



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.....	6
1.1	Terminology.....	6
1.2	Normative References.....	6
1.3	Non-Normative References.....	7
2	Operation Selection and Data Binding.....	8
3	Messaging Bindings.....	9
4	JMS Binding Schema.....	10
5	Default Operation Selection and Data Binding behaviorOperation Selectors and Wire Formats.....	15
5.1	Default Operation Selection.....	15
5.2	Default Data BindingWire Format.....	15
6	Policy.....	17
7	Message Exchange Patterns	18
7.1	One-way message exchange (no Callbacks)	18
7.2	Request/response message exchange (no Callbacks)	18
7.3	JMS User Properties.....	18
7.4	Callbacks	19
7.4.1	Invocation of operations on a bidirectional interface	19
7.4.2	Invocation of operations on a callback interface	19
7.4.3	Use of JMSReplyTo for callbacks for non-SCA JMS applications	19
7.5	Conversations	20
7.5.1	Starting a conversation	20
7.5.2	Continuing a conversation	20
7.5.3	Ending a conversation	20
8	Examples.....	22
8.1	Minimal Binding Example.....	22
8.2	URI Binding Example.....	22
8.3	Binding with Existing Resources Example.....	22
8.4	Resource Creation Example.....	23
8.5	Request/Response Example.....	23
8.6	Use of Predefined Definitions Example.....	24
8.7	Subscription with Selector Example	24
8.8	Policy Set Example.....	25
9	Conformance.....	26
A.	JMS Binding Schema.....	27
B.	Acknowledgements.....	31
C.	Non-Normative Text.....	32
D.	Revision History.....	33
1	Introduction.....	5
1.1	Terminology.....	5
1.2	Normative References.....	5
1.3	Non-Normative References.....	6
2	Operation Selection and Data Binding.....	7

3	Messaging Bindings.....	8
4	JMS Binding Schema.....	9
5	Default Operation Selection and Data Binding behavior.....	13
	5.1 Default Operation Selection.....	13
	5.2 Default Data Binding.....	13
6	Policy.....	14
7	Message Exchange Patterns.....	15
	7.1 One-way message exchange (no Callbacks).....	15
	7.2 Request/response message exchange (no Callbacks).....	15
	7.3 JMS User Properties.....	15
	7.4 Callbacks.....	16
	7.4.1 Invocation of operations on a bidirectional interface.....	16
	7.4.2 Invocation of operations on a callback interface.....	16
	7.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications.....	16
	7.5 Conversations.....	17
	7.5.1 Starting a conversation.....	17
	7.5.2 Continuing a conversation.....	17
	7.5.3 Ending a conversation.....	17
8	Examples.....	18
	8.1 Minimal Binding Example.....	18
	8.2 URI Binding Example.....	18
	8.3 Binding with Existing Resources Example.....	18
	8.4 Resource Creation Example.....	19
	8.5 Request/Response Example.....	19
	8.6 Use of Predefined Definitions Example.....	20
	8.7 Subscription with Selector Example.....	20
	8.8 Policy Set Example.....	20
9	Conformance.....	22
A.	JMS Binding Schema.....	23
B.	Acknowledgements.....	26
C.	Non-Normative Text.....	27
D.	Revision History.....	28

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

21

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/>
- [IETFJMS] IETF URI Scheme for Java™ Message Service 1.0 <http://www.ietf.org/internet-drafts/draft-merrick-jms-uri-05.txt>¹
- [SCA-Assembly] <http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.html>

