Service Component Architecture JMS Binding Specification Version 1.1

Committee Draft 01 revision 4

21st January, 2009

Specification URIs:
This Version:
http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec-cd01-rev4.html
http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec-cd01-rev4.doc
http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec-cd01-rev4.pdf (Authoritative)

Previous Version:
http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd01.html
http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd01.doc
http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-cd01.pdf (Authoritative)

Latest Version:
http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec.html
http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec.doc
http://docs.oasis-open.org/opencsa/sca-bindings/sca-binding-jms-1.1-spec.pdf (Authoritative)

Latest Approved Version:

Technical Committee:
OASIS Service Component Architecture / Bindings (SCA-Bindings) TC

Chair(s):
Simon Holdsworth, IBM

Editor(s):
Simon Holdsworth, IBM
Khanderao Kand, Oracle
Anish Karmarkar, Oracle
Sanjay Patil, SAP
Piotr Przybylski, IBM

Related work:

This specification replaces or supercedes:

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

This specification is related to:

- Service Component Architecture Assembly Model Specification Version 1.1

- Service Component Architecture Policy Framework Specification Version 1.1

Declared XML Namespace(s):

http://docs.oasis-open.org/ns/opencsa/sca/200712

Abstract:

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

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

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

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

Status:

Notices

Table of Contents

1 Introduction

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

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

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

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

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

Table 1-1 Prefixes and Namespaces used in this specification

Prefix	***Namespace***	***Notes***
xs	"http://www.w3.org/2001/XMLSchema"	Defined by XML Schema 1.0 specification
sca	"http://docs.oasis-open.org/ns/opencsa/sca/200712"	Defined by the SCA specifications

1.2 Normative References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[JMS] JMS Specification http://java.sun.com/products/jms/

[WSDL] E. Christensen et al, *Web Service Description Language (WSDL) 1.1*, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, W3C Note, March 15 2001.

R. Chinnici et al, *Web Service Description Language (WSDL) Version 2.0 Part 1: Core Language*, http://www.w3.org/TR/2007/REC-wsdl20-20070626/, W3C Recommendation, June 26 2007.

[JCA15] Java Connector Architecture Specification Version 1.5 http://java.sun.com/j2ee/connector/

[IETFJMS] IETF URI Scheme for Java™ Message Service 1.0 http://www.ietf.org/internet-drafts/draft-merrick-jms-uri-05.txt[1]

