"?
    <property name="NMTOKEN" type="NMTOKEN"?>*
  <headers type="string"?
```



```

120         deliveryMode="persistent or nonpersistent"?
121         timeToLive="long"?
122         priority="0 .. 9"?>
123     <property name="NMTOKEN" type="NMTOKEN"?>*
124 </headers?>
125 </operationProperties>*
126
127 <wireFormat ... />?
128 <operationSelector ... />?
129 </binding.jms>

```

130

131 The binding can be used in one of two ways, either identifying existing [JMS \[JMS\]](#) resources using JNDI
 132 names, or providing the required information to enable the JMS resources to be created.

133 The *binding.jms* element has the following attributes:

- 134 • */binding.jms* – This is the JMS binding element. The element is extensible so that JMS binding
 135 implementers can add additional JMS provider-specific attributes and elements although such
 136 extensions are not guaranteed to be portable across runtimes.
- 137 • */binding.jms/@uri* – as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#). This attribute
 138 identifies the destination, connection factory or activation spec, and other properties to be used to
 139 send/receive the JMS message. There is an implicit *@create="never"* for the resources referred to
 140 in the *@uri* attribute.

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

143 The following illustrates the structure of the URI and the set of property names that have specific
 144 semantics - all other property names are treated as user property names:

```

145 - jms:jndi:<jms-dest>?  