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

36 **1.3 Non-Normative References**

37 **TBD** TBD

38 WSDL

39 **2 Messaging Bindings**

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

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

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

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

493 JMS Binding Schema

50The JMS binding element is defined by the following schema.

```
51 <binding.jms correlationScheme="QName"?
52     initialContextFactory="xs:anyURI"?
53     jndiURL="xs:anyURI"?
54     requestConnection="QName"?
55     responseConnection="QName"?
56     operationProperties="QName"?
57     name="NCName"?
58     requires="list of QName"?
59     uri="xs:anyURI"?
60     ... >
61     <destination jndiName="xs:anyURI" type="queue or topic"?
62         create="always or never or ifnotexist"?>
63         <property name="NMTOKEN" type="NMTOKEN"?>*
64     </destination?>
65     <connectionFactory jndiName="xs:anyURI"
66         create="always or never or ifnotexist"?>
67         <property name="NMTOKEN" type="NMTOKEN"?>*
68     </connectionFactory?>
69     <activationSpec jndiName="xs:anyURI"
70         create="always or never or ifnotexist"?>
71         <property name="NMTOKEN" type="NMTOKEN"?>*
72     </activationSpec?>
73
74     <response>
75         <destination jndiName="xs:anyURI" type="queue or topic"?
76             create="always or never or ifnotexist"?>
77             <property name="NMTOKEN" type="NMTOKEN"?>*
78         </destination?>
79         <connectionFactory jndiName="xs:anyURI"
80             create="always or never or ifnotexist"?>
81             <property name="NMTOKEN" type="NMTOKEN"?>*
82         </connectionFactory?>
83         <activationSpec jndiName="xs:anyURI"
84             create="always or never or ifnotexist"?>
85             <property name="NMTOKEN" type="NMTOKEN"?>*
86         </activationSpec?>
87         <wireFormat/>?
88     </response?>
89
90     <resourceAdapter name="NMTOKEN"?>
91         <property name="NMTOKEN" type="NMTOKEN"?>*
92     </resourceAdapter?>
93
94     <headers JMSType="string"?
95
96         JMSDeliveryMode="PERSISTENT or NON_PERSISTENT"?
97         JMSTimeToLive="long"?
98         JMSPriority="0 .. 9"?>
99         <property name="NMTOKEN" type="NMTOKEN"?>*
100 </headers?>
101
102     <subscriptionHeaders JMSSelector="string"?>
103         <property name="NMTOKEN" type="NMTOKEN"?>*
104 </subscriptionHeaders?>
105
```

```

106 <operationProperties name="string" nativeOperation="string"?>
107   <property name="NMTOKEN" type="NMTOKEN"?>*
108   <headers JMSType="string"?
109
110       JMSDeliveryMode="PERSISTENT or NON_PERSISTENT"?
111       JMSTimeToLive="long"?
112       JMSPriority="0 .. 9"?>
113   <property name="NMTOKEN" type="NMTOKEN"?>*
114   </headers?>
115 </operationProperties>*
116
117 <wireFormat/>?
118 <operationSelector/>?
119 </binding.jms>

```

120

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

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

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

129

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

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

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

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

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

- 207• **/binding.jms/response/connectionFactory** – identifies the connection factory that the binding uses to process response messages. The attributes of this element follow those defined for the **destination** element. A **response** element MUST NOT include both this element and an **activationSpec** element.
- 208
209
210
- 211• **/binding.jms/response/activationSpec** – identifies the activation spec that the binding uses to connect to a JMS destination to process response messages. The attributes of this element follow those defined for the **destination** element. If a response **destination** element is also specified it MUST refer to the same JMS destination as the **activationSpec**. This element MUST NOT be present when the binding is being used for an SCA service.
- 212
213
214
215
- 216• **/binding.jms/response/wireFormat** – identifies the wire format used by responses sent or received by this binding. This value overrides the **wireFormat** specified at the binding level.
- 217
- 218• **/binding.jms/headers** – this element specifies values for standard JMS headers that the SCA runtime MUST set to the given values for all operations. These values apply to requests from a reference and responses from a service.
- 219
220
- 221• **/binding.jms/headers/@JMSType, @JMSDeliveryMode, @JMSTimeToLive, @JMSPriority** – specifies the value to use for the JMS header property. The value of the **@uri** attribute MUST NOT include values for these properties if they are specified using these attributes. Valid values for **@JMSDeliveryMode** are “**PERSISTENT**” and “**NON_PERSISTENT**”; valid values for **@JMSPriority** are “**0**” to “**9**”.
- 222
223
224
225
- 226• **/binding.jms/headers/property** – specifies the value that the SCA runtime MUST set for the specified JMS user property when creating messages..
- 227
- 228• **/binding.jms/subscriptionHeaders** - this element allows JMS subscription options to be set. These values apply to a service subscribing to the destination or for a reference subscribing to the callback or reply-to destinations.
- 229
230
- 231• **/binding.jms/subscriptionHeaders/@JMSSelector** - specifies the value to use for the JMS selector. The value of the **@uri** attribute MUST NOT include values for this property if it is specified using this attribute.
- 232
233
- 234• **/binding.jms/resourceAdapter** – specifies name, type and properties of the Resource Adapter Java bean. This element MUST be present when the JMS resources are to be created for a JMS provider that implements the JCA 1.5 specification JCA15, and is ignored otherwise. SCA runtimes MAY place restrictions on the properties of the RA Java bean that can be set. For JMS providers that do not implement the JCA 1.5 specification, information necessary for resource creation can be added in provider-specific elements or attributes allowed by the extensibility of the **binding.jms** element.
- 235
236
237
238
239
- 240• **/binding.jms/operationProperties** – specifies various properties that are specific to the processing of a particular operation.
- 241
- 242• **/binding.jms/operationProperties/@name** – The name of the operation in the interface.
- 243• **/binding.jms/operationProperties/@selectedOperation** – The value generated by the **operationSelector** that corresponds to the operation in the service or reference interface identified by the **operationProperties/@name** attribute. If this attribute is omitted then the value defaults to the value of the **operationProperties/@name** attribute. The value of this attribute MUST be unique across the containing **binding.jms** element..
- 244
245
246
247
- 248• **/binding.jms/operationProperties/property** – specifies properties specific to this operation. These properties are intended to be used to parameterize the **wireFormat** identified for the binding for a particular operation. The SCA runtime SHOULD make the **operationProperties** element corresponding to the **selectedOperation** available to the **wireFormat** implementation.
- 249
250
251
- 252• **/binding.jms/operationProperties/headers** – this element specifies values for standard JMS headers that the SCA runtime MUST set to the given values for the given operation. These values apply to requests from a reference and responses from a service.
- 253
254
- 255• **/binding.jms/operationProperties/headers/@JMSType, @JMSDeliveryMode, @JMSTimeToLive, @JMSPriority** – specifies the value to use for the JMS header property. The
- 256

