



# Service Component Architecture JMS Binding Specification Version 1.1

Committee Draft 01 revision 4

21st January, 2009

**Specification URIs:**

**This Version:**

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec-cd01-rev4.html>  
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec-cd01-rev4.doc>  
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec-cd01-rev4.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-binding-jms-1.1-spec.html>  
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec.doc>  
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-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 supercedes:

- 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/200712>

**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, 2008. 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
	.....	<b>Error! Bookmark not defined.</b>
1.2	Normative References .....	5
1.3	Non-Normative References .....	6
2	Messaging Bindings .....	7
3	JMS Binding Schema .....	8
4	Operation Selectors and Wire Formats .....	13
4.1	Default Operation Selection.....	13
4.2	Default Wire Format.....	13
5	Policy .....	15
6	Message Exchange Patterns .....	16
6.1	One-way message exchange (no Callbacks).....	16
6.2	Request/response message exchange (no Callbacks).....	16
6.3	JMS User Properties.....	16
6.4	Callbacks .....	17
6.4.1	Invocation of operations on a bidirectional interface.....	17
6.4.2	Invocation of operations on a callback interface .....	17
6.4.3	Use of JMSReplyTo for callbacks for non-SCA JMS applications.....	17
6.5	Conversations.....	18
6.5.1	Starting a conversation.....	18
6.5.2	Continuing a conversation.....	18
6.5.3	Ending a conversation.....	18
7	Examples.....	19
7.1	Minimal Binding Example .....	19
7.2	URI Binding Example.....	19
7.3	Binding with Existing Resources Example .....	19
7.4	Resource Creation Example.....	20
7.5	Request/Response Example .....	20
7.6	Use of Predefined Definitions Example .....	21
7.7	Subscription with Selector Example .....	21
7.8	Policy Set Example.....	21
8	Conformance .....	23
A.	JMS Binding Schema .....	24
B.	Acknowledgements .....	27
C.	Non-Normative Text .....	28
D.	Revision History.....	29

# 1 Introduction

This document defines the concept and behavior of a messaging binding, and a concrete JMS-based [JMS] 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 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/200712"	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] JMS 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] Java 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://www.ietf.org/internet-drafts/draft-merrick-jms-uri-05.txt><sup>1</sup>  
34        **[SCA-Assembly]**    <http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.html>

### 35    **1.3 Non-Normative References**

- 36        **TBD**                TBD

---

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

---

## 37 **2 Messaging Bindings**

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

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

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

45 In addition, each operation may have specific properties defined, that may influence the way native  
46 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"?
  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 JMSType="string"?
  JMSDeliveryMode="PERSISTENT or NON_PERSISTENT"?
  JMSTimeToLive="long"?
  JMSPriority="0 .. 9"?>
  <property name="NMTOKEN" type="NMTOKEN"?>*>
</headers>?

<subscriptionHeaders JMSSelector="string"?>
  <property name="NMTOKEN" type="NMTOKEN"?>*>
</headers>?

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



106  
107  
108  
109  
110  
111  
112  
113  
114  
115

```
JMSDeliveryMode="PERSISTENT or NON_PERSISTENT"?
JMSTimeToLive="long"?
JMSPriority="0 .. 9"?>
  <property name="NMTOKEN" type="NMTOKEN" ?>*
  </headers>?
</operationProperties>*

  <wireFormat/>?
  <operationSelector/>?
</binding.jms>
```

116

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

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

- 120 • **/binding.jms** – This is the generic JMS binding type. The type is extensible so that JMS binding  
121 implementers can add additional JMS provider-specific attributes and elements although such  
122 extensions are not guaranteed to be portable across runtimes.
- 123 • **/binding.jms/@uri** – (from binding) URI that identifies the destination, connection factory or activation  
124 spec, and other properties to be used to send/receive the JMS message

125  
126  
127  
128  
129

The value of the **@uri** attribute MUST have the following format, defined by the IETF URI Scheme for Java™ Message Service 1.0 **IETFJMS**. The following illustrates the structure of the URI and the set of property names that have specific semantics - all other property names are treated as user property names:

- 130 – **jms:<jms-dest>?**  
131 **connectionFactoryName=<Connection-Factory-Name> &**  
132 **destinationType={queue|topic}**  
133 **deliveryMode=<Delivery-Mode> &**  
134 **timeToLive=<Time-To-Live> &**  
135 **priority=<Priority> &**  
136 **selector=<Selector> &**  
137 **<User-Property>=<User-Property-Value> & ...**

138 When the **@uri** attribute is specified, the SCA runtime MUST raise an error if the referenced  
139 resources do not already exist.

