



Service Component Architecture JMS Binding Specification Version 1.1

Committee Specification Draft 05 /
Public Review Draft 03

8 November 2010

Specification URIs:

This Version:

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

Previous Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd04.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd04.doc>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd04.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)

Technical Committee:

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

Chair(s):

Simon Holdsworth, IBM <simon_holdsworth@uk.ibm.com>

Editor(s):

Simon Holdsworth, IBM <simon_holdsworth@uk.ibm.com>
Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>

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(s):

<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” 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>).

Citation Format:

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

SCA-JMSBINDING-v1.1 OASIS Committee Specification Draft 05, *Service Component Architecture JMS Binding Specification Version 1.1*, November 2010.
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csd05.pdf>

Notices

Copyright © OASIS® 2006, 2010. 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	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
3.2	JMS Message Headers and User Properties	13
3.3	JMS Message Selection	13
4	Operation Selectors and Wire Formats	14
4.1	Default Operation Selection.....	14
4.2	Default Wire Format	15
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	Subscription with Selector Example	23
7.7	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-1.1.xsd.....	26
B.	Conformance Items	30
C.	Acknowledgements	35
D.	Revision History	36

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://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/>

31	[IETFJMS]	M. Phillips, P. Easton, D. Rokicki, E. Johnson, <i>URI Scheme for Java™ Message Service 1.0</i> http://tools.ietf.org/id/draft-merrick-jms-uri-09.txt , IETF Internet-Draft September 2010 ¹
32		
33		
34	[SCA-Assembly]	OASIS Committee Draft 05, <i>Service Component Architecture Assembly Model Specification Version 1.1</i> , January 2010
35		
36		http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd05.pdf
37		
38	[SCA-Policy]	OASIS Committee Draft 04, <i>SCA Policy Framework Specification Version 1.1</i> , September 2010
39		
40		http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd04.pdf

41 **1.3 Non-Normative References**

42 N/A

43 **1.4 Naming Conventions**

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

- 45 • The naming conventions defined by section 1.3 of the [SCA Assembly Specification \[SCA-Assembly\]](#).
- 46 • Where the names of elements and attributes consist partially or wholly of acronyms, the letters of the
47 acronyms use the same case. When the acronym appears at the start of the name of an element or
48 an attribute, or after a period, it is in lower case. If it appears elsewhere in the name of an element or
49 an attribute, it is in upper case. For example, an attribute might be named "uri" or "jndiURL".
- 50 • Where the names of types consist partially or wholly of acronyms, the letters of the acronyms are in
51 all upper case. For example, an XML Schema type might be named "JCABinding" or "MessageID".
- 52 • Values, including local parts of QName values, follow the rules for names of elements and attributes
53 as stated above, with the exception that the letters of acronyms are in all upper case. For example, a
54 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

55 **2 Messaging Bindings**

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

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

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

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

66

3 JMS Binding Schema

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

```

68 <binding.jms correlationScheme="QName"?
69     initialContextFactory="xs:anyURI"?
70     jndiURL="xs:anyURI"?
71     name="NCName"?
72     requires="list of QName"?
73     policySets="list of QName"?
74     uri="xs:anyURI"? >
75 <destination jndiName="xs:anyURI"? type="queue or topic"?
76     create="always or never or ifNotExist"?
77     <property name="NMTOKEN" type="string"?>*
78 </destination>?
79 <connectionFactory jndiName="xs:anyURI"?
80     create="always or never or ifNotExist"?
81     <property name="NMTOKEN" type="string"?>*
82 </connectionFactory>?
83 <activationSpec jndiName="xs:anyURI"?
84     create="always or never or ifNotExist"?
85     <property name="NMTOKEN" type="string"?>*
86 </activationSpec>?
87
88 <response>
89     <destination jndiName="xs:anyURI"? type="queue or topic"?
90         create="always or never or ifNotExist"?
91         <property name="NMTOKEN" type="string"?>*
92     </destination>?
93     <connectionFactory jndiName="xs:anyURI"?
94         create="always or never or ifNotExist"?
95         <property name="NMTOKEN" type="string"?>*
96     </connectionFactory>?
97     <activationSpec jndiName="xs:anyURI"?
98         create="always or never or ifNotExist"?
99         <property name="NMTOKEN" type="string"?>*
100     </activationSpec>?
101     <wireFormat/>?
102 </response>?
103
104 <resourceAdapter name="NMTOKEN">?
105     <property name="NMTOKEN" type="string"?>*
106 </resourceAdapter>?
107
108 <headers type="string"?
109     deliveryMode="persistent or nonpersistent"?
110     timeToLive="long"?
111     priority="0 .. 9"?
112     <property name="NMTOKEN" type="boolean or byte or .. or String"?>*
113 </headers>?
114
115 <messageSelection selector="string"?
116     <property name="NMTOKEN" type="string"?>*
117 </messageSelection>?
118
119 <operationProperties name="string" selectedOperation="string"?
120     <property name="NMTOKEN" type="string"?>*
121     <headers type="string"?
122         deliveryMode="persistent or nonpersistent"?
123         timeToLive="long"?
124         priority="0 .. 9"?