- 257 SCA runtime MUST use values specified for particular operations in preference to those defined for
258 all operations in the ***binding.jms/headers*** element or via the binding's ***@uri*** attribute.
- 259• ***/binding.jms/operationProperties/headers/property*** – specifies the value that the SCA runtime
260 MUST set for the specified JMS user property when creating messages.
- 261• ***/binding.jms/wireFormat*** – identifies the wire format used by requests and responses sent or
262 received by this binding.
- 263• ***/binding.jms/operationSelector*** – identifies the operation selector used when receiving requests for
264 a service. If specified for a reference this provides the default operation selector for callbacks if not
265 specified via a callback service element.
- 266• ***/binding.jms/@{any}*** - this is an extensibility mechanism to allow extensibility via attributes.
- 267• ***/binding.jms/any*** – this is an extensibility mechanism to allow extensibility via elements.
- 268Deployers/assemblers can configure ***NON_PERSISTENT*** for ***@JMSDeliveryMode*** in order to provide
269higher performance with a decreased quality of service. A ***binding.jms*** element configured in this way
270cannot satisfy either of the "***atLeastOnce***" and "***exactlyOnce***" policy intents. The SCA Runtime MUST
271raise an error for this invalid combination at deployment time.

2724 Operation Selectors and Wire Formats

273In general messaging providers deal with message formats and destinations. There is not usually a built-
274in concept of “operation” that corresponds to that defined in a WSDL portType [WSDL]. Messages have a
275wire format which corresponds in some way to the schema of an input or output message of an operation
276in the interface of a service or reference, however additional information is required in order for an SCA
277runtime to know how to identify the operation and understand the wire format of messages.

278The process of identifying the operation to be invoked is *operation selection*; the information that
279describes the contents of messages is a *wire format*. The *binding* element as described in the SCA
280Assembly specification [SCA-Assembly] provides the means to identify specific operation selection via the
281*operationSelector* element and the wire format of messages received and to be sent using the
282*wireFormat* element.

283No standard means is provided for linking the *wireFormat* or *operationSelector* elements with the
284runtime components that implement their behaviour.

285This section describes the default *operationSelector* and *wireFormat* for a JMS binding. The SCA
286runtime MUST support this default behavior, and MAY provide additional means to override it.

2874.1 Default Operation Selection

288When receiving a request at a service, or a callback at a reference, the selected operation name is
289determined as follows:

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

297The selected operation name is then mapped to an operation in the service’s interface via a matching
298*operationProperties* element in the JMS binding. If there is no matching element, the operation name is
299assumed to be the same as the selected operation name.

