Service Component Architecture JMS Binding Specification Version 1.1

Committee Specification Draft 06 / Public Review Draft 04

14 July 2011

Specification URIs:

This version:

http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd04.pdf (Authoritative)
http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd04.html
http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd04.doc

Previous version:

http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd03.pdf (Authoritative)
http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd03.html
http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd03.doc

Latest version:

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

Technical Committee:

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

Chair:

Simon Holdsworth (simon_holdsworth@uk.ibm.com), IBM

Editors:

Simon Holdsworth (simon_holdsworth@uk.ibm.com), IBM
Anish Karmarkar (Anish.Karmarkar@oracle.com), Oracle

Related work:

This specification replaces or supersedes:

- Service Component Architecture JMS Binding Specification Version 1.00

This specification is related to:

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

Declared XML namespace:

http://docs.oasis-open.org/ns/opencsa/sca/200912

Abstract:

This document specifies the means by which SCA composites and components, as defined in the SCA Assembly Specification **[SCA-Assembly]**, connect to and access services using a messaging protocol. The connectivity is based on the Java Messaging Service **[JMS]** and is provided by a binding.jms element which applies to the references and services of an SCA component or composite.

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.

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.

Status:

Citation format:

When referencing this specification the following citation format should be used:

[SCA-JMSBinding]

Service Component Architecture JMS Binding Specification Version 1.1. 14 July 2011. OASIS Committee Specification Draft 06 / Public Review Draft 04. http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-spec-csprd04.html.

Notices

Table of Contents

1 Introduction

This document specifies the means by which SCA composites and components, as defined in the SCA Assembly Specification **[SCA-Assembly]**, connect to and access services using a messaging protocol. The connectivity is based on the Java Messaging Service **[JMS]** and is provided by a binding.jms element which applies to the references and services of an SCA component or composite.

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.

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.

1.1 Terminology

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

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

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/200912"	Defined by the SCA specifications

Table 1-1: Prefixes and Namespaces used in this specification

1.2 Normative References

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

[JMS] Java™ Message Service Specification v1.1 http://www.oracle.com/technetwork/java/jms/index.html

[JNDI] Java™ Naming and Directory Interface http://www.oracle.com/technetwork/java/jndi/index.html

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

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

[JCA15] J2EE Connector Architecture Specification Version 1.5 http://java.sun.com/j2ee/connector/

[IETFJMS] M. Phillips, P. Adams, D. Rokicki, E. Johnson, *URI Scheme for Java™ Message Service 1.0*, http://www.ietf.org/rfc/rfc6167.txt, IETF RFC 6167, April 2011.

[SCA-Assembly] OASIS Committee Specification Draft 07, *Service Component Architecture Assembly Model Specification Version 1.1*, January 2011 http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-csd07.pdf

[SCA-Policy] OASIS Committee Draft 04, *SCA Policy Framework Specification Version 1.1*, September 2010 http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd04.pdf

1.3 Non-Normative References

[SCA-JMSTest] OASIS Committee Specification Draft 01, *SCA JMS Binding v1.1 TestCases Version 1.0*, November 2010, http://docs.oasis-open.org/opencsa/sca-bindings/sca-jmsbinding-1.1-testcases-1.0-csd01.pdf

1.4 Naming Conventions

The naming conventions used by artefacts defined in this specification are:

- The naming conventions defined by section 1.3 of the SCA Assembly Specification [SCA-Assembly].
- Where the names of elements and attributes consist partially or wholly of acronyms, the letters of the acronyms use the same case. When the acronym appears at the start of the name of an element or an attribute, or after a period, it is in lower case. If it appears elsewhere in the name of an element or an attribute, it is in upper case. For example, an attribute might be named "uri" or "jndiURL".
- Where the names of types consist partially or wholly of acronyms, the letters of the acronyms are in all upper case. For example, an XML Schema type might be named "JCABinding" or "MessageID".
- Values, including local parts of QName values, follow the rules for names of elements and attributes as stated above, with the exception that the letters of acronyms are in all upper case. For example, a value might be "JMSDefault" or "namespaceURI".

1.5 Testcases

SCA JMS Binding TestCases Version 1.1 **[SCA-JMSTest]** defines test cases for this specification. The TestCases represent a series of tests that SCA runtimes are expected to pass in order to claim conformance to the requirements of this specification.

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 specified.

In addition, each operation in the interface associated with the service or reference can have properties specified, that 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 pseudo-schema in Snippet 3-1.

