



Service Component Architecture JMS Binding Specification Version 1.1

Committee Draft 03 Revision 1 plus proposed changes for issues 95, 105 and 106

25 January 2010

Specification URIs:

This Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd03.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd03.doc>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd03.pdf>
(Authoritative)

Previous Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd03.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd03.doc>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd03.pdf>
(Authoritative)

Latest Version:

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

Latest Approved Version:

Technical Committee:

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

Chair(s):

Simon Holdsworth, IBM

Editor(s):

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

Related work:

This specification replaces or supersedes:

- Service Component Architecture JMS Binding Specification Version 1.00, March 21 2007

This specification is related to:

- OASIS Committee Draft 03, "Service Component Architecture Assembly Model Specification Version 1.1", March 2009
<http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd03.pdf>
- OASIS Committee Draft 02, "SCA Policy Framework Version 1.1", February 2009
<http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd02.pdf>

Declared XML Namespace(s):

<http://docs.oasis-open.org/ns/opencsa/sca/200903>

Abstract:

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

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

The messaging binding describes a common pattern of behavior that may be followed by messaging-related bindings, including the JMS binding. In particular it describes the manner in which operations are selected based on message content, and the manner in which messages are mapped into the runtime representation. These are specified in a language-neutral manner. The JMS binding provides JMS-specific details of the connection to the required JMS resources. It supports the use of Queue and Topic type destinations.

Status:

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

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

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

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

Notices

Copyright © OASIS® 2006, 2010. All Rights Reserved.

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

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

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

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

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

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

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

The names "OASIS", [insert specific trademarked names and abbreviations here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

Table of Contents

1	Introduction	5
1.1	Terminology	5
1.2	Normative References	5
1.3	Non-Normative References	6
1.4	Naming Conventions	6
2	Messaging Bindings	7
3	JMS Binding Schema	8
3.1	Extensibility	12
4	Operation Selectors and Wire Formats	12
4.1	Default Operation Selection	14
4.2	Default Wire Format	15
4.2.1	Example of default wire format	15
5	Policy	17
6	Message Exchange Patterns	18
6.1	One-way message exchange (no Callbacks)	18
6.2	Request/response message exchange (no Callbacks)	18
6.3	JMS User Properties	19
6.4	Callbacks	19
6.4.1	Invocation of operations on a bidirectional interface	19
6.4.2	Invocation of operations on a callback interface	20
6.4.3	Use of JMSReplyTo for callbacks for non-SCA JMS applications	20
7	Examples	21
7.1	Minimal Binding Example	21
7.2	URI Binding Example	21
7.3	Binding with Existing Resources Example	21
7.4	Resource Creation Example	22
7.5	Request/Response Example	22
7.6	Subscription with Selector Example	23
7.7	Policy Set Example	23
8	Conformance	25
8.1	SCA JMS Binding XML Document	25
8.2	SCA Runtime	25
A.	JMS XML Binding Schema: sca-binding-jms-1.1.xsd	26
B.	Conformance Items	29
C.	Acknowledgements	34
D.	Revision History	35

1 Introduction

This document defines the concept and behavior of a messaging binding, and a concrete [Java Message Service \[JMS\]](#) based binding that provides that behavior. The binding specified in this document applies to an SCA composite's services and references. The binding is especially well suited for use by services and references of composites that are directly deployed, as opposed to composites that are used as implementations of higher-level components. Services and references of deployed composites become system-level services and references, which are intended to be used by non-SCA clients.

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

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

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC Keywords \[RFC2119\]](#).

This specification uses predefined namespace prefixes throughout; they are given in the following list. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 1-1 Prefixes and Namespaces used in this specification

Prefix	Namespace	Notes
xs	"http://www.w3.org/2001/XMLSchema"	Defined by XML Schema 1.0 specification
sca	"http://docs.oasis-open.org/ns/opencsa/sca/200903"	Defined by the SCA specifications

1.2 Normative References

- [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- [JMS] Java™ Message Service Specification v1.1 <http://java.sun.com/products/jms/>
- [WSDL] E. Christensen et al, *Web Service Description Language (WSDL) 1.1*, <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, W3C Note, March 15 2001.
- R. Chinnici et al, *Web Service Description Language (WSDL) Version 2.0 Part 1: Core Language*, <http://www.w3.org/TR/2007/REC-wsdl20-20070626/>, W3C Recommendation, June 26 2007.
- [JCA15] J2EE Connector Architecture Specification Version 1.5 <http://java.sun.com/j2ee/connector/>

[IETFJMS]	M. Phillips, P. Easton, D. Rokicki, E. Johnson, <i>URI Scheme for Java™ Message Service 1.0</i> http://www.ietf.org/draft-merrick-jms-uri-06.txt , IETF Internet-Draft June 2009 ¹
[SCA-Assembly]	OASIS Committee Draft 03, "Service Component Architecture Assembly Model Specification Version 1.1", March 2009 http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd03.pdf
[SCA-Policy]	OASIS Committee Draft 02, "SCA Policy Framework Specification Version 1.1", February 2009 http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd02.pdf

1.3 Non-Normative References

TBD	TBD
------------	-----

1.4 Naming Conventions

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

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

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

2 Messaging Bindings

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

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

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

In addition, each operation may have specific properties defined, that may influence the way native 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"?
  name="NCName"?
  requires="list of QName"?
  policySets="list of QName"?
  uri="xs:anyURI"? >
  <destination jndiName="xs:anyURI"? type="queue or topic"?
    create="always or never or ifNotExist"?>
    <property name="NMTOKEN" type="NMTOKEN"?>*
  </destination>?
  <connectionFactory jndiName="xs:anyURI"?
    create="always or never or ifNotExist"?>
    <property name="NMTOKEN" type="NMTOKEN"?>*
  </connectionFactory>?
  <activationSpec jndiName="xs:anyURI"?
    create="always or never or ifNotExist"?>
    <property name="NMTOKEN" type="NMTOKEN"?>*
  </activationSpec>?