300The use of this operation selector can be explicitly specified in a *binding.jms* using the
301*operationSelector.jmsdefault* element; if no *operationSelector* element is specified then SCA runtimes
302MUST use this as the default.

303

3044.2 Default Wire Format

305The default wire format maps between a *JMSMessage* and the object(s) expected by the component
306implementation. We encourage component implementers to avoid exposure of JMS APIs to component
307implementations, however in the case of an existing implementation that expects a *JMSMessage*, this
308provides for simple reuse of that as an SCA component.

309The message body is mapped to the parameters or return value of the target operation as follows:

- 310• If there is a single parameter that is a *JMSMessage*, then the *JMSMessage* is passed as is.
- 311• Otherwise, the *JMSMessage* must be a JMS text message or bytes message containing XML; an
312 SCA runtime MUST be able to receive both forms. When sending messages either form may be
313 used; an SCA runtime MAY provide additional configuration to allow one or other to be selected.

- 314• If there is a single parameter, or for the return value, the JMS text or bytes XML payload is the XML
315 serialization of that parameter according to the WSDL schema for the message.
 - 316• If there are multiple parameters, then they are encoded in XML using the document wrapped style,
317 according to the WSDL schema for the message.
 - 318• When sending request messages, if there is a single parameter and the interface includes more than
319 one operation, the SCA runtime **MUST** set the JMS user property "**scaOperationName**" to the name
320 of the operation being invoked.
- 321The use of this wire format can be explicitly specified in a **binding.jms** using the **wireFormat.jmsdefault**
322element; if no **wireFormat** element is specified then SCA runtimes **MUST** use this as the default.
- 323For example, for the following interface definition:

```

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

```

353

354When the **setCoordinates** operation is invoked via a reference with a JMS binding that uses the default
355wire format, the message sent from the JMS binding is a JMS text or bytes message with the following
356content:

```

357 <setCoordinates xmlns="http://tempuri.org/coordinates">
358   <x>10</x>
359   <y>5</y>
360 </setCoordinates>

```

361 5 Policy

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

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

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

370 The *atLeastOnce*, *atMostOnce* and *ordered* intent are defined in the SCA Policy Specification document
371 in section 8, "Reliability Policy". The *conversational* intent is defined in the SCA Assembly Specification
372 document in section 8.3, "Conversational Interfaces".

373 6 Message Exchange Patterns

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

380 6.1 One-way message exchange (no Callbacks)

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

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

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

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

391 6.2 Request/response message exchange (no Callbacks)

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

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

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

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

410 6.3 JMS User Properties

411 This protocol assigns specific behavior to JMS user properties:

- 412 • "*scaCallbackDestination*" holds the name of the JMS Destination to which callback messages are
413 sent.
- 414 • "*scaConversationStart*" indicates that a conversation is to be started, its value is the identifier for the
415 conversation.

- 416• **"scaConversationMaxIdleTime"** defines the maximum time that should be allowed between
417 operations in the conversation.
- 418• **"scaConversationId"** holds the identifier for the conversation.

419**6.4 Callbacks**

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

428**6.4.1 Invocation of operations on a bidirectional interface**

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

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

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

441**6.4.2 Invocation of operations on a callback interface**

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

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

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

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

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

456**6.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications**

457 When interacting with non-SCA JMS applications, the assembler can choose to model a
458 request/response message exchange using a bidirectional interface. In this case it is likely that the non-
459 SCA JMS application does not support the use of the **scaCallbackDestination** user property. To support

460this, for one-way messages the **JMSReplyTo** header can be used to identify the destination to be used to
461deliver callback messages, as described in sections 7.4.1 and 7.4.2.

4626.5 Conversations

463A conversation is a sequence of operations between two parties that have a common context. The
464conversation can include a mixture of operations in either direction between the two parties, if the
465interface is also bidirectional. Interfaces are marked as conversational in order to ensure that the runtime
466manages the lifecycle of this context. Component implementation specifications define the manner in
467which the context that is associated with the conversation identifier is made available to component
468implementations.

4696.5.1 Starting a conversation

470A conversation is started when an operation is invoked on a conversational interface and there is no
471active conversation with the target of the invocation. When this happens the SCA runtime MUST supply
472an identifier for the conversation, if the client component has not already supplied an identifier, and the
473SCA runtime MUST set the **scaConversationStart** user property to this value in the JMS message that it
474sends for the request, and associate a new runtime context with this conversation identifier.

