

SCA Service Component Architecture

JMS Binding Specification

SCA Version 1.00, March 21 2007

Technical Contacts:	Rajith Attapattu	Red Hat
	Henning Blohm	SAP AG
	Simon Holdsworth	IBM Corporation
	Eric Johnson	TIBCO Software Inc.
	Anish Karmarkar	Oracle
	Michael Rowley	BEA Systems, Inc.

Copyright Notice

© Copyright BEA Systems, Inc., Cape Clear Software, International Business Machines Corp, Interface21, IONA Technologies, Oracle, Primeton Technologies, Progress Software, Red Hat, Rogue Wave Software, SAP AG., Siemens AG., Software AG., Sun Microsystems, Inc., Sybase Inc., TIBCO Software Inc., 2005, 2007. All rights reserved.

License

The Service Component Architecture Specification is being provided by the copyright holders under the following license. By using and/or copying this work, you agree that you have read, understood and will comply with the following terms and conditions:

Permission to copy, display and distribute the Service Component Architecture Specification and/or portions thereof, without modification, in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of the Service Component Architecture Specification, or portions thereof, that you make:

1. A link or URL to the Service Component Architecture Specification at this location:
 - <http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications>
2. The full text of the copyright notice as shown in the Service Component Architecture Specification.

BEA, Cape Clear, IBM, Interface21, IONA, Oracle, Primeton, Progress Software, Red Hat, Rogue Wave, SAP, SIEMENS AG, Software AG., Sun Microsystems, Sybase, TIBCO (collectively, the "Authors") agree to grant you a royalty-free license, under reasonable, non-discriminatory terms and conditions to patents that they deem necessary to implement the Service Component Architecture Specification.

THE Service Component Architecture SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, REGARDING THIS SPECIFICATION AND THE IMPLEMENTATION OF ITS CONTENTS, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OR TITLE.

THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE Service Components Architecture SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the Service Component Architecture Specification or its contents without specific, written prior permission. Title to copyright in the Service Component Architecture Specification will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

Status of this Document

This specification may change before final release and you are cautioned against relying on the content of this specification. The authors are currently soliciting your contributions and suggestions. Licenses are available for the purposes of feedback and (optionally) for implementation.

IBM is a registered trademark of International Business Machines Corporation in the United States, other countries, or both.

BEA is a registered trademark of BEA Systems, Inc.

Cape Clear is a registered trademark of Cape Clear Software

IONA and IONA Technologies are registered trademarks of IONA Technologies plc.

Oracle is a registered trademark of Oracle USA, Inc.

Progress is a registered trademark of Progress Software Corporation

Primeton is a registered trademark of Primeton Technologies, Ltd.

Red Hat is a registered trademark of Red Hat Inc.

Rogue Wave is a registered trademark of Quovadx, Inc

SAP is a registered trademark of SAP AG.

SIEMENS is a registered trademark of SIEMENS AG

Software AG is a registered trademark of Software AG

Sun and Sun Microsystems are registered trademarks of Sun Microsystems, Inc.

Sybase is a registered trademark of Sybase, Inc.

TIBCO is a registered trademark of TIBCO Software, Inc.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Table of Contents

SCA Service Component Architecture.....	i
Copyright Notice	ii
License.....	ii
Status of this Document	iii
Table of Contents.....	i
1 Messaging and JMS Bindings.....	1
1.1 Introduction	1
1.2 Operation Selection and Data Binding	1
1.3 Messaging Bindings	1
1.4 JMS Binding Schema	2
1.5 Default Operation Selection and Data Binding behavior.....	5
1.5.1 Default Operation Selection.....	5
1.5.2 Default Data Binding.....	5
1.6 Policy	6
1.7 Callback and Conversation Protocol	6
1.7.1 JMS User Properties	6
1.7.2 Callbacks	6
1.7.3 Conversations	6
1.8 Examples.....	7
1.8.1 Minimal Binding Example	7
1.8.2 URI Binding Example	8
1.8.3 Binding with Existing Resources Example	8
1.8.4 Resource Creation Example.....	8
1.8.5 Request/Response Example	9
1.8.6 Use of Predefined Definitions Example	9
1.8.7 Policy Set Example	10
2 JMS Binding Schema	12
3 References.....	15

1 Messaging and JMS Bindings

3 1.1 Introduction

4 This document defines the concept and behavior of a messaging binding, and a concrete JMS-
5 based [1] binding that provides that behavior.