```



```

125         <property name="NMTOKEN" type="boolean or byte or .. or String"?>*
126     </headers>?
127 </operationProperties>*
128
129     <wireFormat ... />?
130     <operationSelector ... />?
131 </binding.jms>

```

132 *Snippet 3-1: binding.jms Pseudo-Schema*

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

135 The **binding.jms** element has the attributes:

- 136 • **/binding.jms** – This is the JMS binding element. The element is extensible so that JMS binding
 137 implementers can add additional JMS provider-specific attributes and elements although such
 138 extensions are not guaranteed to be portable across runtimes.
- 139 • **/binding.jms/@uri** – as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#). This attribute
 140 identifies the destination, connection factory or activation spec, and other properties to be used to
 141 send/receive the JMS message. There is an implicit **@create="never"** for the resources referred to
 142 in the **@uri** attribute. Message header properties and the message selector set via the **@uri** attribute
 143 take precedence over those specified in binding elements as defined in section 3.2.

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

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

```

148 jms:jndi:<jms-dest>?
149 jndiURL=<jndi-url> &
150 jndiInitialContextFactory=<jndi-initial-context-factory> &
151 jndiConnectionFactoryName=<Connection-Factory-Name> &
152 deliveryMode=<Delivery-Mode> &
153 timeToLive=<Time-To-Live> &
154 priority=<Priority> &
155 selector=<Message-Selector> &
156 <param-name>=<param-value> & ...

```

157 *Snippet 3-2: JMS URI Structure*

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

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

- 161 • **/binding.jms/@name** - as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- 162 • **/binding.jms/@requires** - as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- 163 • **/binding.jms/@policySets** - as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- 164 • **/binding.jms/@correlationScheme** – identifies the correlation scheme used when sending reply or
 165 callback messages, default value is **"sca:messageID"**. Three specific behaviours are provided.
 166 **"sca:messageID"** indicates that response messages can be correlated with their requests by looking
 167 for the request's messageID header value in the response's correlationID header;
 168 **"sca:correlationID"** indicates that response messages can be correlated with their requests by
 169 looking for the request's correlationID header value in the response's correlationID header;
 170 **"sca:none"** indicates that the response's correlationID header is not to be used for this purpose and
 171 some other means is used for the correlation.