146   jndiURL=<jndi-url> &  

147   jndiInitialContextFactory=<jndi-initial-context-factory> &  

148   jndiConnectionFactoryName=<Connection-Factory-Name> &  

149   deliveryMode=<Delivery-Mode> &  

150   timeToLive=<Time-To-Live> &  

151   priority=<Priority> &  

152   <param-name>=<param-value> & ...

```

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

155 When the *@uri* attribute is specified, the *destination* element MUST NOT be present [BJM30034].

156 An SCA runtime MUST use the values specified in the *@uri* attribute in preference to corresponding
 157 attributes and elements in the binding [BJM30035].

- 158 • */binding.jms/@name* - as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- 159 • */binding.jms/@requires* - as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- 160 • */binding.jms/@policySets* - as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- 161 • */binding.jms/@correlationScheme* – identifies the correlation scheme used when sending reply or
 162 callback messages, default value is "sca:messageID".

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

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

167 If the value of the *@correlationScheme* attribute is "*sca:none*" the SCA runtime MUST NOT set the
 168 correlation ID [BJM30005].

169 SCA runtimes MAY allow other values of the *@correlationScheme* attribute to indicate other
 170 correlation schemes [BJM30006].

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

- 172 • **/binding.jms/@jndiURL** – the URL for the JNDI provider.
- 173 • **/binding.jms/@requestConnection** – identifies a **binding.jms** element that is present in a definition
174 document, whose **destination**, **connectionFactory**, **activationSpec** and **resourceAdapter** children
175 are used to define the values for this binding.
176 If the **@requestConnection** attribute is specified, the **binding.jms** element MUST NOT contain a
177 **destination**, **connectionFactory**, **activationSpec** or **resourceAdapter** element [BJM30007].
- 178 • **/binding.jms/@responseConnection** – identifies a **binding.jms** element that is present in a
179 definition document, whose **response** child element is used to define the values for this binding.
180 If the **@responseConnection** attribute is specified, the **binding.jms** element MUST NOT contain a
181 **response** element [BJM30008].
- 182 • **/binding.jms/@operationProperties** – identifies a **binding.jms** element that is present in a definition
183 document, whose **operationProperties** children are used to define the values for this binding.
184 If the **@operationProperties** attribute is specified, the **binding.jms** element MUST NOT contain an
185 **operationProperties** element [BJM30009].
- 186 • **/binding.jms/destination** – identifies the destination that is to be used to process requests by this
187 binding.
- 188 • **/binding.jms/destination/@type** - the type of the request destination. Valid values are “**queue**” and
189 “**topic**”. The default value is “**queue**”.
190 Whatever the value of the **destination/@type** attribute, the runtime MUST ensure a single response
191 is delivered for request/response operations [BJM30010].
- 192 • **binding.jms/destination/@jndiName** – the JNDI name of the JMS Destination that the binding uses
193 to send or receive messages. The behaviour of this attribute is determined by the value of the
194 **@create** attribute as follows:
 - 195 – If the **@create** attribute value for a destination, connectionFactory or activationSpec element is
196 “**always**” then the **@jndiName** attribute is optional; if the resource cannot be created at the
197 specified location then the SCA runtime MUST raise an error [BJM30011].
198 If the **@jndiName** attribute is omitted this specification places no restriction on the JNDI location
199 of the created resource.
 - 200 – If the **@create** attribute value for a destination, connectionFactory or activationSpec element is
201 “**ifNotExist**” then the **@jndiName** attribute MUST specify the location of the possibly existing
202 resource [BJM30012].
203 If the destination, connectionFactory or activationSpec does not exist at the location identified by
204 the **@jndiName** attribute, but cannot be created there then the SCA runtime MUST raise an error
205 [BJM30013].
206 If the destination, connectionFactory or activationSpec's **@jndiName** attribute refers to an
207 existing resource that is not a JMS Destination of the appropriate type, a JMS connection factory
208 or a JMS activation spec respectively then the SCA runtime MUST raise an error [BJM30014].
 - 209 – If the **@create** attribute value for a destination, connectionFactory or activationSpec element is
210 “**never**” then the **@jndiName** attribute MUST specify the location of the existing resource
211 [BJM30015].
212 If the destination, connection factory or activation spec is not present at the location identified by
213 the **@jndiName** attribute, or the location refers to a resource of an incorrect type then the SCA
214 runtime MUST raise an error [BJM30016].
- 215 • **/binding.jms/destination/@create** – indicates whether the destination should be created when the
216 containing composite is deployed. Valid values are “**always**”, “**never**” and “**ifNotExist**”. The default
217 value is “**ifNotExist**”.
- 218 • **/binding.jms/destination/property** – defines properties to be used to create the destination, if
219 required.
- 220 • **/binding.jms/connectionFactory** – identifies the connection factory that the binding uses to process
221 request messages. The attributes of this element follow the rules defined for the **destination**
222 element.
223 A **binding.jms** element MUST NOT include both a **connectionFactory** element and an

- 224 **activationSpec** element [BJM30017].
- 225 When the **connectionFactory** element is present, then the destination MUST be defined either by
- 226 the **destination** element or the **@uri** attribute [BJM30018].
- 227 • **/binding.jms/activationSpec** – identifies the activation spec that the binding uses to connect to a
 - 228 JMS destination to process request messages. The attributes of this element follow the rules defined
 - 229 for the **destination** element.
 - 230 If the **activationSpec** element is present and the destination is also specified via a **destination**
 - 231 element or the **@uri** attribute then it MUST refer to the same JMS destination as the **activationSpec**
 - 232 [BJM30019].
 - 233 The **activationSpec** element MUST NOT be present when the binding is being used for an SCA
 - 234 reference [BJM30020].
 - 235 • **/binding.jms/response** – defines the resources used for handling response messages (receiving
 - 236 responses for a reference, and sending responses from a service).
 - 237 • **/binding.jms/response/destination** – identifies the destination that is to be used to process
 - 238 responses by this binding. Attributes follow the rules defined for the parent's **destination** element.
 - 239 For a service, this destination is used to send responses to messages that have a null value for the
 - 240 **JMSReplyTo** destination. For a reference, this destination is used to receive reply messages
 - 241 • **/binding.jms/response/connectionFactory** – identifies the connection factory that the binding uses
 - 242 to process response messages. The attributes of this element follow those defined for the
 - 243 **destination** element.
 - 244 A **response** element MUST NOT include both a **connectionFactory** element and an **activationSpec**
 - 245 element [BJM30021].
 - 246 • **/binding.jms/response/activationSpec** – identifies the activation spec that the binding uses to
 - 247 connect to a JMS destination to process response messages. The attributes of this element follow
 - 248 those defined for the **destination** element.
 - 249 If a **response/destination** and **response/activationSpec** element are both specified they MUST
 - 250 refer to the same JMS destination [BJM30022].
 - 251 The **response/activationSpec** element MUST NOT be present when the binding is being used for an
 - 252 SCA service [BJM30023].
 - 253 • **/binding.jms/response/wireFormat** – identifies the wire format used by responses sent or received
 - 254 by this binding. This value overrides the **wireFormat** specified at the binding level. Wire formats for
 - 255 this binding are described in Section 4.
 - 256 • **/binding.jms/headers** – this element specifies values for standard JMS headers.
 - 257 The SCA runtime MUST set JMS headers in messages that it creates to the values specified by the
 - 258 **headers** element unless overridden for the operation being invoked. [BJM30024].
 - 259 These values apply to requests from a reference and responses from a service.
 - 260 • **/binding.jms/headers/@type, @deliveryMode, @timeToLive, @priority** – specifies the value to
 - 261 use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive or JMSPriority
 - 262 respectively.
 - 263 If the **@uri** attribute includes values for the type, delivery mode, time to live or priority properties then
 - 264 the **@uri** values are used and the **headers** and **operationProperties/headers @type,**