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

11 Further work is needed for specifying the simplifications that are possible for messaging bindings
12 used for SCA wires (see section 3: Open Issues).

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

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

19 1.2 Operation Selection and Data Binding

20 In general messaging providers deal with message formats and destinations. There is not
21 usually a built-in concept of "operation" that corresponds to that defined in a WSDL port type
22 [3]. Messages have a format which corresponds in some way to the schema of an input or
23 output message of an operation in the interface of a service or reference, however some means
24 is required in order to identify the specific operation and map the message information in to the
25 required form.

26 No standard means for service providers and consumers to declare and exchange message
27 format information is provided.

28 The process of identifying the operation to be invoked is ***operation selection***; that of mapping
29 message information to the required runtime form is ***data binding***. The JMS binding defines
30 default operation selection and data binding behavior; SCA providers may provide extensions for
31 custom behavior.

32 1.3 Messaging Bindings

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

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

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

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

42 **1.4 JMS Binding Schema**

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

```

44 <binding.jms correlationScheme="string"?
45   initialContextFactory="xs:anyURI"?
46   jndiURL="xs:anyURI"?
47   requestConnection="QName"?
48   responseConnection="QName"?
49   operationProperties="QName"?
50   ... >
51   <destination name="xs:anyURI" type="string"? create="string"?>
52     <property name="NMOKEN" type="NMOKEN">*</property>?
53   </destination>?
54   <connectionFactory name="xs:anyURI" create="string"?>
55     <property name="NMOKEN" type="NMOKEN">*</property>?
56   </connectionFactory>?
57   <activationSpec name="xs:anyURI" create="string"?>
58     <property name="NMOKEN" type="NMOKEN">*</property>?
59   </activationSpec>?

60   <response>
61     <destination name="xs:anyURI" type="string"? create="string"?>
62       <property name="NMOKEN" type="NMOKEN">*</property>?
63     </destination>?
64     <connectionFactory name="xs:anyURI" create="string"?>
65       <property name="NMOKEN" type="NMOKEN">*</property>?
66     </connectionFactory>?
67     <activationSpec name="xs:anyURI" create="string"?>
68       <property name="NMOKEN" type="NMOKEN">*</property>?
69     </activationSpec>?
70   </response>?

71   <resourceAdapter name="NMOKEN">?
72     <property name="NMOKEN" type="NMOKEN">*</property>?
73   </resourceAdapter>?

74   <headers JMSType="string"?
75     JMSCorrelationId="string"?
76     JMSDeliveryMode="string"?
77     JMSTimeToLive="int"?
78     JMSPriority="string"?>
79     <property name="NMOKEN" type="NMOKEN">*</property>?
80   </headers>?

81   <operationProperties name="string" nativeOperation="string"?>
82     <property name="NMOKEN" type="NMOKEN">*</property>?
83     <headers JMSType="string"?
84       JMSCorrelationId="string"?
85       JMSDeliveryMode="string"?
86       JMSTimeToLive="int"?
87       JMSPriority="string"?>
88       <property name="NMOKEN" type="NMOKEN">*</property>?
89     </headers>?
90   </operationProperties>?
91
92
93
94

```

95 </binding.jms>

96

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

99 The binding.jms element has the following attributes:

100 • **/binding.jms** – This is the generic JMS binding type. The type is extensible so that JMS
101 binding implementers can add additional JMS provider-specific attributes and elements
102 although such extensions are not guaranteed to be portable across runtimes.