[SCA-Assembly] http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.html

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

1.3 Non-Normative References

TBD TBD

WSDL

2 Messaging Bindings

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

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

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

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

3 JMS Binding Schema

The JMS binding element is defined by the following schema.

```
<binding.jms correlationScheme="QName"?
             initialContextFactory="xs:anyURI"?
             jndiURL="xs:anyURI"?
             requestConnection="QName"?
             responseConnection="QName"?
             operationProperties="QName"?
             name="NCName"?
             requires="list of QName"?
             uri="xs:anyURI"?
             ... >
    <destination jndiName="xs:anyURI" type="queue or topic"?
                 create="always or never or ifnotexist"?>
        <property name="NMTOKEN" type="NMTOKEN"?>*
    </destination>?
    <connectionFactory jndiName="xs:anyURI"
                       create="always or never or ifnotexist"?>
        <property name="NMTOKEN" type="NMTOKEN"?>*
    </connectionFactory>?
    <activationSpec jndiName="xs:anyURI"
                    create="always or never or ifnotexist"?>
        <property name="NMTOKEN" type="NMTOKEN"?>*
    </activationSpec>?

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

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

    <headers JMSType="string"?

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

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

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

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

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

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

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

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

  – ***jms:<jms-dest>?***
    connectionFactoryName=<Connection-Factory-Name> &
    destinationType={queue|topic}
    deliveryMode=<Delivery-Mode> &
    timeToLive=<Time-To-Live> &
    priority=<Priority> &
    selector=<Selector> &
    <User-Property>=<User-Property-Value> & …

  When the ***@uri*** attribute is specified, the SCA runtime MUST raise an error if the referenced resources do not already exist.
- ***/binding.jms/@name*** - as defined in the SCA Assembly specification in Section 9, "Binding"
- ***/binding.jms/@requires*** - as defined in the SCA Assembly specification in Section 9, "Binding"
- ***/binding.jms/@correlationScheme*** – identifies the correlation scheme used when sending reply or callback messages. Possible values for the ***@correlationScheme*** attribute are "***sca:MessageID***" (the default) where the SCA runtime MUST set the correlation ID of replies to the message ID of the corresponding request; "***sca:CorrelationID***" where the SCA runtime MUST set the correlation ID of replies to the correlation ID of the corresponding request, and "***sca:None***" which indicates that the SCA runtime MUST NOT set the correlation ID. SCA runtimes MAY allow other values to indicate other correlation schemes.
- ***/binding.jms/@initialContextFactory*** – the name of the JNDI initial context factory.
- ***/binding.jms/@jndiURL*** – the URL for the JNDI provider.
- ***/binding.jms/@requestConnection*** – identifies a ***binding.jms*** element that is present in a definition document, whose ***destination***, ***connectionFactory***, ***activationSpec*** and ***resourceAdapter*** children

are used to define the values for this binding. In this case this ***binding.jms*** element MUST NOT also contain the corresponding elements.

- ***/binding.jms/@responseConnection*** – identifies a ***binding.jms*** element that is present in a definition document, whose ***response*** child element is used to define the values for this binding. In this case this ***binding.jms*** element MUST NOT contain a ***response*** element.
- ***/binding.jms/@operationProperties*** – identifies a ***binding.jms*** element that is present in a definition document, whose ***operationProperties*** children are used to define the values for this binding. In this case this ***binding.jms*** element MUST NOT contain an ***operationProperties*** element.
- ***/binding.jms/destination*** – identifies the destination that is to be used to process requests by this binding.
- ***/binding.jms/destination/@type*** - the type of the request destination. Valid values are "***queue***" and "***topic***". The default value is "***queue***". In either case the runtime MUST ensure a single response is delivered for request/response operations.
-
- ***binding.jms/destination/@jndiName*** – the JNDI name of the JMS Destination that the binding uses to send or receive messages. The behaviour of this attribute is determined by the value of the ***@create*** attribute as follows:
  - If the ***@create*** attribute value is "always" then the ***@jndiName*** attribute is optional; if the destination cannot be created at the specified location then the SCA runtime MUST raise an error. If the ***@jndiName*** attribute is omitted this specification places no restriction on the JNDI location of the created resource.
  - If the ***@create*** attribute value is "ifnotexist" then the ***@jndiName*** attribute MUST specify the location of the possibly existing destination; if the destination does not exist at this location, but cannot be created there then the SCA runtime MUST raise an error. If the ***@jndiName*** refers to an existing resource other than a JMS Destination of the specified type then the SCA runtime MUST raise an error.
  - If the ***@create*** attribute value is "never" then the ***@jndiName*** attribute MUST specify the location of the existing destination; If the destination is not present at the location, or the location refers to a resource other than a JMS Destination of the specified type then the SCA runtime MUST raise an error.
- ***/binding.jms/destination/@create*** – indicates whether the destination should be created when the containing composite is deployed. Valid values are "***always***", "***never***" and "***ifnotexist***". The default value is "***ifnotexist***"..
- ***/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. The attributes of this element follow those defined for the ***destination*** element. A ***binding.jms*** element MUST NOT include both this element and an ***activationSpec*** element. When this element is present, the ***destination*** element MUST also be present
- ***/binding.jms/activationSpec*** – identifies the activation spec that the binding uses to connect to a JMS destination to process request messages. The attributes of this element follow those defined for the ***destination*** element. If a ***destination*** element is also specified it MUST refer to the same JMS destination as the ***activationSpec***. This element MUST NOT be present when the binding is being used for an SCA reference.
- ***/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. For a service, this destination is used to send responses to messages that have a null value for the ***JMSReplyTo*** destination. For a reference, this destination is used to receive reply messages

- ***/binding.jms/response/connectionFactory*** – identifies the connection factory that the binding uses to process response messages. The attributes of this element follow those defined for the ***destination*** element. A ***response*** element MUST NOT include both this element and an ***activationSpec*** element.
- ***/binding.jms/response/activationSpec*** – identifies the activation spec that the binding uses to connect to a JMS destination to process response messages. The attributes of this element follow those defined for the ***destination*** element. If a response ***destination*** element is also specified it MUST refer to the same JMS destination as the ***activationSpec***. This element MUST NOT be present when the binding is being used for an SCA service.
- ***/binding.jms/response/wireFormat*** – identifies the wire format used by responses sent or received by this binding. This value overrides the ***wireFormat*** specifed at the binding level.
- ***/binding.jms/headers*** – this element specifies values for standard JMS headers that the SCA runtime MUST set to the given values for all operations. These values apply to requests from a reference and responses from a service.
- ***/binding.jms/headers/@JMSType, @JMSDeliveryMode, @JMSTimeToLive, @JMSPriority*** – specifies the value to use for the JMS header property. The value of the ***@uri*** attribute MUST NOT include values for these properties if they are specified using these attributes. Valid values for ***@JMSDeliveryMode*** are "***PERSISTENT***" and "***NON_PERSISTENT***"; valid values for ***@JMSPriority*** are "***0***" to "***9***".
- ***/binding.jms/headers/property*** – specifies the value that the SCA runtime MUST set for the specified JMS user property when creating messages..