- 140 • **/binding.jms/@name** - as defined in the SCA Assembly specification in Section 9, “Binding”
- 141 • **/binding.jms/@requires** - as defined in the SCA Assembly specification in Section 9, “Binding”
- 142 • **/binding.jms/@correlationScheme** – identifies the correlation scheme used when sending reply or  
143 callback messages. Possible values for the **@correlationScheme** attribute are “**sca:MessageID**”  
144 (the default) where the SCA runtime MUST set the correlation ID of replies to the message ID of the  
145 corresponding request; “**sca:CorrelationID**” where the SCA runtime MUST set the correlation ID of  
146 replies to the correlation ID of the corresponding request, and “**sca:None**” which indicates that the  
147 SCA runtime MUST NOT set the correlation ID. SCA runtimes MAY allow other values to indicate  
148 other correlation schemes.
- 149 • **/binding.jms/@initialContextFactory** – the name of the JNDI initial context factory.
- 150 • **/binding.jms/@jndiURL** – the URL for the JNDI provider.
- 151 • **/binding.jms/@requestConnection** – identifies a **binding.jms** element that is present in a definition  
152 document, whose **destination**, **connectionFactory**, **activationSpec** and **resourceAdapter** children  
153 are used to define the values for this binding. In this case this **binding.jms** element MUST NOT also  
154 contain the corresponding elements.
- 155 • **/binding.jms/@responseConnection** – identifies a **binding.jms** element that is present in a  
156 definition document, whose **response** child element is used to define the values for this binding. In  
157 this case this **binding.jms** element MUST NOT contain a **response** element.

- 158 • **/binding.jms/@operationProperties** – identifies a **binding.jms** element that is present in a definition  
159 document, whose **operationProperties** children are used to define the values for this binding. In this  
160 case this **binding.jms** element MUST NOT contain an **operationProperties** element.
- 161 • **/binding.jms/destination** – identifies the destination that is to be used to process requests by this  
162 binding.
- 163 • **/binding.jms/destination/@type** - the type of the request destination. Valid values are “**queue**” and  
164 “**topic**”. The default value is “**queue**”. In either case the runtime MUST ensure a single response is  
165 delivered for request/response operations.
- 166 • **binding.jms/destination/@jndiName** – the JNDI name of the JMS Destination that the binding uses  
167 to send or receive messages. The behaviour of this attribute is determined by the value of the  
168 **@create** attribute as follows:
  - 169 – If the **@create** attribute value is "always" then the **@jndiName** attribute is optional; if the  
170 destination cannot be created at the specified location then the SCA runtime MUST raise an  
171 error. If the **@jndiName** attribute is omitted this specification places no restriction on the JNDI  
172 location of the created resource.
  - 173 – If the **@create** attribute value is "ifnotexist" then the **@jndiName** attribute MUST specify the  
174 location of the possibly existing destination; if the destination does not exist at this location, but  
175 cannot be created there then the SCA runtime MUST raise an error. If the **@jndiName** refers to  
176 an existing resource other than a JMS Destination of the specified type then the SCA runtime  
177 MUST raise an error.
  - 178 – If the **@create** attribute value is "never" then the **@jndiName** attribute MUST specify the location  
179 of the existing destination; If the destination is not present at the location, or the location refers to  
180 a resource other than a JMS Destination of the specified type then the SCA runtime MUST raise  
181 an error.
- 182 • **/binding.jms/destination/@create** – indicates whether the destination should be created when the  
183 containing composite is deployed. Valid values are “**always**”, “**never**” and “**ifnotexist**”. The default  
184 value is “**ifnotexist**”.
- 185 • **/binding.jms/destination/property** – defines properties to be used to create the destination, if  
186 required.
- 187 • **/binding.jms/connectionFactory** – identifies the connection factory that the binding uses to process  
188 request messages. The attributes of this element follow those defined for the **destination** element.  
189 A **binding.jms** element MUST NOT include both this element and an **activationSpec** element. When  
190 this element is present, the **destination** element MUST also be present
- 191 • **/binding.jms/activationSpec** – identifies the activation spec that the binding uses to connect to a  
192 JMS destination to process request messages. The attributes of this element follow those defined for  
193 the **destination** element. If a **destination** element is also specified it MUST refer to the same JMS  
194 destination as the **activationSpec**. This element MUST NOT be present when the binding is being  
195 used for an SCA reference.
- 196 • **/binding.jms/response** – defines the resources used for handling response messages (receiving  
197 responses for a reference, and sending responses from a service).
- 198 • **/binding.jms/response/destination** – identifies the destination that is to be used to process  
199 responses by this binding. Attributes are as for the parent’s **destination** element. For a service, this  
200 destination is used to send responses to messages that have a null value for the **JMSReplyTo**  
201 destination. For a reference, this destination is used to receive reply messages
- 202 • **/binding.jms/response/connectionFactory** – identifies the connection factory that the binding uses  
203 to process response messages. The attributes of this element follow those defined for the  
204 **destination** element. A **response** element MUST NOT include both this element and an  
205 **activationSpec** element.
- 206 • **/binding.jms/response/activationSpec** – identifies the activation spec that the binding uses to  
207 connect to a JMS destination to process response messages. The attributes of this element follow  
208 those defined for the **destination** element. If a response **destination** element is also specified it