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

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

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

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

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

    <operationProperties name="string" selectedOperation="string"?>
        <property name="NMTOKEN" type="string"?>*
        <headers type="string"?
                 deliveryMode="persistent or nonpersistent"?
                 timeToLive="long"?
                 priority="0 .. 9"?>
            <property name=”NMTOKEN” type=”boolean or byte or .. or String”?>*
        </headers>?
    </operationProperties>*

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

Snippet 3-1: binding.jms Pseudo-Schema

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

The `binding.jms` element has the attributes:

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

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

  Snippet 3-2 illustrates the structure of the URI and the set of property names that have specific semantics:

  ```
  jms:jndi:<jms-dest>?
  jndiURL=<jndi-url> &
  jndiInitialContextFactory=<jndi-initial-context-factory> &
  jndiConnectionFactoryName=<Connection-Factory-Name> &
  deliveryMode=<Delivery-Mode> &
  timeToLive=<Time-To-Live> &
  priority=<Priority> &
  selector=<Message-Selector> &
  <param-name>=<param-value> & …
  ```

Snippet 3-2: JMS URI Structure

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

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

- ***/binding.jms/@name*** - as defined in the SCA Assembly Specification [SCA-Assembly].
- ***/binding.jms/@requires*** - as defined in the SCA Assembly Specification [SCA-Assembly].
- ***/binding.jms/@policySets*** - as defined in the SCA Assembly Specification [SCA-Assembly].
- ***/binding.jms/@correlationScheme*** – identifies the correlation scheme used when sending reply or callback messages, default value is “`sca:messageID`”. Three specific behaviours are provided. "`sca:messageID`" indicates that response messages can be correlated with their requests by looking for the request's messageID header value in the response's correlationID header; "`sca:correlationID`" indicates that response messages can be correlated with their requests by looking for the request's correlationID header value in the response's correlationID header; "`sca:none`" indicates that the response's correlationID header is not to be used for this purpose and some other means is used for the correlation.

  If the value of the `@correlationScheme` attribute is “`sca:messageID`” the SCA runtime MUST set the correlation ID of replies to the message ID of the corresponding request [BJM30003].

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

If the value of the `@correlationScheme` attribute is "`sca:correlationID`" the SCA runtime MUST set a non-null correlation ID value in requests that it sends [BJM30007].

If the value of the `@correlationScheme` attribute is "`sca:none`" the SCA runtime MUST NOT set the correlation ID in responses that it sends[BJM30005].

SCA runtimes supporting other correlation schemes can allow additional values for the `@correlationScheme` attribute.

- ***/binding.jms/@initialContextFactory*** – the name of the JNDI initial context factory.
- ***/binding.jms/@jndiURL*** – the URL for the JNDI provider.
- ***/binding.jms/destination*** – identifies the destination that is to be used to process requests by this binding.
- ***/binding.jms/destination/@type*** - the type of the request destination. Valid values are "`queue`" and "`topic`". The default value is "`queue`".

  Whatever the value of the `destination/@type` attribute, the SCA runtime MUST ensure a single response is delivered for request/response operations [BJM30010].

- ***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 for a `destination`, `connectionFactory` or `activationSpec` element is "`always`" and the `@jndiName` attribute is present and the resource cannot be created at the location specified by the `@jndiName` attribute then the SCA runtime MUST raise an error [BJM30011].

    If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "always" and the `@jndiName` attribute is not present and the resource cannot be created, then the SCA runtime MUST raise an error [BJM30037].

    If the `@jndiName` attribute is omitted this specification places no restriction on the JNDI location of the created resource.
  - If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" then the `@jndiName` attribute MUST specify the location of the possibly existing resource [BJM30012].

    If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" and the resource does not exist at the location identified by the `@jndiName` attribute and cannot be created there then the SCA runtime MUST raise an error [BJM30013].

    If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" and the `@jndiName` attribute refers to an existing resource that is not a JMS Destination of the approprate type, a JMS connection factory or a JMS activation spec respectively then the SCA runtime MUST raise an error [BJM30014].
  - If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`never`" and the `@jndiName` attribute is not specified, or the resource is not present at the location identified by the `@jndiName` attribute, or the location refers to a resource of an incorrect type then the SCA runtime MUST raise an error [BJM30015].
- ***/binding.jms/destination/@create*** – indicates whether the destination should be created when the containing composite is deployed. Valid values are "`always`", "`never`" and "`ifNotExist`". `"always"` indicates that new resources are created for use by this binding; `"never"` indicates that existing resources are used and none created; `"ifNotExist"` indicates that if the resources

already exist those are used, otherwise new ones are created. Refer to the `destination/@jndiName` attribute for a detailed definition of each case. 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 the rules defined for the `destination` element.

  A `binding.jms` element MUST NOT include both a `connectionFactory` element and an `activationSpec` element [BJM30017].