475When a message is received that contains a value for the **scaConversationStart** user property, the SCA
476runtime MUST associate a new runtime context with the given conversation identifier.

477The SCA runtime MAY include in the message that starts the conversation the
478**scaConversationMaxIdleTime** user property; if this value is not present the SCA runtime MUST derive
479the maximum idle time for the conversation by subtracting the current time from the value of the
480**JMSExpiration** property, unless the **JMSExpiration** property value is zero, in which case the maximum
481idle time is unlimited.

482The SCA runtime MUST consider operations invoked on or by other parties to be outside of a
483conversation with a given party, and MUST use different conversation identifiers if those operations are
484conversational.

4856.5.2 Continuing a conversation

486When creating messages for subsequent operations between the sender and receiver that are part of this
487conversation, the SCA runtime MUST include the **scaConversationId** user property in the JMS message,
488set to the conversation identifier. The SCA runtime MAY also include an updated value of the
489**scaConversationMaxIdleTime** property. Once a conversation has been started, the SCA runtime MUST
490use the initial value of the **scaCallbackDestination** user property for all messages in the conversation,
491and MUST ignore the value of the **scaCallbackDestination** user property in subsequent messages in the
492same conversation.

493The SCA runtime MUST deal with messages received either containing a conversation identifier that does
494not correspond to a started conversation, or containing the **scaConversationStart** user property with a
495conversation identifier that matches an active conversation, by raising an error, and MUST NOT deliver
496such messages.

4976.5.3 Ending a conversation

498When an operation is invoked by either party that is marked as "**endsConversation**", or the maximum
499idle time is exceeded, then the SCA runtime MUST discard the conversation identifier and associated
500context after the operation has been processed. The idle time is defined as the amount of time since the
501SCA runtime last completed processing of an operation that is part of the conversation. There may be
502times when one party ends the conversation before the other does. In that case if one party does invoke
503an operation on the other, the SCA runtime MUST NOT deliver the message and SHOULD raise an error.

504The SCA runtime MAY reuse conversation identifiers. In particular, the SCA runtime does not have to
505guarantee unique conversation identifiers and does not have to be able to identify an ended conversation
506indefinitely, although it MAY do so for some period after the conversation ends. Due to the long-running

507nature of conversations, the SCA runtime SHOULD ensure conversation context is available across
508server restarts, although it MAY choose to treat a server restart as implicitly ending the conversation.

5097 Examples

510The following snippets show the *sca.composite* file for the *MyValueComposite* file containing the
511*service* element for the MyValueService and a *reference* element for the StockQuoteService. Both the
512service and the reference use a JMS binding.

5137.1 Minimal Binding Example

514The following example shows the JMS binding being used with no further attributes or elements. In this
515case, it is left to the deployer to identify the resources to which the binding is connected.

```
516 <?xml version="1.0" encoding="ASCII"?>
517 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
518           name="MyValueComposite">
519
520     <service name="MyValueService">
521       <interface.java interface="services.myvalue.MyValueService"/>
522       <binding.jms/>
523     </service>
524
525     <reference name="StockQuoteService">
526       <interface.java interface="services.stockquote.StockQuoteService"/>
527       <binding.jms/>
528     </reference>
529 </composite>
```

5307.2 URI Binding Example

531The following example shows the JMS binding using the *@uri* attribute to specify the connection type and
532its information:

```
533 <?xml version="1.0" encoding="ASCII"?>
534 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
535           name="MyValueComposite">
536
537     <service name="MyValueService">
538       <interface.java interface="services.myvalue.MyValueService"/>
539       <binding.jms uri="jms:MyValueServiceQueue?
540                   activationSpecName=MyValueServiceAS&
541                   ... "/>
542     </service>
543
544     <reference name="StockQuoteService">
545       <interface.java interface="services.stockquote.StockQuoteService"/>
546       <binding.jms uri="jms:StockQuoteServiceQueue?
547                   connectionFactoryName=StockQuoteServiceQCF&
548                   deliveryMode=1&
549                   ... "/>
550     </reference>
551 </composite>
```