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

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

- 176 If the value of the **@correlationScheme** attribute is "**sca:correlationID**" the SCA runtime MUST set
 177 a non-null correlation ID value in requests that it sends [BJM30007].
- 178 If the value of the **@correlationScheme** attribute is "**sca:none**" the SCA runtime MUST NOT set the
 179 correlation ID in responses that it sends [BJM30005].
- 180 SCA runtimes MAY allow other values of the **@correlationScheme** attribute to indicate other
 181 correlation schemes [BJM30006].
- 182 • **/binding.jms/@initialContextFactory** – the name of the JNDI initial context factory.
 - 183 • **/binding.jms/@jndiURL** – the URL for the JNDI provider.
 - 184 • **/binding.jms/destination** – identifies the destination that is to be used to process requests by this
 185 binding.
 - 186 • **/binding.jms/destination/@type** - the type of the request destination. Valid values are "**queue**" and
 187 "**topic**". The default value is "**queue**".
- 188 Whatever the value of the **destination/@type** attribute, the SCA runtime MUST ensure a single
 189 response is delivered for request/response operations [BJM30010].
- 190 • **binding.jms/destination/@jndiName** – the JNDI name of the JMS Destination that the binding uses
 191 to send or receive messages. The behaviour of this attribute is determined by the value of the
 192 **@create** attribute as follows:
 - 193 – If the **@create** attribute value for a **destination**, **connectionFactory** or **activationSpec** element
 194 is "**always**" and the **@jndiName** attribute is present and the resource cannot be created at the
 195 location specified by the **@jndiName** attribute then the SCA runtime MUST raise an error
 196 [BJM30011].
 - 197 If the **@create** attribute value for a **destination**, **connectionFactory** or **activationSpec** element
 198 is "always" and the **@jndiName** attribute is not present and the resource cannot be created, then
 199 the SCA runtime MUST raise an error [BJM30037].
 - 200 If the **@jndiName** attribute is omitted this specification places no restriction on the JNDI location
 201 of the created resource.
 - 202 – If the **@create** attribute value for a **destination**, **connectionFactory** or **activationSpec** element
 203 is "**ifNotExist**" then the **@jndiName** attribute MUST specify the location of the possibly existing
 204 resource [BJM30012].
 - 205 If the **@create** attribute value for a **destination**, **connectionFactory** or **activationSpec** element
 206 is "**ifNotExist**" and the resource does not exist at the location identified by the **@jndiName**
 207 attribute and cannot be created there then the SCA runtime MUST raise an error [BJM30013].
 - 208 If the **@create** attribute value for a **destination**, **connectionFactory** or **activationSpec** element
 209 is "**ifNotExist**" and the **@jndiName** attribute refers to an existing resource that is not a JMS
 210 Destination of the appropriate type, a JMS connection factory or a JMS activation spec
 211 respectively then the SCA runtime MUST raise an error [BJM30014].
 - 212 – If the **@create** attribute value for a **destination**, **connectionFactory** or **activationSpec** element
 213 is "**never**" and the **@jndiName** attribute is not specified, or the resource is not present at the
 214 location identified by the **@jndiName** attribute, or the location refers to a resource of an incorrect
 215 type then the SCA runtime MUST raise an error [BJM30015].
 - 216 • **/binding.jms/destination/@create** – indicates whether the destination should be created when the
 217 containing composite is deployed. Valid values are "**always**", "**never**" and "**ifNotExist**". "**always**"
 218 indicates that new resources are created for use by this binding; "**never**" indicates that existing
 219 resources are used and none created; "**ifNotExist**" indicates that if the resources already exist those
 220 are used, otherwise new ones are created. Refer to the **destination/@jndiName** attribute for a
 221 detailed definition of each case. The default value is "**ifNotExist**".
 - 222 • **/binding.jms/destination/property** – defines properties to be used to create the destination, if
 223 required.

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

227 A **binding.jms** element MUST NOT include both a **connectionFactory** element and an
228 **activationSpec** element [BJM30017].

229 When the **connectionFactory** element is present as a child of the **binding.jms** element, then the
230 destination MUST be defined either by the **destination** element child of the **binding.jms** element or
231 the **@uri** attribute of the **binding.jms** element [BJM30018].

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

235 If the **activationSpec** element is present as a child of the **binding.jms** element and the destination is
236 also specified via a **destination** element child of the **binding.jms** element or the **@uri** attribute of the
237 **binding.jms** element then it MUST refer to the same JMS destination as the **activationSpec**
238 [BJM30019].

239 The **activationSpec** element MUST NOT be present when the binding is being used for an SCA
240 reference [BJM30020].

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

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

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

250 A **response** element MUST NOT include both a **connectionFactory** element and an **activationSpec**
251 element [BJM30021].

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

255 If a **response/destination** and **response/activationSpec** element are both specified they MUST
256 refer to the same JMS destination [BJM30022].

257 The **response/activationSpec** element MUST NOT be present when the binding is being used for an
258 SCA service [BJM30023].

259 • **/binding.jms/response/wireFormat** – identifies the wire format used by responses sent or received
260 by this binding. This value overrides the **wireFormat** specified at the binding level. Wire formats for
261 this binding are described in Section 4.

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

265 • **/binding.jms/headers/@type, @deliveryMode, @timeToLive, @priority** – specifies the value to
266 use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive or JMSPriority
267 respectively. Valid values for **@deliveryMode** are **"persistent"** and **"nonpersistent"**, corresponding
268 to the values defined in the JMS Specification [JMS] for the JMSDeliveryMode message header, with
269 **"persistent"** being the default; valid values for **@priority** are **"0"** to **"9"**, where **"0"** indicates lowest
270 priority and **"9"** highest priority, with **"4"** being the default; valid values for **@timeToLive** are positive
271 integers, with 0 indicating unlimited time and being the default value.