  When the `connectionFactory` element is present as a child of the `binding.jms` element, then the destination MUST be defined either by the `destination` element child of the `binding.jms` element or the `@uri` attribute of the `binding.jms` element [BJM30018].

- ***/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 the rules defined for the `destination` element.

  If the `activationSpec` element is present as a child of the `binding.jms` element and the destination is also specified via a `destination` element child of the `binding.jms` element or the `@uri` attribute of the `binding.jms` element then it MUST refer to the same JMS destination as the `activationSpec` [BJM30019].

  The `activationSpec` element MUST NOT be present when the binding is being used for an SCA reference [BJM30020].

- ***/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 follow the rules defined 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 a `connectionFactory` element and an `activationSpec` element [BJM30021].

- ***/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` and `response/activationSpec` element are both specified they MUST refer to the same JMS destination [BJM30022].

  The `response/activationSpec` element MUST NOT be present when the binding is being used for an SCA service [BJM30023].

- ***/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. Wire formats for this binding are described in Section 4.
- ***/binding.jms/headers*** – this element specifies values to be set for standard JMS headers. These values apply to requests from a reference and responses from a service. Section 3.2 defines the priority rules for determining the values for JMS headers and user properties.
- ***/binding.jms/headers/@type, @deliveryMode, @timeToLive, @priority*** – specifies the value to use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive or JMSPriority

respectively. Valid values for `@deliveryMode` are `"persistent"` and `"nonpersistent"`, corresponding to the values defined in the JMS Specification **[JMS]** for the JMSDeliveryMode message header, with `"persistent"` being the default; valid values for `@priority` are `"0"` to `"9"`, where `"0"` indicates lowest priority and `"9"` highest priority, with `"4"` being the default; valid values for `@timeToLive` are positive integers, with 0 indicating unlimited time and being the default value.

- ***/binding.jms/headers/property*** – specifies the value and type for the named JMS user property.
- ***/binding.jms/messageSelection*** - this element specifies JMS message selection options. This element applies to a service receiving messages from the request destination or for a reference receiving messages from the callback or reply-to destination.
- ***/binding.jms/messageSelection/@selector*** - specifies the value to use for the JMS message selector. Section 3.3 defines the priority rules for determining the values for the message selector.
- ***/binding.jms/resourceAdapter*** – specifies name, type and properties of the Resource Adapter Java bean. The resource adapter and SCA runtime together define the set of valid properties for configuring the resource adapter via the JMS binding.

  The `resourceAdapter` element MUST be present when JMS resources are to be created for a JMS provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise [BJM30031].

  For JMS providers that do not implement the JCA 1.5 specification [JCA15], 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 the ***operationProperties/@selectedOperation*** attribute MUST be unique across the containing `binding.jms` element [BJM30029].

- ***/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.
- ***/binding.jms/operationProperties/headers*** – this element specifies values to be set for standard JMS headers. These values apply to requests from a reference and responses from a service. Section 3.2 defines the priority rules for determining the values for JMS headers and user properties.
- ***/binding.jms/operationProperties/headers/@type, @deliveryMode, @timeToLive, @priority*** – specifies the value to use for the JMS header property JMSType, JMSDeliveryMode, JMSTimeToLive or JMSPriority, respectively. Refer to the description of the `binding.jms/headers` element for the valid values for these attributes.
- ***/binding.jms/operationProperties/headers/property*** – specifies the value and type for the named JMS user property.
- ***/binding.jms/wireFormat*** – identifies the wire format used by requests and responses sent or received by this binding. Wire formats for this binding are described in Section 4.
- ***/binding.jms/operationSelector*** – identifies the operation selector used when receiving requests for a service. If specified for a reference this provides the default operation selector for callbacks if not specified via a callback service element. Operation selectors for this binding are described in Section 3.2.

The `binding.jms` element MUST conform to the XML schema defined in sca-binding-jms-1.1.xsd [BJM30036].

3.1 Extensibility

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

3.2 JMS Message Headers and User Properties

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

The type of the JMS user property is specified via the property/@type attribute using one of the values define in the JMS specification **[JMS]**: "boolean", "byte", "short", "int", "long", "float", "double", "String" (the default), or "xs:string". "xs:string" and "String" both represent the String user property type, "xs:string" is for backward compatibility only and its use is deprecated.

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

1) the value for the header specified in the `@uri` attribute (highest priority);

2) the value for the header specified in the `operationProperties/headers` element matching the operation being invoked;

3) the value for the header specified in the `headers` element;

4) the default value for the header as specified by the definition of the `binding.jms/headers` element (lowest priority) [BJM30024].