- 209 MUST refer to the same JMS destination as the **activationSpec**. This element MUST NOT be  
210 present when the binding is being used for an SCA service.
- 211 • **/binding.jms/response/wireFormat** – identifies the wire format used by responses sent or received  
212 by this binding. This value overrides the **wireFormat** specified at the binding level.
  - 213 • **/binding.jms/headers** – this element specifies values for standard JMS headers that the SCA  
214 runtime MUST set to the given values for all operations. These values apply to requests from a  
215 reference and responses from a service.
  - 216 • **/binding.jms/headers/@JMSType, @JMSDeliveryMode, @JMSTimeToLive, @JMSPriority** –  
217 specifies the value to use for the JMS header property. The value of the **@uri** attribute MUST NOT  
218 include values for these properties if they are specified using these attributes. Valid values for  
219 **@JMSDeliveryMode** are “**PERSISTENT**” and “**NON\_PERSISTENT**”; valid values for **@JMSPriority**  
220 are “**0**” to “**9**”.
  - 221 • **/binding.jms/headers/property** – specifies the value that the SCA runtime MUST set for the  
222 specified JMS user property when creating messages..
  - 223 • **/binding.jms/subscriptionHeaders** - this element allows JMS subscription options to be set. These  
224 values apply to a service subscribing to the destination or for a reference subscribing to the callback  
225 or reply-to destinations.
  - 226 • **/binding.jms/subscriptionHeaders/@JMSSelector** - specifies the value to use for the JMS selector.  
227 The value of the **@uri** attribute MUST NOT include values for this property if it is specified using this  
228 attribute.
  - 229 • **/binding.jms/resourceAdapter** – specifies name, type and properties of the Resource Adapter Java  
230 bean. This element MUST be present when the JMS resources are to be created for a JMS provider  
231 that implements the JCA 1.5 specification **JCA15**, and is ignored otherwise. SCA runtimes MAY place  
232 restrictions on the properties of the RA Java bean that can be set. For JMS providers that do not  
233 implement the JCA 1.5 specification, information necessary for resource creation can be added in  
234 provider-specific elements or attributes allowed by the extensibility of the **binding.jms** element.
  - 235 • **/binding.jms/operationProperties** – specifies various properties that are specific to the processing  
236 of a particular operation.
  - 237 • **/binding.jms/operationProperties/@name** – The name of the operation in the interface.
  - 238 • **/binding.jms/operationProperties/@selectedOperation** – The value generated by the  
239 **operationSelector** that corresponds to the operation in the service or reference interface identified  
240 by the **operationProperties/@name** attribute. If this attribute is omitted then the value defaults to  
241 the value of the **operationProperties/@name** attribute. The value of this attribute MUST be unique  
242 across the containing **binding.jms** element..
  - 243 • **/binding.jms/operationProperties/property** – specifies properties specific to this operation. These  
244 properties are intended to be used to parameterize the **wireFormat** identified for the binding for a  
245 particular operation. The SCA runtime SHOULD make the **operationProperties** element  
246 corresponding to the **selectedOperation** available to the **wireFormat** implementation.
  - 247 • **/binding.jms/operationProperties/headers** – this element specifies values for standard JMS  
248 headers that the SCA runtime MUST set to the given values for the given operation. These values  
249 apply to requests from a reference and responses from a service.
  - 250 • **/binding.jms/operationProperties/headers/@JMSType, @JMSDeliveryMode, @JMSTimeToLive,**  
251 **@JMSPriority** – specifies the value to use for the JMS header property. The SCA runtime MUST  
252 use values specified for particular operations in preference to those defined for all operations in the  
253 **binding.jms/headers** element or via the binding’s **@uri** attribute.
  - 254 • **/binding.jms/operationProperties/headers/property** – specifies the value that the SCA runtime  
255 MUST set for the specified JMS user property when creating messages.
  - 256 • **/binding.jms/wireFormat** – identifies the wire format used by requests and responses sent or  
257 received by this binding.

- 258 • ***/binding.jms/operationSelector*** – identifies the operation selector used when receiving requests for  
259 a service. If specified for a reference this provides the default operation selector for callbacks if not  
260 specified via a callback service element.
- 261 • ***/binding.jms/@{any}*** - this is an extensibility mechanism to allow extensibility via attributes.
- 262 • ***/binding.jms/any*** – this is an extensibility mechanism to allow extensibility via elements.
- 263 Deployers/assemblers can configure ***NON\_PERSISTENT*** for ***@JMSDeliveryMode*** in order to provide  
264 higher performance with a decreased quality of service. A ***binding.jms*** element configured in this way  
265 cannot satisfy either of the "***atLeastOnce***" and "***exactlyOnce***" policy intents. The SCA Runtime MUST  
266 raise an error for this invalid combination at deployment time.

---

## 267 4 Operation Selectors and Wire Formats

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

273 The process of identifying the operation to be invoked is *operation selection*; the information that  
274 describes the contents of messages is a *wire format*. The **binding** element as described in the SCA  
275 Assembly specification [SCA-Assembly] provides the means to identify specific operation selection via the  
276 **operationSelector** element and the wire format of messages received and to be sent using the  
277 **wireFormat** element.

278 No standard means is provided for linking the **wireFormat** or **operationSelector** elements with the  
279 runtime components that implement their behaviour.

280 This section describes the default **operationSelector** and **wireFormat** for a JMS binding. The SCA  
281 runtime MUST support this default behavior, and MAY provide additional means to override it.

### 282 4.1 Default Operation Selection

283 When receiving a request at a service, or a callback at a reference, the selected operation name is  
284 determined as follows:

- 285 • If there is only one operation on the service’s interface, then that operation is assumed as the  
286 selected operation name.
- 287 • Otherwise, if the JMS user property “**scaOperationName**” is present, then its value is used as the  
288 selected operation name.
- 289 • Otherwise, if the message is a JMS text or bytes message containing XML, then the selected  
290 operation name is taken from the local name of the root element of the XML payload.
- 291 • Otherwise, the selected operation name is assumed to be “**onMessage**”.