- ***/binding.jms/subscriptionHeaders*** - this element allows JMS subscription options to be set. These values apply to a service subscribing to the destination or for a reference subscribing to the callback or reply-to destinations.
- ***/binding.jms/subscriptionHeaders/@JMSSelector*** - specifies the value to use for the JMS selector. The value of the ***@uri*** attribute MUST NOT include values for this property if it is specified using this attribute.
- ***/binding.jms/resourceAdapter*** – specifies name, type and properties of the Resource Adapter Java bean. This element MUST be present when the JMS resources are to be created for a JMS provider that implements the JCA 1.5 specification JCA15, and is ignored otherwise. SCA runtimes MAY place restrictions on the properties of the RA Java bean that can be set. For JMS providers that do not implement the JCA 1.5 specification, information necessary for resource creation can be added in provider-specific elements or attributes allowed by the extensibility of the ***binding.jms*** element.
- ***/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/@selectedOperation*** – The value generated by the ***operationSelector*** that corresponds to the operation in the service or reference interface identified by the ***operationProperties/@name*** attribute. If this attribute is omitted then the value defaults to the value of the ***operationProperties/@name*** attribute. The value of this attribute MUST be unique across the containing ***binding.jms*** element..
- ***/binding.jms/operationProperties/property*** – specifies properties specific to this operation. These properties are intended to be used to parameterize the ***wireFormat*** identified for the binding for a particular operation. The SCA runtime SHOULD make the ***operationProperties*** element corresponding to the ***selectedOperation*** available to the ***wireFormat*** implementation.
- ***/binding.jms/operationProperties/headers*** – this element specifies values for standard JMS headers that the SCA runtime MUST set to the given values for the given operation. These values apply to requests from a reference and responses from a service.
- ***/binding.jms/operationProperties/headers/@JMSType, @JMSDeliveryMode, @JMSTimeToLive, @JMSPriority*** – specifies the value to use for the JMS header property. The

SCA runtime MUST use values specified for particular operations in preference to those defined for all operations in the ***binding.jms/headers*** element or via the binding's ***@uri*** attribute.

- ***/binding.jms/operationProperties/headers/property*** – specifies the value that the SCA runtime MUST set for the specified JMS user property when creating messages.
- ***/binding.jms/wireFormat*** – identifies the wire format used by requests and responses sent or received by this binding.
- ***/binding.jms/operationSelector*** – identifies the operation selector used when receiving requests for a service. If specified for a reference this provides the default operation selector for callbacks if not specified via a callback service element.
- ***/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.

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

4 Operation Selectors and Wire Formats

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

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

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

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

4.1 Default Operation Selection

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

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

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

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

4.2 Default Wire Format

The default wire format maps between a ***JMSMessage*** and the object(s) expected by the component implementation. We encourage component implementers to avoid exposure of JMS 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 that is a ***JMSMessage***, then the ***JMSMessage*** is passed as is.
- Otherwise, the ***JMSMessage*** must be a JMS text message or bytes message containing XML; an SCA runtime MUST be able to receive both forms. When sending messages either form may be used; an SCA runtime MAY provide additional configuration to allow one or other to be selected.

- If there is a single parameter, or for the return value, the JMS text or bytes XML payload is the XML serialization of that parameter according to the WSDL schema for the message.
- If there are multiple parameters, then they are encoded in XML using the document wrapped style, according to the WSDL schema for the message.
- When sending request messages, if there is a single parameter and the interface includes more than one operation, the SCA runtime MUST set the JMS user property "***scaOperationName***" to the name of the operation being invoked.

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

For example, for the following interface definition:

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

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

  <wsdl:portType name="Coordinates">
    <wsdl:operation name="setCoordinates">
      <wsdl:input message="tns:setCoordinatesRequestMsg"
name="setCoordinatesRequest"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

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

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

5 Policy

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

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

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

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

6 Message Exchange Patterns

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

6.1 One-way message exchange (no Callbacks)

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

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

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

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

6.2 Request/response message exchange (no Callbacks)

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

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

When a response message is sent by a service with a JMS binding for a request/response MEP, the SCA runtime MUST send the response message to the destination identified by the request message's ***JMSReplyTo*** header value if it is not null, otherwise the SCA runtime MUST send the response message to the destination identified by the JMS binding's ***response*** element if specified. If there is no destination defined by either means then an error SHOULD be raised by the SCA runtime. 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.

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

6.3 JMS User Properties

This protocol assigns specific behavior to JMS user properties:

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

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

6.4 Callbacks

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

6.4.1 Invocation of operations on a bidirectional interface

When a request message is sent by a reference with a JMS binding for a one-way MEP with a bidirectional interface, 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. The SCA runtime MAY also set the ***JMSReplyTo*** destination to this value.

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

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

6.4.2 Invocation of operations on a callback interface

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

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

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

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

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

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

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

6.5 Conversations

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

6.5.1 Starting a conversation

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

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

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

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

6.5.2 Continuing a conversation

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

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

6.5.3 Ending a conversation

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

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

7 Examples

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

7.1 Minimal Binding Example

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

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

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

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

7.2 URI Binding Example

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

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

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

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

7.3 Binding with Existing Resources Example

The following example shows the JMS binding using existing resources:

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

