



# FUSE Services Framework

Using the JMS Transport

Version 2.0  
December 2007

Making Software Work Together™

---

# Using the JMS Transport

IONA Technologies

Version 2.0

Published 22 May 2008

Copyright © 2001-2008 IONA Technologies PLC

## Trademark and Disclaimer Notice

IONA Technologies PLC and/or its subsidiaries may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this publication. Except as expressly provided in any written license agreement from IONA Technologies PLC, the furnishing of this publication does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Any rights not expressly granted herein are reserved.

IONA, IONA Technologies, the IONA logo, Orbix, High Performance Integration, Artix, FUSE, and Making Software Work Together are trademarks or registered trademarks of IONA Technologies PLC and/or its subsidiaries.

Java and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. CORBA is a trademark or registered trademark of the Object Management Group, Inc. in the United States and other countries. All other trademarks that appear herein are the property of their respective owners.

While the information in this publication is believed to be accurate, IONA Technologies PLC makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IONA shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

## Copyright Notice

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of IONA Technologies PLC. No third-party intellectual property right liability is assumed with respect to the use of the information contained herein. IONA Technologies PLC assumes no responsibility for errors or omissions contained in this publication. This publication and features described herein are subject to change without notice. Portions of this document may include Apache Foundation documentation, all rights reserved.

---

---

# Table of Contents

<b>JMS Namespaces</b> .....	<b>9</b>
<b>Configuring a JMS Endpoint</b> .....	<b>11</b>
Basic Endpoint Configuration .....	12
Using WSDL .....	13
Using Configuration .....	16
Consumer Endpoint Configuration .....	18
Using Configuration .....	19
Using WSDL .....	20
Provider Endpoint Configuration .....	21
Using Configuration .....	22
Using WSDL .....	24
<b>JMS Runtime Configuration</b> .....	<b>25</b>
JMS Session Pool Configuration .....	26
Consumer Specific Runtime Configuration .....	27
Provider Specific Runtime Configuration .....	28
Index .....	29



---

## List of Tables

1. JMS Endpoint Attributes .....	13
2. <code>messageType</code> Values .....	19
3. JMS Client WSDL Extensions .....	20
4. Provider Endpoint Configuration .....	22
5. JMS Provider Endpoint WSDL Extensions .....	24
6. Attributes for Configuring the JMS Session Pool .....	26



---

## List of Examples

1. JMS Extension Namespace .....	9
2. JMS Configuration Namespaces .....	9
3. JMS WSDL Port Specification .....	15
4. Addressing Information in a FUSE Services Framework Configuration File .....	16
5. Configuration for a JMS Consumer Endpoint .....	19
6. WSDL for a JMS Consumer Endpoint .....	20
7. Configuration for a Provider Endpoint .....	22
8. WSDL for a JMS Provider Endpoint .....	24
9. JMS Session Pool Configuration .....	26
10. JMS Consumer Endpoint Runtime Configuration .....	27
11. Provider Endpoint Runtime Configuration .....	28



---

# JMS Namespaces

## Summary

The XML elements used to configure an endpoint to use the FUSE Services Framework's JMS transport are defined in a single namespace. Before you can use the elements, you must add them to the document defining the endpoint.

### WSDL Namespace

---

The WSDL extensions for defining a JMS endpoint are defined in the namespace `http://cxf.apache.org/transport/jms`. In order to use the JMS extensions you will need to add the line shown in Example 1, "JMS Extension Namespace" to the definitions element of your contract.

#### Example 1. JMS Extension Namespace

```
xmlns:jms="http://cxf.apache.org/transport/jms"
```

---

### Configuration Namespace

The FUSE Services Framework JMS endpoint configuration properties are specified under the `http://cxf.apache.org/transport/jms` namespace. In order to use the JMS configuration properties you will need to add the line shown in Example 2, "JMS Configuration Namespaces" to the beans element of your configuration.

#### Example 2. JMS Configuration Namespaces

```
xmlns:jms="http://cxf.apache.org/transport/jms"
```



---

# Configuring a JMS Endpoint

## Summary

To use JMS as a transport, your endpoints need the information needed to connect to the JMS broker and the JMS destinations. This information can be provided by either adding a WSDL `port` element to the endpoint's WSDL document or by adding the JMS configuration into the endpoint's configuration document.

