



Service Component Architecture JMS Binding Specification Version 1.1

Committee Draft 03 / Public Review 01

10 July 2009

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://www.oasis-open.org/committees/download.php/31233/sca-binding-jms-1.1-spec-cd02.doc>
<http://www.oasis-open.org/committees/download.php/31234/sca-binding-jms-1.1-spec-cd02.pdf>
(Authoritative)

Latest Version:

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

Technical Committee:

[OASIS Service Component Architecture / Bindings \(SCA-Bindings\) TC](#)

Chair(s):

Simon Holdsworth, IBM

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, 2009. All Rights Reserved.

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

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

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

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

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

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

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

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

Table of Contents

| | | |
|-------|---|----|
| 1 | Introduction..... | 5 |
| 1.1 | Terminology | 5 |
| 1.2 | Normative References | 5 |
| 1.3 | Non-Normative References | 6 |
| 1.4 | Naming Conventions | 6 |
| 2 | Messaging Bindings | 7 |
| 3 | JMS Binding Schema | 8 |
| 3.1 | Extensibility | 13 |
| 4 | Operation Selectors and Wire Formats..... | 14 |
| 4.1 | Default Operation Selection..... | 14 |
| 4.2 | Default Wire Format..... | 14 |
| 4.2.1 | Example of default wire format..... | 15 |
| 5 | Policy | 17 |
| 6 | Message Exchange Patterns | 18 |
| 6.1 | One-way message exchange (no Callbacks)..... | 18 |
| 6.2 | Request/response message exchange (no Callbacks) | 18 |
| 6.3 | JMS User Properties..... | 19 |
| 6.4 | Callbacks | 19 |
| 6.4.1 | Invocation of operations on a bidirectional interface..... | 19 |
| 6.4.2 | Invocation of operations on a callback interface | 19 |
| 6.4.3 | Use of JMSReplyTo for callbacks for non-SCA JMS applications..... | 20 |
| 7 | Examples..... | 21 |
| 7.1 | Minimal Binding Example | 21 |
| 7.2 | URI Binding Example..... | 21 |
| 7.3 | Binding with Existing Resources Example | 21 |
| 7.4 | Resource Creation Example..... | 22 |
| 7.5 | Request/Response Example | 22 |
| 7.6 | Use of Predefined Definitions Example | 23 |
| 7.7 | Subscription with Selector Example | 23 |
| 7.8 | Policy Set Example | 23 |
| 8 | Conformance | 25 |
| 8.1 | SCA JMS Binding XML Document | 25 |
| 8.2 | SCA Runtime | 25 |
| A. | JMS XML Binding Schema: sca-binding-jms.xsd..... | 26 |
| B. | Conformance Items | 29 |
| C. | Acknowledgements | 34 |
| D. | Revision History..... | 35 |

1 Introduction

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

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

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

1.1 Terminology

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

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

Table 1-1 Prefixes and Namespaces used in this specification

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

1.2 Normative References

- [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- [JMS] Java™ Message Service Specification 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/>

| | | |
|----|-----------------------|--|
| 32 | [IETFJMS] | M. Phillips, P. Easton, D. Rokicki, E. Johnson, <i>URI Scheme for Java™ Message Service 1.0</i> http://www.ietf.org/id/draft-merrick-jms-uri-06.txt , IETF Internet-Draft June 2009 ¹ |
| 33 | | |
| 34 | | |
| 35 | [SCA-Assembly] | OASIS Committee Draft 03, "Service Component Architecture Assembly Model Specification Version 1.1", March 2009 |
| 36 | | |
| 37 | | http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd03.pdf |
| 38 | | |
| 39 | [SCA-Policy] | OASIS Committee Draft 02, "SCA Policy Framework Specification Version 1.1", February 2009 |
| 40 | | |
| 41 | | http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd02.pdf |

42 1.3 Non-Normative References

43 TBD TBD

44 1.4 Naming Conventions

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

- 48 • Where the names of elements and attributes consist partially or wholly of acronyms, the letters of the
 49 acronyms use the same case. When the acronym appears at the start of the name of an element or
 50 an attribute, or after a period, it is in lower case. If it appears elsewhere in the name of an element or
 51 an attribute, it is in upper case. For example, an attribute might be named "uri" or "jndiURL".
- 52 • Where the names of types consist partially or wholly of acronyms, the letters of the acronyms are in
 53 all upper case. For example, an XML Schema type might be named "JCABinding" or "MessageID".
- 54 • Values, including local parts of QName values, follow the rules for names of elements and attributes
 55 as stated above, with the exception that the letters of acronyms are in all upper case. For example, a
 56 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