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

- 273 • **/binding.jms/messageSelection** - this element specifies JMS message selection options. This
 274 element applies to a service receiving messages from the request destination or for a reference
 275 receiving messages from the callback or reply-to destination.
- 276 • **/binding.jms/messageSelection/@selector** - specifies the value to use for the JMS message
 277 selector. Section 3.3 defines the priority rules for determining the values for the message selector.
- 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 to be set for standard
 303 JMS headers. These values apply to requests from a reference and responses from a service.
 304 Section 3.2 defines the priority rules for determining the values for JMS headers and user properties.
- 305 • **/binding.jms/operationProperties/headers/@type, @deliveryMode, @timeToLive, @priority** –
 306 specifies the value to use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive
 307 or JMSPriority, respectively. Refer to the description of the **binding.jms/headers** element for the
 308 valid values for these attributes.
- 309 • **/binding.jms/operationProperties/headers/property** – specifies the value and type for the named
 310 JMS user property.
- 311 • **/binding.jms/wireFormat** – identifies the wire format used by requests and responses sent or
 312 received by this binding. Wire formats for this binding are described in Section 4.
- 313 • **/binding.jms/operationSelector** – identifies the operation selector used when receiving requests for
 314 a service. If specified for a reference this provides the default operation selector for callbacks if not
 315 specified via a callback service element. Operation selectors for this binding are described in Section
 316 3.2.
- 317 The **binding.jms** element MUST conform to the XML schema defined in **sca-binding-jms-1.1.xsd**
 318 [BJM30036].

319 3.1 Extensibility

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

324 3.2 JMS Message Headers and User Properties

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

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

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

- 335 1) the value for the header specified in the `@uri` attribute (highest priority);
- 336 2) the value for the header specified in the `operationProperties/headers` element matching the
337 operation being invoked;
- 338 3) the value for the header specified in the `headers` element;
- 339 4) the default value for the header as specified by the definition of the `binding.jms/headers` element
340 (lowest priority) [BJM30024].

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

- 343 1) the type and value for the named user property specified in an
344 `operationProperties/headers/property` element matching the name of the operation being invoked
345 (highest priority);
- 346 2) the type and value for the named user property specified in a `headers/property` element (lowest
347 priority) [BJM30025].

348 3.3 JMS Message Selection

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

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

- 354 1) the value for the message selector specified in the `@uri` attribute value's "selector" parameter
355 (highest priority);
- 356 2) the value for the message selector specified in the `messageSelection/@selector` attribute;
- 357 3) otherwise no message selector is used (lowest priority) [BJM30026].

358 4 Operation Selectors and Wire Formats

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

364 The process of identifying the operation to be invoked is *operation selection*; the information that
365 describes the contents of messages is a *wire format*. The **binding** element as described in the [SCA
366 Assembly Specification \[SCA-Assembly\]](#) provides the means to identify specific operation selection via
367 the **operationSelector** element and the wire format of messages received and to be sent using the
368 **wireFormat** element.

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

- 373 • If the selected operation name generated by the **operationSelector** matches the value of an
374 **operationProperties/@selectedOperation** attribute then the resolved operation name is the value of
375 the **operationProperties/@name** attribute.
- 376 • Otherwise the resolved operation name is the selected operation name generated by the
377 **operationSelector**.

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

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

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

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

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

388 4.1 Default Operation Selection

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

- 392 • If there is only one operation on the service's interface, then that operation is the selected operation
393 name;
- 394 • Otherwise, if the JMS user property “**scaOperationName**” is present, then the value of that user
395 property is used as the selected operation name;
- 396 • Otherwise, if the message is a JMS text or bytes message containing XML, then the selected
397 operation name is the local name of the root element of the XML payload;
- 398 • Otherwise, the selected operation name is “**onMessage**”.

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

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

4.2 Default Wire Format

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

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

- If there is a single parameter that is a **JMSMessage**, then the **JMSMessage** is passed as is.
- Otherwise, if the **JMSMessage** is not a JMS text message or bytes message containing XML it is invalid.
- Otherwise if there is a single parameter, or for the return value, the JMS text or bytes XML payload is the XML serialization of that parameter according to the WSDL schema for the message.
- Otherwise the multiple parameters are encoded in XML using the document wrapped style, according to the WSDL schema for the message.