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

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

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

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

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



```

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

```

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

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

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

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

The following illustrates the structure of the URI and the set of property names that have specific semantics:

```

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

```

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

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

- **/binding.jms/@name** – as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- **/binding.jms/@requires** – as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- **/binding.jms/@policySets** – as defined in the [SCA Assembly Specification \[SCA-Assembly\]](#).
- **/binding.jms/@correlationScheme** – identifies the correlation scheme used when sending reply or callback messages, default value is "sca:messageID".
If the value of the **@correlationScheme** attribute is "sca:messageID" the SCA runtime MUST set the correlation ID of replies to the message ID of the corresponding request [BJM30003].
If the value of the **@correlationScheme** attribute is "sca:correlationID" the SCA runtime MUST set the correlation ID of replies to the correlation ID of the corresponding request [BJM30004].
If the value of the **@correlationScheme** attribute is "sca:none" the SCA runtime MUST NOT set the correlation ID [BJM30005].
SCA runtimes MAY allow other values of the **@correlationScheme** attribute to indicate other correlation schemes [BJM30006].
- **/binding.jms/@initialContextFactory** – the name of the JNDI initial context factory.
- **/binding.jms/@jndiURL** – the URL for the JNDI provider.
- **/binding.jms/destination** – identifies the destination that is to be used to process requests by this binding.

Deleted: - all other property names are treated as user property names

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

- 178 • **/binding.jms/destination/@type** - the type of the request destination. Valid values are “**queue**” and
179 “**topic**”. The default value is “**queue**”.
180 Whatever the value of the **destination/@type** attribute, the runtime MUST ensure a single response
181 is delivered for request/response operations [BJM30010].
- 182 • **binding.jms/destination/@jndiName** – the JNDI name of the JMS Destination that the binding uses
183 to send or receive messages. The behaviour of this attribute is determined by the value of the
184 **@create** attribute as follows:
 - 185 – If the **@create** attribute value for a destination, connectionFactory or activationSpec element is
186 “**always**” and the **@jndiName** attribute is present and the resource cannot be created at the
187 location specified by the **@jndiName** attribute then the SCA runtime MUST raise an error
188 [BJM30011].
189 If the **@create** attribute value for a **destination**, **connectionFactory** or **activationSpec** element
190 is “**always**” and the **@jndiName** attribute is not present and the resource cannot be created, then
191 the SCA runtime MUST raise an error [BJM30037].
192 If the **@jndiName** attribute is omitted this specification places no restriction on the JNDI location
193 of the created resource.
 - 194 – If the **@create** attribute value for a destination, connectionFactory or activationSpec element is
195 “**ifNotExist**” then the **@jndiName** attribute MUST specify the location of the possibly existing
196 resource [BJM30012].
197 If the destination, connectionFactory or activationSpec does not exist at the location identified by
198 the **@jndiName** attribute, but cannot be created there then the SCA runtime MUST raise an error
199 [BJM30013].
200 If the destination, connectionFactory or activationSpec’s **@jndiName** attribute refers to an
201 existing resource that is not a JMS Destination of the appropriate type, a JMS connection factory
202 or a JMS activation spec respectively then the SCA runtime MUST raise an error [BJM30014].
 - 203 – If the **@create** attribute value for a destination, connectionFactory or activationSpec element is
204 “**never**” then the **@jndiName** attribute MUST specify the location of the existing resource
205 [BJM30015].
206 If the destination, connection factory or activation spec is not present at the location identified by
207 the **@jndiName** attribute, or the location refers to a resource of an incorrect type then the SCA
208 runtime MUST raise an error [BJM30016].
- 209 • **/binding.jms/destination/@create** – indicates whether the destination should be created when the
210 containing composite is deployed. Valid values are “**always**”, “**never**” and “**ifNotExist**”. The default
211 value is “**ifNotExist**”.
- 212 • **/binding.jms/destination/property** – defines properties to be used to create the destination, if
213 required.
- 214 • **/binding.jms/connectionFactory** – identifies the connection factory that the binding uses to process
215 request messages. The attributes of this element follow the rules defined for the **destination**
216 element.
217 A **binding.jms** element MUST NOT include both a **connectionFactory** element and an
218 **activationSpec** element [BJM30017].
219 When the **connectionFactory** element is present, then the destination MUST be defined either by
220 the **destination** element or the **@uri** attribute [BJM30018].
- 221 • **/binding.jms/activationSpec** – identifies the activation spec that the binding uses to connect to a
222 JMS destination to process request messages. The attributes of this element follow the rules defined
223 for the **destination** element.
224 If the **activationSpec** element is present and the destination is also specified via a **destination**
225 element or the **@uri** attribute then it MUST refer to the same JMS destination as the **activationSpec**
226 [BJM30019].
227 The **activationSpec** element MUST NOT be present when the binding is being used for an SCA
228 reference [BJM30020].
- 229 • **/binding.jms/response** – defines the resources used for handling response messages (receiving
230 responses for a reference, and sending responses from a service).

