

FUSETM Services Framework

Using the JMS Transport

Version 2.1.x
May 2008

Using the JMS Transport

Version 2.1.x

Published 22 Jan 2009

Copyright © 2008 IONA Technologies PLC, a wholly-owned subsidiary of Progress Software Corporation.

Legal Notices

Progress Software Corporation 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 Progress Software Corporation, 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.

Progress, IONA, IONA Technologies, the IONA logo, Orbix, High Performance Integration, Artix, FUSE, and Making Software Work Together are trademarks or registered trademarks of Progress Software Corporation and/or its subsidiaries in the US and other countries.

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 US 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 Progress Software Corporation 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. Progress Software Corporation 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.

All products or services mentioned in this manual are covered by the trademarks, service marks, or product names as designated by the companies who market those products.

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

JMS Namespaces	9
Configuring a JMS Endpoint	11
Basic Endpoint Configuration	12
Using Configuration	13
Using WSDL	17
Using a Named Reply Destination	18
Consumer Endpoint Configuration	19
Using Configuration	20
Using WSDL	21
Provider Endpoint Configuration	22
Using Configuration	23
Using WSDL	25
JMS Runtime Configuration	27
JMS Session Pool Configuration	28
Consumer Specific Runtime Configuration	29
Provider Specific Runtime Configuration	30
Index	31

List of Tables

1. JMS Endpoint Attributes	14
2. messageType Values	20
3. JMS Client WSDL Extensions	21
4. Provider Endpoint Configuration	23
5. JMS Provider Endpoint WSDL Extensions	25
6. Attributes for Configuring the JMS Session Pool	28

List of Examples

1. JMS Extension Namespace	9
2. JMS Configuration Namespaces	9
3. Addressing Information in a FUSE Services Framework Configuration File	15
4. JMS WSDL Port Specification	17
5. JMS Consumer Specification Using a Named Reply Queue	18
6. Configuration for a JMS Consumer Endpoint	20
7. WSDL for a JMS Consumer Endpoint	21
8. Configuration for a Provider Endpoint	23
9. WSDL for a JMS Provider Endpoint	25
10. JMS Session Pool Configuration	28
11. JMS Consumer Endpoint Runtime Configuration	29
12. Provider Endpoint Runtime Configuration	30

JMS Namespaces

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 on page 9](#) 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 on page 9](#) to the beans element of your configuration.

Example 2. JMS Configuration Namespaces

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


Configuring a JMS Endpoint

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 added a WSDL `port` element to the endpoint's WSDL document or by adding the JMS configuration into the endpoint's configuration document.

Basic Endpoint Configuration	12
Using Configuration	13
Using WSDL	17
Using a Named Reply Destination	18
Consumer Endpoint Configuration	19
Using Configuration	20
Using WSDL	21
Provider Endpoint Configuration	22
Using Configuration	23
Using WSDL	25

Basic Endpoint Configuration

Using Configuration	13
Using WSDL	17
Using a Named Reply Destination	18

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:

- [Configuration](#)
- [WSDL](#)

Using Configuration

Overview

JMS endpoints are configured using Spring configuration. You can configure the server-side and consumer-side transports independently.

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



Note

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 uses the attributes described in [Table 1 on page 14](#) 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>jmsDestinationName</code>	Specifies the JMS name of the JMS destination to which requests are sent.
<code>jmsReplyDestinationName</code>	Specifies the JMS name of 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 on page 18 .
<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 on page 18 .
<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.

Example

[Example 3 on page 15](#) shows a FUSE Services Framework configuration entry for configuring the addressing information for a JMS consumer endpoint.

Example 3. 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/transports/jms"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd"
    http://cxf.apache.org/schemas/jaxws.xsd
    http://cxf.apache.org/transports/jms http://cxf.apache.org/schem
as/configuration/jms.xsd">
  <jms:conduit name="{http://cxf.apache.org/jms_endpt>HelloWorldJMSPort.jms-conduit">
    <jms:address destinationStyle="queue"
      jndiConnectionFactoryName="myConnectionFactory"
```

Configuring a JMS Endpoint

```
        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>
```

Using WSDL

Overview

If you prefer to configure your endpoint using WSDL, you can specify JMS endpoints as a part of a WSDL service definition. The `.jms:address` element is a child of the WSDL `port` element.



Important

Information in the configuration file will override the information in the endpoint's WSDL file.

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 used in WSDL is identical to the one used in the configuration file. Its attributes are listed in [Table 1 on page 14](#). Like the `.jms:address` element in the configuration file, the `.jms:address` WSDL element also uses a `.jms:JMSNamingProperties` child element to specify additional information needed to connect to a JNDI provider.

Example

[Example 4 on page 17](#) shows an example of a JMS WSDL `port` specification.

Example 4. 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 a Named Reply Destination

Overview

By default, FUSE Services Framework endpoints using JMS create a temporary queue for sending replies back and forth. If you prefer to use named queues, you can configure the queue used to send replies as part of an endpoint's JMS configuration.

Setting the reply destination name

You specify the reply destination using either the `jmsReplyDestinationName` attribute or the `jndiReplyDestinationName` attribute in the endpoint's JMS configuration. 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 5 on page 18](#) shows the configuration for a JMS client endpoint.

Example 5. JMS Consumer Specification Using a Named Reply Queue