57 **2 Messaging Bindings**

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

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

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

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

67

3 JMS Binding Schema

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

```

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

```

```

126         deliveryMode="persistent or nonpersistent"?
127         timeToLive="long"?
128         priority="0 .. 9"?>
129     <property name="NMTOKEN" type="NMTOKEN"?>*
130     </headers?>
131 </operationProperties>*
132
133     <wireFormat ... />?
134     <operationSelector ... />?
135 </binding.jms>

```

136

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

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

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

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

- 281 • ***/binding.jms/messageSelection/@selector*** - specifies the value to use for the JMS selector. If the
 282 ***@uri*** attribute includes a value for the message selector then the ***@uri*** value is used and the
 283 ***messageSelection/@selector*** attribute is ignored [BJM30027].
- 284 • ***/binding.jms/resourceAdapter*** – specifies name, type and properties of the Resource Adapter Java
 285 bean.
 286 The ***resourceAdapter*** element MUST be present when JMS resources are to be created for a JMS
 287 provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise
 288 [BJM30031].
 289 SCA runtimes MAY place restrictions on the properties of the resource adapter Java bean that can be
 290 set using the ***resourceAdapter*** element [BJM30028].
 291 For JMS providers that do not implement the JCA 1.5 specification [JCA15], information necessary for
 292 resource creation can be added in provider-specific elements or attributes allowed by the extensibility
 293 of the ***binding.jms*** element.
- 294 • ***/binding.jms/operationProperties*** – specifies various properties that are specific to the processing
 295 of a particular operation.
- 296 • ***/binding.jms/operationProperties/@name*** – The name of the operation in the interface.
- 297 • ***/binding.jms/operationProperties/@selectedOperation*** – The value generated by the
 298 ***operationSelector*** that corresponds to the operation in the service or reference interface identified
 299 by the ***operationProperties/@name*** attribute. If this attribute is omitted then the value defaults to
 300 the value of the ***operationProperties/@name*** attribute.
 301 The value of the ***operationProperties/@selectedOperation*** attribute MUST be unique across the
 302 containing ***binding.jms*** element [BJM30029].
- 303 • ***/binding.jms/operationProperties/property*** – specifies properties specific to this operation. These
 304 properties are intended to be used to parameterize the ***wireFormat*** identified for the binding for a
 305 particular operation.
 306 The SCA runtime SHOULD make the ***operationProperties*** element corresponding to the
 307 ***selectedOperation*** available to the ***wireFormat*** implementation [BJM30030].
- 308 • ***/binding.jms/operationProperties/headers*** – this element specifies values for standard JMS
 309 headers. These values apply to requests from a reference and responses from a service.
 310 The SCA runtime MUST set JMS headers in messages it creates when the operation identified by the
 311 ***operationProperties/@name*** attribute is invoked to the values specified by the corresponding
 312 ***operationProperties/headers*** element [BJM30032].
- 313 • ***/binding.jms/operationProperties/headers/@type, @deliveryMode, @timeToLive, @priority*** –
 314 specifies the value to use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive
 315 or JMSPriority, respectively,
- 316 • ***/binding.jms/operationProperties/headers/property*** – specifies the value for the given JMS user
 317 property.
 318 For each ***operationProperties/headers/property*** element the SCA runtime MUST set the named
 319 JMS user property to the given value in messages it creates when the operation identified by the
 320 ***operationProperties/@name*** attribute is invoked [BJM30033].
- 321 • ***/binding.jms/wireFormat*** – identifies the wire format used by requests and responses sent or
 322 received by this binding. Wire formats for this binding are described in Section 4.
- 323 • ***/binding.jms/operationSelector*** – identifies the operation selector used when receiving requests for
 324 a service. If specified for a reference this provides the default operation selector for callbacks if not
 325 specified via a callback service element. Operation selectors for this binding are described in Section
 326 4.