292 The selected operation name is then mapped to an operation in the service’s interface via a matching  
293 **operationProperties** element in the JMS binding. If there is no matching element, the operation name is  
294 assumed to be the same as the selected operation name.

295 The use of this operation selector can be explicitly specified in a **binding.jms** using the  
296 **operationSelector.jmsdefault** element; if no **operationSelector** element is specified then SCA runtimes  
297 MUST use this as the default.

### 298 4.2 Default Wire Format

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

303 The message body is mapped to the parameters or return value of the target operation as follows:

- 304 • If there is a single parameter that is a **JMSMessage**, then the **JMSMessage** is passed as is.
- 305 • Otherwise, the **JMSMessage** must be a JMS text message or bytes message containing XML; an  
306 SCA runtime MUST be able to receive both forms. When sending messages either form may be  
307 used; an SCA runtime MAY provide additional configuration to allow one or other to be selected.
- 308 • If there is a single parameter, or for the return value, the JMS text or bytes XML payload is the XML  
309 serialization of that parameter according to the WSDL schema for the message.

- 310 • If there are multiple parameters, then they are encoded in XML using the document wrapped style,  
311 according to the WSDL schema for the message.
- 312 • When sending request messages, if there is a single parameter and the interface includes more than  
313 one operation, the SCA runtime MUST set the JMS user property "**scaOperationName**" to the name  
314 of the operation being invoked.

315 The use of this wire format can be explicitly specified in a **binding.jms** using the **wireFormat.jmsdefault**  
316 element; if no **wireFormat** element is specified then SCA runtimes MUST use this as the default.

317 For example, for the following interface definition:

```
318 <wsdl:definitions name="Coordinates"  
319 targetNamespace="http://tempuri.org/coordinates"  
320 xmlns:tns="http://tempuri.org/coordinates"  
321 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  
322 xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
323 <wsdl:types>  
324 <xsd:schema targetNamespace="http://tempuri.org/coordinates">  
325 <xsd:element name="setCoordinates">  
326 <xsd:complexType>  
327 <xsd:sequence>  
328 <xsd:element name="x" type="xsd:int"/>  
329 <xsd:element name="y" type="xsd:int"/>  
330 </xsd:sequence>  
331 </xsd:complexType>  
332 </xsd:element>  
333 </xsd:schema>  
334 </wsdl:types>  
335  
336 <wsdl:message name="setCoordinatesRequestMsg">  
337 <wsdl:part element="tns:setCoordinates" name="setCoordinatesParameters"/>  
338 </wsdl:message>  
339  
340 <wsdl:portType name="Coordinates">  
341 <wsdl:operation name="setCoordinates">  
342 <wsdl:input message="tns:setCoordinatesRequestMsg"  
343 name="setCoordinatesRequest"/>  
344 </wsdl:operation>  
345 </wsdl:portType>  
346 </wsdl:definitions>
```

347

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

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

355

---

## 5 Policy

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

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

361  
362  
363

```
<bindingType type="binding.jms"  
            alwaysProvides="jms"  
            mayProvide="atLeastOnce atMostOnce ordered conversational"/>
```

364 The atLeastOnce, atMostOnce and ordered intent are defined in the SCA Policy Specification document  
365 in section 8, "Reliability Policy". The conversational intent is defined in the SCA Assembly Specification  
366 document in section 8.3, "Conversational Interfaces".

---

## 367 6 Message Exchange Patterns

368 This section describes the message exchange patterns that are possible when using the JMS binding,  
369 including one-way, request/response, callbacks and conversations. JMS has a looser concept of  
370 message exchange patterns than WSDL, so this section explains how JMS messages that are sent and  
371 received by the SCA runtime relate to the WSDL input/output messages. Each operation in a WSDL  
372 interface is either one-way or request/response. Callback interfaces may include both one-way and  
373 request/response operations.

### 374 6.1 One-way message exchange (no Callbacks)

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

378 When a request message is sent by a reference with a JMS binding for a one-way MEP, the SCA runtime  
379 SHOULD NOT set the **JMSReplyTo** destination header in the JMS message that it creates, regardless of  
380 whether the JMS binding has a **response** element with a **destination** defined.

381 When a request message is received by a service with a JMS binding for a one-way MEP, the SCA  
382 runtime MUST ignore the **JMSReplyTo** destination header in the JMS message, and MUST NOT raise  
383 an error.

384 The use of one-way exchanges when using a bidirectional interface is described in section 7.4.

### 385 6.2 Request/response message exchange (no Callbacks)

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

389 When a request message is sent by a reference with a JMS binding for a request/response MEP, the  
390 SCA runtime MUST set a non-null value for the **JMSReplyTo** header in the JMS message it creates for  
391 the request. If the JMS binding has a **response** element with a **destination** defined, then the SCA  
392 runtime MUST use that destination for the **JMSReplyTo** header value, otherwise the SCA runtime MUST  
393 provide an appropriate destination on which to receive response messages. The SCA runtime MAY  
394 choose to receive the response message on the basis of its correlation ID as defined by the binding's  
395 **@correlationScheme** attribute, or use a unique destination for each response.