 - 265 **@deliveryMode, @timeToLive** or **@priority** attributes are ignored [BJM30025].
 - 266 Valid values for **@deliveryMode** are "**persistent**" and "**nonpersistent**"; valid values for **@priority**
 - 267 are "**0**" to "**9**".
 - 268 • **/binding.jms/headers/property** – specifies the value for the given JMS user property.
 - 269 For each **header/properties** element the SCA runtime MUST set the named JMS user property to
 - 270 the given value in messages it creates unless overridden for the operation being invoked
 - 271 [BJM30026].
 - 272 • **/binding.jms/messageSelection** - this element allows JMS message selection options to be set.
 - 273 These values apply to a service receiving messages from the request destination or for a reference
 - 274 receiving messages from the callback or reply-to destination.

- 275 • **/binding.jms/messageSelection/@selector** - specifies the value to use for the JMS selector. If the
276 **@uri** attribute includes a value for the message selector then the **@uri** value is used and the
277 **messageSelection/@selector** attribute is ignored [BJM30027].
- 278 • **/binding.jms/resourceAdapter** – specifies name, type and properties of the Resource Adapter Java
279 bean.
280 The **resourceAdapter** element MUST be present when JMS resources are to be created for a JMS
281 provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise
282 [BJM30031].
283 SCA runtimes MAY place restrictions on the properties of the resource adapter Java bean that can be
284 set using the **resourceAdapter** element [BJM30028].
285 For JMS providers that do not implement the JCA 1.5 specification [JCA15], information necessary for
286 resource creation can be added in provider-specific elements or attributes allowed by the extensibility
287 of the **binding.jms** element.
- 288 • **/binding.jms/operationProperties** – specifies various properties that are specific to the processing
289 of a particular operation.
- 290 • **/binding.jms/operationProperties/@name** – The name of the operation in the interface.
- 291 • **/binding.jms/operationProperties/@selectedOperation** – The value generated by the
292 **operationSelector** that corresponds to the operation in the service or reference interface identified
293 by the **operationProperties/@name** attribute. If this attribute is omitted then the value defaults to
294 the value of the **operationProperties/@name** attribute.
295 The value of the **operationProperties/@selectedOperation** attribute MUST be unique across the
296 containing **binding.jms** element [BJM30029].
- 297 • **/binding.jms/operationProperties/property** – specifies properties specific to this operation. These
298 properties are intended to be used to parameterize the **wireFormat** identified for the binding for a
299 particular operation.
300 The SCA runtime SHOULD make the **operationProperties** element corresponding to the
301 **selectedOperation** available to the **wireFormat** implementation [BJM30030].
- 302 • **/binding.jms/operationProperties/headers** – this element specifies values for standard JMS
303 headers. These values apply to requests from a reference and responses from a service.
304 The SCA runtime MUST set JMS headers in messages it creates when the operation identified by the
305 **operationProperties/@name** attribute is invoked to the values specified by the corresponding
306 **operationProperties/headers** element [BJM30032].
- 307 • **/binding.jms/operationProperties/headers/@type, @deliveryMode, @timeToLive, @priority** –
308 specifies the value to use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive
309 or JMSPriority, respectively,
- 310 • **/binding.jms/operationProperties/headers/property** – specifies the value for the given JMS user
311 property.
312 For each **operationProperties/headers/property** element the SCA runtime MUST set the named
313 JMS user property to the given value in messages it creates when the operation identified by the
314 **operationProperties/@name** attribute is invoked [BJM30033].
- 315 • **/binding.jms/wireFormat** – identifies the wire format used by requests and responses sent or
316 received by this binding. Wire formats for this binding are described in Section 4.
- 317 • **/binding.jms/operationSelector** – identifies the operation selector used when receiving requests for
318 a service. If specified for a reference this provides the default operation selector for callbacks if not
319 specified via a callback service element. Operation selectors for this binding are described in Section
320 4.

321 The **binding.jms** element MUST conform to the XML schema defined in **sca-binding-jms.xsd**
322 [BJM30036].

323 Deployers/assemblers can configure **nonpersistent** for **@deliveryMode** in order to provide higher
324 performance with a decreased quality of service. A **binding.jms** element configured in this way cannot
325 satisfy either of the "**atLeastOnce**" and "**exactlyOnce**" policy intents. The SCA Runtime MUST raise an
326 error for this invalid combination at deployment time.

327 **3.1 Extensibility**

328 The JMS binding allows further customization of the binding element and its subelements with vendor
329 specific attributes or elements. This is done by providing extension points in the schema; refer to
330 Appendix **Error! Reference source not found.**, "**Error! Reference source not found.**" for the locations
331 of these extension points.

332 4 Operation Selectors and Wire Formats

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

338 The process of identifying the operation to be invoked is *operation selection*; the information that
339 describes the contents of messages is a *wire format*. The **binding** element as described in the [SCA
340 Assembly Specification \[SCA-Assembly\]](#) provides the means to identify specific operation selection via
341 the **operationSelector** element and the wire format of messages received and to be sent using the
342 **wireFormat** element. When the JMS binding receives a message, the **operationSelector** is used to
343 generate a selected operation name from the message content. The selected operation name is then
344 mapped to an operation in the service’s interface via a matching **operationProperties** element in the
345 JMS binding. If there is no matching element, the operation name is assumed to be the same as the
346 selected operation name.

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

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

350 The SCA runtime MUST support the default JMS wire format and operation selector behavior, and MAY
351 provide additional means to override it [BJM40001].

352 4.1 Default Operation Selection

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

- 356 • If there is only one operation on the service’s interface, then that operation is the selected operation
357 name;
- 358 • Otherwise, if the JMS user property “**scaOperationName**” is present, then the value of that user
359 property is used as the selected operation name;
- 360 • Otherwise, if the message is a JMS text or bytes message containing XML, then the selected
361 operation name is the local name of the root element of the XML payload;
- 362 • Otherwise, the selected operation name is “**onMessage**”.

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

365 If no **operationSelector** element is specified then SCA runtimes MUST use
366 **operationSelector.jmsDefault** as the default [BJM40002].

367 4.2 Default Wire Format

368 The default wire format maps between a **JMSMessage** and the object(s) expected by the component
369 implementation. We encourage component implementers to avoid exposure of [JMS \[JMS\]](#) APIs to
370 component implementations, however in the case of an existing implementation that expects a
371 **JMSMessage**, this provides for simple reuse of that as an SCA component.

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

- 374 • If there is a single parameter that is a **JMSMessage**, then the **JMSMessage** is passed as is.
- 375 • Otherwise, if the **JMSMessage** is not a JMS text message or bytes message containing XML it is
376 invalid.

- 377 • Otherwise if there is a single parameter, or for the return value, the JMS text or bytes XML payload is
378 the XML serialization of that parameter according to the WSDL schema for the message.
- 379 • Otherwise the multiple parameters are encoded in XML using the document wrapped style, according
380 to the WSDL schema for the message.

381 When a **binding.jms** element specifies the **wireFormat.jmsDefault** element, the SCA runtime MUST use
382 the default wire format [BJM40009].

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

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

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

390 When using the default wire format an SCA runtime MAY provide additional configuration to allow
391 selection between JMS text or bytes messages to be sent [BJM40007].

392 If no **wireFormat** element is specified in a JMS binding then SCA runtimes MUST use
393 **wireFormat.jmsDefault** as the default [BJM40004].

394 4.2.1 Example of default wire format

395 For the following interface definition:

```

396 <wsdl:definitions name="Coordinates"
397 targetNamespace="http://tempuri.org/coordinates"
398 xmlns:tns="http://tempuri.org/coordinates"
399 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
400 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
401   <wsdl:types>
402     <xsd:schema targetNamespace="http://tempuri.org/coordinates">
403       <xsd:element name="setCoordinates">
404         <xsd:complexType>
405           <xsd:sequence>
406             <xsd:element name="x" type="xsd:int"/>
407             <xsd:element name="y" type="xsd:int"/>
408           </xsd:sequence>
409         </xsd:complexType>
410       </xsd:element>
411     </xsd:schema>
412   </wsdl:types>
413
414   <wsdl:message name="setCoordinatesRequestMsg">
415     <wsdl:part element="tns:setCoordinates" name="setCoordinatesParameters"/>
416   </wsdl:message>
417
418   <wsdl:portType name="Coordinates">
419     <wsdl:operation name="setCoordinates">
420       <wsdl:input message="tns:setCoordinatesRequestMsg"
421 name="setCoordinatesRequest"/>
422     </wsdl:operation>
423   </wsdl:portType>
424 </wsdl:definitions>

```

425 When the **setCoordinates** operation is invoked via a reference with a JMS binding that uses the default
426 wire format, the message sent from the JMS binding is a JMS text or bytes message with the following
427 content:

```

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

```

```
</setCoordinates>
```

432

5 Policy

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

436 JMS binding implementations MAY support the following standard intents, as defined by the JMS
437 binding's ***bindingType***:

```
438 <bindingType type="binding.jms"  
439             alwaysProvides="JMS"  
440             mayProvide="atLeastOnce atMostOnce ordered"/>
```

441 The atLeastOnce, atMostOnce and ordered intent are defined in the [SCA Policy Specification \[SCA-
442 Policy\]](#) document in section 8, "Reliability Policy".

443

6 Message Exchange Patterns

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

6.1 One-way message exchange (no Callbacks)

451 A one-way message exchange is one where a request message is sent that does not require or expect a
452 corresponding response message. These are represented in WSDL as an operation with an **input**
453 element and no **output** elements and no **fault** elements.

454 For an SCA reference with a JMS binding, when a request message is sent as part of a one-way MEP,
455 the SCA runtime SHOULD NOT set the **JMSReplyTo** destination header in the JMS message that it
456 creates, regardless of whether the JMS binding has a **response** element with a **destination** defined
457 [BJM60001].

458 For an SCA service with a JMS binding, when a request message is received as part of a one-way MEP,
459 the SCA runtime MUST ignore the **JMSReplyTo** destination header in the JMS message, and not raise
460 an error [BJM60002].

461 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)

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

466 For an SCA reference with a JMS binding, when a request message is sent as part of a request/response
467 MEP, the SCA runtime MUST set a non-null value for the **JMSReplyTo** header in the JMS message it
468 creates for the request [BJM60003].

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

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

477 For an SCA reference with a JMS binding, the SCA runtime MAY choose to receive response messages
478 on the basis of their correlation ID as defined by the binding's **@correlationScheme** attribute, or use a
479 unique destination for each response [BJM60006].

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

483 For an SCA service with a JMS binding, when a response message is sent as part of a request/response
484 MEP where the request message included a null **JMSReplyTo** destination and the JMS binding includes
485 a **response/destination** element the SCA runtime MUST send the response message to that destination
486 [BJM60008].

487 For an SCA service with a JMS binding, when a response message is sent as part of a request/response
488 MEP where the request message included a null **JMSReplyTo** destination and the JMS binding does not
489 include a **response/destination** then an error SHOULD be raised by the SCA runtime [BJM60009].

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

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

494 6.3 JMS User Properties

495 This protocol assigns specific behavior to JMS user properties:

- 496 • "**scaCallbackDestination**" holds the name of the JMS Destination to which callback messages are
497 sent.

498 6.4 Callbacks

499 Callbacks are SCA's way of representing bidirectional interfaces, where messages are sent in both
500 directions between a client and a service. A callback is the invocation of an operation on a service's
501 callback interface. A callback operation can be one-way or request/response. Messages that correspond
502 to one-way or request/response operations on a bidirectional interface use either the
503 **scaCallbackDestination** user property or the **JMSReplyTo** destination, or both, to identify the
504 destination to which messages are to be sent when operations are invoked on the callback interface. The
505 use of **JMSReplyTo** for this purpose is to enable interaction with non-SCA JMS applications, as
506 described below.

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

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

511 6.4.1 Invocation of operations on a bidirectional interface

512 For an SCA reference with a JMS binding and a bidirectional interface, when a request message is sent
513 the SCA runtime MUST set the destination to which callback messages are to be sent as the value of the
514 **scaCallbackDestination** user property in the message it creates [BJM60011].

515 For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent the
516 SCA runtime MAY set the **JMSReplyTo** destination to the same value as the **scaCallbackDestination**
517 user property [BJM60012].

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

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

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

526 6.4.2 Invocation of operations on a callback interface

527 An SCA service with a callback interface can invoke operations on that callback interface by sending
528 messages to the destination identified by the **scaCallbackDestination** user property in a message that it
529 has received, the **JMSReplyTo** destination of a one-way message that it has received, or the destination
530 identified by the service's callback reference JMS binding.

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

- 533 • The **scaCallbackDestination** identified by an earlier request, if not null;
- 534 • the **JMSReplyTo** destination identified by an earlier one-way request, if not null;
- 535 • the request destination of the service's callback reference JMS binding, if specified

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

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

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

545 6.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications

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

551 7 Examples

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

555 7.1 Minimal Binding Example

556 The following example shows the JMS binding being used with no further attributes or elements. In this
557 case, it is left to the deployer to identify the resources to which the binding is connected.