- 231 • **/binding.jms/response/destination** – identifies the destination that is to be used to process
 232 responses by this binding. Attributes follow the rules defined for the parent's **destination** element.
 233 For a service, this destination is used to send responses to messages that have a null value for the
 234 **JMSReplyTo** destination. For a reference, this destination is used to receive reply messages
- 235 • **/binding.jms/response/connectionFactory** – identifies the connection factory that the binding uses
 236 to process response messages. The attributes of this element follow those defined for the
 237 **destination** element.
 238 A **response** element MUST NOT include both a **connectionFactory** element and an **activationSpec**
 239 element [BJM30021].
- 240 • **/binding.jms/response/activationSpec** – identifies the activation spec that the binding uses to
 241 connect to a JMS destination to process response messages. The attributes of this element follow
 242 those defined for the **destination** element.
 243 If a **response/destination** and **response/activationSpec** element are both specified they MUST
 244 refer to the same JMS destination [BJM30022].
 245 The **response/activationSpec** element MUST NOT be present when the binding is being used for an
 246 SCA service [BJM30023].
- 247 • **/binding.jms/response/wireFormat** – identifies the wire format used by responses sent or received
 248 by this binding. This value overrides the **wireFormat** specified at the binding level. Wire formats for
 249 this binding are described in Section 3.2.
- 250 • **/binding.jms/headers** – this element specifies values to be set for standard JMS headers. These
 251 values apply to requests from a reference and responses from a service. Section 3.2 defines the
 252 priority rules for determining the values for JMS headers and user properties.
- 253 • **/binding.jms/headers/@type, @deliveryMode, @timeToLive, @priority** – specifies the value to
 254 use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive or JMSPriority
 255 respectively. Valid values for @deliveryMode are "persistent" and "nonpersistent" with "persistent"
 256 being the default; valid values for @priority are "0" to "9", with "4" being the default; valid values for
 257 @timeToLive are positive integers, with 0 indicating unlimited time and being the default value.
- 258 • **/binding.jms/headers/property** – specifies the value and type for the given JMS user property.
- 259 • **/binding.jms/messageSelection** - this element allows JMS message selection options to be set.
 260 These values apply to a service receiving messages from the request destination or for a reference
 261 receiving messages from the callback or reply-to destination.
- 262 • **/binding.jms/messageSelection/@selector** - specifies the value to use for the JMS message
 263 selector. Section 3.3 defines the priority rules for determining the values for the message selector.
- 264 • **/binding.jms/resourceAdapter** – specifies name, type and properties of the Resource Adapter Java
 265 bean.
 266 The **resourceAdapter** element MUST be present when JMS resources are to be created for a JMS
 267 provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise
 268 [BJM30031].
 269 SCA runtimes MAY place restrictions on the properties of the resource adapter Java bean that can be
 270 set using the **resourceAdapter** element [BJM30028].
 271 For JMS providers that do not implement the JCA 1.5 specification [JCA15], information necessary for
 272 resource creation can be added in provider-specific elements or attributes allowed by the extensibility
 273 of the **binding.jms** element.
- 274 • **/binding.jms/operationProperties** – specifies various properties that are specific to the processing
 275 of a particular operation.
- 276 • **/binding.jms/operationProperties/@name** – The name of the operation in the interface.
- 277 • **/binding.jms/operationProperties/@selectedOperation** – The value generated by the
 278 **operationSelector** that corresponds to the operation in the service or reference interface identified
 279 by the **operationProperties/@name** attribute. If this attribute is omitted then the value defaults to
 280 the value of the **operationProperties/@name** attribute.
 281 The value of the **operationProperties/@selectedOperation** attribute MUST be unique across the
 282 containing **binding.jms** element [BJM30029].

Deleted: .
 The SCA runtime MUST set JMS headers in messages that it creates to the values specified by the **headers** element unless overridden for the operation being invoked. [BJM30024].

Deleted: .
 If the **@uri** attribute includes values for the type, delivery mode, time to live or priority properties then the **@uri** values are used and the **headers** and **operationProperties/headers @type, @deliveryMode, @timeToLive** or **@priority** attributes are ignored [BJM30025].

Deleted: .
 For each **header/properties** element the SCA runtime MUST set the named JMS user property to the given value in messages it creates unless overridden for the operation being invoked [BJM30026].

Deleted: If the **@uri** attribute includes a value for the message selector then the **@uri** value is used and the **messageSelection/@selector** attribute is ignored [BJM30027].

- **/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 [BJM30030].
 - **/binding.jms/operationProperties/headers** – this element specifies values **to be set** for standard JMS headers. These values apply to requests from a reference and responses from a service.
Section 3.2 defines the priority rules for determining the values for JMS headers and user properties.
 - **/binding.jms/operationProperties/headers/@type, @deliveryMode, @timeToLive, @priority** – specifies the value to use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive or JMSPriority, respectively,
 - **/binding.jms/operationProperties/headers/property** – specifies the value **and type** for the given JMS user property.
 - **/binding.jms/wireFormat** – identifies the wire format used by requests and responses sent or received by this binding. Wire formats for this binding are described in Section 4.
 - **/binding.jms/operationSelector** – identifies the operation selector used when receiving requests for a service. If specified for a reference this provides the default operation selector for callbacks if not specified via a callback service element. Operation selectors for this binding are described in Section 3.2.
- The **binding.jms** element MUST conform to the XML schema defined in **sca-binding-jms.xsd** [BJM30036].