103 • **/binding.jms/@uri** – (from binding) URI that identifies the destination, connection factory
104 or activation spec, and other properties to be used to send/receive the JMS message

105 The URI has the following format:

106 o jms:<jms-dest>?
107 connectionFactoryName=<Connection-Factory-Name> &
108 destinationType={queue|topic}
109 deliveryMode=<Delivery-Mode> &
110 timeToLive=<Time-To-Live> &
111 priority=<Priority> &
112 <User-Property>=<User-Property-Value> & ...

113 When the URI is used, it is assumed that the referenced resources already exist.

114 • **/binding.jms/@correlationScheme** – identifies the correlation scheme used when sending
115 reply or callback messages. Valid values are "RequestMsgIDToCorrelID" (the default),
116 "RequestCorrelIDToCorrelID", and "None".

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

118 • **/binding.jms/@jndiURL** – the URL for the JNDI provider.

119 • **/binding.jms/@requestConnection** – identifies a binding.jms element that is present in a
120 definition document, whose destination, connectionFactory, activationSpec and
121 resourceAdapter children are used to define the values for this binding. In this case the
122 corresponding elements must not be present within this binding element.

123 • **/binding.jms/@responseConnection** – identifies a binding.jms element that is present in
124 a definition document, whose response child element is used to define the values for this
125 binding. In this case no response element must be present within this binding element.

126 • **/binding.jms/@operationProperties** – identifies a binding.jms element that is present in
127 a definition document, whose operationProperties children are used to define the values for
128 this binding. In this case no operationProperties elements must be present within this binding
129 element.

130 • **/binding.jms/destination** – identifies the destination that is to be used to process requests
131 by this binding.

132 • **/binding.jms/destination/@type** - the type of the request destination. Must take one of
133 the values "queue" or "topic". The default value is "queue". When "topic" is specified, then
134 all the operations in the interface that corresponds to the binding must be one-way.

135 • **/binding.jms/destination/@name** – the name of the destination to which the binding is
136 connected. This may be a JNDI name or a plain destination name.

137 • **/binding.jms/destination/@create** – indicates whether the destination should be created
138 when the containing composite is deployed. Valid values are "always", "never" and
139 "ifnotexist". The default value is "ifnotexist". If "always" is specified and the corresponding
140 resource already exists, then this should be considered an error.

- **/binding.jms/destination/property** – defines properties to be used to create the destination, if required.
- **/binding.jms/connectionFactory** – identifies the connection factory that the binding uses to process request messages. This may be a JNDI name or a plain connection factory name. The attributes of this element follow those defined for the destination element. This element is mutually exclusive with the **activationSpec** element.
- **/binding.jms/activationSpec** – identifies the activation spec that the binding uses to connect to a JMS destination to process request messages. This may be a JNDI name or a plain activation spec name. The attributes of this element follow those defined for the destination element.
- **/binding.jms/response** – defines the resources used for handling response messages (receiving responses for a reference, and sending responses from a service).
- **/binding.jms/response/destination** – identifies the destination that is to be used to process responses by this binding. Attributes are as for the parent's destination element.
- **/binding.jms/response/connectionFactory** – identifies the connection factory that the binding uses to process response messages. This may be a JNDI name or a plain connection factory name. The attributes of this element follow those defined for the destination element. This element is mutually exclusive with the **activationSpec** element.
- **/binding.jms/response/activationSpec** – identifies the activation spec that the binding uses to connect to a JMS destination to process response messages. This may be a JNDI name or a plain activation spec name. The attributes of this element follow those defined for the destination element.
- **/binding.jms/headers** – this element allows JMS headers to be set to the given values for all operations. These values apply to requests from a reference and responses from a service.
- **/binding.jms/headers/@JMSType, @JMSCorrelationID, @JMSDeliveryMode, @JMSTimeToLive, @JMSPriority** – specifies the value to use for the JMS header property. If these attributes are specified they must not appear in the URI.
- **/binding.jms/headers/property** – specifies the value to use for the specified JMS user property.
- **/binding.jms/resourceAdapter** – specifies name, type and properties of the Resource Adapter Java bean. This is required when the JMS resources are to be created for a JCA 1.5-compliant JMS provider [4], and is ignored otherwise. There may be a restriction, depending on the deployment platform, about specifying properties of the RA Java Bean. For non-JCA 1.5-compliant JMS providers, information necessary for resource creation must be done in provider-specific elements or attributes allowed by the extensibility of the binding.jms element.
- **/binding.jms/operationProperties** – specifies various properties that are specific to the processing of a particular operation.
- **/binding.jms/operationProperties/@name** – The name of the operation in the interface.
- **/binding.jms/operationProperties/@nativeOperation** – The name of the native operation that corresponds to this operation in the interface.
- **/binding.jms/operationProperties/property** – specifies properties specific to this operation.
- **/binding.jms/operationProperties/headers** – this element allows JMS headers to be set to the given values for the given operation. These values apply to requests from a reference and responses from a service.