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

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

333 **3.1 Extensibility**

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

337 4 Operation Selectors and Wire Formats

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

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

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

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

357 **Error! Not a valid bookmark self-reference.** [BJM40010].

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

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

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

363 4.1 Default Operation Selection

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

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

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

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

378 4.2 Default Wire Format

379 The default wire format maps between a **JMSMessage** and the object(s) expected by the component
380 implementation. We encourage component implementers to avoid exposure of [JMS \[JMS\]](#) APIs to

381 component implementations, however in the case of an existing implementation that expects a
382 **JMSMessage**, this provides for simple reuse of that as an SCA component.

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

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

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

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

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

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

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

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

405 4.2.1 Example of default wire format

406 For the following interface definition:

```
407 <wsdl:definitions name="Coordinates"  
408 targetNamespace="http://tempuri.org/coordinates"  
409 xmlns:tns="http://tempuri.org/coordinates"  
410 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  
411 xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
412 <wsdl:types>  
413 <xsd:schema targetNamespace="http://tempuri.org/coordinates">  
414 <xsd:element name="setCoordinates">  
415 <xsd:complexType>  
416 <xsd:sequence>  
417 <xsd:element name="x" type="xsd:int"/>  
418 <xsd:element name="y" type="xsd:int"/>  
419 </xsd:sequence>  
420 </xsd:complexType>  
421 </xsd:element>  
422 </xsd:schema>  
423 </wsdl:types>  
424  
425 <wsdl:message name="setCoordinatesRequestMsg">  
426 <wsdl:part element="tns:setCoordinates" name="setCoordinatesParameters"/>  
427 </wsdl:message>  
428  
429 <wsdl:portType name="Coordinates">  
430 <wsdl:operation name="setCoordinates">  
431 <wsdl:input message="tns:setCoordinatesRequestMsg"  
432 name="setCoordinatesRequest"/>  
433 </wsdl:operation>  
434 </wsdl:portType>
```

435 `</wsdl:definitions>`

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

```
439 <setCoordinates xmlns="http://tempuri.org/coordinates">  
440   <x>10</x>  
441   <y>5</y>  
442 </setCoordinates>
```

443

5 Policy

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

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

```
449 <bindingType type="binding.jms"  
450           alwaysProvides="JMS"  
451           mayProvide="atLeastOnce atMostOnce ordered"/>
```

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

454 6 Message Exchange Patterns

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

461 6.1 One-way message exchange (no Callbacks)

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

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

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

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

473 6.2 Request/response message exchange (no Callbacks)

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

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

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

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

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

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

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

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

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

502 6.3 JMS User Properties

503 This protocol assigns specific behavior to JMS user properties:

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

506 6.4 Callbacks

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

515 SCA runtimes MUST follow the behavior described in section 6.4 and its subsections when **binding.jms**
516 [BJM60018].

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

519 6.4.1 Invocation of operations on a bidirectional interface

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

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

526 For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as
527 [BJM60013].
528

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

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

534 6.4.2 Invocation of operations on a callback interface

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

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

- 541 • The **scaCallbackDestination** identified by an earlier request, if not null;
- 542 • the **JMSReplyTo** destination identified by an earlier one-way request, if not null;

- 543 • the request destination of the service's callback reference JMS binding, if specified
- 544 For an SCA service with a JMS binding, when a callback request message is sent for either a one-way or
545 request/response MEP, the SCA runtime MUST send the callback request message to the callback
546 destination. [BJM60015].
- 547 For an SCA service with a JMS binding, when a callback request message is sent and no callback
548 destination can be identified then the SCA runtime SHOULD raise an error, and MUST throw an
549 exception to the caller of the callback operation [BJM60016].
- 550 For an SCA service with a JMS binding, when a callback request message is sent the SCA runtime
551
552 [BJM60017].