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

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 [BJM40003].

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

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

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 [BJM40007].

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

4.2.1 Example of default wire format

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">
    <wsdl:part element="tns:setCoordinates" name="setCoordinatesParameters"/>
  </wsdl:message>

  <wsdl:portType name="Coordinates">
    <wsdl:operation name="setCoordinates">
```

```
456     <wsdl:input message="tns:setCoordinatesRequestMsg"
457         name="setCoordinatesRequest"/>
458     </wsdl:operation>
459 </wsdl:portType>
460 </wsdl:definitions>
```

461 *Snippet 4-1: Example WSDL Interface Definition*

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

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

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

470 5 Policy

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

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

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

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

- 479 • *atLeastOnce*
- 480 • *atMostOnce*
- 481 • *ordered*

482 The *atLeastOnce*, *atMostOnce* and *ordered* intents are defined in the [SCA Policy Specification \[SCA-](#)
483 [Policy\]](#) document in section 8, "Reliability Policy".

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

491 6 Message Exchange Patterns

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

498 6.1 One-way message exchange (no Callbacks)

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

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

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

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

510 6.2 Request/response message exchange (no Callbacks)

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

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

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

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

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

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

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

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

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

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

543 6.3 JMS User Properties

544 This protocol assigns specific behavior to JMS user properties:

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

548 6.4 Callbacks

549 Callbacks are SCA's way of representing bidirectional interfaces, where messages are sent in both
550 directions between a client and a service. A callback is the invocation of an operation on a service's
551 callback interface. A callback operation can be one-way or request/response. Messages that correspond
552 to one-way or request/response operations on a bidirectional interface use either the
553 **scaCallbackDestination** user property (for request/response) or the **JMSReplyTo** destination (for one-
554 way) to identify the destination to which messages are to be sent when operations are invoked on the
555 callback interface. The use of **JMSReplyTo** for this purpose is to enable interaction with non-SCA JMS
556 applications, as described below.

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

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

561 6.4.1 Invocation of operations on a bidirectional interface

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

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

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

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

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

578 6.4.2 Invocation of operations on a callback interface

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

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

- 584 • The **scaCallbackDestination** identified by an earlier request/response operation, if not null;
- 585 • the **JMSReplyTo** destination identified by an earlier one-way operation, if not null;
- 586 • the request destination of the service's callback reference JMS binding, if specified

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

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

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

596 **6.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications**

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

603 7 Examples

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

607 7.1 Minimal Binding Example

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

```
610 <?xml version="1.0" encoding="UTF-8"?>
611 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
612     name="MyValueComposite">
613
614     <service name="MyValueService">
615         <interface.java interface="services.myvalue.MyValueService"/>
616         <binding.jms/>
617     </service>
618
619     <reference name="StockQuoteService">
620         <interface.java interface="services.stockquote.StockQuoteService"/>
621         <binding.jms/>
622     </reference>
623 </composite>
```

624 *Snippet 7-1: Minimal Binding Example*

625 7.2 URI Binding Example

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

```
628 <?xml version="1.0" encoding="UTF-8"?>
629 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
630     name="MyValueComposite">
631
632     <service name="MyValueService">
633         <interface.java interface="services.myvalue.MyValueService"/>
634         <binding.jms uri="jms:MyValueServiceQueue?
635             activationSpecName=MyValueServiceAS&
636             ... "/>
637     </service>
638
639     <reference name="StockQuoteService">
640         <interface.java interface="services.stockquote.StockQuoteService"/>
641         <binding.jms uri="jms:StockQuoteServiceQueue?
642             connectionFactoryName=StockQuoteServiceQCF&
643             deliveryMode=1&
644             ... "/>
645     </reference>
646 </composite>
```

647 *Snippet 7-2: Binding Example with URI Specified*

648 7.3 Binding with Existing Resources Example

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

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

```

653
654     <service name="MyValueService">
655         <interface.java interface="services.myvalue.MyValueService"/>
656         <binding.jms>
657             <destination jndiName="MyValueServiceQ" create="never"/>
658             <activationSpec jndiName="MyValueServiceAS" create="never"/>
659         </binding.jms>
660     </service>
661 </composite>

```

662 *Snippet 7-3: Binding Example Using Existing Resources*

663 7.4 Resource Creation Example

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