396 When a response message is sent by a service with a JMS binding for a request/response MEP, the SCA  
397 runtime MUST send the response message to the destination identified by the request message's  
398 **JMSReplyTo** header value if it is not null, otherwise the SCA runtime MUST send the response message  
399 to the destination identified by the JMS binding's **response** element if specified. If there is no destination  
400 defined by either means then an error SHOULD be raised by the SCA runtime. The SCA runtime MUST  
401 set the correlation identifier in the JMS message that it creates for the response as defined by the JMS  
402 binding's **@correlationScheme** attribute.

403 The use of request/response exchanges when using a bidirectional interface is described in section 7.4.

### 404 6.3 JMS User Properties

405 This protocol assigns specific behavior to JMS user properties:

- 406 • "**scaCallbackDestination**" holds the name of the JMS Destination to which callback messages are  
407 sent.
- 408 • "**scaConversationStart**" indicates that a conversation is to be started, its value is the identifier for the  
409 conversation.



- 410 • "**scaConversationMaxIdleTime**" defines the maximum time that should be allowed between  
411 operations in the conversation.
- 412 • "**scaConversationId**" holds the identifier for the conversation.

## 413 6.4 Callbacks

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

### 422 6.4.1 Invocation of operations on a bidirectional interface

423 When a request message is sent by a reference with a JMS binding for a one-way MEP with a  
424 bidirectional interface, the SCA runtime **MUST** set the destination to which callback messages are to be  
425 sent as the value of the **scaCallbackDestination** user property in the message it creates. The SCA  
426 runtime **MAY** also set the **JMSReplyTo** destination to this value.

427 When a request message is sent by a reference with a JMS binding for a request/response MEP with a  
428 bidirectional interface, the SCA runtime **MUST** set the **scaCallbackDestination** user property in the  
429 message it creates to identify the destination from which it will read callback messages. The SCA runtime  
430 **MUST** set the **JMSReplyTo** header in the message it creates as described in section 7.2.

431 For both one-way and request/response operations, if the reference has a callback service element with a  
432 JMS binding with a request destination, then the SCA runtime **MUST** use that destination as the one to  
433 which callback messages are to be sent, otherwise the SCA runtime **MUST** provide an appropriate  
434 destination for this purpose.

### 435 6.4.2 Invocation of operations on a callback interface

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

440 When a callback request message is sent by a service with a JMS binding for either a one-way or  
441 request/response MEP, the SCA runtime **MUST** send the callback request message to the JMS  
442 destination identified as follows, in order of priority:

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

446 If no destination is identified then the SCA runtime **SHOULD** raise an error, and **MUST** throw an  
447 exception to the caller of the callback operation.

448 The SCA runtime **MUST** set the **JMSReplyTo** destination and correlation identifier in the callback request  
449 message as defined in sections 7.1 or 7.2 as appropriate for the type of the callback operation invoked.

### 450 6.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications

451 When interacting with non-SCA JMS applications, the assembler can choose to model a  
452 request/response message exchange using a bidirectional interface. In this case it is likely that the non-  
453 SCA JMS application does not support the use of the **scaCallbackDestination** user property. To support  
454 this, for one-way messages the **JMSReplyTo** header can be used to identify the destination to be used to  
455 deliver callback messages, as described in sections 7.4.1 and 7.4.2.

## 456 6.5 Conversations

457 A conversation is a sequence of operations between two parties that have a common context. The  
458 conversation can include a mixture of operations in either direction between the two parties, if the  
459 interface is also bidirectional. Interfaces are marked as conversational in order to ensure that the runtime  
460 manages the lifecycle of this context. Component implementation specifications define the manner in  
461 which the context that is associated with the conversation identifier is made available to component  
462 implementations.

### 463 6.5.1 Starting a conversation

464 A conversation is started when an operation is invoked on a conversational interface and there is no  
465 active conversation with the target of the invocation. When this happens the SCA runtime MUST supply  
466 an identifier for the conversation, if the client component has not already supplied an identifier, and the  
467 SCA runtime MUST set the **scaConversationStart** user property to this value in the JMS message that it  
468 sends for the request, and associate a new runtime context with this conversation identifier.

469 When a message is received that contains a value for the **scaConversationStart** user property, the SCA  
470 runtime MUST associate a new runtime context with the given conversation identifier.

471 The SCA runtime MAY include in the message that starts the conversation the  
472 **scaConversationMaxIdleTime** user property; if this value is not present the SCA runtime MUST derive  
473 the maximum idle time for the conversation by subtracting the current time from the value of the  
474 **JMSExpiration** property, unless the **JMSExpiration** property value is zero, in which case the maximum  
475 idle time is unlimited.

476 The SCA runtime MUST consider operations invoked on or by other parties to be outside of a  
477 conversation with a given party, and MUST use different conversation identifiers if those operations are  
478 conversational.

### 479 6.5.2 Continuing a conversation

480 When creating messages for subsequent operations between the sender and receiver that are part of this  
481 conversation, the SCA runtime MUST include the **scaConversationId** user property in the JMS message,  
482 set to the conversation identifier. The SCA runtime MAY also include an updated value of the  
483 **scaConversationMaxIdleTime** property. Once a conversation has been started, the SCA runtime MUST  
484 use the initial value of the **scaCallbackDestination** user property for all messages in the conversation,  
485 and MUST ignore the value of the **scaCallbackDestination** user property in subsequent messages in the  
486 same conversation.