- */binding.jms/operationProperties/headers/@JMSType, @JMSCorrelationID, @JMSDeliveryMode, @JMSTimeToLive, @JMSPriority* – specifies the value to use for the JMS header property. Values specified for particular operations take precedence over those defined on the binding or via the URI.
- */binding.jms/operationProperties/headers/property* – specifies the value to use for the specified JMS user property.
- */binding.jms/@{any}* - this is an extensibility mechanism to allow extensibility via attributes.
- */binding.jms/any* – this is an extensibility mechanism to allow extensibility via elements.

1.5 Default Operation Selection and Data Binding behavior

This section describes the default behavior for operation selection and data binding for a JMS binding.

1.5.1 Default Operation Selection

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

- If there is only one operation on the service's interface, then that operation is assumed as the native operation name.
- Otherwise, if the JMS user property "scaOperationName" is present, then its value is used as the native operation name.
- Otherwise, the native operation name is assumed to be "onMessage".

The native operation name may then be mapped to an operation in the service's interface via a matching operation element in the JMS binding. If there is no matching element, the operation name is assumed to be the same as the native operation name.

When sending a request from a reference, or a callback from a service, if the interface includes more than one operation then the "scaOperationName" JMS user property is set to the operation being invoked.

To support any other means of function selection, the SCA runtime may provide the means for supplying and identifying alternative function selection behaviors.

1.5.2 Default Data Binding

The default data binding behavior maps between a JMSMessage and the object(s) expected by the component implementation. We encourage component implementers to avoid exposure of 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.

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

- If there is a single parameter or return value that is a JMSMessage, then the JMSMessage is passed as is.
- Otherwise, the JMSMessage must be a JMS text message containing XML.
- If there is a single parameter, or for the return value, the JMS text XML payload is the XML serialization of that parameter according to the WSDL schema for the message.
- If there are multiple parameters, then they are encoded in XML using the document wrapped style, according to the WSDL schema for the message.

231 To support any other type of JMS message, the SCA runtime should provide the means for
 232 supplying and identifying alternative data binding behaviors.

233 **1.6 Policy**

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

238 JMS binding implementations may natively provide support for some standard intents, as defined
 239 by the JMS binding's bindingType:

```
240 <bindingType type="binding.jms"
241   alwaysProvides="jms"
242   mayProvide="atLeastOnce atMostOnce ordered conversation"/>
```

243 **1.7 Callback and Conversation Protocol**

244 This section describes the protocol that is used to support callbacks and conversational behavior
 245 when using the JMS binding. These apply to a JMS binding on a service or reference with a
 246 bidirectional interface.