## Table of Contents

Basic Endpoint Configuration .....	12
Using WSDL .....	13
Using Configuration .....	16
Consumer Endpoint Configuration .....	18
Using Configuration .....	19
Using WSDL .....	20
Provider Endpoint Configuration .....	21
Using Configuration .....	22
Using WSDL .....	24

# Basic Endpoint Configuration

## Table of Contents

Using WSDL .....	13
Using Configuration .....	16

JMS endpoints need to know certain basic information about how to establish a connection to the proper destination. This information can be provided in one of two places:

- WSDL
- Configuration

## Using WSDL

The JMS destination information is provided using the `jms:address` element and its child, the `jms:JMSNamingProperties` element. The `jms:address` element's attributes specify the information needed to identify the JMS broker and the destination. The `jms:JMSNamingProperties` element specifies the Java properties used to connect to the JNDI service.

---

### The address element

The basic configuration for a JMS endpoint is done by using a `jms:address` element as the child of your service's port element. The `jms:address` element uses the attributes described in Table 1, "JMS Endpoint Attributes" to configure the connection to the JMS broker.

**Table 1. JMS Endpoint Attributes**

Attribute	Description
<code>destinationStyle</code>	Specifies if the JMS destination is a JMS queue or a JMS topic.
<code>jndiConnectionFactoryName</code>	Specifies the JNDI name bound to the JMS connection factory to use when connecting to the JMS destination.
<code>jndiDestinationName</code>	Specifies the JNDI name bound to the JMS destination to which requests are sent.
<code>jndiReplyDestinationName</code>	Specifies the JNDI name bound to the JMS destinations where replies are sent. This attribute allows you to use a user defined destination for replies. For more details see Using a named reply destination.
<code>connectionUserName</code>	Specifies the user name to use when connecting to a JMS broker.
<code>connectionPassword</code>	Specifies the password to use when connecting to a JMS broker.

### The JMSNamingProperties element

To increase interoperability with JMS and JNDI providers, the `jms:address` element has a child element, `jms:JMSNamingProperties`, that allows you to specify the values used to populate the properties used when connecting to the JNDI provider. The `jms:JMSNamingProperties` element has two attributes: `name` and `value`. `name` specifies the name of the property to set. `value` attribute specifies the value for the specified property.

`jms:JMSNamingProperties` element can also be used for specification of provider specific properties.

The following is a list of common JNDI properties that can be set:

1. `java.naming.factory.initial`
2. `java.naming.provider.url`
3. `java.naming.factory.object`
4. `java.naming.factory.state`
5. `java.naming.factory.url.pkgs`
6. `java.naming.dns.url`
7. `java.naming.authoritative`
8. `java.naming.batchsize`
9. `java.naming.referral`
10. `java.naming.security.protocol`
11. `java.naming.security.authentication`
12. `java.naming.security.principal`
13. `java.naming.security.credentials`
14. `java.naming.language`
15. `java.naming.applet`

For more details on what information to use in these attributes, check your JNDI provider's documentation and consult the Java API reference material.

---

### Using a named reply destination

By default, FUSE Services Framework endpoints using JMS create a temporary queue for sending replies back and forth. You can change this behavior by

setting the `jndiReplyDestinationName` attribute in the endpoint's contract. A client endpoint will listen for replies on the specified destination and it will specify the value of the attribute in the `ReplyTo` field of all outgoing requests. A service endpoint will use the value of the `jndiReplyDestinationName` attribute as the location for placing replies if there is no destination specified in the request's `ReplyTo` field.

---

### Example

Example 3, "JMS WSDL Port Specification" shows an example of a JMS WSDL port specification.

### Example 3. JMS WSDL Port Specification

```
<service name="JMSService">
  <port binding="tns:Greeter_SOAPBinding" name="SoapPort">
    <jms:address jndiConnectionFactoryName="ConnectionFactory"
      jndiDestinationName="dynamicQueues/test.Celtix.jmstransport" >
      <jms:JMSNamingProperty name="java.naming.factory.initial"
        value="org.activemq.jndi.ActiveMQInitialContextFactory" />
      <jms:JMSNamingProperty name="java.naming.provider.url"
        value="tcp://localhost:61616" />
    </jms:address>
  </port>
</service>
```

## Using Configuration

In addition to using the WSDL file to specify the connection information for a JMS endpoint, you can supply it in the endpoint's configuration file. The information in the configuration file will override the information in the endpoint's WSDL file.