Deleted: .
The SCA runtime MUST set JMS headers in messages it creates when the operation identified by the **operationProperties/@name** attribute is invoked to the values specified by the corresponding **operationProperties/headers** element [BJM30032].

Deleted: .
For each **operationProperties/headers/property** element the SCA runtime MUST set the named JMS user property to the given value in messages it creates when the operation identified by the **operationProperties/@name** attribute is invoked [BJM30033].

Deleted: 4

Formatted: Bullets and Numbering

3.1 Extensibility

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

3.2 JMS Message Headers and User Properties

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

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

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

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

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

3.3 JMS Message Selection

Message selectors allow the JMS binding to receive a specific subset of messages from a given destination, such that only messages that match the selector are delivered to a given JMS binding. This allows more than one JMS binding to share a destination.

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

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

4 Operation Selectors and Wire Formats

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

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

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

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

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

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

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

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

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

Comment [SAJH1]: Note that this reference is broken in CD03-rev1

Deleted: Error! Not a valid bookmark self-reference. When a message is received at an SCA service with JMS binding, the SCA runtime MUST invoke the target component using the resolved operation name

4.1 Default Operation Selection

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

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

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

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

4.2 Default Wire Format

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

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

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

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

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

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

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

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

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

4.2.1 Example of default wire format

For the following interface definition:

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

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

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

```
435     <wsdl:operation name="setCoordinates">
436         <wsdl:input message="tns:setCoordinatesRequestMsg"
437         name="setCoordinatesRequest"/>
438     </wsdl:operation>
439 </wsdl:portType>
440 </wsdl:definitions>
```

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

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

5 Policy

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

JMS binding implementations MUST support the JMS intent [BJM50001]. The JMS intent MUST always be included in the @alwaysProvides attribute of the JMS bindingType [BJM50002]

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

- atLeastOnce
- atMostOnce
- ordered

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

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

6 Message Exchange Patterns

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

6.1 One-way message exchange (no Callbacks)

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

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

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

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

6.2 Request/response message exchange (no Callbacks)

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

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

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

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

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

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

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

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

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

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

520 6.3 JMS User Properties

521 This protocol assigns specific behavior to JMS user properties:

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

525 6.4 Callbacks

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

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

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

538 6.4.1 Invocation of operations on a bidirectional interface

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

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

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

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

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

555 6.4.2 Invocation of operations on a callback interface

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

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

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

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

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

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

573 6.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications

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

7 Examples

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

7.1 Minimal Binding Example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
  name="MyValueComposite">

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

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

7.2 URI Binding Example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
  name="MyValueComposite">

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

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

7.3 Binding with Existing Resources Example

The following example shows the JMS binding using existing resources:

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
  name="MyValueComposite">
```

```

629     <service name="MyValueService">
630         <interface.java interface="services.myvalue.MyValueService"/>
631         <binding.jms>
632             <destination jndiName="MyValueServiceQ" create="never"/>
633             <activationSpec jndiName="MyValueServiceAS" create="never"/>
634         </binding.jms>
635     </service>
636 </composite>

```

7.4 Resource Creation Example

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

```

640 <?xml version="1.0" encoding="UTF-8"?>
641 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
642     name="MyValueComposite">
643
644     <service name="MyValueService">
645         <interface.java interface="services.myvalue.MyValueService"/>
646         <binding.jms>
647             <destination jndiName="MyValueServiceQueue" create="always">
648                 <property name="prop1" type="string">XYZ</property>
649                 <property name="destName" type="string">MyValueDest</property>
650             </destination>
651             <activationSpec jndiName="MyValueServiceAS" create="always"/>
652             <resourceAdapter jndiName="com.example.JMSRA"/>
653         </binding.jms>
654     </service>
655
656     <reference name="StockQuoteService">
657         <interface.java interface="services.stockquote.StockQuoteService"/>
658         <binding.jms>
659             <destination jndiName="StockQuoteServiceQueue"/>
660             <connectionFactory jndiName="StockQuoteServiceQCF"/>
661             <resourceAdapter name="com.example.JMSRA"/>
662         </binding.jms>
663     </reference>
664 </composite>

```

7.5 Request/Response Example

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

```

669 <?xml version="1.0" encoding="UTF-8"?>
670 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
671     name="MyValueComposite">
672
673     <service name="MyValueService">
674         <interface.java interface="services.myvalue.MyValueService"/>
675         <binding.jms correlationScheme="sca:messageID">
676             <destination jndiName="MyValueServiceQ" create="never"/>
677             <activationSpec jndiName="MyValueServiceAS" create="never"/>
678         </binding.jms>
679     </service>
680
681     <reference name="StockQuoteService">
682         <interface.java interface="services.stockquote.StockQuoteService"/>
683         <binding.jms correlationScheme="sca:messageID">
684             <destination jndiName="StockQuoteServiceQueue"/>
685             <connectionFactory jndiName="StockQuoteServiceQCF"/>

```

```

686         <response>
687             <destination jndiName="MyValueResponseQueue"/>
688             <activationSpec jndiName="MyValueResponseAS"/>
689         </response>
690     </binding.jms>
691 </reference>
692 </composite>

```

7.6 Subscription with Selector Example

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