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

1) the type and value for the named user property specified in an `operationProperties/headers/property` element matching the name of the operation being invoked (highest priority);

2) the type and value for the named user property specified in a `headers/property` element (lowest priority) [BJM30025].

3.3 JMS Message Selection

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

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

1) the value for the message selector specified in the `@uri` attribute value’s “selector” parameter (highest priority);

2) the value for the message selector specified in the `messageSelection/@selector` attribute;

3) otherwise no message selector is used (lowest priority) [BJM30026].

4 Operation Selectors and Wire Formats

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

The process of identifying the operation to be invoked is *operation selection*; the information that describes the contents of messages is a *wire format*. The `binding` element as described in the SCA Assembly Specification [SCA-Assembly] provides the means to identify specific operation selection via the `operationSelector` element and the wire format of messages received and to be sent using the `wireFormat` element. The `operationSelector` and `wireFormat` elements allow a binding element to specify behaviour defined by the binding specification or custom behaviour provided by an SCA runtime.

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

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

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

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

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

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

4.1 Default Operation Selection

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

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

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

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

4.2 Default Wire Format

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

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

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

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

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

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

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

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

The default wire format allows a choice of text or bytes format when sending messages; an SCA runtime can restrict this to one or other via additional configuration.

4.2.1 Example of default wire format

For the interface definition in Snippet 4-1:

```
<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>
```

Snippet 4-1: Example WSDL Interface Definition

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 content shown in Snippet 4-2:

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

Snippet 4-2: JMS Message Content for setCoordinates Operation of Snippet 4-1

5 Policy

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

JMS binding implementations MUST support the JMS intent [BJM50001].

The JMS intent MUST always be included in the `@alwaysProvides` attribute of the JMS `bindingType` [BJM50002]

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

- atLeastOnce
- atMostOnce
- ordered

The `atLeastOnce`, `atMostOnce` and `ordered` intents are defined in the SCA Policy Specification [SCA-Policy] document in section 8, "Reliability Policy".

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

6 Message Exchange Patterns

This section describes the message exchange patterns that are possible when using the JMS binding, including one-way, request/response and callbacks. JMS [JMS] has a looser concept of message exchange patterns than WSDL, so this section explains how JMS messages that are sent and received by the SCA runtime relate to the WSDL input/output messages. Each operation in a WSDL interface is either one-way or request/response. Callback interfaces can 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. The JMS specification provides the `JMSReplyTo` header as the way for a JMS application to identify the destination on which replies or other messages are to be placed that relate to the one being sent. For one-way requests sent by SCA references with unidirectional interfaces, the `JMSReplyTo` will not usually be set as no reply or other related message is expected.

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

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

6.2 Request/response message exchange (no Callbacks)

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

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

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

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

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

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

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

For an SCA service with a JMS binding, when a request message is received as part of a request/response MEP where the request message includes a null `JMSReplyTo` destination and the JMS binding does not include a response/destination then the SCA runtime MUST NOT process the request and MUST raise an error [BJM60009].

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

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

6.3 JMS User Properties

This protocol assigns specific behavior to JMS user properties:

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

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 (for request/response) or the `JMSReplyTo` destination (for one-way) 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.

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

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

6.4.1 Invocation of operations on a bidirectional interface

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

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

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

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

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

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, the `JMSReplyTo` destination, or the destination identified by the service's callback reference JMS binding.

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

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

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

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

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

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 with a one-way operation in the forward and callback interfaces. 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 is used to identify the destination to be used to deliver callback messages, as described in sections 6.4.1 and 6.4.2.

7 Examples

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

7.1 Minimal Binding Example

Snippet 7-1 shows the JMS binding being used with no further attributes or elements. In this case, it is left to the deployer to identify the resources to which the binding is connected.

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           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>
```

Snippet 7-1: Minimal Binding Example

7.2 URI Binding Example

Snippet 7-2 shows the JMS binding using the `@uri` attribute to specify the connection type and its information:

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           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>
```

Snippet 7-2: Binding Example with URI Specified

7.3 Binding with Existing Resources Example

Snippet 7-3 shows the JMS binding using existing resources:

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
```

```
           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>
```

Snippet 7-3: Binding Example Using Existing Resources

7.4 Resource Creation Example

Snippet 7-4 shows the JMS binding providing information to create JMS resources rather than using existing ones:

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           name="MyValueComposite">

    <service name="MyValueService">
        <interface.java interface="services.myvalue.MyValueService"/>
        <binding.jms>
            <destination jndiName="MyValueServiceQueue" create="always">
                <property name="prop1">XYZ</property>
                <property name="destName">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>
```