553 6.4.3 Use of JMSReplyTo for callbacks for non-SCA JMS applications

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

559 7 Examples

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

563 7.1 Minimal Binding Example

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

```
566 <?xml version="1.0" encoding="UTF-8"?>
567 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
568           name="MyValueComposite">
569
570     <service name="MyValueService">
571       <interface.java interface="services.myvalue.MyValueService"/>
572       <binding.jms/>
573     </service>
574
575     <reference name="StockQuoteService">
576       <interface.java interface="services.stockquote.StockQuoteService"/>
577       <binding.jms/>
578     </reference>
579 </composite>
```

580 7.2 URI Binding Example

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

```
583 <?xml version="1.0" encoding="UTF-8"?>
584 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
585           name="MyValueComposite">
586
587     <service name="MyValueService">
588       <interface.java interface="services.myvalue.MyValueService"/>
589       <binding.jms uri="jms:MyValueServiceQueue?
590                   activationSpecName=MyValueServiceAS&
591                   ... "/>
592     </service>
593
594     <reference name="StockQuoteService">
595       <interface.java interface="services.stockquote.StockQuoteService"/>
596       <binding.jms uri="jms:StockQuoteServiceQueue?
597                   connectionFactoryName=StockQuoteServiceQCF&
598                   deliveryMode=1&
599                   ... "/>
600     </reference>
601 </composite>
```

602 7.3 Binding with Existing Resources Example

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

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

```

608     <service name="MyValueService">
609         <interface.java interface="services.myvalue.MyValueService" />
610         <binding.jms>
611             <destination jndiName="MyValueServiceQ" create="never" />
612             <activationSpec jndiName="MyValueServiceAS" create="never" />
613         </binding.jms>
614     </service>
615 </composite>

```

616 7.4 Resource Creation Example

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

```

619 <?xml version="1.0" encoding="UTF-8"?>
620 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
621     name="MyValueComposite">
622
623     <service name="MyValueService">
624         <interface.java interface="services.myvalue.MyValueService" />
625         <binding.jms>
626             <destination jndiName="MyValueServiceQueue" create="always">
627                 <property name="prop1" type="string">XYZ</property>
628                 <property name="destName" type="string">MyValueDest</property>
629             </destination>
630             <activationSpec jndiName="MyValueServiceAS" create="always" />
631             <resourceAdapter jndiName="com.example.JMSRA" />
632         </binding.jms>
633     </service>
634
635     <reference name="StockQuoteService">
636         <interface.java interface="services.stockquote.StockQuoteService" />
637         <binding.jms>
638             <destination jndiName="StockQuoteServiceQueue" />
639             <connectionFactory jndiName="StockQuoteServiceQCF" />
640             <resourceAdapter name="com.example.JMSRA" />
641         </binding.jms>
642     </reference>
643 </composite>

```

644 7.5 Request/Response Example

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

```

648 <?xml version="1.0" encoding="UTF-8"?>
649 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
650     name="MyValueComposite">
651
652     <service name="MyValueService">
653         <interface.java interface="services.myvalue.MyValueService" />
654         <binding.jms correlationScheme="sca:messageId">
655             <destination jndiName="MyValueServiceQ" create="never" />
656             <activationSpec jndiName="MyValueServiceAS" create="never" />
657         </binding.jms>
658     </service>
659
660     <reference name="StockQuoteService">
661         <interface.java interface="services.stockquote.StockQuoteService" />
662         <binding.jms correlationScheme="sca:messageId">
663             <destination jndiName="StockQuoteServiceQueue" />
664             <connectionFactory jndiName="StockQuoteServiceQCF" />

```

```

665         <response>
666             <destination jndiName="MyValueResponseQueue" />
667             <activationSpec jndiName="MyValueResponseAS" />
668         </response>
669     </binding.jms>
670 </reference>
671 </composite>

```

672 7.6 Use of Predefined Definitions Example

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

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

```

676 <?xml version="1.0" encoding="UTF-8"?>
677 <definitions targetNamespace="http://acme.com"
678             xmlns="http://docs.oasis-open.org/ns/opencsa/sca/2007903">
679     <binding.jms name="StockQuoteService">
680         <destination jndiName="StockQuoteServiceQueue" create="never" />
681         <connectionFactory jndiName="StockQuoteServiceQCF" create="never" />
682     </binding.jms>
683 </definitions>

```

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

```

685 <?xml version="1.0" encoding="UTF-8"?>
686 <composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200903"
687           xmlns:acme="http://acme.com"
688           name="MyValueComposite">
689     <reference name="MyValueService">
690         <interface.java interface="services.myvalue.MyValueService" />
691         <binding.jms requestConnection="acme:StockQuoteService" />
692     </reference>
693 </composite>