```
558 <?xml version="1.0" encoding="ASCII"?>
559 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
560           name="MyValueComposite">
561
562     <service name="MyValueService">
563       <interface.java interface="services.myvalue.MyValueService"/>
564       <binding.jms/>
565     </service>
566
567     <reference name="StockQuoteService">
568       <interface.java interface="services.stockquote.StockQuoteService"/>
569       <binding.jms/>
570     </reference>
571 </composite>
```

572 7.2 URI Binding Example

573 The following example shows the JMS binding using the **@uri** attribute to specify the connection type and
574 its information:

```
575 <?xml version="1.0" encoding="ASCII"?>
576 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
577           name="MyValueComposite">
578
579     <service name="MyValueService">
580       <interface.java interface="services.myvalue.MyValueService"/>
581       <binding.jms uri="jms:MyValueServiceQueue?
582                   activationSpecName=MyValueServiceAS&
583                   ... "/>
584     </service>
585
586     <reference name="StockQuoteService">
587       <interface.java interface="services.stockquote.StockQuoteService"/>
588       <binding.jms uri="jms:StockQuoteServiceQueue?
589                   connectionFactoryName=StockQuoteServiceQCF&
590                   deliveryMode=1&
591                   ... "/>
592     </reference>
593 </composite>
```

594 7.3 Binding with Existing Resources Example

595 The following example shows the JMS binding using existing resources:

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

```

600     <service name="MyValueService">
601         <interface.java interface="services.myvalue.MyValueService"/>
602         <binding.jms>
603             <destination jndiName="MyValueServiceQ" create="never"/>
604             <activationSpec jndiName="MyValueServiceAS" create="never"/>
605         </binding.jms>
606     </service>
607 </composite>

```

608 7.4 Resource Creation Example

609 The following example shows the JMS binding providing information to create JMS resources rather than
610 using existing ones:

```

611 <?xml version="1.0" encoding="ASCII"?>
612 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
613     name="MyValueComposite">
614
615     <service name="MyValueService">
616         <interface.java interface="services.myvalue.MyValueService"/>
617         <binding.jms>
618             <destination jndiName="MyValueServiceQueue" create="always">
619                 <property name="prop1" type="string">XYZ</property>
620                 <property name="destName" type="string">MyValueDest</property>
621             </destination>
622             <activationSpec jndiName="MyValueServiceAS" create="always"/>
623             <resourceAdapter jndiName="com.example.JMSRA"/>
624         </binding.jms>
625     </service>
626
627     <reference name="StockQuoteService">
628         <interface.java interface="services.stockquote.StockQuoteService"/>
629         <binding.jms>
630             <destination jndiName="StockQuoteServiceQueue"/>
631             <connectionFactory jndiName="StockQuoteServiceQCF"/>
632             <resourceAdapter name="com.example.JMSRA"/>
633         </binding.jms>
634     </reference>
635 </composite>

```

636 7.5 Request/Response Example

637 The following example shows the JMS binding using existing resources to support request/response
638 operations. The service uses the **JMSReplyTo** destination to send response messages, and does not
639 specify a response queue:

```

640 <?xml version="1.0" encoding="ASCII"?>
641 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
642     name="MyValueComposite">
643
644     <service name="MyValueService">
645         <interface.java interface="services.myvalue.MyValueService"/>
646         <binding.jms correlationScheme="sca:messageId">
647             <destination jndiName="MyValueServiceQ" create="never"/>
648             <activationSpec jndiName="MyValueServiceAS" create="never"/>
649         </binding.jms>
650     </service>
651
652     <reference name="StockQuoteService">
653         <interface.java interface="services.stockquote.StockQuoteService"/>
654         <binding.jms correlationScheme="sca:messageId">
655             <destination jndiName="StockQuoteServiceQueue"/>
656             <connectionFactory jndiName="StockQuoteServiceQCF"/>

```

```

657         <response>
658             <destination jndiName="MyValueResponseQueue"/>
659             <activationSpec jndiName="MyValueResponseAS"/>
660         </response>
661     </binding.jms>
662 </reference>
663 </composite>

```

664 7.6 Use of Predefined Definitions Example

665 This example shows the case where there is common connection information shared by more than one
666 reference.

667 The common connection information is defined in a separate definitions file:

```

668 <?xml version="1.0" encoding="ASCII"?>
669 <definitions targetNamespace="http://acme.com"
670             xmlns="http://docs.oasis-open.org/ns/opencsa/sca/2007903">
671     <binding.jms name="StockQuoteService">
672         <destination jndiName="StockQuoteServiceQueue" create="never"/>
673         <connectionFactory jndiName="StockQuoteServiceQCF" create="never"/>
674     </binding.jms>
675 </definitions>

```

676 Any *binding.jms* element may then refer to that definition:

```

677 <?xml version="1.0" encoding="ASCII"?>
678 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
679           xmlns:acme="http://acme.com"
680           name="MyValueComposite">
681     <reference name="MyValueService">
682         <interface.java interface="services.myvalue.MyValueService"/>
683         <binding.jms requestConnection="acme:StockQuoteService"/>
684     </reference>
685 </composite>

```

686 7.7 Subscription with Selector Example

687 The following example shows how the JMS binding is used in order to consume messages from existing
688 JMS infrastructure. The JMS binding subscribes using selector:

```

689 <?xml version="1.0" encoding="ASCII"?>
690 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
691           name="MyValueComposite">
692     <service name="MyValueService">
693         <interface.java interface="services.myvalue.MyValueService"/>
694         <binding.jms>
695             <destination jndiName="MyValueServiceTopic" create="never"/>
696             <connectionFactory jndiName="StockQuoteServiceTCF"
697 create="never"/>
698             <messageSelection selector="Price>1000"/>
699         </binding.jms>
700     </service>
701 </composite>

```

702 7.8 Policy Set Example

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