247 **1.7.1 JMS User Properties**

248 This protocol assigns specific behavior to JMS user properties:

- 249 • "scaCallbackQueue" holds the name of the queue to which callback messages are sent.
- 250 • "scaConversationStart" indicates that a conversation is to be started, its value is the identifier
 251 for the conversation.
- 252 • "scaConversationMaxIdleTime" defines the maximum time that should be allowed between
 253 operations in the conversation.
- 254 • "scaConversationId" holds the identifier for the conversation.

255 **1.7.2 Callbacks**

256 A callback is the invocation of an operation on a service's callback interface.

257 When an SCA component with a reference with a bidirectional interface and JMS binding ("the
 258 sender") invokes an operation on that interface, the JMS message that is sent may identify the
 259 target for callbacks using the "scaCallbackQueue" user property, or for one-way operations the
 260 JMS replyTo header.

261 The invoked SCA component ("the receiver") can only invoke operations on the callback interface
 262 during the execution of the target operation for such a message, or when the service's callback
 263 binding identifies a fixed callback queue. The sender's callback queue can be specified on the
 264 reference's JMS callback binding, or it can be left to the runtime to provide one, by omitting the
 265 callbackService element, the JMS callback binding, or omitting the uri and destination from the
 266 JMS callback binding.

267 **1.7.3 Conversations**

268 A conversation is a sequence of operations between two parties that have a common context.
 269 The conversation may include a mixture of operations in either direction between the two parties.
 270 Interfaces must be marked as conversational in order to ensure that the runtime manages the
 271 lifecycle of this context.

272 Either the sender or receiver must start a conversation when an operation is invoked on a
 273 conversational interface and there is no active conversation with the other party. This is done by
 274 including the "scaConversationStart" user property in the JMS message with the value set to the
 275 required conversation identifier. A new runtime context is associated with the conversation
 276 identifier in both the sender and receiver.

277 The message that starts the conversation may also include the "scaConversationMaxIdleTime"
 278 user property; if not present the maximum idle time for the conversation is derived by
 279 subtracting the current time from the value of the JMSExpiration property, unless the
 280 JMSExpiration property value is zero, in which case the maximum idle time is unlimited. The
 281 sender may provide a specific callback queue for the identified conversation by including a value
 282 for the "scaCallbackQueue" user property.

283 Subsequent operations between the sender and receiver that are part of this conversation must
 284 include the "scaConversationId" user property in the JMS message, set to the conversation
 285 identifier. The message may also include an updated value of the "scaConversationMaxIdleTime"
 286 property. The value of "scaCallbackQueue" is ignored within a conversation in messages after
 287 the one that starts the conversation.

288 When an operation is invoked either by the sender or receiver that is marked as
 289 "endsConversation", or the maximum idle time is exceeded, then the conversation identifier and
 290 associated context is discarded after the operation has been processed. The idle time is defined
 291 within the sender and receiver as the amount of time since the sender/receiver last completed
 292 processing of an operation that is part of the conversation. There may be times when the sender
 293 or receiver ends the conversation before the other does. In that case if one party does invoke an
 294 operation on the other, it is treated as being after the conversation has ended and is an error.

295 Operations invoked on other parties must not be considered part of this conversation and must
 296 use different conversation identifiers.

297 Messages received containing a conversation identifier that does not correspond to a started
 298 conversation, or containing a start conversation property with a conversation identifier that
 299 matches an active conversation, should be treated as errors and should not be processed.
 300 Conversation identifiers may be reused. In particular, runtimes do not have to guarantee unique
 301 conversation identifiers and do not have to be able to identify an ended conversation indefinitely,
 302 although they may do for some period after the conversation ends. Due to the long-running
 303 nature of conversations, runtimes should ensure conversation context is available across server
 304 restarts, although they may choose to treat a restart as implicitly ending the conversation.

305 Component implementation specifications define the manner in which the context that is
 306 associated with the conversation identifier is made available to component implementations.