```

666 <?xml version="1.0" encoding="UTF-8"?>
667 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
668     name="MyValueComposite">
669
670     <service name="MyValueService">
671         <interface.java interface="services.myvalue.MyValueService"/>
672         <binding.jms>
673             <destination jndiName="MyValueServiceQueue" create="always">
674                 <property name="prop1">XYZ</property>
675                 <property name="destName">MyValueDest</property>
676             </destination>
677             <activationSpec jndiName="MyValueServiceAS" create="always"/>
678             <resourceAdapter jndiName="com.example.JMSRA"/>
679         </binding.jms>
680     </service>
681
682     <reference name="StockQuoteService">
683         <interface.java interface="services.stockquote.StockQuoteService"/>
684         <binding.jms>
685             <destination jndiName="StockQuoteServiceQueue"/>
686             <connectionFactory jndiName="StockQuoteServiceQCF"/>
687             <resourceAdapter name="com.example.JMSRA"/>
688         </binding.jms>
689     </reference>
690 </composite>

```

691 *Snippet 7-4: Binding Example that Creates a Resource*

692 7.5 Request/Response Example

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

```

696 <?xml version="1.0" encoding="UTF-8"?>
697 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
698     name="MyValueComposite">
699
700     <service name="MyValueService">
701         <interface.java interface="services.myvalue.MyValueService"/>
702         <binding.jms correlationScheme="sca:messageID">
703             <destination jndiName="MyValueServiceQ" create="never"/>
704             <activationSpec jndiName="MyValueServiceAS" create="never"/>
705         </binding.jms>
706     </service>
707
708     <reference name="StockQuoteService">
709         <interface.java interface="services.stockquote.StockQuoteService"/>

```

```

710     <binding.jms correlationScheme="sca:messageID">
711         <destination jndiName="StockQuoteServiceQueue"/>
712         <connectionFactory jndiName="StockQuoteServiceQCF"/>
713         <response>
714             <destination jndiName="MyValueResponseQueue"/>
715             <activationSpec jndiName="MyValueResponseAS"/>
716         </response>
717     </binding.jms>
718 </reference>
719 </composite>

```

720 *Snippet 7-5: Binding Example with a Response*

721 7.6 Subscription with Selector Example

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

```

724 <?xml version="1.0" encoding="UTF-8"?>
725 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
726     name="MyValueComposite">
727     <service name="MyValueService">
728         <interface.java interface="services.myvalue.MyValueService"/>
729         <binding.jms>
730             <destination jndiName="MyValueServiceTopic" create="never"/>
731             <connectionFactory jndiName="StockQuoteServiceTCF"
732                 create="never"/>
733             <messageSelection selector="Price>1000"/>
734         </binding.jms>
735     </service>
736 </composite>

```

737 *Snippet 7-6: Binding Example with a Selector*

738 7.7 Policy Set Example

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

```

742 <policySet name="JMSPolicy"
743     provides="priority log"
744     appliesTo="binding.jms">
745
746     <intentMap provides="priority" default="medium">
747         <qualifier name="high">
748             <headers priority="9"/>
749         </qualifier>
750         <qualifier name="medium">
751             <headers priority="4"/>
752         </qualifier>
753         <qualifier name="low">
754             <headers priority="0"/>
755         </qualifier>
756     </intentMap>
757
758     <intentMap provides="log">
759         <qualifier>
760             <headers>
761                 <property name="user_example_log">logged</property>
762             </headers>
763         </qualifier>
764     </intentMap>
765 </policySet>

```

766 *Snippet 7-7: Example Policy Set*

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

```
769 <reference name="StockQuoteService" requires="priority.high log">  
770   <interface.java interface="services.stockquote.StockQuoteService"/>  
771   <binding.jms>  
772     <destination name="StockQuoteServiceQueue"/>  
773     <connectionFactory name="StockQuoteServiceQCF"/>  
774   </binding.jms>  
775 </reference>
```

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

777 8 Conformance

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

- 781 a) SCA JMS Binding XML Document
- 782 b) SCA Runtime

783 8.1 SCA JMS Binding XML Document

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

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

791 8.2 SCA Runtime

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

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

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