---

### Configuration elements

You configure a JMS endpoint using one of the following configuration elements:

`jms:conduit`

The `jms:conduit` element contains the configuration for a consumer endpoint. It has one attribute, `name`, whose value takes the form

`{WSDLNamespace}WSDLPortName.jms-conduit`.

`jms:destination`

The `jms:destination` element contains the configuration for a provider endpoint. It has one attribute, `name`, whose value takes the form

`{WSDLNamespace}WSDLPortName.jms-destination`.

---

### The address element

JMS connection information is specified by adding a `jms:address` child to the base configuration element. The `jms:address` element used in the configuration file is identical to the one used in the WSDL file. Its attributes are listed in Table 1, "JMS Endpoint Attributes". Like the `jms:address` element in the WSDL file, the `jms:address` configuration element also has a `jms:JMSPNamingProperties` child element that is used to specify additional information used to connect to a JNDI provider.

---

### Example

Example 4, "Addressing Information in a FUSE Services Framework Configuration File" shows a FUSE Services Framework configuration entry for configuring the addressing information for a JMS consumer endpoint.

### Example 4. Addressing Information in a FUSE Services Framework Configuration File

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ct="http://cxf.apache.org/configuration/types"
  xmlns:jms="http://cxf.apache.org/transport/jms"
```

```
    xsi:schemaLocation="http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans.xsd">
<jms:conduit name="{http://cxf.apache.org/jms_endpt}HelloWorldJMSPort.jms-conduit">
  <jms:address destinationStyle="queue"
              jndiConnectionFactoryName="myConnectionFactory"
              jndiDestinationName="myDestination"
              jndiReplyDestinationName="myReplyDestination"
              connectionUserName="testUser"
              connectionPassword="testPassword">
    <jms:JMSNamingProperty name="java.naming.factory.initial"
                          value="org.apache.cxf.transport.jms.MyInitialContextFactory"
    />
  <jms:JMSNamingProperty name="java.naming.provider.url"
                          value="tcp://localhost:61616" />
  </jms:address>
</jms:conduit>
</beans>
```

# Consumer Endpoint Configuration

## Table of Contents

Using Configuration .....	19
Using WSDL .....	20

JMS consumer endpoints specify the type of messages they use. JMS consumer endpoint can use either a JMS `ByteMessage` or a JMS `TextMessage`. When using an `ObjectMessage` the consumer endpoint uses a `byte[]` as the method for storing data into and retrieving data from the JMS message body. When messages are sent, the message data, including any formatting information, is packaged into a `byte[]` and placed into the message body before it is placed on the wire. When messages are received, the consumer endpoint will attempt to unmarshal the data stored in the message body as if it were packed in a `byte[]`.

When using a `TextMessage`, the consumer endpoint uses a string as the method for storing and retrieving data from the message body. When messages are sent, the message information, including any format-specific information, is converted into a string and placed into the JMS message body. When messages are received the consumer endpoint will attempt to unmarshal the data stored in the JMS message body as if it were packed into a string.

When native JMS applications interact with FUSE Services Framework consumers, the JMS application is responsible for interpreting the message and the formatting information. For example, if the FUSE Services Framework contract specifies that the binding used for a JMS endpoint is SOAP, and the messages are packaged as `TextMessage`, the receiving JMS application will get a text message containing all of the SOAP envelope information.

A consumer endpoint can be configured in one of two ways:

- Configuration
- WSDL



### Tip

The recommended method is to place the consumer endpoint specific information into the FUSE Services Framework configuration file for the endpoint.

## Using Configuration

### Specifying the message type

Consumer endpoint configuration is specified using the `jms:conduit` element. Using this configuration element, you specify the message type supported by the consumer endpoint using the `jms:runtimePolicy` child element. The message type is specified using the `messageType` attribute. The `messageType` attribute has two possible values:

**Table 2. messageType Values**

<code>text</code>	Specifies that the data will be packaged as a <code>TextMessage</code> .
<code>binary</code>	specifies that the data will be packaged as an <code>ByteMessage</code> .

### Example

Example 5, “Configuration for a JMS Consumer Endpoint” shows a configuration entry for configuring a JMS consumer endpoint.

### Example 5. Configuration for a JMS Consumer Endpoint

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ct="http://cxf.apache.org/configuration/types"
  xmlns:jms="http://cxf.apache.org/transports/jms"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">
  ...
  <jms:conduit name="{http://cxf.apache.org/jms_endpt}HelloWorldJMSPort.jms-conduit">
    <jms:address ... >
      ...
    </jms:address>
    ...
    <jms:runtimePolicy messageType="binary"/>
    ...
  </jms:conduit>
  ...
</beans>
```