```
           name=”MyValueComposite”>

    <service name=”MyValueService”>
        <interface.java interface=”services.myvalue.MyValueService”/>
        <binding.jms>
            <destination jndiName=”MyValueServiceQ” create=”never”/>
            <activationSpec jndiName=”MyValueServiceAS” create=”never”/>
        </binding.jms>
    </service>
</composite>
```

7.4 Resource Creation Example

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

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

    <service name=”MyValueService”>
        <interface.java interface=”services.myvalue.MyValueService”/>
        <binding.jms>
            <destination jndiName=”MyValueServiceQueue” create=”always”>
                <property name=”prop1” type=”string”>XYZ</property>
                <property name=”destName” type=”string”>MyValueDest</property>
            </destination>
            <activationSpec jndiName=”MyValueServiceAS”/ create=”always”>
            <resourceAdapter jndiName=”com.example.JMSRA”/>
        </binding.jms>
    </service>

    <reference name=”StockQuoteService”>
        <interface.java interface=”services.stockquote.StockQuoteService”/>
        <binding.jms>
            <destination jndiName=”StockQuoteServiceQueue”/>
            <connectionFactory jndiName=”StockQuoteServiceQCF”/>
            <resourceAdapter name=”com.example.JMSRA”/>
        </binding.jms>
    </reference>
</composite>
```

7.5 Request/Response Example

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

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

    <service name=”MyValueService”>
        <interface.java interface=”services.myvalue.MyValueService”/>
        <binding.jms correlationScheme=”sca:MessageId”>
            <destination jndiName=”MyValueServiceQ” create=”never”/>
            <activationSpec jndiName=”MyValueServiceAS” create=”never”/>
        </binding.jms>
    </service>

    <reference name=”StockQuoteService”>
```

```
        <interface.java interface="services.stockquote.StockQuoteService"/>
        <binding.jms correlationScheme="sca:MessageId">
            <destination jndiName="StockQuoteServiceQueue"/>
            <connectionFactory jndiName="StockQuoteServiceQCF"/>
            <response>
                <destination jndiName="MyValueResponseQueue"/>
                <activationSpec jndiName="MyValueResponseAS"/>
            </response>
        </binding.jms>
    </reference>
</composite>
```

7.6 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 definitions file:

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

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

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
           xmlns:acme="http://acme.com"
           name="MyValueComposite">
    <reference name="MyValueService">
        <interface.java interface="services.myvalue.MyValueService"/>
        <binding.jms requestConnection="acme:StockQuoteService"/>
    </reference>
</composite>
```