487 The SCA runtime MUST deal with messages received either containing a conversation identifier that does  
488 not correspond to a started conversation, or containing the **scaConversationStart** user property with a  
489 conversation identifier that matches an active conversation, by raising an error, and MUST NOT deliver  
490 such messages.

### 491 6.5.3 Ending a conversation

492 When an operation is invoked by either party that is marked as "**endsConversation**", or the maximum  
493 idle time is exceeded, then the SCA runtime MUST discard the conversation identifier and associated  
494 context after the operation has been processed. The idle time is defined as the amount of time since the  
495 SCA runtime last completed processing of an operation that is part of the conversation. There may be  
496 times when one party ends the conversation before the other does. In that case if one party does invoke  
497 an operation on the other, the SCA runtime MUST NOT deliver the message and SHOULD raise an error.

498 The SCA runtime MAY reuse conversation identifiers. In particular, the SCA runtime does not have to  
499 guarantee unique conversation identifiers and does not have to be able to identify an ended conversation  
500 indefinitely, although it MAY do so for some period after the conversation ends. Due to the long-running  
501 nature of conversations, the SCA runtime SHOULD ensure conversation context is available across  
502 server restarts, although it MAY choose to treat a server restart as implicitly ending the conversation.

---

## 503 7 Examples

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

### 507 7.1 Minimal Binding Example

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

```
510 <?xml version="1.0" encoding="ASCII"?>
511 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
512           name="MyValueComposite">
513
514     <service name="MyValueService">
515       <interface.java interface="services.myvalue.MyValueService"/>
516       <binding.jms/>
517     </service>
518
519     <reference name="StockQuoteService">
520       <interface.java interface="services.stockquote.StockQuoteService"/>
521       <binding.jms/>
522     </reference>
523 </composite>
```

### 524 7.2 URI Binding Example

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

```
527 <?xml version="1.0" encoding="ASCII"?>
528 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
529           name="MyValueComposite">
530
531     <service name="MyValueService">
532       <interface.java interface="services.myvalue.MyValueService"/>
533       <binding.jms uri="jms:MyValueServiceQueue?
534                   activationSpecName=MyValueServiceAS&
535                   ... "/>
536     </service>
537
538     <reference name="StockQuoteService">
539       <interface.java interface="services.stockquote.StockQuoteService"/>
540       <binding.jms uri="jms:StockQuoteServiceQueue?
541                   connectionFactoryName=StockQuoteServiceQCF&
542                   deliveryMode=1&
543                   ... "/>
544     </reference>
545 </composite>
```

### 546 7.3 Binding with Existing Resources Example

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

```
548 <?xml version="1.0" encoding="ASCII"?>
549 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
550           name="MyValueComposite">
551
```

```

552 <service name="MyValueService">
553   <interface.java interface="services.myvalue.MyValueService" />
554   <binding.jms>
555     <destination jndiName="MyValueServiceQ" create="never" />
556     <activationSpec jndiName="MyValueServiceAS" create="never" />
557   </binding.jms>
558 </service>
559 </composite>

```

## 560 7.4 Resource Creation Example

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

```

563 <?xml version="1.0" encoding="ASCII"?>
564 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
565   name="MyValueComposite">
566
567   <service name="MyValueService">
568     <interface.java interface="services.myvalue.MyValueService" />
569     <binding.jms>
570       <destination jndiName="MyValueServiceQueue" create="always">
571         <property name="prop1" type="string">XYZ</property>
572         <property name="destName" type="string">MyValueDest</property>
573       </destination>
574       <activationSpec jndiName="MyValueServiceAS" / create="always">
575       <resourceAdapter jndiName="com.example.JMSRA" />
576     </binding.jms>
577   </service>
578
579   <reference name="StockQuoteService">
580     <interface.java interface="services.stockquote.StockQuoteService" />
581     <binding.jms>
582       <destination jndiName="StockQuoteServiceQueue" />
583       <connectionFactory jndiName="StockQuoteServiceQCF" />
584       <resourceAdapter name="com.example.JMSRA" />
585     </binding.jms>
586   </reference>
587 </composite>

```

## 588 7.5 Request/Response Example

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

```

592 <?xml version="1.0" encoding="ASCII"?>
593 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
594   name="MyValueComposite">
595
596   <service name="MyValueService">
597     <interface.java interface="services.myvalue.MyValueService" />
598     <binding.jms correlationScheme="sca:MessageId">
599       <destination jndiName="MyValueServiceQ" create="never" />
600       <activationSpec jndiName="MyValueServiceAS" create="never" />
601     </binding.jms>
602   </service>
603
604   <reference name="StockQuoteService">
605     <interface.java interface="services.stockquote.StockQuoteService" />
606     <binding.jms correlationScheme="sca:MessageId">
607       <destination jndiName="StockQuoteServiceQueue" />
608       <connectionFactory jndiName="StockQuoteServiceQCF" />

```

```

609         <response>
610             <destination jndiName="MyValueResponseQueue" />
611             <activationSpec jndiName="MyValueResponseAS" />
612         </response>
613     </binding.jms>
614 </reference>
615 </composite>