## Using WSDL

### Specifying the message type

The type of messages accepted by a JMS consumer endpoint is configured using the optional `jms:client` element. The `jms:client` element is a child of the WSDL `port` element and has one attribute:

**Table 3. JMS Client WSDL Extensions**

<code>messageType</code>	Specifies how the message data will be packaged as a JMS message. <code>text</code> specifies that the data will be packaged as a <code>TextMessage</code> . <code>binary</code> specifies that the data will be packaged as an <code>ByteMessage</code> .
--------------------------	--

### Example

Example 6, “WSDL for a JMS Consumer Endpoint” shows the WSDL for configuring a JMS consumer endpoint.

### Example 6. WSDL for a JMS Consumer Endpoint

```
<service name="JMSService">
  <port binding="tns:Greeter_SOAPBinding" name="SoapPort">
    <jms:address jndiConnectionFactoryName="ConnectionFactory"
      jndiDestinationName="dynamicQueues/test.Celtix.jmstransport" >
      <jms:JMSNamingProperty name="java.naming.factory.initial"
        value="org.activemq.jndi.ActiveMQInitialContextFactory" />
      <jms:JMSNamingProperty name="java.naming.provider.url"
        value="tcp://localhost:61616" />
    </jms:address>
    <jms:client messageType="binary" />
  </port>
</service>
```

# Provider Endpoint Configuration

## Table of Contents

Using Configuration .....	22
Using WSDL .....	24

JMS provider endpoints have a number of behaviors that are configurable. These include:

- how messages are correlated
- the use of durable subscriptions
- if the service uses local JMS transactions
- the message selectors used by the endpoint

Service endpoints can be configured in one of two ways:

- Configuration
- WSDL



### Tip

The recommended method is to place the provider endpoint specific information into the FUSE Services Framework configuration file for the endpoint.

## Using Configuration

### Specifying configuration data

Provider endpoint configuration is specified using the `.jms:destination` configuration element. Using this configuration element, you can specify the provider endpoint's behaviors using the `.jms:runtimePolicy` element. When configuring a provider endpoint you can use the following `.jms:runtimePolicy` attributes:

**Table 4. Provider Endpoint Configuration**

Attribute	Description
<code>useMessageIDAsCorrealationID</code>	Specifies whether the JMS broker will use the message ID to correlate messages. The default is <code>false</code> .
<code>durableSubscriberName</code>	Specifies the name used to register a durable subscription.
<code>messageSelector</code>	Specifies the string value of a message selector to use. For more information on the syntax used to specify message selectors, see the JMS 1.1 specification.
<code>transactional</code>	Specifies whether the local JMS broker will create transactions around message processing. The default is <code>false</code> . <sup>a</sup>

<sup>a</sup>Currently, setting the `transactional` attribute to `true` is not supported by the runtime.

### Example

Example 7, “Configuration for a Provider Endpoint” shows a FUSE Services Framework configuration entry for configuring a provider endpoint.

### Example 7. Configuration for a Provider Endpoint

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ct="http://cxf.apache.org/configuration/types"
  xmlns:jms="http://cxf.apache.org/transports/jms"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">
  ...
  <jms:destination name="{http://cxf.apache.org/jms_endpt>HelloWorldJMSPort.jms-destination">
    ...
    <jms:runtimePolicy messageSelector="cxf_message_selector"
      useMessageIDAsCorrelationID="true"
      transactional="true">
```

```
        durableSubscriberName="cxf_subscriber" />
    ...
</jms:destination>
    ...
</beans>
```

## Using WSDL

### Configuring the endpoint

Provider endpoint behaviors are configured using the optional `jms:server` element. The `jms:server` element is a child of the WSDL `wSDL:port` element and has the following attributes:

**Table 5. JMS Provider Endpoint WSDL Extensions**

Attribute	Description
<code>useMessageIDAsCorrelationID</code>	Specifies whether JMS will use the message ID to correlate messages. The default is <code>false</code> .
<code>durableSubscriberName</code>	Specifies the name used to register a durable subscription.
<code>messageSelector</code>	Specifies the string value of a message selector to use. For more information on the syntax used to specify message selectors, see the JMS 1.1 specification.
<code>transactional</code>	Specifies whether the local JMS broker will create transactions around message processing. The default is <code>false</code> . <sup>a</sup>

<sup>a</sup>Currently, setting the `transactional` attribute to `true` is not supported by the runtime.

### Example

Example 8, “WSDL for a JMS Provider Endpoint” shows the WSDL for configuring a JMS provider endpoint.

### Example 8. WSDL for a JMS Provider Endpoint

