Service Component Architecture JMS Binding Specification Version 1.1

Committee Draft 01

1st August, 2008

Specification URIs:
This 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)

Previous Version:

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

Latest Approved Version:

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

Chair(s):
Simon Holdsworth, IBM

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

Related work:
This specification replaces or supercedes:

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

This specification is related to:

- Service Component Architecture Assembly Model Specification Version 1.1
- Service Component Architecture Policy Framework Specification Version 1.1

Declared XML Namespace(s):
http://docs.oasis-open.org/ns/opencsa/sca/200712

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

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

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

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

Status:

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

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

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

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

Notices

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

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].

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-03.txt[1]

1.3 Non-Normative References

TBD	TBD

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

2 Operation Selection and Data Binding

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 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 some means is required in order to identify the specific operation and map the message information in to the required form.

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

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

3 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 data binding components 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.

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

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

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

    <headers JMSType="string"?
             JMSCorrelationId="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"?
```

```
                JMSCorrelationId="string"?
                JMSDeliveryMode="PERSISTENT or NON_PERSISTENT"?
                JMSTimeToLive="long"?
                JMSPriority="0 .. 9"?>
            <property name="NMTOKEN" type="NMTOKEN"?>*
        </headers>?
    </operationProperties>*
</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 IEFT 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, it is assumed that the referenced resources 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 correlation ID of replies is set to the message ID of the corresponding request; "***sca:CorrelationID***" where the correlation ID of replies is set to the correlation ID of the corresponding request, and "***sca:None***" which indicates that the correlation ID is not set. 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/@name*** – the name of the destination to which the binding is connected. This may be a JNDI name or a plain destination name.
- ***/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***". If "***always***" is specified and the corresponding resource already exists, then the SCA runtime SHOULD considered this as 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 can either be a JNDI name or a plain connection factory name. 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.
- ***/binding.jms/activationSpec*** – identifies the activation spec that the binding uses to connect to a JMS destination to process request messages. This can either 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 can either be a JNDI name or a plain connection factory name. 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. This can either 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. 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 to use for the specified JMS user property.
- ***/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 is required 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/@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 in the ***binding.jms/headers*** element or via the binding's ***@uri*** attribute.
- ***/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.

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 generate an error for this invalid combination at deployment time.

5 Default Operation Selection and Data Binding behavior

This section describes the default behavior for operation selection and data binding for a JMS binding. The SCA runtime MUST support this default behaviour, and MAY provide additional means to override it.

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, if the message is a JMS text or bytes message containing XML, then the operation name is taken from the local name of the root element of the XML payload.
- Otherwise, the native operation name is assumed to be "***onMessage***".

The native 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 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 name of the operation being invoked.

The SCA runtime MAY provide the means for supplying and identifying alternative function selection behaviors.

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 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.
- 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.

The SCA runtime SHOULD provide the means for supplying and identifying alternative data binding behaviors to support any other type of JMS message.

6 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"/>
```

7 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.

7.1 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 treat this as an error.

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

7.2 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 recorded 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.

7.3 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.

7.4 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.

7.4.1 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.

7.4.2 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 record 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.

7.4.3 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.

7.5 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.

7.5.1 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.

7.5.2 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 consider 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, as an error, and MUST NOT deliver such messages.

7.5.3 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 report 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.

8 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.

8.1 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>
```

8.2 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>
```

8.3 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 name="MyValueServiceQ" create="never"/>
            <activationSpec name="MyValueServiceAS" create="never"/>
        </binding.jms>
    </service>
</composite>
```

8.4 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 name="MyValueServiceQueue" create="always">
                <property name="prop1" type="string">XYZ</property>
            </destination>
            <activationSpec name="MyValueServiceAS"/ create="always">
            <resourceAdapter name="com.example.JMSRA"/>
        </binding.jms>
    </service>

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

8.5 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 name="MyValueServiceQ" create="never"/>
            <activationSpec name="MyValueServiceAS" create="never"/>
        </binding.jms>
    </service>

    <reference name="StockQuoteService">
```

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

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 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 name="StockQuoteServiceQueue" create="never"/>
        <connectionFactory name="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>
```

8.7 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 name="MyValueServiceTopic" create="never"/>
            <connectionFactory name="StockQuoteServiceTCF" create="never"/>
            <subscriptionHeaders JMSSelector="Price&gt;1000"/>
        </binding.jms>
    </service>
</composite>
```

8.8 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>
```

9 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>
               <element name="destination" type="sca:Destination" minOccurs="0"/>
               <element name="connectionFactory" type="sca:ConnectionFactory"
                        minOccurs="0"/>
               <element name="activationSpec" type="sca:ActivationSpec"
                        minOccurs="0"/>
               <element name="response" type="sca:Response" minOccurs="0"/>
               <element name="headers" type="sca:Headers" minOccurs="0"/>
               <element name="subscriptionHeaders " type="sca:SubscriptionHeaders"
                        minOccurs="0"/>
               <element name="resourceAdapter" type="sca:ResourceAdapter"
                        minOccurs="0"/>
               <element name="operationProperties" type="sca:OperationProperties"
                        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="Destination">
      <sequence>
         <element name="property" type="sca:BindingProperty"
                  minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="name" 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="ConnectionFactory">
   <sequence>
      <element name="property" type="sca:BindingProperty"
               minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="anyURI" use="required"/>
   <attribute name="create" type="sca:CreateResource"
              use="optional" default="ifnotexist"/>
</complexType>

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

<complexType name="Response">
   <sequence>
      <element name="destination" type="sca:Destination" minOccurs="0"/>
      <element name="connectionFactory" type="sca:ConnectionFactory"
               minOccurs="0"/>
      <element name="activationSpec" type="sca:ActivationSpec" minOccurs="0"/>
   </sequence>
</complexType>

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

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

   <complexType name="OperationProperties">
      <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"/>
</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

D. Revision History

[optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
1	2007-09-25	Anish Karmarkar	Applied the OASIS template + related changes to the Submission
2	2008-03-12	Simon Holdsworth	Updated text for RFC2119 conformance Updates to resolve following issues: BINDINGS-1 BINDINGS-5 BINDINGS-6 BINDINGS-12 BINDINGS-14 BINDINGS-18 BINDINGS-26 Applied updates discussed at Bindings TC meeting of 27th March
3	2008-06-19	Simon Holdsworth	* Applied most of the editorial changes from Eric Johnson's review
4	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)