```

## 616 7.6 Use of Predefined Definitions Example

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

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

```

620 <?xml version="1.0" encoding="ASCII"?>
621 <definitions targetNamespace="http://acme.com"
622             xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712">
623     <binding.jms name="StockQuoteService">
624         <destination jndiName="StockQuoteServiceQueue" create="never" />
625         <connectionFactory jndiName="StockQuoteServiceQCF" create="never" />
626     </binding.jms>
627 </definitions>

```

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

```

629 <?xml version="1.0" encoding="ASCII"?>
630 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
631           xmlns:acme="http://acme.com"
632           name="MyValueComposite">
633     <reference name="MyValueService">
634         <interface.java interface="services.myvalue.MyValueService" />
635         <binding.jms requestConnection="acme:StockQuoteService" />
636     </reference>
637 </composite>

```

## 638 7.7 Subscription with Selector Example

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

```

641 <?xml version="1.0" encoding="ASCII"?>
642 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
643           name="MyValueComposite">
644     <service name="MyValueService">
645         <interface.java interface="services.myvalue.MyValueService" />
646         <binding.jms>
647             <destination jndiName="MyValueServiceTopic" create="never" />
648             <connectionFactory jndiName="StockQuoteServiceTCF"
649 create="never" />
650             <subscriptionHeaders JMSSelector="Price>1000" />
651         </binding.jms>
652     </service>
653 </composite>

```

## 654 7.8 Policy Set Example

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

```

658 <policySet name="JMSPolicy"
659           provides="priority log"

```

```
660         appliesTo="binding.jms">
661
662     <intentMap provides="priority" default="medium">
663         <qualifier name="high">
664             <headers JMSPriority="9"/>
665         </qualifier>
666         <qualifier name="medium">
667             <headers JMSPriority="4"/>
668         </qualifier>
669         <qualifier name="low">
670             <headers JMSPriority="0"/>
671         </qualifier>
672     </intentMap>
673
674     <intentMap provides="log">
675         <qualifier>
676             <headers>
677                 <property name="user_example_log">logged</property>
678             </headers>
679         </qualifier>
680     </intentMap>
681 </policySet>
```

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

```
683 <reference name="StockQuoteService" requires="priority.high log">
684     <interface.java interface="services.stockquote.StockQuoteService"/>
685     <binding.jms>
686         <destination name="StockQuoteServiceQueue"/>
687         <connectionFactory name="StockQuoteServiceQCF"/>
688     </binding.jms>
689 </reference>
```

---

## 690 8 Conformance

691 Any SCA runtime that claims to support this binding MUST abide by the requirements of this specification.

692 The XML schema available at the namespace URI, defined by this specification, is considered to be  
693 authoritative and takes precedence over the XML Schema defined in the appendix of this document.

694 Within this specification, the following conformance targets are used:

- 695 • XML document elements and attributes, including binding.jms and its children, and bindingType
- 696 • The SCA runtime – this refers to the implementation that provides the functionality to support the SCA  
697 specifications, including that specific to the JMS binding as well as other SCA capabilities
- 698 • JMS objects, including Destinations, ConnectionFactories and ActivationSpecs
- 699 • WSDL documents



## A. JMS Binding Schema

```

701 <?xml version="1.0" encoding="UTF-8"?>
702 <!-- (c) Copyright OASIS 2006, 2008 -->
703 <schema xmlns="http://www.w3.org/2001/XMLSchema"
704         targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200712"
705         xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200712"
706         elementFormDefault="qualified">
707
708     <include schemaLocation="sca-core.xsd"/>
709
710     <complexType name="JMSBinding">
711         <complexContent>
712             <extension base="sca:Binding">
713                 <sequence>
714                     <choice minOccurs="0" maxOccurs="1">
715                         <sequence>
716                             <element name="destination" type="sca:JMSDestination"/>
717                             <element name="connectionFactory"
718                                     type="sca:JMSConnectionFactory"/>
719                         </sequence>
720                         <sequence>
721                             <element name="destination"
722                                     type="sca:JMSDestination" minOccurs="0"/>
723                             <element name="activationSpec" type="sca:JMSActivationSpec"/>
724                         </sequence>
725                     </choice>
726                     <element name="response" type="sca:JMSResponse" minOccurs="0"/>
727                     <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
728                     <element name="subscriptionHeaders"
729                             type="sca:JMSSubscriptionHeaders"
730                             minOccurs="0"/>
731                     <element name="resourceAdapter" type="sca:JMSResourceAdapter"
732                             minOccurs="0"/>
733                     <element name="operationProperties"
734                             type="sca:JMSOperationProperties"
735                             minOccurs="0" maxOccurs="unbounded"/>
736                     <any namespace="##other" processContents="lax"
737                             minOccurs="0" maxOccurs="unbounded"/>
738                 </sequence>
739                 <attribute name="correlationScheme" type="QName"
740                             default="sca:MessageId"/>
741                 <attribute name="initialContextFactory" type="anyURI"/>
742                 <attribute name="jndiURL" type="anyURI"/>
743                 <attribute name="requestConnection" type="QName"/>
744                 <attribute name="responseConnection" type="QName"/>
745                 <attribute name="operationProperties" type="QName"/>
746                 <anyAttribute/>
747             </extension>
748         </complexContent>
749     </complexType>
750
751     <simpleType name="CreateResource">
752         <restriction base="string">
753             <enumeration value="always"/>
754             <enumeration value="never"/>
755             <enumeration value="ifnotexist"/>
756         </restriction>
757     </simpleType>
758
759     <complexType name="JMSDestination">