307 1.8 Examples

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

311 1.8.1 Minimal Binding Example

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

```
314 <?xml version="1.0" encoding="ASCII"?>
315 <composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
316           name="MyValueComposite">
317
318   <service name="MyValueService">
319     <interface.java interface="services.myvalue.MyValueService"/>
```

```

320      <binding.jms/>
321  </service>
322
323  <reference name="StockQuoteService">
324      <interface.java interface="services.stockquote.StockQuoteService"/>
325      <binding.jms/>
326  </reference>
327 </composite>
328

```

1.8.2 URI Binding Example

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

```

332  <?xml version="1.0" encoding="ASCII"?>
333  <composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
334      name="MyValueComposite">
335
336      <service name="MyValueService">
337          <interface.java interface="services.myvalue.MyValueService"/>
338          <binding.jms uri="jms:MyValueServiceQueue?
339              activationSpecName=MyValueServiceAS&
340              ... "/>
341      </service>
342
343      <reference name="StockQuoteService">
344          <interface.java interface="services.stockquote.StockQuoteService"/>
345          <binding.jms uri="jms:StockQuoteServiceQueue?
346              connectionFactoryName=StockQuoteServiceQCF&
347              deliveryMode=1&
348              ... "/>
349      </reference>
350  </composite>
351

```

1.8.3 Binding with Existing Resources Example

The following example shows the JMS binding using existing resources:

```

354  <?xml version="1.0" encoding="ASCII"?>
355  <composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
356      name="MyValueComposite">
357
358      <service name="MyValueService">
359          <interface.java interface="services.myvalue.MyValueService"/>
360          <binding.jms>
361              <destination name="MyValueServiceQ" create="never"/>
362              <activationSpec name="MyValueServiceAS" create="never"/>
363          </binding.jms>
364      </service>
365  </composite>
366

```

1.8.4 Resource Creation Example

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

```

370  <?xml version="1.0" encoding="ASCII"?>
371  <composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
372      name="MyValueComposite">
373

```

```

374 <service name="MyValueService">
375   <interface.java interface="services.myvalue.MyValueService"/>
376   <binding.jms>
377     <destination name="MyValueServiceQueue" create="always">
378       <property name="prop1" type="string">XYZ</property>
379     </destination>
380     <activationSpec name="MyValueServiceAS" / create="always">
381       <resourceAdapter name="com.example.JMSRA"/>
382     </binding.jms>
383   </service>
384
385   <reference name="StockQuoteService">
386     <interface.java interface="services.stockquote.StockQuoteService"/>
387     <binding.jms>
388       <destination name="StockQuoteServiceQueue" />
389       <connectionFactory name="StockQuoteServiceQCF" />
390       <resourceAdapter name="com.example.JMSRA"/>
391     </binding.jms>
392   </reference>
393 </composite>
394

```

1.8.5 Request/Response Example

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

```

399 <?xml version="1.0" encoding="ASCII"?>
400 <composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
401           name="MyValueComposite">
402
403   <service name="MyValueService">
404     <interface.java interface="services.myvalue.MyValueService"/>
405     <binding.jms correlationScheme="RequestMsgIdToCorrelId">
406       <destination name="MyValueServiceQ" create="never" />
407       <activationSpec name="MyValueServiceAS" create="never" />
408     </binding.jms>
409   </service>
410
411   <reference name="StockQuoteService">
412     <interface.java interface="services.stockquote.StockQuoteService"/>
413     <binding.jms correlationScheme="RequestMsgIdToCorrelId">
414       <destination name="StockQuoteServiceQueue" />
415       <connectionFactory name="StockQuoteServiceQCF" />
416       <response>
417         <destination name="MyValueResponseQueue" />
418         <activationSpec name="MyValueResponseAS" />
419       </response>
420     </binding.jms>
421   </reference>
422 </composite>
423