```

696 <?xml version="1.0" encoding="UTF-8"?>
697 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
698     name="MyValueComposite">
699     <service name="MyValueService">
700         <interface.java interface="services.myvalue.MyValueService"/>
701         <binding.jms>
702             <destination jndiName="MyValueServiceTopic" create="never"/>
703             <connectionFactory jndiName="StockQuoteServiceTCF"
704 create="never"/>
705             <messageSelection selector="Price>1000"/>
706         </binding.jms>
707     </service>
708 </composite>

```

7.7 Policy Set Example

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

```

713 <policySet name="JMSPolicy"
714     provides="priority log"
715     appliesTo="binding.jms">
716
717     <intentMap provides="priority" default="medium">
718         <qualifier name="high">
719             <headers priority="9"/>
720         </qualifier>
721         <qualifier name="medium">
722             <headers priority="4"/>
723         </qualifier>
724         <qualifier name="low">
725             <headers priority="0"/>
726         </qualifier>
727     </intentMap>
728
729     <intentMap provides="log">
730         <qualifier>
731             <headers>
732                 <property name="user_example_log">logged</property>
733             </headers>
734         </qualifier>
735     </intentMap>
736 </policySet>

```

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

```

738 <reference name="StockQuoteService" requires="priority,high log">
739     <interface.java interface="services.stockquote.StockQuoteService"/>
740     <binding.jms>

```

```
741         <destination name="StockQuoteServiceQueue"/>
742         <connectionFactory name="StockQuoteServiceQCF"/>
743     </binding.jms>
744 </reference>
```

8 Conformance

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

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

8.1 SCA JMS Binding XML Document

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

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

8.2 SCA Runtime

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

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

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

```

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

```

```

828         <enumeration value="topic"/>
829     </restriction>
830 </simpleType>
831 </attribute>
832 <attribute name="create" type="sca:JMSCreateResource"
833     use="optional" default="ifNotExist"/>
834 </complexType>
835
836 <complexType name="JMSConnectionFactory">
837     <sequence>
838         <element name="property" type="sca:BindingProperty"
839             minOccurs="0" maxOccurs="unbounded"/>
840     </sequence>
841     <attribute name="jndiName" type="anyURI"/>
842     <attribute name="create" type="sca:JMSCreateResource"
843         use="optional" default="ifNotExist"/>
844 </complexType>
845
846 <complexType name="JMSActivationSpec">
847     <sequence>
848         <element name="property" type="sca:BindingProperty"
849             minOccurs="0" maxOccurs="unbounded"/>
850     </sequence>
851     <attribute name="jndiName" type="anyURI"/>
852     <attribute name="create" type="sca:JMSCreateResource"
853         use="optional" default="ifNotExist"/>
854 </complexType>
855
856 <complexType name="JMSResponse">
857     <sequence>
858         <element name="wireFormat" type="sca:WireFormatType" minOccurs="0"/>
859         <element name="destination" type="sca:JMSDestination" minOccurs="0"/>
860         <choice minOccurs="0">
861             <element name="connectionFactory" type="sca:JMSConnectionFactory"/>
862             <element name="activationSpec" type="sca:JMSActivationSpec"/>
863         </choice>
864     </sequence>
865 </complexType>
866
867 <complexType name="JMSHeaders">
868     <sequence>
869         <element name="property" type="sca:BindingProperty"
870             minOccurs="0" maxOccurs="unbounded"/>
871     </sequence>
872     <attribute name="type" type="string"/>
873     <attribute name="deliveryMode" default="persistent">
874         <simpleType>
875             <restriction base="string">
876                 <enumeration value="persistent"/>
877                 <enumeration value="nonpersistent"/>
878             </restriction>
879         </simpleType>
880     </attribute>
881     <attribute name="timeToLive" type="long" default="0"/>
882     <attribute name="priority" default="4">
883         <simpleType>
884             <restriction base="string">
885                 <enumeration value="0"/>
886                 <enumeration value="1"/>
887                 <enumeration value="2"/>
888                 <enumeration value="3"/>
889                 <enumeration value="4"/>
890                 <enumeration value="5"/>
891                 <enumeration value="6"/>

```