5527.3 Binding with Existing Resources Example

553The following example shows the JMS binding using existing resources:

```
554 <?xml version="1.0" encoding="ASCII"?>
555 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
```

```

556         name="MyValueComposite">
557
558     <service name="MyValueService">
559         <interface.java interface="services.myvalue.MyValueService"/>
560         <binding.jms>
561             <destination jndiName="MyValueServiceQ" create="never"/>
562             <activationSpec jndiName="MyValueServiceAS" create="never"/>
563         </binding.jms>
564     </service>
565 </composite>

```

5667.4 Resource Creation Example

567The following example shows the JMS binding providing information to create JMS resources rather than
568using existing ones:

```

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

```

5947.5 Request/Response Example

595The following example shows the JMS binding using existing resources to support request/response
596operations. The service uses the **JMSReplyTo** destination to send response messages, and does not
597specify a response queue:

```

598 <?xml version="1.0" encoding="ASCII"?>
599 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
600     name="MyValueComposite">
601
602     <service name="MyValueService">
603         <interface.java interface="services.myvalue.MyValueService"/>
604         <binding.jms correlationScheme="sca:MessageId">
605             <destination jndiName="MyValueServiceQ" create="never"/>
606             <activationSpec jndiName="MyValueServiceAS" create="never"/>
607         </binding.jms>
608     </service>
609
610     <reference name="StockQuoteService">

```

```

611     <interface.java interface="services.stockquote.StockQuoteService"/>
612     <binding.jms correlationScheme="sca:MessageId">
613         <destination jndiName="StockQuoteServiceQueue"/>
614         <connectionFactory jndiName="StockQuoteServiceQCF"/>
615         <response>
616             <destination jndiName="MyValueResponseQueue"/>
617             <activationSpec jndiName="MyValueResponseAS"/>
618         </response>
619     </binding.jms>
620 </reference>
621 </composite>

```

6227.6 Use of Predefined Definitions Example

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

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

```

626 <?xml version="1.0" encoding="ASCII"?>
627 <definitions targetNamespace="http://acme.com"
628     xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712">
629     <binding.jms name="StockQuoteService">
630         <destination jndiName="StockQuoteServiceQueue" create="never"/>
631         <connectionFactory jndiName="StockQuoteServiceQCF" create="never"/>
632     </binding.jms>
633 </definitions>

```

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

```

635 <?xml version="1.0" encoding="ASCII"?>
636 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
637     xmlns:acme="http://acme.com"
638     name="MyValueComposite">
639     <reference name="MyValueService">
640         <interface.java interface="services.myvalue.MyValueService"/>
641         <binding.jms requestConnection="acme:StockQuoteService"/>
642     </reference>
643 </composite>

```

6447.7 Subscription with Selector Example

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

```

647 <?xml version="1.0" encoding="ASCII"?>
648 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
649     name="MyValueComposite">
650     <service name="MyValueService">
651         <interface.java interface="services.myvalue.MyValueService"/>
652         <binding.jms>
653             <destination jndiName="MyValueServiceTopic" create="never"/>
654             <connectionFactory jndiName="StockQuoteServiceTCF" create="never"/>
655         >
656             <subscriptionHeaders JMSSelector="Price>1000"/>
657         </binding.jms>
658     </service>
659 </composite>

```

6607.8 Policy Set Example

661A policy set defines the manner in which intents map to JMS binding properties. The following illustrates
662an example of a policy set that defines values for the **@JMSpriority** attribute using the “**priority**” intent,
663and also allows setting of a value for a user JMS property using the “**log**” intent.

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

688Given this policy set, the intents can be required on a service or reference:

```
689 <reference name="StockQuoteService" requires="priority.high log">
690     <interface.java interface="services.stockquote.StockQuoteService"/>
691     <binding.jms>
692         <destination name="StockQuoteServiceQueue"/>
693         <connectionFactory name="StockQuoteServiceQCF"/>
694     </binding.jms>
695 </reference>
```