```

694 7.7 Subscription with Selector Example

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

```

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

```

710 7.8 Policy Set Example

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

```

714 <policySet name="JMSPolicy"
715           provides="priority log"

```

```

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

```

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

```

739 <reference name="StockQuoteService" requires="priority.high log">
740     <interface.java interface="services.stockquote.StockQuoteService"/>
741     <binding.jms>
742         <destination name="StockQuoteServiceQueue"/>
743         <connectionFactory name="StockQuoteServiceQCF"/>
744     </binding.jms>
745 </reference>

```

746

8 Conformance

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

750 a) SCA JMS Binding XML Document

751 b) SCA Runtime

752 8.1 SCA JMS Binding XML Document

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

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

761 8.2 SCA Runtime

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

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

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

```

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

```

```

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

```

```

894         <enumeration value="4" />
895         <enumeration value="5" />
896         <enumeration value="6" />
897         <enumeration value="7" />
898         <enumeration value="8" />
899         <enumeration value="9" />
900     </restriction>
901 </simpleType>
902 </attribute>
903 </complexType>
904
905 <complexType name="JMSMessageSelection">
906     <sequence>
907         <element name="property" type="sca:BindingProperty"
908             minOccurs="0" maxOccurs="unbounded" />
909     </sequence>
910     <attribute name="selector" type="string" />
911 </complexType>
912
913 <complexType name="JMSResourceAdapter">
914     <sequence>
915         <element name="property" type="sca:BindingProperty"
916             minOccurs="0" maxOccurs="unbounded" />
917     </sequence>
918     <attribute name="name" type="string" use="required" />
919 </complexType>
920
921 <complexType name="JMSOperationProperties">
922     <sequence>
923         <element name="property" type="sca:BindingProperty"
924             minOccurs="0" maxOccurs="unbounded" />
925         <element name="headers" type="sca:JMSHeaders" />
926     </sequence>
927     <attribute name="name" type="string" use="required" />
928     <attribute name="nativeOperation" type="string" />
929 </complexType>
930
931 <complexType name="BindingProperty">
932     <simpleContent>
933         <extension base="string">
934             <attribute name="name" type="NMTOKEN" />
935             <attribute name="type" type="string" use="optional"
936                 default="xs:string" />
937         </extension>
938     </simpleContent>
939 </complexType>
940
941 <element name="binding.jms" type="sca:JMSBinding"
942     substitutionGroup="sca:binding" />
943
944 <element name="wireFormat.jmsDefault" type="sca:WireFormatType"
945     substitutionGroup="sca:wireFormat" />
946
947 <element name="operationSelector.jmsDefault" type="sca:OperationSelectorType"
948     substitutionGroup="sca:operationSelector" />
949 </schema>

```

B. Conformance Items

951 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 |
| [BJM30007] | If the @requestConnection attribute is specified, the binding.jms element MUST NOT contain a destination , connectionFactory , activationSpec or resourceAdapter element |
| [BJM30008] | If the @responseConnection attribute is specified, the binding.jms element MUST NOT contain a response element |
| [BJM30009] | If the @operationProperties attribute is specified, the binding.jms element MUST NOT contain an operationProperties element |
| [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 " then the @jndiName attribute is optional; if the resource cannot be created at the specified location 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] | The SCA runtime MUST set JMS headers in messages that it creates to the values specified by the headers element unless overridden for the operation being invoked. |
| [BJM30025] | If the @uri attribute includes values for the type, delivery mode, time to live or priority properties then the @uri values are used and the headers and operationProperties/headers @type, @deliveryMode, @timeToLive or @priority attributes are ignored |
| [BJM30026] | For each header/properties element the SCA runtime MUST set the named JMS user property to the given value in messages it creates unless overridden for the operation being invoked |
| [BJM30027] | If the @uri attribute includes a value for the message selector then the @uri value is used and the messageSelection/@selector attribute is ignored |
| [BJM30028] | SCA runtimes MAY place restrictions on the properties of the resource adapter Java bean that can be set using the resourceAdapter element |
| [BJM30029] | The value of the operationProperties/@selectedOperation attribute MUST be unique across the containing binding.jms element |
| [BJM30030] | The SCA runtime SHOULD make the operationProperties element corresponding to the selectedOperation available to the wireFormat implementation |
| [BJM30031] | The resourceAdapter element MUST be present when JMS resources are to be created for a JMS provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise |
| [BJM30032] | The SCA runtime MUST set JMS headers in messages it creates when the operation identified by the operationProperties/@name attribute is invoked to |