```

892         <enumeration value="7"/>
893         <enumeration value="8"/>
894         <enumeration value="9"/>
895     </restriction>
896 </simpleType>
897 </attribute>
898 </complexType>
899
900 <complexType name="JMSMessageSelection">
901     <sequence>
902         <element name="property" type="sca:BindingProperty"
903             minOccurs="0" maxOccurs="unbounded"/>
904     </sequence>
905     <attribute name="selector" type="string"/>
906 </complexType>
907
908 <complexType name="JMSResourceAdapter">
909     <sequence>
910         <element name="property" type="sca:BindingProperty"
911             minOccurs="0" maxOccurs="unbounded"/>
912     </sequence>
913     <attribute name="name" type="string" use="required"/>
914 </complexType>
915
916 <complexType name="JMSOperationProperties">
917     <sequence>
918         <element name="property" type="sca:BindingProperty"
919             minOccurs="0" maxOccurs="unbounded"/>
920         <element name="headers" type="sca:JMSHeaders"/>
921     </sequence>
922     <attribute name="name" type="string" use="required"/>
923     <attribute name="nativeOperation" type="string"/>
924 </complexType>
925
926 <complexType name="BindingProperty">
927     <simpleContent>
928         <extension base="string">
929             <attribute name="name" type="NMTOKEN" use="required"/>
930             <attribute name="type" type="string" use="optional"
931                 default="xs:string"/>
932         </extension>
933     </simpleContent>
934 </complexType>
935
936 <element name="binding.jms" type="sca:JMSBinding"
937     substitutionGroup="sca:binding"/>
938
939 <element name="wireFormat.jmsDefault" type="sca:WireFormatType"
940     substitutionGroup="sca:wireFormat"/>
941
942 <element name="operationSelector.jmsDefault" type="sca:OperationSelectorType"
943     substitutionGroup="sca:operationSelector"/>
944 </schema>

```

945 **B. Conformance Items**

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

Conformance ID	Description
[BJM30001]	The value of the @uri attribute MUST have the format defined by the IETF URI Scheme for Java™ Message Service 1.0 [IETFJMS]
[BJM30002]	When the @uri attribute is specified, the SCA runtime MUST raise an error if the referenced resources do not already exist
[BJM30003]	If the value of the @correlationScheme attribute is " sca:messageID " the SCA runtime MUST set the correlation ID of replies to the message ID of the corresponding request
[BJM30004]	If the value of the @correlationScheme attribute is " sca:correlationID " the SCA runtime MUST set the correlation ID of replies to the correlation ID of the corresponding request
[BJM30005]	If the value of the @correlationScheme attribute is " sca:none " the SCA runtime MUST NOT set the correlation ID
[BJM30006]	SCA runtimes MAY allow other values of the @correlationScheme attribute to indicate other correlation schemes
[BJM30010]	Whatever the value of the destination/@type attribute, the runtime MUST ensure a single response is delivered for request/response operations
[BJM30011]	If the @create attribute value for a destination, connectionFactory or activationSpec element is " always " and the @jndiName attribute is present and the resource cannot be created at the location specified by the @jndiName attribute then the SCA runtime MUST raise an error
[BJM30012]	If the @create attribute value for a destination, connectionFactory or activationSpec element is " ifNotExist " then the @jndiName attribute MUST specify the location of the possibly existing resource
[BJM30013]	If the destination, connectionFactory or activationSpec does not exist at the location identified by the @jndiName attribute, but cannot be created there then the SCA runtime MUST raise an error
[BJM30014]	If the destination, connectionFactory or activationSpec's @jndiName attribute refers to an existing resource that is not a JMS Destination of the appropriate type, a JMS connection factory or a JMS activation spec respectively then the SCA runtime MUST raise an error
[BJM30015]	If the @create attribute value for a destination, connectionFactory or activationSpec element is " never " then the @jndiName attribute MUST specify the location of the existing resource
[BJM30016]	If the destination, connection factory or activation spec is not present at the location identified by the @jndiName attribute, or the location refers to a resource of an incorrect type then the SCA runtime MUST raise an error
[BJM30017]	A binding:jms element MUST NOT include both a connectionFactory element and an activationSpec element

[BJM30018]	When the connectionFactory element is present, then the destination MUST be defined either by the destination element or the @uri attribute
[BJM30019]	If the activationSpec element is present and the destination is also specified via a destination element or the @uri attribute then it MUST refer to the same JMS destination as the activationSpec
[BJM30020]	The activationSpec element MUST NOT be present when the binding is being used for an SCA reference
[BJM30021]	A response element MUST NOT include both a connectionFactory element and an activationSpec element
[BJM30022]	If a response/destination and response/activationSpec element are both specified they MUST refer to the same JMS destination
[BJM30023]	The response/activationSpec element MUST NOT be present when the binding is being used for an SCA service
[BJM30024]	When sending messages for a JMS binding, the SCA runtime MUST set each of the JMSType, JMSDeliveryMode, JMSTimeToLive and JMSPriority headers to values specified in the binding definition in the following priority order: 1) the value for the header specified in the @uri attribute (highest priority); 2) the value for the header specified in the operationProperties/headers element matching the operation being invoked; 3) the value for the header specified in the headers element; 4) the default value for the header as specified by the definition of the binding.jms/headers element (lowest priority).
[BJM30025]	When sending messages for a JMS binding, the SCA runtime MUST set each named user property with type and value specified in the binding definition in the following priority order: 1) the type and value for the named user property specified in an operationProperties/headers/property element matching the name of the operation being invoked (highest priority); 2) the type and value for the named user property specified in a headers/property element (lowest priority).