Snippet 7-4: Binding Example that Creates a Resource

7.5 Request/Response Example

Snippet 7-5 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="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           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>
```

Snippet 7-5: Binding Example with a Response

7.6 Subscription with Selector Example

Snippet 7-6 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="UTF-8"?>
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           name="MyValueComposite">
    <service name="MyValueService">
        <interface.java interface="services.myvalue.MyValueService"/>
        <binding.jms>
            <destination jndiName="MyValueServiceTopic" create="never"/>
            <connectionFactory jndiName="StockQuoteServiceTCF"
                create="never"/>
            <messageSelection selector="Price&gt;1000"/>
        </binding.jms>
    </service>
</composite>
```

Snippet 7-6: Binding Example with a Selector

7.7 Policy Set Example

A policy set defines the manner in which intents map to JMS binding properties. Snippet 7-7 illustrates an example of a policy set that defines values for the `@priority` attribute using the "`priority`" intent, and also allows setting of a value for a user JMS property using the "`log`" intent.

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

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

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

Snippet 7-7: Example Policy Set

Given the policy set in Snippet 7-7, the intents can be required on a service or reference as shown in Snippet 7-8:

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

Snippet 7-8: Binding Example with Intents

8 Conformance

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

a) SCA JMS Binding XML Document

b) SCA Runtime

8.1 SCA JMS Binding XML Document

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

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

8.2 SCA Runtime

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

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright(C) OASIS(R) 2005,2010. All Rights Reserved.
     OASIS trademark, IPR and other policies apply.  -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
        xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912"
        elementFormDefault="qualified">

   <include schemaLocation="sca-core-1.1-cd05.xsd"/>

   <complexType name="JMSBinding">
      <complexContent>
         <extension base="sca:Binding">
            <sequence>
               <element name="destination" type="sca:JMSDestination"
                        minOccurs="0"/>
               <choice minOccurs="0" maxOccurs="1">
                  <element name="connectionFactory"
                           type="sca:JMSConnectionFactory"/>
                  <element name="activationSpec" type="sca:JMSActivationSpec"/>
               </choice>
               <element name="response" type="sca:JMSResponse" minOccurs="0"/>
               <element name="headers" type="sca:JMSHeaders" minOccurs="0"/>
               <element name="messageSelection" type="sca:JMSMessageSelection"
                        minOccurs="0"/>
               <element name="resourceAdapter" type="sca:JMSResourceAdapter"
                        minOccurs="0"/>
               <element name="operationProperties"
                        type="sca:JMSOperationProperties"
                        minOccurs="0" maxOccurs="unbounded"/>
               <element ref="sca:extensions" minOccurs="0" maxOccurs="1"/>
            </sequence>
            <attribute name="correlationScheme" type="QName"
                       default="sca:messageID"/>
            <attribute name="initialContextFactory" type="anyURI"/>
            <attribute name="jndiURL" type="anyURI"/>
         </extension>
      </complexContent>
   </complexType>

   <simpleType name="JMSCreateResource">
      <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"/>
      <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:JMSCreateResource"
                 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"/>
      <attribute name="create" type="sca:JMSCreateResource"
                 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"/>
      <attribute name="create" type="sca:JMSCreateResource"
                 use="optional" default="ifNotExist"/>
   </complexType>

   <complexType name="JMSResponse">
      <sequence>
         <element ref="sca:wireFormat" minOccurs="0" maxOccurs="1"/>
         <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:JMSUserProperty"
                  minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="type" type="string"/>
      <attribute name="deliveryMode" default="persistent">
         <simpleType>
            <restriction base="string">
               <enumeration value="persistent"/>
               <enumeration value="nonpersistent"/>
            </restriction>
         </simpleType>
      </attribute>
      <attribute name="timeToLive" type="long" default="0"/>
      <attribute name="priority" default="4">
         <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="JMSMessageSelection">
       <sequence>
          <element name="property" type="sca:BindingProperty"
                   minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
       <attribute name="selector" 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:JMSHeaders" minOccurs="0"/>
       </sequence>
       <attribute name="name" type="string" use="required"/>
       <attribute name="selectedOperation" type="string"/>
    </complexType>

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

    <simpleType name="JMSUserPropertyType">
       <restriction base="string">
          <enumeration value="boolean"/>
          <enumeration value="byte"/>
          <enumeration value="short"/>
          <enumeration value="int"/>
          <enumeration value="long"/>
          <enumeration value="float"/>
          <enumeration value="double"/>
          <enumeration value="String"/>
          <enumeration value="xs:string"/>
       </restriction>
    </simpleType>