7.7 Subscription with Selector Example

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

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200712"
           name="MyValueComposite">
    <service name="MyValueService">
        <interface.java interface="services.myvalue.MyValueService"/>
        <binding.jms>
            <destination jndiName="MyValueServiceTopic" create="never"/>
            <connectionFactory jndiName="StockQuoteServiceTCF" create="never"/
>
            <subscriptionHeaders JMSSelector="Price&gt;1000"/>
        </binding.jms>
    </service>
</composite>
```

7.8 Policy Set Example

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

```
<policySet name="JMSPolicy"
           provides="priority log"
           appliesTo="binding.jms">

    <intentMap provides="priority" default="medium">
        <qualifier name="high">
            <headers JMSPriority="9"/>
        </qualifier>
        <qualifier name="medium">
            <headers JMSPriority="4"/>
        </qualifier>
        <qualifier name="low">
            <headers JMSPriority="0"/>
        </qualifier>
    </intentMap>

    <intentMap provides="log">
        <qualifier>
            <headers>
                <property name="user_example_log">logged</property>
            </headers>
        </qualifier>
    </intentMap>
</policySet>
```

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

```
<reference name="StockQuoteService" requires="priority.high log">
    <interface.java interface="services.stockquote.StockQuoteService"/>
    <binding.jms>
        <destination name="StockQuoteServiceQueue"/>
        <connectionFactory name="StockQuoteServiceQCF"/>
    </binding.jms>
</reference>
```

8 Conformance

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

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

Within this specification, the following conformance targets are used:

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

A. JMS Binding Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) Copyright OASIS 2006, 2008 -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200712"
        xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200712"
        elementFormDefault="qualified">

   <include schemaLocation="sca-core.xsd"/>

   <complexType name="JMSBinding">
      <complexContent>
         <extension base="sca:Binding">
            <sequence>
               <choice minOccurs="0" maxOccurs="1">
                  <sequence>
                     <element name="destination" type="sca:JMSDestination"/>
                     <element name="connectionFactory"
                              type="sca:JMSConnectionFactory"/>
                  </sequence>
                  <sequence>
                     <element name="destination"
                              type="sca:JMSDestination" minOccurs="0"/>
                     <element name="activationSpec" type="sca:JMSActivationSpec"/>
                  </sequence>
               </choice>

               <element name="response" type="sca:JMSResponse" minOccurs="0"/>
               <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
               <element name="subscriptionHeaders "
                              type="sca:JMSSubscriptionHeaders"
                        minOccurs="0"/>
               <element name="resourceAdapter" type="sca:JMSResourceAdapter"
                        minOccurs="0"/>
               <element name="operationProperties"
                        type="sca:JMSOperationProperties"
                        minOccurs="0" maxOccurs="unbounded"/>
               <any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="correlationScheme" type="QName"
                       default="sca:MessageId"/>
            <attribute name="initialContextFactory" type="anyURI"/>
            <attribute name="jndiURL" type="anyURI"/>
            <attribute name="requestConnection" type="QName"/>
            <attribute name="responseConnection" type="QName"/>
            <attribute name="operationProperties" type="QName"/>
            <anyAttribute/>
         </extension>
      </complexContent>
   </complexType>

   <simpleType name="CreateResource">
      <restriction base="string">
         <enumeration value="always"/>
         <enumeration value="never"/>
         <enumeration value="ifnotexist"/>
      </restriction>
   </simpleType>