696 **8 Conformance**

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

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

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

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

706 A. JMS Binding Schema

```
707<?xml version="1.0" encoding="UTF-8"?>
708<!-- (c) Copyright OASIS 2006, 2008 -->
709<schema xmlns="http://www.w3.org/2001/XMLSchema"
710       targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200712"
711       xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200712"
712       elementFormDefault="qualified">
713
714   <include schemaLocation="sca-core.xsd"/>
715
716   <complexType name="JMSBinding">
717     <complexContent>
718       <extension base="sca:Binding">
719         <sequence>
720           <choice minOccurs="0" maxOccurs="1">
721             <sequence>
722               <element name="destination" type="sca:JMSDestination"/>
723               <element name="connectionFactory"
724                 type="sca:JMSConnectionFactory"/>
725             </sequence>
726             <sequence>
727               <element name="destination"
728                 type="sca:JMSDestination" minOccurs="0"/>
729               <element name="activationSpec" type="sca:JMSActivationSpec"/>
730             </sequence>
731           </choice>
732
733           <element name="response" type="sca:JMSResponse" minOccurs="0"/>
734           <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
735           <element name="subscriptionHeaders"
736             type="sca:JMSSubscriptionHeaders"
737             minOccurs="0"/>
738           <element name="resourceAdapter" type="sca:JMSResourceAdapter"
739             minOccurs="0"/>
740           <element name="operationProperties"
741             type="sca:JMSOperationProperties"
742             minOccurs="0" maxOccurs="unbounded"/>
743           <any namespace="##other" processContents="lax"
744             minOccurs="0" maxOccurs="unbounded"/>
745         </sequence>
746         <attribute name="correlationScheme" type="QName"
747           default="sca:MessageId"/>
748         <attribute name="initialContextFactory" type="anyURI"/>
749         <attribute name="jndiURL" type="anyURI"/>
750         <attribute name="requestConnection" type="QName"/>
751         <attribute name="responseConnection" type="QName"/>
752         <attribute name="operationProperties" type="QName"/>
753       </anyAttribute/>
754     </extension>
755   </complexContent>
756 </complexType>
757
758 <simpleType name="CreateResource">
759   <restriction base="string">
760     <enumeration value="always"/>
761     <enumeration value="never"/>
762     <enumeration value="ifnotexist"/>
763   </restriction>
764 </simpleType>
765
```

```

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

```

```

829     </simpleType>
830 </attribute>
831 <attribute name="JMSTimeToLive" type="long"/>
832 <attribute name="JMSPriority">
833   <simpleType>
834     <restriction base="string">
835       <enumeration value="0"/>
836       <enumeration value="1"/>
837       <enumeration value="2"/>
838       <enumeration value="3"/>
839       <enumeration value="4"/>
840       <enumeration value="5"/>
841       <enumeration value="6"/>
842       <enumeration value="7"/>
843       <enumeration value="8"/>
844       <enumeration value="9"/>
845     </restriction>
846   </simpleType>
847 </attribute>
848 </complexType>
849
850 <complexType name="JMSSubscriptionHeaders">
851   <sequence>
852     <element name="property" type="sca:BindingProperty"
853       minOccurs="0" maxOccurs="unbounded"/>
854   </sequence>
855   <attribute name="JMSSelector" type="string"/>
856 </complexType>
857
858 <complexType name="JMSResourceAdapter">
859   <sequence>
860     <element name="property" type="sca:BindingProperty"
861       minOccurs="0" maxOccurs="unbounded"/>
862   </sequence>
863   <attribute name="name" type="string" use="required"/>
864 </complexType>
865
866 <complexType name="JMSOperationProperties">
867   <sequence>
868     <element name="property" type="sca:BindingProperty"
869       minOccurs="0" maxOccurs="unbounded"/>
870     <element name="headers" type="sca:Headers"/>
871   </sequence>
872   <attribute name="name" type="string" use="required"/>
873   <attribute name="nativeOperation" type="string"/>
874 </complexType>
875
876 <complexType name="BindingProperty">
877   <simpleContent>
878     <extension base="string">
879       <attribute name="name" type="NMTOKEN"/>
880       <attribute name="type" type="string" use="optional"
881         default="xs:string"/>
882     </extension>
883   </simpleContent>
884 </complexType>
885
886 <element name="binding.jms" type="sca:JMSBinding"
887   substitutionGroup="sca:binding"/>
888
889 <element name="wireFormat.jmsdefault" type="sca:WireFormatType"
890   substitutionGroup="sca:wireFormat"/>
891

```

```
892 <element name="operationSelector.jmsdefault" type="sca:OperationSelectorType"
893 substitutionGroup="sca:operationSelector"/>
894</schema>
```

895

896 **B. Acknowledgements**

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

899 **Participants:**

900 [Participant Name, Affiliation | Individual Member]

901 [Participant Name, Affiliation | Individual Member]

902

903 **C. Non-Normative Text**

904 D. Revision History

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

906

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
--	--	--	---

907