```

706 <policySet name="JMSPolicy"
707           provides="priority log"

```

```

708     appliesTo="binding.jms">
709
710     <intentMap provides="priority" default="medium">
711         <qualifier name="high">
712             <headers priority="9"/>
713         </qualifier>
714         <qualifier name="medium">
715             <headers priority="4"/>
716         </qualifier>
717         <qualifier name="low">
718             <headers priority="0"/>
719         </qualifier>
720     </intentMap>
721
722     <intentMap provides="log">
723         <qualifier>
724             <headers>
725                 <property name="user_example_log">logged</property>
726             </headers>
727         </qualifier>
728     </intentMap>
729 </policySet>

```

730 Given this policy set, the intents can be required on a service or reference:

```

731 <reference name="StockQuoteService" requires="priority.high log">
732     <interface.java interface="services.stockquote.StockQuoteService"/>
733     <binding.jms>
734         <destination name="StockQuoteServiceQueue"/>
735         <connectionFactory name="StockQuoteServiceQCF"/>
736     </binding.jms>
737 </reference>

```

738 8 Conformance

739 The XML schema pointed to by the RDDL document at the namespace URI, defined by this specification,
740 are considered to be authoritative and take precedence over the XML schema defined in the appendix of
741 this document. There are two categories of artifacts for which this specification defines conformance:

- 742 a) SCA JMS Binding XML Document
- 743 b) SCA Runtime

744 8.1 SCA JMS Binding XML Document

745 An SCA JMS Binding XML document is an SCA Composite Document, an SCA Definitions Document or
746 an SCA ComponentType Document, as defined by the [SCA Assembly Specification \[SCA-Assembly\]](#)
747 Section 13.1 that uses the *binding.jms* element.

748 An SCA JMS Binding XML document MUST be a conformant SCA Composite Document, SCA
749 Definitions Document or a SCA ComponentType Document, as defined by the [SCA Assembly
750 Specification \[SCA-Assembly\]](#), and MUST comply with all the applicable requirements specified in this
751 specification.

752 8.2 SCA Runtime

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

- 755 1. The implementation MUST comply with all statements in Appendix B: Conformance Items related
756 to an SCA Runtime, notably all "MUST" statements have to be implemented
- 757 2. The implementation MUST conform to the [SCA Assembly Model Specification Version 1.1 \[SCA-
758 Assembly\]](#), and to the [SCA Policy Framework Version 1.1 \[SCA-Policy\]](#)
- 759 3. The implementation MUST reject an SCA JMS Binding XML Document that is not conformant per
760 Section 8.1

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

```

762 <?xml version="1.0" encoding="UTF-8"?>
763 <!-- Copyright (C) OASIS(R) 2005,2009. All Rights Reserved.
764 OASIS trademark, IPR and other policies apply. -->
765 <schema xmlns="http://www.w3.org/2001/XMLSchema"
766 targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200903"
767 xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"
768 elementFormDefault="qualified">
769
770 <include schemaLocation="sca-core-1.1-cd03.xsd"/>
771
772 <complexType name="JMSBinding">
773 <complexContent>
774 <extension base="sca:Binding">
775 <sequence>
776 <element name="destination" type="sca:JMSDestination"
777 minOccurs="0"/>
778 <choice minOccurs="0" maxOccurs="1">
779 <element name="connectionFactory"
780 type="sca:JMSConnectionFactory"/>
781 <element name="activationSpec" type="sca:JMSActivationSpec"/>
782 </choice>
783 <element name="response" type="sca:JMSResponse" minOccurs="0"/>
784 <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
785 <element name="messageSelection" type="sca:JMSMessageSelection"
786 minOccurs="0"/>
787 <element name="resourceAdapter" type="sca:JMSResourceAdapter"
788 minOccurs="0"/>
789 <element name="operationProperties"
790 type="sca:JMSOperationProperties"
791 minOccurs="0" maxOccurs="unbounded"/>
792 <any namespace="##other" processContents="lax"
793 minOccurs="0" maxOccurs="unbounded"/>
794 </sequence>
795 <attribute name="correlationScheme" type="QName"
796 default="sca:messageId"/>
797 <attribute name="initialContextFactory" type="anyURI"/>
798 <attribute name="jndiURL" type="anyURI"/>
799 <attribute name="requestConnection" type="QName"/>
800 <attribute name="responseConnection" type="QName"/>
801 <attribute name="operationProperties" type="QName"/>
802 </extension>
803 </complexContent>
804 </complexType>
805
806 <simpleType name="JMSCreateResource">
807 <restriction base="string">
808 <enumeration value="always"/>
809 <enumeration value="never"/>
810 <enumeration value="ifNotExist"/>
811 </restriction>
812 </simpleType>
813
814 <complexType name="JMSDestination">
815 <sequence>
816 <element name="property" type="sca:BindingProperty"
817 minOccurs="0" maxOccurs="unbounded"/>
818 </sequence>
819 <attribute name="jndiName" type="anyURI" use="required"/>
820 <attribute name="type" use="optional" default="queue">

```