```

801 <?xml version="1.0" encoding="UTF-8"?>
802 <!-- Copyright(C) OASIS(R) 2005,2010. All Rights Reserved.
803      OASIS trademark, IPR and other policies apply. -->
804 <schema xmlns="http://www.w3.org/2001/XMLSchema"
805       targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
806       xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912"
807       elementFormDefault="qualified">
808
809   <include schemaLocation="sca-core-1.1-cd05.xsd"/>
810
811   <complexType name="JMSBinding">
812     <complexContent>
813       <extension base="sca:Binding">
814         <sequence>
815           <element name="destination" type="sca:JMSDestination"
816             minOccurs="0"/>
817           <choice minOccurs="0" maxOccurs="1">
818             <element name="connectionFactory"
819               type="sca:JMSConnectionFactory"/>
820             <element name="activationSpec" type="sca:JMSActivationSpec"/>
821           </choice>
822           <element name="response" type="sca:JMSResponse" minOccurs="0"/>
823           <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
824           <element name="messageSelection" type="sca:JMSMessageSelection"
825             minOccurs="0"/>
826           <element name="resourceAdapter" type="sca:JMSResourceAdapter"
827             minOccurs="0"/>
828           <element name="operationProperties"
829             type="sca:JMSOperationProperties"
830             minOccurs="0" maxOccurs="unbounded"/>
831           <element ref="sca:extensions" minOccurs="0" maxOccurs="1"/>
832         </sequence>
833         <attribute name="correlationScheme" type="QName"
834           default="sca:messageID"/>
835         <attribute name="initialContextFactory" type="anyURI"/>
836         <attribute name="jndiURL" type="anyURI"/>
837       </extension>
838     </complexContent>
839   </complexType>
840
841   <simpleType name="JMSCreateResource">
842     <restriction base="string">
843       <enumeration value="always"/>
844       <enumeration value="never"/>
845       <enumeration value="ifNotExist"/>
846     </restriction>
847   </simpleType>
848
849   <complexType name="JMSDestination">
850     <sequence>
851       <element name="property" type="sca:BindingProperty"
852         minOccurs="0" maxOccurs="unbounded"/>
853     </sequence>
854     <attribute name="jndiName" type="anyURI"/>
855     <attribute name="type" use="optional" default="queue">
856       <simpleType>
857         <restriction base="string">
858           <enumeration value="queue"/>
859           <enumeration value="topic"/>

```

```

860         </restriction>
861     </simpleType>
862 </attribute>
863     <attribute name="create" type="sca:JMSCreateResource"
864         use="optional" default="ifNotExist"/>
865 </complexType>
866
867 <complexType name="JMSConnectionFactory">
868     <sequence>
869         <element name="property" type="sca:BindingProperty"
870             minOccurs="0" maxOccurs="unbounded"/>
871     </sequence>
872     <attribute name="jndiName" type="anyURI"/>
873     <attribute name="create" type="sca:JMSCreateResource"
874         use="optional" default="ifNotExist"/>
875 </complexType>
876
877 <complexType name="JMSActivationSpec">
878     <sequence>
879         <element name="property" type="sca:BindingProperty"
880             minOccurs="0" maxOccurs="unbounded"/>
881     </sequence>
882     <attribute name="jndiName" type="anyURI"/>
883     <attribute name="create" type="sca:JMSCreateResource"
884         use="optional" default="ifNotExist"/>
885 </complexType>
886
887 <complexType name="JMSResponse">
888     <sequence>
889         <element ref="sca:wireFormat" minOccurs="0" maxOccurs="1"/>
890         <element name="destination" type="sca:JMSDestination" minOccurs="0"/>
891         <choice minOccurs="0">
892             <element name="connectionFactory" type="sca:JMSConnectionFactory"/>
893             <element name="activationSpec" type="sca:JMSActivationSpec"/>
894         </choice>
895     </sequence>
896 </complexType>
897
898 <complexType name="JMSHeaders">
899     <sequence>
900         <element name="property" type="sca:JMSUserProperty"
901             minOccurs="0" maxOccurs="unbounded"/>
902     </sequence>
903     <attribute name="type" type="string"/>
904     <attribute name="deliveryMode" default="persistent">
905         <simpleType>
906             <restriction base="string">
907                 <enumeration value="persistent"/>
908                 <enumeration value="nonpersistent"/>
909             </restriction>
910         </simpleType>
911     </attribute>
912     <attribute name="timeToLive" type="long" default="0"/>
913     <attribute name="priority" default="4">
914         <simpleType>
915             <restriction base="string">
916                 <enumeration value="0"/>
917                 <enumeration value="1"/>
918                 <enumeration value="2"/>
919                 <enumeration value="3"/>
920                 <enumeration value="4"/>
921                 <enumeration value="5"/>
922                 <enumeration value="6"/>
923                 <enumeration value="7"/>

```