```

1.8.6 Use of Predefined Definitions Example

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

The common connection information is defined in a separate resource file:

```
<?xml version="1.0" encoding="ASCII"?>
```

```

429 <definitions targetNamespace="http://acme.com"
430   xmlns="http://www.osoa.org/xmlns/sca/1.0">
431   <binding.jms name="StockQuoteService">
432     <destination name="StockQuoteServiceQueue" create="never"/>
433     <connectionFactory name="StockQuoteServiceQCF" create="never"/>
434   </binding.jms>
435 </definitions>

```

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

```

437 <?xml version="1.0" encoding="ASCII"?>
438 <composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
439   xmlns:acme="http://acme.com"
440   name="MyValueComposite">
441
442   <reference name="MyValueService">
443     <interface.java interface="services.myvalue.MyValueService"/>
444     <binding.jms requestConnection="acme:StockQuoteService"/>
445   </reference>
446 </composite>

```

447

1.8.7 Policy Set Example

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

```

452 <policySet name="JMSPolicy"
453   provides="priority log"
454   appliesTo="binding.jms">
455
456   <intentMap provides="priority" default="medium">
457     <qualifier name="high">
458       <headers JMSPriority="9"/>
459     </qualifier>
460     <qualifier name="medium">
461       <headers JMSPriority="4"/>
462     </qualifier>
463     <qualifier name="low">
464       <headers JMSPriority="0"/>
465     </qualifier>
466   </intentMap>
467
468   <intentMap provides="log">
469     <qualifier>
470       <headers>
471         <property name="user_example_log">logged</property>
472       </headers>
473     </qualifier>
474   </intentMap>
475 </policySet>

```

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

```

477 <reference name="StockQuoteService" requires="priority.high log">
478   <interface.java interface="services.stockquote.StockQuoteService"/>
479   <binding.jms>
480     <destination name="StockQuoteServiceQueue"/>
481     <connectionFactory name="StockQuoteServiceQCF"/>
482   </binding.jms>

```

484
485

</reference>

486 2 JMS Binding Schema

```

487
488 <?xml version="1.0" encoding="UTF-8"?>
489 <!-- (c) Copyright SCA Collaboration 2006 -->
490 <schema xmlns="http://www.w3.org/2001/XMLSchema"
491   targetNamespace="http://www.osoa.org/xmlns/sca/1.0"
492   xmlns:sca="http://www.osoa.org/xmlns/sca/1.0"
493   elementFormDefault="qualified">
494
495   <include schemaLocation="sca-core.xsd"/>
496
497   <complexType name="JMSBinding">
498     <complexContent>
499       <extension base="sca:Binding">
500         <sequence>
501           <element name="destination" type="sca:Destination" minOccurs="0"/>
502           <element name="connectionFactory" type="sca:ConnectionFactory"
503             minOccurs="0"/>
504           <element name="activationSpec" type="sca:ActivationSpec"
505             minOccurs="0"/>
506           <element name="response" type="sca:Response" minOccurs="0"/>
507           <element name="headers" type="sca:Headers" minOccurs="0"/>
508           <element name="resourceAdapter" type="sca:ResourceAdapter"
509             minOccurs="0"/>
510           <element name="operationProperties" type="sca:OperationProperties"
511             minOccurs="0" maxOccurs="unbounded"/>
512           <any namespace="#other" processContents="lax"
513             minOccurs="0" maxOccurs="unbounded"/>
514         </sequence>
515         <attribute name="correlationScheme"
516           default="RequestMsgIDToCorrelID">
517           <simpleType>
518             <restriction base="string">
519               <enumeration value="RequestMsgIDToCorrelID"/>
520               <enumeration value="RequestCorrelIDToCorrelID"/>
521               <enumeration value="None"/>
522             </restriction>
523           </simpleType>
524         </attribute>
525
526         <attribute name="initialContextFactory" type="anyURI"/>
527         <attribute name="jndiURL" type="anyURI"/>
528         <attribute name="requestConnection" type="QName"/>
529         <attribute name="responseConnection" type="QName"/>
530         <attribute name="operationProperties" type="QName"/>
531         <anyAttribute/>
532       </extension>
533     </complexContent>
534   </complexType>
535
536   <simpleType name="CreateResource">
537     <restriction base="string">
538       <enumeration value="always"/>
539       <enumeration value="never"/>
540       <enumeration value="ifnotexist"/>
541     </restriction>