```

821     <simpleType>
822         <restriction base="string">
823             <enumeration value="queue"/>
824             <enumeration value="topic"/>
825         </restriction>
826     </simpleType>
827 </attribute>
828 <attribute name="create" type="sca:JMSCreateResource"
829     use="optional" default="ifNotExist"/>
830 </complexType>
831
832 <complexType name="JMSConnectionFactory">
833     <sequence>
834         <element name="property" type="sca:BindingProperty"
835             minOccurs="0" maxOccurs="unbounded"/>
836     </sequence>
837     <attribute name="jndiName" type="anyURI" use="required"/>
838     <attribute name="create" type="sca:JMSCreateResource"
839         use="optional" default="ifNotExist"/>
840 </complexType>
841
842 <complexType name="JMSActivationSpec">
843     <sequence>
844         <element name="property" type="sca:BindingProperty"
845             minOccurs="0" maxOccurs="unbounded"/>
846     </sequence>
847     <attribute name="jndiName" type="anyURI" use="required"/>
848     <attribute name="create" type="sca:JMSCreateResource"
849         use="optional" default="ifNotExist"/>
850 </complexType>
851
852 <complexType name="JMSResponse">
853     <sequence>
854         <element name="wireFormat" type="sca:WireFormatType" minOccurs="0"/>
855         <element name="destination" type="sca:JMSDestination" minOccurs="0"/>
856         <choice minOccurs="0">
857             <element name="connectionFactory" type="sca:JMSConnectionFactory"/>
858             <element name="activationSpec" type="sca:JMSActivationSpec"/>
859         </choice>
860     </sequence>
861 </complexType>
862
863 <complexType name="JMSHeaders">
864     <sequence>
865         <element name="property" type="sca:BindingProperty"
866             minOccurs="0" maxOccurs="unbounded"/>
867     </sequence>
868     <attribute name="type" type="string"/>
869     <attribute name="deliveryMode">
870         <simpleType>
871             <restriction base="string">
872                 <enumeration value="persistent"/>
873                 <enumeration value="nonpersistent"/>
874             </restriction>
875         </simpleType>
876     </attribute>
877     <attribute name="timeToLive" type="long"/>
878     <attribute name="priority">
879         <simpleType>
880             <restriction base="string">
881                 <enumeration value="0"/>
882                 <enumeration value="1"/>
883                 <enumeration value="2"/>
884                 <enumeration value="3"/>

```

