SCA Service Component Architecture

JMS Binding Specification

SCA Version 1.00, March 21 2007

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

Copyright Notice

License

Status of this Document

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

Table of Contents

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

1 Messaging and JMS Bindings

1.1 Introduction

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

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

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.

1.2 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 port type [3]. 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.

1.3 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.

1.4 JMS Binding Schema

The JMS binding element is defined by the following schema.

```
<binding.jms correlationScheme="string"?
             initialContextFactory="xs:anyURI"?
             jndiURL="xs:anyURI"?
             requestConnection="QName"?
             responseConnection="QName"?
             operationProperties="QName"?
             ... >
    <destination name="xs:anyURI" type="string"? create="string"?>
        <property name="NMTOKEN" type="NMTOKEN">*
    </destination>?
    <connectionFactory name="xs:anyURI" create="string"?>
        <property name="NMTOKEN" type="NMTOKEN">*
    </connectionFactory>?
    <activationSpec name="xs:anyURI" create="string"?>
        <property name="NMTOKEN" type="NMTOKEN">*
    </activationSpec>?

    <response>
        <destination name="xs:anyURI" type="string"? create="string"?>
            <property name="NMTOKEN" type="NMTOKEN">*
        </destination>?
        <connectionFactory name="xs:anyURI" create="string"?>
            <property name="NMTOKEN" type="NMTOKEN">*
        </connectionFactory>?
        <activationSpec name="xs:anyURI" create="string"?>
            <property name="NMTOKEN" type="NMTOKEN">*
        </activationSpec>?
    </response>?

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

    <headers JMSType="string"?
             JMSCorrelationId="string"?
             JMSDeliveryMode="string"?
             JMSTimeToLive="int"?
             JMSPriority="string"?>
        <property name="NMTOKEN" type="NMTOKEN">*
    </headers>?

    <operationProperties name="string" nativeOperation="string"?>
        <property name="NMTOKEN" type="NMTOKEN">*
        <headers JMSType="string"?
                 JMSCorrelationId="string"?
                 JMSDeliveryMode="string"?
                 JMSTimeToLive="int"?
                 JMSPriority="string"?>
            <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 URI has the following format:

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

  When the URI is used, it is assumed that the referenced resources already exist.
- ***/binding.jms/@correlationScheme*** – identifies the correlation scheme used when sending reply or callback messages. Valid values are "RequestMsgIDToCorrelID" (the default), "RequestCorrelIDToCorrelID", and "None".
- ***/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 the corresponding elements must not be present within this binding element.
- ***/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 no response element must be present within this binding 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 no operationProperties elements must be present within this binding 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. Must take one of the values "queue" or "topic". The default value is "queue". When "topic" is specified, then all the operations in the interface that corresponds to the binding must be one-way.
- ***/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 this should be considered an error.

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

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

1.5 Default Operation Selection and Data Binding behavior

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

1.5.1 Default Operation Selection

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

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

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

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

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

1.5.2 Default Data Binding

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

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

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

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

1.6 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 natively provide support for some standard intents, as defined by the JMS binding's bindingType:

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

1.7 Callback and Conversation Protocol

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

1.7.1 JMS User Properties

This protocol assigns specific behavior to JMS user properties:

- "scaCallbackQueue" holds the name of the queue 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.

1.7.2 Callbacks

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

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

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

1.7.3 Conversations

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

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

The message that starts the conversation may also include the “scaConversationMaxIdleTime” user property; if not present the maximum idle time for the conversation is derived 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 sender may provide a specific callback queue for the identified conversation by including a value for the “scaCallbackQueue” user property.

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

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

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

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

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

1.8 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.

1.8.1 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://www.osoa.org/xmlns/sca/1.0”
           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>
```

1.8.2 URI Binding Example

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

```
    <?xml version=”1.0” encoding="ASCII”?>
    <composite xmlns=”http://www.osoa.org/xmlns/sca/1.0”
               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>
```

1.8.3 Binding with Existing Resources Example

The following example shows the JMS binding using existing resources:

```
    <?xml version=”1.0” encoding=”ASCII”?>
    <composite xmlns=”http://www.osoa.org/xmlns/sca/1.0”
               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>
```

1.8.4 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://www.osoa.org/xmlns/sca/1.0”
               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>
```

1.8.5 Request / Response Example

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

```
    <?xml version=”1.0” encoding=”ASCII”?>
    <composite xmlns=”http://www.osoa.org/xmlns/sca/1.0”
               name=”MyValueComposite”>

        <service name=”MyValueService”>
            <interface.java interface=”services.myvalue.MyValueService”/>
            <binding.jms correlationScheme=”RequestMsgIdToCorrelId”>
                <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=”RequestMsgIdToCorrelId”>
                <destination name=”StockQuoteServiceQueue”/>
                <connectionFactory name=”StockQuoteServiceQCF”/>
                <response>
                    <destination name=”MyValueResponseQueue”/>
                    <activationSpec name=”MyValueResponseAS”/>
                </response>
            </binding.jms>
        </reference>
    </composite>
```

1.8.6 Use of Predefined Definitions Example

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

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

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

```
<definitions targetNamespace="http://acme.com"
             xmlns="http://www.osoa.org/xmlns/sca/1.0">
    <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://www.osoa.org/xmlns/sca/1.0"
           xmlns:acme="http://acme.com"
           name="MyValueComposite">

    <reference name="MyValueService">
        <interface.java interface="services.myvalue.MyValueService"/>
        <binding.jms requestConnection="acme:StockQuoteService"/>
    </reference>
</composite>
```

1.8.7 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 "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 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>
```

2 JMS Binding Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) Copyright SCA Collaboration 2006 -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.osoa.org/xmlns/sca/1.0"
        xmlns:sca="http://www.osoa.org/xmlns/sca/1.0"
        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="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"
                       default="RequestMsgIDToCorrelID">
               <simpleType>
                  <restriction base="string">
                     <enumeration value="RequestMsgIDToCorrelID"/>
                     <enumeration value="RequestCorrelIDToCorrelID"/>
                     <enumeration value="None"/>
                  </restriction>
               </simpleType>
            </attribute>

            <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="string"
                   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="string"
                   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="string"
                   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="string"
                   minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
       <attribute name="JMSType" type="string"/>
       <attribute name="JMSCorrelationID" type="string"/>
       <attribute name="JMSDeliveryMode" type="string"/>
       <attribute name="JMSTimeToLive" type="int"/>
       <attribute name="JMSPriority" type="string"/>
```

```
   </complexType>

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

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

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

3 References

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

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

[3] WSDL Specification

WSDL 1.1: http://www.w3.org/TR/wsdl

WSDL 2.0: http://www.w3.org/TR/wsdl20/

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