| | |
|------------|---|
| | the values specified by the corresponding <i>operationProperties/headers</i> element |
| [BJM30033] | For each <i>operationProperties/headers/property</i> 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 <i>operationProperties/@name</i> attribute is invoked |
| [BJM30034] | When the <i>@uri</i> attribute is specified, the <i>destination</i> element MUST NOT be present |
| [BJM30035] | An SCA runtime MUST use the values specified in the <i>@uri</i> attribute in preference to corresponding attributes and elements in the binding |
| [BJM30036] | The <i>binding.jms</i> element MUST conform to the XML schema defined in <i>sca-binding-jms.xsd</i> |
| [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 <i>operationSelector</i> element is specified then SCA runtimes MUST use <i>operationSelector.jmsDefault</i> 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 " <i>scaOperationName</i> " to the name of the operation being invoked |
| [BJM40004] | If no <i>wireFormat</i> element is specified in a JMS binding then SCA runtimes MUST use <i>wireFormat.jmsDefault</i> 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 <i>binding.jms</i> element specifies the <i>operationSelector.jmsDefault</i> element, the SCA runtime MUST use the default operation selection algorithm to determine the selected operation |
| [BJM40009] | When a <i>binding.jms</i> element specifies the <i>wireFormat.jmsDefault</i> element, the SCA runtime MUST use the default wire format |
| [BJM40010] | Error! Not a valid bookmark self-reference. |
| [BJM60001] | For an SCA reference with a JMS binding and unidirectional interface, when a request message is sent as part of a one-way MEP, the SCA runtime SHOULD NOT set the <i>JMSReplyTo</i> destination header in the JMS message that it creates, regardless of whether the JMS binding has a <i>response</i> element with a <i>destination</i> defined |
| [BJM60002] | For an SCA service with a JMS binding, when a request message is received as part of a one-way MEP, the SCA runtime MUST ignore the <i>JMSReplyTo</i> destination header in the JMS message, and not raise an error |
| [BJM60004] | For an SCA reference with a JMS binding, when a request message is sent as |

| | |
|------------|---|
| | part of a request/response MEP, and the JMS binding has a response element with a destination defined, then the SCA runtime MUST use that destination for the JMSReplyTo header in the JMS message it creates for the request |
| [BJM60005] | For an SCA reference with a JMS binding, when a request message is sent as part of a request/response MEP, and the JMS binding does not have a response element with a destination defined, the SCA runtime MUST provide an appropriate destination on which to receive response messages and use that destination for the JMSReplyTo header in the JMS message it creates for the request |
| [BJM60006] | For an SCA reference with a JMS binding, the SCA runtime MAY choose to receive response messages on the basis of their correlation ID as defined by the binding's @correlationScheme attribute, or use a unique destination for each response |
| [BJM60007] | For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a non-null JMSReplyTo destination, the SCA runtime MUST send the response message to that destination |
| [BJM60008] | For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a null JMSReplyTo destination and the JMS binding includes a response/destination element the SCA runtime MUST send the response message to that destination |
| [BJM60009] | For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a null JMSReplyTo destination and the JMS binding does not include a response/destination then an error SHOULD be raised by the SCA runtime |
| [BJM60010] | For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP the SCA runtime MUST set the correlation identifier in the JMS message that it creates for the response as defined by the JMS binding's @correlationScheme attribute |
| [BJM60011] | For an SCA reference with a JMS binding and a bidirectional interface, when a request message is sent the SCA runtime MUST set the destination to which callback messages are to be sent as the value of the scaCallbackDestination user property in the message it creates |
| [BJM60012] | For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent the SCA runtime MAY set the JMSReplyTo destination to the same value as the scaCallbackDestination user property |
| [BJM60013] | For an SCA reference with a JMS binding and bidirectional interface, when a |
| [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 |
| [BJM60018] | SCA runtimes MUST follow the behavior described in section 6.4 and its |

953

C. Acknowledgements

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

956 **Participants:**

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

957

D. Revision History

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

959

| 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 |