```

885         <enumeration value="4"/>
886         <enumeration value="5"/>
887         <enumeration value="6"/>
888         <enumeration value="7"/>
889         <enumeration value="8"/>
890         <enumeration value="9"/>
891     </restriction>
892 </simpleType>
893 </attribute>
894 </complexType>
895
896 <complexType name="JMSMessageSelection">
897     <sequence>
898         <element name="property" type="sca:BindingProperty"
899             minOccurs="0" maxOccurs="unbounded"/>
900     </sequence>
901     <attribute name="selector" type="string"/>
902 </complexType>
903
904 <complexType name="JMSResourceAdapter">
905     <sequence>
906         <element name="property" type="sca:BindingProperty"
907             minOccurs="0" maxOccurs="unbounded"/>
908     </sequence>
909     <attribute name="name" type="string" use="required"/>
910 </complexType>
911
912 <complexType name="JMSOperationProperties">
913     <sequence>
914         <element name="property" type="sca:BindingProperty"
915             minOccurs="0" maxOccurs="unbounded"/>
916         <element name="headers" type="sca:JMSHeaders"/>
917     </sequence>
918     <attribute name="name" type="string" use="required"/>
919     <attribute name="nativeOperation" type="string"/>
920 </complexType>
921
922 <complexType name="BindingProperty">
923     <simpleContent>
924         <extension base="string">
925             <attribute name="name" type="NMTOKEN"/>
926             <attribute name="type" type="string" use="optional"
927                 default="xs:string"/>
928         </extension>
929     </simpleContent>
930 </complexType>
931
932 <element name="binding.jms" type="sca:JMSBinding"
933     substitutionGroup="sca:binding"/>
934
935 <element name="wireFormat.jmsDefault" type="sca:WireFormatType"
936     substitutionGroup="sca:wireFormat"/>
937
938 <element name="operationSelector.jmsDefault" type="sca:OperationSelectorType"
939     substitutionGroup="sca:operationSelector"/>
940 </schema>

```

941

B. Conformance Items

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

Conformance ID	Description
[BJM30001]	The value of the <i>@uri</i> attribute MUST have the format defined by the IETF URI Scheme for Java™ Message Service 1.0 [IETFJMS]
[BJM30002]	When the <i>@uri</i> attribute is specified, the SCA runtime MUST raise an error if the referenced resources do not already exist
[BJM30003]	If the value of the <i>@correlationScheme</i> attribute is " <i>sca:messageID</i> " the SCA runtime MUST set the correlation ID of replies to the message ID of the corresponding request
[BJM30004]	If the value of the <i>@correlationScheme</i> attribute is " <i>sca:correlationID</i> " the SCA runtime MUST set the correlation ID of replies to the correlation ID of the corresponding request
[BJM30005]	If the value of the <i>@correlationScheme</i> attribute is " <i>sca:none</i> " the SCA runtime MUST NOT set the correlation ID
[BJM30006]	SCA runtimes MAY allow other values of the <i>@correlationScheme</i> attribute to indicate other correlation schemes
[BJM30007]	If the <i>@requestConnection</i> attribute is specified, the <i>binding.jms</i> element MUST NOT contain a <i>destination</i> , <i>connectionFactory</i> , <i>activationSpec</i> or <i>resourceAdapter</i> element
[BJM30008]	If the <i>@responseConnection</i> attribute is specified, the <i>binding.jms</i> element MUST NOT contain a <i>response</i> element
[BJM30009]	If the <i>@operationProperties</i> attribute is specified, the <i>binding.jms</i> element MUST NOT contain an <i>operationProperties</i> element
[BJM30010]	Whatever the value of the <i>destination/@type</i> attribute, the runtime MUST ensure a single response is delivered for request/response operations
[BJM30011]	If the <i>@create</i> attribute value for a destination, connectionFactory or activationSpec element is " <i>always</i> " then the <i>@jndiName</i> attribute is optional; if the resource cannot be created at the specified location then the SCA runtime MUST raise an error
[BJM30012]	If the <i>@create</i> attribute value for a destination, connectionFactory or activationSpec element is " <i>ifNotExist</i> " then the <i>@jndiName</i> attribute MUST specify the location of the possibly existing resource
[BJM30013]	If the destination, connectionFactory or activationSpec does not exist at the location identified by the <i>@jndiName</i> attribute, but cannot be created there then the SCA runtime MUST raise an error
[BJM30014]	If the destination, connectionFactory or activationSpec's <i>@jndiName</i> attribute refers to an existing resource that is not a JMS Destination of the appropriate type, a JMS connection factory or a JMS activation spec respectively then the SCA runtime MUST raise an error
[BJM30015]	If the <i>@create</i> attribute value for a destination, connectionFactory or

	activationSpec element is " never " then the @jndiName attribute MUST specify the location of the existing resource
[BJM30016]	If the destination, connection factory or activation spec 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, then the destination MUST be defined either by the destination element or the @uri attribute
[BJM30019]	If the activationSpec element is present and the destination is also specified via a destination element or the @uri attribute 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]	The SCA runtime MUST set JMS headers in messages that it creates to the values specified by the headers element unless overridden for the operation being invoked.
[BJM30025]	If the @uri attribute includes values for the type, delivery mode, time to live or priority properties then the @uri values are used and the headers and operationProperties/headers @type, @deliveryMode, @timeToLive or @priority attributes are ignored
[BJM30026]	For each header/properties element the SCA runtime MUST set the named JMS user property to the given value in messages it creates unless overridden for the operation being invoked
[BJM30027]	If the @uri attribute includes a value for the message selector then the @uri value is used and the messageSelection/@selector attribute is ignored
[BJM30028]	SCA runtimes MAY place restrictions on the properties of the resource adapter Java bean that can be set using the resourceAdapter element
[BJM30029]	The value of the operationProperties/@selectedOperation attribute MUST be unique across the containing binding.jms element
[BJM30030]	The SCA runtime SHOULD make the operationProperties element corresponding to the selectedOperation available to the wireFormat implementation
[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
[BJM30032]	The SCA runtime MUST set JMS headers in messages it creates when the operation identified by the operationProperties/@name attribute is invoked to

	the values specified by the corresponding operationProperties/headers element
[BJM30033]	For each operationProperties/headers/property element the SCA runtime MUST set the named JMS user property to the given value in messages it creates when the operation identified by the operationProperties/@name attribute is invoked
[BJM30034]	When the @uri attribute is specified, the destination element MUST NOT be present
[BJM30035]	An SCA runtime MUST use the values specified in the @uri attribute in preference to corresponding attributes and elements in the binding
[BJM30036]	The binding.jms element MUST conform to the XML schema defined in sca-binding-jms.xsd
[BJM40001]	The SCA runtime MUST support the default JMS wire format and operation selector behavior, and MAY provide additional means to override it
[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
[BJM40007]	When using the default wire format an SCA runtime MAY provide additional configuration to allow selection between JMS text or bytes messages to be sent
[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
[BJM60001]	For an SCA reference with a JMS binding, when a request message is sent as part of a one-way MEP, the SCA runtime SHOULD NOT set the JMSReplyTo destination header in the JMS message that it creates, regardless of whether the JMS binding has a response element with a destination defined
[BJM60002]	For an SCA service with a JMS binding, 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, when a request message is sent as part of a request/response MEP, the SCA runtime MUST set a non-null value for the JMSReplyTo header in the JMS message it creates for the request

[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, the SCA runtime MAY choose to receive response messages on the basis of their correlation ID as defined by the binding's @correlationScheme attribute, or use a unique destination for each response
[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 response message is sent as part of a request/response MEP where the request message included a null JMSReplyTo destination and the JMS binding does not include a response/destination then an error SHOULD be raised by the SCA runtime
[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 the SCA runtime MUST set the destination to which callback messages are to be sent as the value of the scaCallbackDestination user property in the message it creates
[BJM60012]	For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent the SCA runtime MAY set the JMSReplyTo destination to the same value as the scaCallbackDestination user property
[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 SHOULD raise an error, and MUST 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 and correlation identifier 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

943

944 **C. Acknowledgements**

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

947 **Participants:**

Participant Name	Affiliation
Bryan Aupperle	IBM
Ron Barack	SAP AG
Michael Beisiegel	IBM
Henning Blohm	SAP AG
David Booz	IBM
Martin Chapman	Oracle Corporation
Jean-Sebastien Delfino	IBM
Laurent Domenech	TIBCO Software Inc.
Jacques Durand	Fujitsu Limited
Mike Edwards	IBM
Billy Feng	Primeton Technologies, Inc.
Nimish Hathalia	TIBCO Software Inc.
Simon Holdsworth	IBM
Eric Johnson	Software Inc.
Uday Joshi	Oracle Corporation
Khanderao Kand	Oracle Corporation
Anish Karmarkar	Oracle Corporation
Nickolaos Kavantzias	Oracle Corporation
Mark Little	Red Hat
Ashok Malhotra	Oracle Corporation
Jim Marino	Individual
Jeff Mischkinisky	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.

948

D. Revision History

949 [optional; should not be included in OASIS Standards]

950

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 27 th 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