    <complexType name="JMSUserProperty">
       <simpleContent>
          <extension base="string">
             <attribute name="name" type="NMTOKEN" use="required"/>
             <attribute name="type" type="sca:JMSUserPropertyType"
                        use="optional" default="String"/>
          </extension>
```

```
      </simpleContent>
   </complexType>

   <complexType name="JMSDefaultWireFormatType">
      <complexContent>
         <extension base="sca:WireFormatType"/>
      </complexContent>
   </complexType>

   <complexType name="JMSDefaultOperationSelectorType">
      <complexContent>
         <extension base="sca:OperationSelectorType"/>
      </complexContent>
   </complexType>

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

   <element name="wireFormat.jmsDefault"
            type="sca:JMSDefaultWireFormatType"
            substitutionGroup="sca:wireFormat"/>

   <element name="operationSelector.jmsDefault"
            type="sca:JMSDefaultOperationSelectorType"
            substitutionGroup="sca:operationSelector"/>
</schema>
```

B. Conformance Items

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

Conformance ID	Description
[BJM30001]	The value of the `@uri` attribute MUST have the format defined by the IETF URI Scheme for Java™ Message Service 1.0 [IETFJMS]
[BJM30002]	When the `@uri` attribute is specified, the SCA runtime MUST raise an error if the referenced resources do not already exist
[BJM30003]	If the value of the `@correlationScheme` attribute is “`sca:messageID`” the SCA runtime MUST set the correlation ID of replies to the message ID of the corresponding request
[BJM30004]	If the value of the `@correlationScheme` attribute is “`sca:correlationID`” the SCA runtime MUST set the correlation ID of replies to the correlation ID of the corresponding request
[BJM30005]	If the value of the `@correlationScheme` attribute is “`sca:none`” the SCA runtime MUST NOT set the correlation ID in responses that it sends
[BJM30007]	If the value of the `@correlationScheme` attribute is "`sca:correlationID`" the SCA runtime MUST set a non-null correlation ID value in requests that it sends
[BJM30010]	Whatever the value of the `destination/@type` attribute, the SCA runtime MUST ensure a single response is delivered for request/response operations
[BJM30011]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`always`" and the `@jndiName` attribute is present and the resource cannot be created at the location specified by the `@jndiName` attribute then the SCA runtime MUST raise an error
[BJM30012]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" then the `@jndiName` attribute MUST specify the location of the possibly existing resource
[BJM30013]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" and the resource does not exist at the location identified by the `@jndiName` attribute and cannot be created there then the SCA runtime MUST raise an error
[BJM30014]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`ifNotExist`" and the `@jndiName` attribute refers to an existing resource that is not a JMS Destination of the approprate type, a JMS connection factory or a JMS activation spec respectively then the SCA runtime MUST raise an error
[BJM30015]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "`never`" and the `@jndiName` attribute is not specified, or the resource is not present at the location identified by the `@jndiName` attribute, or the location refers to a resource of an incorrect type then the SCA runtime MUST raise an error

[BJM30017]	A `binding.jms` element MUST NOT include both a `connectionFactory` element and an `activationSpec` element
[BJM30018]	When the `connectionFactory` element is present as a child of the `binding.jms` element, then the destination MUST be defined either by the `destination` element child of the `binding.jms` element or the `@uri` attribute of the `binding.jms` element
[BJM30019]	If the `activationSpec` element is present as a child of the `binding.jms` element and the destination is also specified via a `destination` element child of the `binding.jms` element or the `@uri` attribute of the `binding.jms` element then it MUST refer to the same JMS destination as the `activationSpec`
[BJM30020]	The `activationSpec` element MUST NOT be present when the binding is being used for an SCA reference
[BJM30021]	A `response` element MUST NOT include both a `connectionFactory` element and an `activationSpec` element
[BJM30022]	If a `response/destination` and `response/activationSpec` element are both specified they MUST refer to the same JMS destination
[BJM30023]	The `response/activationSpec` element MUST NOT be present when the binding is being used for an SCA service
[BJM30024]	When sending messages for a JMS binding, the SCA runtime MUST set each of the JMSType, JMSDeliveryMode, JMSTimeToLive and JMSPriority headers to values specified in the binding definition in the following priority order: 1) the value for the header specified in the `@uri` attribute (highest priority); 2) the value for the header specified in the `operationProperties/headers` element matching the operation being invoked; 3) the value for the header specified in the `headers` element; 4) the default value for the header as specified by the definition of the `binding.jms/headers` element (lowest priority)