```
<service name="JMSService">
  <port binding="tns:Greeter_SOAPBinding" name="SoapPort">
    <jms:address jndiConnectionFactoryName="ConnectionFactory"
      jndiDestinationName="dynamicQueues/test.Celtix.jmstransport" >
      <jms:JMSNamingProperty name="java.naming.factory.initial"
        value="org.activemq.jndi.ActiveMQInitialContextFactory" />
      <jms:JMSNamingProperty name="java.naming.provider.url"
        value="tcp://localhost:61616" />
    </jms:address>
    <jms:server messageSelector="cxf_message_selector"
      useMessageIDAsCorrelationID="true"
      transactional="true"
      durableSubscriberName="cxf_subscriber" />
  </port>
</service>
```

---

# JMS Runtime Configuration

## Summary

*The JMS transport has a number of configurable runtime behaviors. Unlike the endpoint configuration, the information used to configure the runtime behaviors must be placed into the endpoint's configuration document.*

## Table of Contents

JMS Session Pool Configuration .....	26
Consumer Specific Runtime Configuration .....	27
Provider Specific Runtime Configuration .....	28

In addition to configuring the externally visible aspects of your JMS endpoint, you can also configure aspects of its internal runtime behavior. There are three types of runtime configuration:

- JMS session pool configuration
- Consumer specific configuration
- Provider specific configuration

## JMS Session Pool Configuration

The JMS configuration allows you to specify the number of JMS sessions an endpoint will keep in a pool.

### Configuration element

You use the `javax:jms:sessionPool` element to specify the session pool configuration for a JMS endpoint. The `javax:jms:sessionPool` element is a child of both the `javax:jms:conduit` element and the `javax:jms:destination` element.

The `javax:jms:sessionPool` element's attributes, listed in Table 6, "Attributes for Configuring the JMS Session Pool", specify the high and low water marks for the endpoint's JMS session pool.

**Table 6. Attributes for Configuring the JMS Session Pool**

Attribute	Description
<code>lowWaterMark</code>	Specifies the minimum number of JMS sessions pooled by the endpoint. The default is 20.
<code>highWaterMark</code>	Specifies the maximum number of JMS sessions pooled by the endpoint. The default is 500.

### Example

Example 9, "JMS Session Pool Configuration" shows an example of configuring the session pool for a FUSE Services Framework JMS provider endpoint.

### Example 9. JMS Session Pool Configuration

```

...
<jms:destination name="{http://cxf.apache.org/jms_endpt>HelloWorldJMSPort.jms-destination">
    <jms:address ... >
        ...
    </jms:address>
    ...
    <jms:sessionPool lowWaterMark="10"
                    highWaterMark="5000" />
    ...
</jms:destination>
...

```

## Consumer Specific Runtime Configuration

The JMS consumer configuration allows you to specify two runtime behaviors:

- the number of milliseconds the consumer will wait for a response.
- the number of milliseconds a request will exist before the JMS broker can remove it.

---

### Configuration element

You configure consumer runtime behavior using the `jms:clientConfig` element. The `jms:clientConfig` element is a child of the `jms:conduit` element. It has two attributes that are used to specify the configurable runtime properties of a consumer endpoint.

---

### Configuring the response timeout interval

You specify the interval, in milliseconds, a consumer endpoint will wait for a response before timing out using the `jms:clientConfig` element's `clientReceiveTimeout` attribute. The default timeout interval is 2000.

---

### Configure the request time to live

You specify the interval, in milliseconds, that a request can remain unreceived before the JMS broker can delete it using the `jms:clientConfig` element's `messageTimeToLive` attribute. The default time to live interval is 0 which specifies that the request has an infinite time to live.

---

### Example

Example 10, “JMS Consumer Endpoint Runtime Configuration” shows a configuration fragment that sets the consumer endpoint's request lifetime to 500 milliseconds and its timeout value to 500 milliseconds.

### Example 10. JMS Consumer Endpoint Runtime Configuration