```
<jms:conduit name="{http://cxf.apache.org/jms_endpt}HelloWorldJMSPort.jms-conduit">
  <jms:address destinationStyle="queue"
    jndiConnectionFactoryName="myConnectionFactory"
    jndiDestinationName="myDestination"
    jndiReplyDestinationName="myReplyDestination" >
    <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>
```

Consumer Endpoint Configuration

Using Configuration	20
Using WSDL	21

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 6 on page 20](#) shows a configuration entry for configuring a JMS consumer endpoint.

Example 6. 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/transport/jms"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd"
    http://cxf.apache.org/jaxws ht
tp://cxf.apache.org/schemas/jaxws.xsd
    http://cxf.apache.org/transport/jms http://cxf.apache.org/schem
as/configuration/jms.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 7 on page 21](#) shows the WSDL for configuring a JMS consumer endpoint.

Example 7. 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

Using Configuration	23
Using WSDL	25

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 configure 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>useMessageIDAsCorrelationID</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> . ^a

^aCurrently, setting the `transactional` attribute to `true` is not supported by the runtime.

Example

[Example 8 on page 23](#) shows a FUSE Services Framework configuration entry for configuring a provider endpoint.

Example 8. 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/transport/jms"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://cxf.apache.org/jaxws ht
tp://cxf.apache.org/schemas/jaxws.xsd
    http://cxf.apache.org/transport/jms http://cxf.apache.org/schem
as/configuration/jms.xsd">
  ...
  <jms:destination name="{http://cxf.apache.org/jms_endpt>HelloWorldJMSPort.jms-destination">
    ...
```

Configuring a JMS Endpoint

```
<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> . ^a

^aCurrently, setting the `transactional` attribute to `true` is not supported by the runtime.

Example

[Example 9 on page 25](#) shows the WSDL for configuring a JMS provider endpoint.

Example 9. 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

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.

JMS Session Pool Configuration	28
Consumer Specific Runtime Configuration	29
Provider Specific Runtime Configuration	30

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

The `jms:sessionPool` element's attributes, listed in [Table 6 on page 28](#), 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 10 on page 28](#) shows an example of configuring the session pool for a FUSE Services Framework JMS provider endpoint.

Example 10. 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 11 on page 29](#) shows a configuration fragment that sets the consumer endpoint's request lifetime to 500 milliseconds and its timeout value to 500 milliseconds.

Example 11. 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 12 on page 30](#) shows a configuration fragment that sets the provider endpoint's response lifetime to 500 milliseconds and its durable subscription client identifier to `javax-test-id`.

Example 12. 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, 29
 - JMS session pool (see `jms:sessionPool`)
 - `jms:address` (see `jms:address`)
 - provider endpoint (see `jms:destination`)
 - provider endpoint properties, 23
 - provider runtime, 30
 - specifying the message type, 20
 - (see also `jms:runtimePolicy`)
- consumer endpoint configuration
 - specifying the message type, 20
 - (see also `jms:runtimePolicy`)
- consumer runtime configuration, 29
 - request time to live, 29
 - response timeout, 29

E

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

J

- JMS
 - specifying the message type, 21
- JMS destination
 - specifying, 14
- `jms:address`, 17
 - `connectionPassword` attribute, 14
 - `connectionUserName` attribute, 14
 - `destinationStyle` attribute, 14
 - `jmsDestinationName` attribute, 14
 - `jmsiReplyDestinationName` attribute, 18
 - `jmsReplyDestinationName` attribute, 14
 - `jndiConnectionFactoryName` attribute, 14
 - `jndiDestinationName` attribute, 14
 - `jndiReplyDestinationName` attribute, 14, 18
- `jms:client`, 21
 - `messageType` attribute, 21
- `jms:clientConfig`, 29

- `clientReceiveTimeout`, 29
- `messageTimeToLive`, 29
- `jms:conduit`, 13
- `jms:destination`, 13
- `jms:JMSNamingProperties`, 14
- `jms:runtimePolicy`
 - consumer endpoint properties, 20
 - `durableSubscriberName`, 23
 - `messageSelector`, 23
 - `messageType` attribute, 20
 - provider configuration, 23
 - transactional, 23
 - `useMessageIDAsCorrelationID`, 23
- `jms:server`, 25
 - `durableSubscriberName`, 25
 - `messageSelector`, 25
 - transactional, 25
 - `useMessageIDAsCorrelationID`, 25
- `jms:serverConfig`, 30
 - `durableSubscriptionClientId`, 30
 - `messageTimeToLive`, 30
- `jms:sessionPool`, 28
 - `highWaterMark`, 28
 - `lowWaterMark`, 28
- JNDI
 - specifying the connection factory, 14

N

- named reply destination
 - specifying in WSDL, 14
 - using, 18

P

- provider endpoint configuration, 23
- provider runtime configuration, 30
 - durable subscriber identification, 30
 - response time to live, 30

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)