[BJM30025]	When sending messages for a JMS binding, the SCA runtime MUST set each named user property with type and value specified in the binding definition in the following priority order: 1) the type and value for the named user property specified in an `operationProperties/headers/property` element matching the name of the operation being invoked (highest priority); 2) the type and value for the named user property specified in a `headers/property` element (lowest priority)
[BJM30026]	When receiving messages for a JMS binding, the SCA runtime MUST use a message selector if specified in the binding definition in the following priority order: 1) the value for the message selector specified in the `@uri` attribute value's "selector" parameter (highest priority); 2) the value for the message selector specified in the `messageSelection/@selector` attribute; 3) otherwise no message selector is used (lowest priority)

[BJM30029]	The value of the ***operationProperties/@selectedOperation*** attribute MUST be unique across the containing `binding.jms` element
[BJM30031]	The `resourceAdapter` element MUST be present when JMS resources are to be created for a JMS provider that implements the JCA 1.5 Specification [JCA15] specification, and is ignored otherwise
[BJM30034]	When the `@uri` attribute is specified, the `destination` element MUST NOT be present
[BJM30036]	The `binding.jms` element MUST conform to the XML schema defined in sca-binding-jms-1.1.xsd
[BJM30037]	If the `@create` attribute value for a `destination`, `connectionFactory` or `activationSpec` element is "always" and the `@jndiName` attribute is not present and the resource cannot be created, then the SCA runtime MUST raise an error
[BJM40002]	If no `operationSelector` element is specified then SCA runtimes MUST use `operationSelector.jmsDefault` as the default
[BJM40003]	When using the default wire format to send request messages, if there is a single parameter and the interface includes more than one operation, the SCA runtime MUST set the JMS user property "`scaOperationName`" to the name of the operation being invoked
[BJM40004]	If no `wireFormat` element is specified in a JMS binding then SCA runtimes MUST use `wireFormat.jmsDefault` as the default
[BJM40005]	When using the default wire format an SCA runtime MUST be able to receive both JMS text and bytes messages
[BJM40006]	When using the default wire format an SCA runtime MUST send either a JMS text or a JMS bytes message
[BJM40008]	When a `binding.jms` element specifies the `operationSelector.jmsDefault` element, the SCA runtime MUST use the default operation selection algorithm to determine the selected operation
[BJM40009]	When a `binding.jms` element specifies the `wireFormat.jmsDefault` element, the SCA runtime MUST use the default wire format
[BJM40010]	When a message is received at an SCA service with JMS binding and the resolved operation name is in the target component's interface, the SCA runtime MUST invoke the target component using the resolved operation name
[BJM40011]	When a message is received at an SCA service with JMS binding and the resolved operation name is not in the target component's interface the SCA runtime MUST raise an error
[BJM50001]	JMS binding implementations MUST support the JMS intent
[BJM50002]	The JMS intent MUST always be included in the `@alwaysProvides` attribute of the JMS `bindingType`
[BJM60002]	For an SCA service with a JMS binding and unidirectional interface, when a request message is received as part of a one-way MEP, the SCA runtime MUST ignore the `JMSReplyTo` destination header in the JMS message, and

	not raise an error
[BJM60003]	For an SCA reference with a JMS binding that has a destination specified via the response element, the SCA runtime MUST receive response messages as defined by the binding's `@correlationScheme` attribute
[BJM60004]	For an SCA reference with a JMS binding, when a request message is sent as part of a request/response MEP, and the JMS binding has a `response` element with a `destination` defined, then the SCA runtime MUST use that destination for the `JMSReplyTo` header in the JMS message it creates for the request
[BJM60005]	For an SCA reference with a JMS binding, when a request message is sent as part of a request/response MEP, and the JMS binding does not have a `response` element with a `destination` defined, the SCA runtime MUST provide an appropriate destination on which to receive response messages and use that destination for the `JMSReplyTo` header in the JMS message it creates for the request
[BJM60006]	For an SCA reference with a JMS binding that does not have a destination specified via the response element, the SCA runtime MUST either receive response messages as defined by the binding's `@correlationScheme` attribute, or use a unique destination for each request/response interaction