```

```

542     </simpleType>
543
544     <complexType name="Destination">
545         <sequence>
546             <element name="property" type="string"
547                 minOccurs="0" maxOccurs="unbounded" />
548         </sequence>
549         <attribute name="name" type="anyURI" use="required"/>
550         <attribute name="type" use="optional" default="queue">
551             <simpleType>
552                 <restriction base="string">
553                     <enumeration value="queue" />
554                     <enumeration value="topic" />
555                 </restriction>
556             </simpleType>
557         </attribute>
558         <attribute name="create" type="sca:CreateResource"
559                 use="optional" default="ifnotexist"/>
560     </complexType>
561
562     <complexType name="ConnectionFactory">
563         <sequence>
564             <element name="property" type="string"
565                 minOccurs="0" maxOccurs="unbounded" />
566         </sequence>
567         <attribute name="name" type="anyURI" use="required"/>
568         <attribute name="create" type="sca:CreateResource"
569                 use="optional" default="ifnotexist"/>
570     </complexType>
571
572     <complexType name="ActivationSpec">
573         <sequence>
574             <element name="property" type="string"
575                 minOccurs="0" maxOccurs="unbounded" />
576         </sequence>
577         <attribute name="name" type="anyURI" use="required"/>
578         <attribute name="create" type="sca:CreateResource"
579                 use="optional" default="ifnotexist"/>
580     </complexType>
581
582     <complexType name="Response">
583         <sequence>
584             <element name="destination" type="sca:Destination" minOccurs="0" />
585             <element name="connectionFactory" type="sca:ConnectionFactory"
586                 minOccurs="0" />
587             <element name="activationSpec" type="sca:ActivationSpec" minOccurs="0" />
588         </sequence>
589     </complexType>
590
591     <complexType name="Headers">
592         <sequence>
593             <element name="property" type="string"
594                 minOccurs="0" maxOccurs="unbounded" />
595         </sequence>
596         <attribute name="JMSType" type="string" />
597         <attribute name="JMSCorrelationID" type="string" />
598         <attribute name="JMSDeliveryMode" type="string" />
599         <attribute name="JMSTimeToLive" type="int" />
600         <attribute name="JMSPriority" type="string" />

```

```
601    </complexType>
602
603    <complexType name="ResourceAdapter">
604        <sequence>
605            <element name="property" type="string"
606                  minOccurs="0" maxOccurs="unbounded" />
607        </sequence>
608        <attribute name="name" type="string" use="required" />
609    </complexType>
610
611    <complexType name="OperationProperties">
612        <sequence>
613            <element name="property" type="string"
614                  minOccurs="0" maxOccurs="unbounded" />
615            <element name="headers" type="sca:Headers" />
616        </sequence>
617        <attribute name="name" type="string" use="required" />
618        <attribute name="nativeOperation" type="string" />
619    </complexType>
620
621    <element name="binding.jms" type="sca:JMSBinding"
622                  substitutionGroup="sca:binding" />
623 </schema>
```

624 3 References

625

626 [1] JMS Specification
627 <http://java.sun.com/products/jms/>

628

629 [2] Java Enterprise Edition 1.4 specification
630 <http://java.sun.com/j2ee/1.4/>

631

632 [3] WSDL Specification
633 WSDL 1.1: <http://www.w3.org/TR/wsdl>
634 WSDL 2.0: <http://www.w3.org/TR/wsdl20/>

635

636 [4] Java Connector Architecture Specification Version 1.5
637 <http://java.sun.com/j2ee/connector/>