```



```

760     <sequence>
761         <element name="property" type="sca:BindingProperty"
762             minOccurs="0" maxOccurs="unbounded" />
763     </sequence>
764     <attribute name="jndiName" type="anyURI" use="required" />
765     <attribute name="type" use="optional" default="queue">
766         <simpleType>
767             <restriction base="string">
768                 <enumeration value="queue" />
769                 <enumeration value="topic" />
770             </restriction>
771         </simpleType>
772     </attribute>
773     <attribute name="create" type="sca:CreateResource"
774         use="optional" default="ifnotexist" />
775 </complexType>
776
777 <complexType name="JMSConnectionFactory">
778     <sequence>
779         <element name="property" type="sca:BindingProperty"
780             minOccurs="0" maxOccurs="unbounded" />
781     </sequence>
782     <attribute name="jndiName" type="anyURI" use="required" />
783     <attribute name="create" type="sca:CreateResource"
784         use="optional" default="ifnotexist" />
785 </complexType>
786
787 <complexType name="JMSActivationSpec">
788     <sequence>
789         <element name="property" type="sca:BindingProperty"
790             minOccurs="0" maxOccurs="unbounded" />
791     </sequence>
792     <attribute name="jndiName" type="anyURI" use="required" />
793     <attribute name="create" type="sca:CreateResource"
794         use="optional" default="ifnotexist" />
795 </complexType>
796
797 <complexType name="JMSResponse">
798     <sequence>
799         <element name="destination" type="sca:JMSDestination" minOccurs="0" />
800         <choice minOccurs="0">
801             <element name="connectionFactory" type="sca:JMSConnectionFactory" />
802             <element name="activationSpec" type="sca:JMSActivationSpec" />
803         </choice>
804     </sequence>
805 </complexType>
806
807 <complexType name="JMSHeaders">
808     <sequence>
809         <element name="property" type="sca:BindingProperty"
810             minOccurs="0" maxOccurs="unbounded" />
811     </sequence>
812     <attribute name="JMSType" type="string" />
813     <attribute name="JMSDeliveryMode">
814         <simpleType>
815             <restriction base="string">
816                 <enumeration value="PERSISTENT" />
817                 <enumeration value="NON_PERSISTENT" />
818             </restriction>
819         </simpleType>
820     </attribute>
821     <attribute name="JMSTimeToLive" type="long" />
822     <attribute name="JMSPriority">
823         <simpleType>

```

```

824         <restriction base="string">
825             <enumeration value="0"/>
826             <enumeration value="1"/>
827             <enumeration value="2"/>
828             <enumeration value="3"/>
829             <enumeration value="4"/>
830             <enumeration value="5"/>
831             <enumeration value="6"/>
832             <enumeration value="7"/>
833             <enumeration value="8"/>
834             <enumeration value="9"/>
835         </restriction>
836     </simpleType>
837 </attribute>
838 </complexType>
839
840 <complexType name="JMSSubscriptionHeaders">
841     <sequence>
842         <element name="property" type="sca:BindingProperty"
843             minOccurs="0" maxOccurs="unbounded"/>
844     </sequence>
845     <attribute name="JMSSelector" type="string"/>
846 </complexType>
847
848 <complexType name="JMSResourceAdapter">
849     <sequence>
850         <element name="property" type="sca:BindingProperty"
851             minOccurs="0" maxOccurs="unbounded"/>
852     </sequence>
853     <attribute name="name" type="string" use="required"/>
854 </complexType>
855
856 <complexType name="JMSOperationProperties">
857     <sequence>
858         <element name="property" type="sca:BindingProperty"
859             minOccurs="0" maxOccurs="unbounded"/>
860         <element name="headers" type="sca:Headers"/>
861     </sequence>
862     <attribute name="name" type="string" use="required"/>
863     <attribute name="nativeOperation" type="string"/>
864 </complexType>
865
866 <complexType name="BindingProperty">
867     <simpleContent>
868         <extension base="string">
869             <attribute name="name" type="NMTOKEN"/>
870             <attribute name="type" type="string" use="optional"
871                 default="xs:string"/>
872         </extension>
873     </simpleContent>
874 </complexType>
875
876 <element name="binding.jms" type="sca:JMSBinding"
877     substitutionGroup="sca:binding"/>
878
879 <element name="wireFormat.jmsdefault" type="sca:WireFormatType"
880     substitutionGroup="sca:wireFormat"/>
881
882 <element name="operationSelector.jmsdefault" type="sca:OperationSelectorType"
883     substitutionGroup="sca:operationSelector"/>
884 </schema>

```

885

---

886 **B. Acknowledgements**

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

889 **Participants:**

890 [Participant Name, Affiliation | Individual Member]

891 [Participant Name, Affiliation | Individual Member]

892

---

## C. Non-Normative Text

894

## D. Revision History

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

896

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 <sup>th</sup> 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
--	--	--	---

897