[BJM60007]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a non-null `JMSReplyTo` destination, the SCA runtime MUST send the response message to that destination
[BJM60008]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP where the request message included a null `JMSReplyTo` destination and the JMS binding includes a `response/destination` element the SCA runtime MUST send the response message to that destination
[BJM60009]	For an SCA service with a JMS binding, when a request message is received as part of a request/response MEP where the request message includes a null `JMSReplyTo` destination and the JMS binding does not include a response/destination then the SCA runtime MUST NOT process the request and MUST raise an error
[BJM60010]	For an SCA service with a JMS binding, when a response message is sent as part of a request/response MEP the SCA runtime MUST set the correlation identifier in the JMS message that it creates for the response as defined by the JMS binding's `@correlationScheme` attribute
[BJM60011]	For an SCA reference with a JMS binding and a bidirectional interface, when a request message is sent as part of a request/response MEP the SCA runtime MUST set the `scaCallbackDestination` user property in the message it creates to a JMS URI string, in the format defined by the IETF URI Scheme for Java™ Message Service 1.0 **[IETFJMS]**, that identifies the destination to which callback messages are to be sent
[BJM60012]	For an SCA reference with a JMS binding and bidirectional interface, when a request message is sent as part of a one-way MEP the SCA runtime MUST set the destination to which callback messages are to be sent as the `JMSReplyTo` destination in the message it creates

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

C. Acknowledgements

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

Participants:

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

D. Revision History

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
cd02	2009-02-16	Simon Holdsworth	Rename and editorial updates
cd02-rev1	2009-05-22	Simon Holdsworth	Updates to resolve issue BINDINGS-62 (conformance statement numbering) Updated assembly namespace to 200903 Fixed errors in schema
cd02-rev2	2009-05-22	Simon Holdsworth	Updates to resolve following issues: BINDINGS-39 BINDINGS-59 BINDINGS-65 BINDINGS-66 BINDINGS-67 BINDINGS-68 BINDINGS-70 BINDINGS-71
cd02-rev3	2009-06-18	Simon Holdsworth	Editorial concerns addressed Added acknowledgements appendix
cd02-rev4	2009-06-19	Simon Holdsworth	Updates to resolve following issues BINDINGS-74 Some editorial updates Fixed normative statement missed in application of BINDINGS-67
cd02-rev5	2009-06-24	Simon Holdsworth	Updates to resolve following issues BINDINGS-77 Renamed document to old form Removed editorial commentary Editorial fixes around external references; changed all links to hyperlinks
cd02-rev6	2009-06-24	Simon Holdsworth	Fixed application of BINDINGS-74 Fixed broken cross reference Changed ASCII to UTF-8 in examples
cd03	2009-06-29	Simon Holdsworth	Updates to resolve following issues BINDINGS-80 BINDINGS-81
cd03-rev1	2010-01-24	Simon Holdsworth	Editorial fix to XML schema name

			Updated to resolve following issues BINDINGS-48 BINDINGS-83 BINDINGS-85 BINDINGS-90 BINDINGS-93 BINDINGS-94 BINDINGS-96 BINDINGS-97 BINDINGS-98 BINDINGS-103 BINDINGS-108 BINDINGS-109 BINDINGS-110
cd03-rev2	2010-02-12	Simon Holdsworth	Editorial fixes to cross-references Fix cd03-rev1 change to add BINDINGS-110 Updated to resolve following issues BINDINGS-95 BINDINGS-104 BINDINGS-105 BINDINGS-106
cd03-rev3	2010-02-17	Bryan Aupperle	Add captions to all diagrams
cd03-rev4	2010-02-22	Simon Holdsworth	Updated assembly namespace to 200912 Editorial updates from action items and issues BINDINGS-101 BINDINGS-102 20091015-3: no change to copyright (currently consistent with all other SCA specs) 20091015-8: removed non-normative references section 20091015-9: cleaned up naming conventions section 20091015-10: cleaned up some phrases that used “may” or “allows” 20091015-12: no changes made (currently consistent with all other SCA specs)
cd03-rev5	2010-03-18	Simon Holdsworth	Fixed application of issue BINDINGS-108 Editorial cleanup Changed assembly reference to CD05
cd03-rev6	2010-04-16	Simon Holdsworth	Applied resolution to BINDINGS-128
cd04	2010-04-30	Simon Holdsworth	Rename and fix acknowledgements, fixup for publication

cd04-rev1	2010-10-05	Simon Holdsworth	Applied resolutions for issues: BINDINGS-134 BINDINGS-135 BINDINGS-136 BINDINGS-138 BINDINGS-139 Updated SCA policy spec reference and IETF JMS URI draft reference
cd04-rev2	2010-10-26	Simon Holdsworth	Applied resolutions for issues: BINDINGS-141
cd04-rev3	2010-10-29	Simon Holdsworth	Applied resolutions for issues: BINDINGS-140
csprd03-rev1	2011-05-22	Simon Holdsworth	Applied resolutions for issues: BINDINGS-144 BINDINGS-149 BINDINGS-154 BINDINGS-156 BINDINGS-157 BINDINGS-158 BINDINGS-159
csdprd03-rev2	2011-07-04	Simon Holdsworth	Applied resolutions for issues: BINDINGS-169