```

```
<complexType name="JMSDestination">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="jndiName" type="anyURI" use="required"/>
   <attribute name="type" use="optional" default="queue">
      <simpleType>
         <restriction base="string">
            <enumeration value="queue"/>
            <enumeration value="topic"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="create" type="sca:CreateResource"
              use="optional" default="ifnotexist"/>
</complexType>

<complexType name="JMSConnectionFactory">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="jndiName" type="anyURI" use="required"/>
   <attribute name="create" type="sca:CreateResource"
              use="optional" default="ifnotexist"/>
</complexType>

<complexType name="JMSActivationSpec">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="jndiName" type="anyURI" use="required"/>
   <attribute name="create" type="sca:CreateResource"
              use="optional" default="ifnotexist"/>
</complexType>

<complexType name="JMSResponse">
   <sequence>
      <element name="destination" type="sca:JMSDestination" minOccurs="0"/>
      <choice minOccurs="0">
         <element name="connectionFactory" type="sca:JMSConnectionFactory"/>
         <element name="activationSpec" type="sca:JMSActivationSpec"/>
      </choice>

   </sequence>
</complexType>


<complexType name="JMSHeaders">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="JMSType" type="string"/>

   <attribute name="JMSDeliveryMode">
      <simpleType>
         <restriction base="string">
            <enumeration value="PERSISTENT"/>
            <enumeration value="NON_PERSISTENT"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="JMSTimeToLive" type="long"/>
   <attribute name="JMSPriority">
      <simpleType>
         <restriction base="string">
            <enumeration value="0"/>
            <enumeration value="1"/>
            <enumeration value="2"/>
            <enumeration value="3"/>
            <enumeration value="4"/>
            <enumeration value="5"/>
            <enumeration value="6"/>
            <enumeration value="7"/>
            <enumeration value="8"/>
            <enumeration value="9"/>
         </restriction>
      </simpleType>
   </attribute>
</complexType>

<complexType name="JMSSubscriptionHeaders">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="JMSSelector" type="string"/>
</complexType>

<complexType name="JMSResourceAdapter">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="string" use="required"/>
</complexType>

<complexType name="JMSOperationProperties">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
      <element name="headers" type="sca:Headers"/>
   </sequence>
   <attribute name="name" type="string" use="required"/>
   <attribute name="nativeOperation" type="string"/>
</complexType>

<complexType name="BindingProperty">
   <simpleContent>
      <extension base="string">
         <attribute name="name" type="NMTOKEN"/>
         <attribute name="type" type="string" use="optional"
                    default="xs:string"/>
      </extension>
   </simpleContent>
</complexType>

<element name="binding.jms" type="sca:JMSBinding"
         substitutionGroup="sca:binding"/>

<element name="wireFormat.jmsdefault" type="sca:WireFormatType"
         substitutionGroup="sca:wireFormat"/>
<element name="operationSelector.jmsdefault" type="sca:OperationSelectorType"
         substitutionGroup="sca:operationSelector"/>
</schema>
```

B. Acknowledgements

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

Participants:

[Participant Name, Affiliation | Individual Member]

[Participant Name, Affiliation | Individual Member]

C. Non-Normative Text

904D. Revision History

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

906

Revision	Date	Editor	Changes Made
1	2007-09-25	Anish Karmarkar	Applied the OASIS template + related changes to the Submission
2	2008-03-12	Simon Holdsworth	Updated text for RFC2119 conformance Updates to resolve following issues: BINDINGS-1 BINDINGS-5 BINDINGS-6 BINDINGS-12 BINDINGS-14 BINDINGS-18 BINDINGS-26 Applied updates discussed at Bindings TC meeting of 27th March
3	2008-06-19	Simon Holdsworth	* Applied most of the editorial changes from Eric Johnson's review
cd01	2008-08-01	Simon Holdsworth	Updates to resolve following issues: BINDINGS-13 (JMS part) BINDINGS-20 (complete) BINDINGS-30 (JMS part) BINDINGS-32 (JMS part) BINDINGS-33 (complete) BINDINGS-34 (complete) BINDINGS-35 (complete) BINDINGS-38 (JMS part)
cd01-rev1	2008-10-16	Simon Holdsworth	Updated text for RFC2119 conformance throughout Updates to resolve following issues: BINDINGS-41 BINDINGS-46 BINDINGS-47
cd01-rev2	2008-12-01	Simon Holdsworth	Added comments identifying those updates that relate to RFC2119 language (issue 52)
cd01-rev3	2008-12-02	Simon Holdsworth	Final RFC2119 language updates BINDINGS-52
cd01-rev4	2009-01-09	Simon Holdsworth	Updates to resolve following issues:

			BINDINGS-7 BINDINGS-31 BINDINGS-40 BINDINGS-42 BINDINGS-44 BINDINGS-50

907