[BJM30026]	When receiving messages for a JMS binding, the SCA runtime MUST use a message selector if specified in the binding definition in the following priority order: 1) the value for the message selector specified in the @uri attribute value's "selector" parameter (highest priority); 2) the value for the message selector specified in the messageSelection/@selector attribute; 3) otherwise no message selector is used (lowest priority).
[BJM30028]	SCA runtimes MAY place restrictions on the properties of the resource adapter Java bean that can be set using the resourceAdapter element
[BJM30029]	The value of the operationProperties/@selectedOperation attribute MUST be unique across the containing binding.jms element
[BJM30030]	The SCA runtime SHOULD make the operationProperties element corresponding to the selectedOperation available to the wireFormat implementation

Deleted: The SCA runtime MUST set JMS headers in messages that it creates to the values specified by the **headers** element unless overridden for the operation being invoked.

Deleted: If the **@uri** attribute includes values for the type, delivery mode, time to live or priority properties then the **@uri** values are used and the **headers** and **operationProperties/headers** **@type**, **@deliveryMode**, **@timeToLive** or **@priority** attributes are ignored

Deleted: For each **header/properties** element the SCA runtime MUST set the named JMS user property to the given value in messages it creates unless overridden for the operation being invoked

Deleted: [BJM30027] ... [1]

[BJM30031]	The resourceAdapter element MUST be present when JMS resources are to be created for a JMS provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise
[BJM30034]	When the @uri attribute is specified, the destination element MUST NOT be present
[BJM30036]	The binding.jms element MUST conform to the XML schema defined in sca-binding-jms.xsd
[BJM30037]	If the @create attribute value for a destination , connectionFactory or activationSpec element is "always" and the @jndiName attribute is not present and the resource cannot be created, then the SCA runtime MUST raise an error
[BJM40001]	The SCA runtime MUST support the default JMS wire format and operation selector behavior, and MAY provide additional means to override it
[BJM40002]	If no operationSelector element is specified then SCA runtimes MUST use operationSelector.jmsDefault as the default
[BJM40003]	When using the default wire format to send request messages, if there is a single parameter and the interface includes more than one operation, the SCA runtime MUST set the JMS user property " scaOperationName " to the name of the operation being invoked
[BJM40004]	If no wireFormat element is specified in a JMS binding then SCA runtimes MUST use wireFormat.jmsDefault as the default
[BJM40005]	When using the default wire format an SCA runtime MUST be able to receive both JMS text and bytes messages
[BJM40006]	When using the default wire format an SCA runtime MUST send either a JMS text or a JMS bytes message
[BJM40007]	When using the default wire format an SCA runtime MAY provide additional configuration to allow selection between JMS text or bytes messages to be sent
[BJM40008]	When a binding.jms element specifies the operationSelector.jmsDefault element, the SCA runtime MUST use the default operation selection algorithm to determine the selected operation
[BJM40009]	When a binding.jms element specifies the wireFormat.jmsDefault element, the SCA runtime MUST use the default wire format
[BJM40010]	When a message is received at an SCA service with JMS binding, the SCA runtime MUST invoke the target component using the resolved operation name
[BJM40011]	When a message is received at an SCA service with JMS binding and the resolved operation name is not in the target component's interface the SCA runtime MUST raise an error
[BJM50001]	JMS binding implementations MUST support the JMS intent
[BJM50002]	The JMS intent MUST always be included in the @alwaysProvides attribute of the JMS bindingType
[BJM60001]	For an SCA reference with a JMS binding and unidirectional interface, when a request message is sent as part of a one-way MEP, the SCA runtime

Deleted: [BJM30032] ... [2]

Deleted: [BJM30035] ... [3]

Deleted: Error! Not a valid bookmark self-reference.

	SHOULD NOT set the JMSReplyTo destination header in the JMS message that it creates, regardless of whether the JMS binding has a response element with a destination defined
[BJM60002]	For an SCA service with a JMS binding and unidirectional interface, when a request message is received as part of a one-way MEP, the SCA runtime MUST ignore the JMSReplyTo destination header in the JMS message, and not raise an error
[BJM60003]	For an SCA reference with a JMS binding that has a destination specified via the response element, the SCA runtime MUST receive response messages as defined by the binding's @correlationScheme attribute
[BJM60004]	For an SCA reference with a JMS binding, when a request message is sent as part of a request/response MEP, and the JMS binding has a response element with a destination defined, then the SCA runtime MUST use that destination for the JMSReplyTo header in the JMS message it creates for the request
[BJM60005]	For an SCA reference with a JMS binding, when a request message is sent as part of a request/response MEP, and the JMS binding does not have a response element with a destination defined, the SCA runtime MUST provide an appropriate destination on which to receive response messages and use that destination for the JMSReplyTo header in the JMS message it creates for the request
[BJM60006]	For an SCA reference with a JMS binding that does not have a destination specified via the response element, the SCA runtime MUST either receive response messages as defined by the binding's @correlationScheme attribute, or use a unique destination for each request/response interaction
[BJM60007]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a non-null JMSReplyTo destination, the SCA runtime MUST send the response message to that destination