```

924         <enumeration value="8"/>
925         <enumeration value="9"/>
926     </restriction>
927 </simpleType>
928 </attribute>
929 </complexType>
930
931 <complexType name="JMSMessageSelection">
932     <sequence>
933         <element name="property" type="sca:BindingProperty"
934             minOccurs="0" maxOccurs="unbounded"/>
935     </sequence>
936     <attribute name="selector" type="string"/>
937 </complexType>
938
939 <complexType name="JMSResourceAdapter">
940     <sequence>
941         <element name="property" type="sca:BindingProperty"
942             minOccurs="0" maxOccurs="unbounded"/>
943     </sequence>
944     <attribute name="name" type="string" use="required"/>
945 </complexType>
946
947 <complexType name="JMSOperationProperties">
948     <sequence>
949         <element name="property" type="sca:BindingProperty"
950             minOccurs="0" maxOccurs="unbounded"/>
951         <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
952     </sequence>
953     <attribute name="name" type="string" use="required"/>
954     <attribute name="selectedOperation" type="string"/>
955 </complexType>
956
957 <complexType name="BindingProperty">
958     <simpleContent>
959         <extension base="string">
960             <attribute name="name" type="NMTOKEN" use="required"/>
961             <attribute name="type" type="string" use="optional"
962                 default="xs:string"/>
963         </extension>
964     </simpleContent>
965 </complexType>
966
967 <simpleType name="JMSUserPropertyType">
968     <restriction base="string">
969         <enumeration value="boolean"/>
970         <enumeration value="byte"/>
971         <enumeration value="short"/>
972         <enumeration value="int"/>
973         <enumeration value="long"/>
974         <enumeration value="float"/>
975         <enumeration value="double"/>
976         <enumeration value="String"/>
977         <enumeration value="xs:string"/>
978     </restriction>
979 </simpleType>
980
981 <complexType name="JMSUserProperty">
982     <simpleContent>
983         <extension base="string">
984             <attribute name="name" type="NMTOKEN" use="required"/>
985             <attribute name="type" type="sca:JMSUserPropertyType"
986                 use="optional" default="String"/>
987         </extension>

```

```
988     </simpleContent>
989 </complexType>
990
991 <complexType name="JMSDefaultWireFormatType">
992   <complexContent>
993     <extension base="sca:WireFormatType"/>
994   </complexContent>
995 </complexType>
996
997 <complexType name="JMSDefaultOperationSelectorType">
998   <complexContent>
999     <extension base="sca:OperationSelectorType"/>
1000   </complexContent>
1001 </complexType>
1002
1003 <element name="binding.jms" type="sca:JMSBinding"
1004   substitutionGroup="sca:binding"/>
1005
1006 <element name="wireFormat.jmsDefault"
1007   type="sca:JMSDefaultWireFormatType"
1008   substitutionGroup="sca:wireFormat"/>
1009
1010 <element name="operationSelector.jmsDefault"
1011   type="sca:JMSDefaultOperationSelectorType"
1012   substitutionGroup="sca:operationSelector"/>
1013 </schema>
```

1014

B. Conformance Items

1015

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
[BJM30006]	SCA runtimes MAY allow other values of the @correlationScheme attribute to indicate other correlation schemes
[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 appropriate 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: <ol style="list-style-type: none"> 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: <ol style="list-style-type: none"> 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: <ol style="list-style-type: none"> 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)
[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
[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
[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
[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
[BJM60001]	For an SCA reference with a JMS binding and unidirectional interface, 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 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 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 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 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 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

1016

1017

C. Acknowledgements

1018 The following individuals have participated in the creation of this specification and are gratefully
1019 acknowledged:

1020 **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 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.

1021

D. Revision History

1022 [optional; should not be included in OASIS Standards]

1023

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
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

1024