```
...
<jms:conduit name="{http://cxf.apache.org/jms_endpt}HelloWorldJMSPort.jms-conduit">
  <jms:address ... >
    ...
  </jms:address>
  ...
  <jms:clientConfig clientReceiveTimeout="500"
                    messageTimeToLive="500" />
  ...
</jms:conduit>
...
```

## Provider Specific Runtime Configuration

The provider specific configuration allows you to specify to runtime behaviors:

- the amount of time a response message can remain unreceived before the JMS broker can delete it.
- the client identifier used when creating and accessing durable subscriptions.

---

### Configuration element

You configure provider runtime behavior using the `javax:jms:serverConfig` element. The `javax:jms:serverConfig` element is a child of the `javax:jms:destination` element. It has two attributes that are used to specify the configurable runtime properties of a provider endpoint.

---

### Configuring the response time to live

The `javax:jms:serverConfig` element's `messageTimeToLive` attribute specifies the amount of time, in milliseconds, that a response can remain unread before the JMS broker is allowed to delete it. The default is 0 which specifies that the message can live forever.

---

### Configuring the durable subscriber identifier

The `javax:jms:serverConfig` element's `durableSubscriptionClientId` attribute specifies the client identifier the endpoint uses to create and access durable subscriptions.

---

### Example

Example 11, "Provider Endpoint Runtime Configuration" shows a configuration fragment that sets the provider endpoint's response lifetime to 500 milliseconds and its durable subscription client identifier to `javax:jms-test-id`.

### Example 11. Provider Endpoint Runtime Configuration

```
...
<jms:destination name="{http://cxf.apache.org/jms_endpt}HelloWorldJMSPort.jms-destination">
    <jms:address ... >
        ...
    </jms:address>
    ...
    <jms:serverConfig messageTimeToLive="500"
                        durableSubscriptionClientId="jms-test-id" />
    ...
</jms:destination>
...
```

---

# Index

## C

- configuration
  - consumer endpoint (see `jms:conduit`)
  - consumer runtime, 27
  - JMS session pool (see `jms:sessionPool`)
  - `jms:address` (see `jms:address`)
  - provider endpoint (see `jms:destination`)
  - provider endpoint properties, 22
  - provider runtime, 28
  - specifying the message type, 19 (see also `jms:runtimePolicy`)
- consumer endpoint configuration
  - specifying the message type, 19 (see also `jms:runtimePolicy`)
- consumer runtime configuration, 27
  - request time to live, 27
  - response timeout, 27

## E

- endpoint address configuration (see `jms:address`)

## J

- JMS
  - specifying the message type, 20
- JMS destination
  - specifying, 13
- `jms:address`, 13
  - `connectionPassword` attribute, 13
  - `connectionUserName` attribute, 13
  - `destinationStyle` attribute, 13
  - `jndiConnectionFactoryName` attribute, 13
  - `jndiDestinationName` attribute, 13
  - `jndiReplyDestinationName` attribute, 13, 14
- `jms:client`, 20
  - `messageType` attribute, 20
- `jms:clientConfig`, 27
  - `clientReceiveTimeout`, 27
  - `messageTimeToLive`, 27
- `jms:conduit`, 16

- `jms:destination`, 16
- `jms:JMSNamingProperties`, 13
- `jms:runtimePolicy`
  - consumer endpoint properties, 19
  - `durableSubscriberName`, 22
  - `messageSelector`, 22
  - `messageType` attribute, 19
  - provider configuration, 22
  - transactional, 22
  - `useMessageDAsCorrelationID`, 22
- `jms:server`, 24
  - `durableSubscriberName`, 24
  - `messageSelector`, 24
  - transactional, 24
  - `useMessageDAsCorrelationID`, 24
- `jms:serverConfig`, 28
  - `durableSubscriptionClientId`, 28
  - `messageTimeToLive`, 28
- `jms:sessionPool`, 26
  - `highWaterMark`, 26
  - `lowWaterMark`, 26
- JNDI
  - specifying the connection factory, 13

## N

- named reply destination
  - specifying in WSDL, 13
  - using, 14

## P

- provider endpoint configuration, 22
- provider runtime configuration, 28
  - durable subscriber identification, 28
  - response time to live, 28

## S

- session pool configuration (see `jms:sessionPool`)

## W

- WSDL extensors
  - `jms:address` (see `jms:address`)
  - `jms:client` (see `jms:client`)

---

jms:JMSNamingProperties (see  
jms:JMSNamingProperties)  
jms:server (see jms:server)