[BJM60008]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a null JMSReplyTo destination and the JMS binding includes a response/destination element the SCA runtime MUST send the response message to that destination
[BJM60009]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a null JMSReplyTo destination and the JMS binding does not include a response/destination then an error SHOULD be raised by the SCA runtime
[BJM60010]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP the SCA runtime MUST set the correlation identifier in the JMS message that it creates for the response as defined by the JMS binding's @correlationScheme attribute
[BJM60011]	For an SCA reference with a JMS binding and a bidirectional interface, when a request message is sent as part of a request/response MEP the SCA runtime MUST set the scaCallbackDestination user property in the message it creates to a JMS URI string, in the format defined by the IETF URI Scheme for Java™ Message Service 1.0 [IETFJMS], that identifies the destination to which callback messages are to be sent

[BJM60012]	For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as part of a one-way MEP the SCA runtime MUST set the destination to which callback messages are to be sent as the JMSReplyTo destination in the message it creates
[BJM60013]	For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as part of a request/response MEP, the SCA runtime MUST set the JMSReplyTo header in the message it creates as described in section 6.2
[BJM60014]	For an SCA reference with a JMS binding and bidirectional interface, the SCA runtime MUST identify the callback destination from the reference's callback service binding if present, or supply a suitable callback destination if not present
[BJM60015]	For an SCA service with a JMS binding, when a callback request message is sent for either a one-way or request/response MEP, the SCA runtime MUST send the callback request message to the callback destination.
[BJM60016]	For an SCA service with a JMS binding, when a callback request message is sent and no callback destination can be identified then the SCA runtime SHOULD raise an error, and MUST throw an exception to the caller of the callback operation
[BJM60017]	For an SCA service with a JMS binding, when a callback request message is sent the SCA runtime MUST set the JMSReplyTo destination in the callback request message as defined in sections 6.1 or 6.2 as appropriate for the type of the callback operation invoked
[BJM60018]	SCA runtimes MUST follow the behavior described in section 6.4 and its subsections when binding.jms is used in both the forward and callback directions

C. Acknowledgements

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

Participants:

Participant Name	Affiliation
Bryan Aupperle	IBM
Ron Barack	SAP AG
Michael Beisiegel	IBM
Henning Blohm	SAP AG
David Booz	IBM
Martin Chapman	Oracle Corporation
Jean-Sebastien Delfino	IBM
Laurent Domenech	TIBCO Software Inc.
Jacques Durand	Fujitsu Limited
Mike Edwards	IBM
Billy Feng	Primeton Technologies, Inc.
Nimish Hathalia	TIBCO Software Inc.
Simon Holdsworth	IBM
Eric Johnson	Software Inc.
Uday Joshi	Oracle Corporation
Khanderao Kand	Oracle Corporation
Anish Karmarkar	Oracle Corporation
Nickolaos Kavantzias	Oracle Corporation
Mark Little	Red Hat
Ashok Malhotra	Oracle Corporation
Jim Marino	Individual
Jeff Mischkinsky	Oracle Corporation
Dale Moberg	Axway Software
Simon Nash	Individual
Sanjay Patil	SAP AG
Plamen Pavlov	SAP AG
Peter Peshev	SAP AG
Piotr Przybylski	IBM
Luciano Resende	IBM
Tom Rutt	Fujitsu Limited
Vladimir Savchenko	SAP AG
Scott Vorthmann	TIBCO Software Inc.
Tim Watson	Oracle Corporation
Owen Williams	Avaya, Inc.
Prasad Yendluri	Software AG, Inc.

D. Revision History

[optional; should not be included in OASIS Standards]

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

cd01-rev4	2009-01-09	Simon Holdsworth	Updates to resolve following issues: BINDINGS-7 BINDINGS-31 BINDINGS-40 BINDINGS-42 BINDINGS-44 BINDINGS-50
cd02	2009-02-16	Simon Holdsworth	Rename and editorial updates
cd02-rev1	2009-05-22	Simon Holdsworth	Updates to resolve issue BINDINGS-62 (conformance statement numbering) Updated assembly namespace to 200903 Fixed errors in schema
cd02-rev2	2009-05-22	Simon Holdsworth	Updates to resolve following issues: BINDINGS-39 BINDINGS-59 BINDINGS-65 BINDINGS-66 BINDINGS-67 BINDINGS-68 BINDINGS-70 BINDINGS-71
cd02-rev3	2009-06-18	Simon Holdsworth	Editorial concerns addressed Added acknowledgements appendix
cd02-rev4	2009-06-19	Simon Holdsworth	Updates to resolve following issues BINDINGS-74 Some editorial updates Fixed normative statement missed in application of BINDINGS-67
cd02-rev5	2009-06-24	Simon Holdsworth	Updates to resolve following issues BINDINGS-77 Renamed document to old form Removed editorial commentary Editorial fixes around external references; changed all links to hyperlinks
cd02-rev6	2009-06-24	Simon Holdsworth	Fixed application of BINDINGS-74 Fixed broken cross reference Changed ASCII to UTF-8 in examples
cd03	2009-06-29	Simon Holdsworth	Updates to resolve following issues BINDINGS-80 BINDINGS-81

cd03-rev1	2010-01-24	Simon Holdsworth	Editorial fix to XML schema name Updated to resolve following issues BINDINGS-48 BINDINGS-83 BINDINGS-85 BINDINGS-90 BINDINGS-93 BINDINGS-94 BINDINGS-96 BINDINGS-97 BINDINGS-98 BINDINGS-103 BINDINGS-108 BINDINGS-109
-----------	------------	------------------	--

Page 30: [1] Deleted		Simon Holdsworth	08/02/2010 14:45:00
[BJM30027]	If the @uri attribute includes a value for the message selector then the @uri value is used and the messageSelection/@selector attribute is ignored		

Page 31: [2] Deleted		Simon Holdsworth	08/02/2010 14:45:00
[BJM30032]	The SCA runtime MUST set JMS headers in messages it creates when the operation identified by the operationProperties/@name attribute is invoked to the values specified by the corresponding operationProperties/headers element		
[BJM30033]	For each operationProperties/headers/property element the SCA runtime MUST set the named JMS user property to the given value in messages it creates when the operation identified by the operationProperties/@name attribute is invoked		

Page 31: [3] Deleted		Simon Holdsworth	08/02/2010 14:44:00
[BJM30035]	An SCA runtime MUST use the values specified in the @uri attribute in preference to corresponding attributes and elements in the binding		