CXF 2.x JMS configuration (removed in CXF 3)

Standard JMS transport configuration in CXF is done by defining a JMSConduit or JMSDestination, discussed below.

JMS Namespaces

WSDL Namespace

The WSDL extensions for defining a JMS endpoint are defined in the namespace `http://cxf.apache.org/transports/jms`. In order to use the JMS extensions you will need to add the namespace definition shown below to the definitions element of your contract.

JMS Extension Namespace

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

Configuration Namespaces

In order to use the JMS configuration properties you will need to add the line shown below to the `beans` element of your configuration.

JMS Configuration Namespaces

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

Basic Endpoint Configuration

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 or XML configuration. The following configuration elements which are described can be used in both the client side Conduits and the server side Destinations.

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 below to configure the connection to the JMS broker.

Attribute	Description
`destinationStyle`	Specifies if the JMS destination is a JMS queue or a JMS topic.
`jndiConnectionFactoryName`	Specifies the JNDI name bound to the JMS connection factory to use when connecting to the JMS destination.
`jndiDestinationName`	Specifies the JNDI name bound to the JMS destination to which requests are sent.
`jndiReplyDestinationName`	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.
`connectionUserName`	Specifies the username to use when connecting to a JMS broker.
`connectionPassword`	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`. The `name` attribute specifies the name of the property to set. The `value` attribute specifies the value for the specified property. The `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, CXF 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.

A static reply queue can not be shared by several instances of the service client. Please use a dynamic reply queue or different queue names per instance instead.

The following example shows an example of a JMS WSDL port specification.

JMS WSDL Port Specification

```
<service name="JMSService">
  <port binding="tns:Greeter_SOAPBinding" name="SoapPort">
    <jms:address jndiConnectionFactoryName="ConnectionFactory"
                 jndiDestinationName="dynamicQueues/test.cxf.jmstransport">
      <jms:JMSNamingProperty name="java.naming.factory.initial"
                             value="org.apache.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 also supply it in the endpoint's XML configuration. 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 the address element's attribute table. Like the jms:address element in the WSDL file, the jms:address configuration element also has a `jms:JMSNamingProperties` child element that is used to specify additional information used to connect to a JNDI provider.

Example

Addressing Information a 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/jaxws
           http://cxf.apache.org/schemas/jaxws.xsd
           http://cxf.apache.org/transports/jms
           http://cxf.apache.org/schemas/configuration/jms.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

JMS consumer endpoints specify the type of messages they use. JMS consumer endpoint can use either a JMS `ObjectMessage` 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 formating information, is packaged into a `byte[]` and placed into the JMS message body before it is placed on the wire. When messages are received, the consumer endpoint will attempt to unmarshall the data stored in the JMS 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 JMS 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 unmashall the data stored in the JMS message body as if it were packed into a string.

When native JMS applications interact with CXF consumers, the JMS application is responsible for interpreting the message and the formatting information. For example, if the CXF 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.

Consumer endpoint can be configured by both XML configuration and via WSDL.

Using Configuration

Specifying the message type

You can specify the message type supported by the consumer endpoint using a `jms:runtimePolicy` element that has a single attribute:

- `messageType` - Specifies how the message data will be packaged as a JMS message. `text` specifies that the data will be packaged as a `TextMessage`. `binary` specifies that the data will be packaged as an `ObjectMessage`.

The following example shows a configuration entry for configuring a JMS consumer endpoint.

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"
           http://cxf.apache.org/jaxws
           http://cxf.apache.org/schemas/jaxws.xsd
           http://cxf.apache.org/transports/jms
           http://cxf.apache.org/schemas/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>
```

The id on the `jms:conduit` is in the form of {*WSDLNamespace*}*WSDLPortName*`.jms-conduit`. This provides CXF with the information so that it can associate the configuration with your service's endpoint.

Using WSDL

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:

- `messageType` - Specifies how the message data will be packaged as a JMS message. `text` specifies that the data will be packaged as a `TextMessage`. `binary` specifies that the data will be packaged as an ObjectMessage.

Service Endpoint Configuration

JMS service endpoints have a number of behaviors that are configurable in the contract. 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

Using Configuration

Specifying configuration data

Using the `jms:destination` elements you can configure your service's endpoint. You can specify the service endpoint's behaviors using the `jms:runtimePolicy` element that has a the following attributes:

Attribute	Description
`useMessageIDAsCorrealationID`	Specifies whether the JMS broker will use the message ID to correlate messages. The default is `false`.
`durableSubscriberName`	Specifies the name used to register a durable subscription.
`messageSelector`	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.
`transactional`	Specifies whether the local JMS broker will create transactions around message processing. The default is `false`.

The following example shows a CXF configuration entry for configuring a JMS service endpoint.

Configuration for a JMS Service 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"
           http://cxf.apache.org/jaxws
           http://cxf.apache.org/schemas/jaxws.xsd
           http://cxf.apache.org/transports/jms
           http://cxf.apache.org/schemas/configuration/jms.xsd">
...
<jms:destination name="{http://cxf.apache.org/jms_endpt}HelloWorldJMSPort.jms-destination">
  <jms:address ... >
    ...
  </jms:address>
  ...
  <jms:runtimePolicy messageSelector="cxf_message_selector"
       useMessageIDAsCorrelationID="true"
       transactional="true"
       durableSubscriberName="cxf_subscriber" />
  ...
</jms:destination>
...
</beans>
```

Using WSDL

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

Attribute	Description
`useMessageIDAsCorrelationID`	Specifies whether JMS will use the message ID to correlate messages. The default is `false`.
`durableSubscriberName`	Specifies the name used to register a durable subscription.
`messageSelector`	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.
`transactional`	Specifies whether the local JMS broker will create transactions around message processing. The default is `false`. Currently, this is not supported by the runtime.

JMS Runtime Configuration

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:

- Session pool configuration (common to both services and consumers)
- Consumer specific configuration
- Service specific configuration

Session Pool Configuration

You configure an endpoint's JMS session pool using the `jms:sessionPoolConfig` element. This property allows you to set a high and low water mark for the number of JMS sessions an endpoint will keep pooled. The endpoint is guaranteed to maintain a pool of sessions equal to the low water mark and to never pool more sessions than specified by the high water mark.
The `jms:sessionPool` element's attributes, listed below, specify the high and low water marks for the endpoint's JMS session pool.

Attribute	Description
`lowWaterMark`	Specifies the minimum number of JMS sessions pooled by the endpoint. The default is 20.

`highWaterMark`	Specifies the maximum number of JMS sessions pooled by the endpoint. The default is 500.

The following example shows an example of configuring the session pool for a CXF JMS service endpoint.

JMS Session Pool Configuration

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

The `jms:sessionPool` element can also be used within a `jms:conduit`.

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.

You use the `jms:clientConfig` element to set JMS consumer runtime behavior. This element's attributes, listed in the following table, specify the configuration values for consumer runtime behavior.

Attribute	Description
`clientReceiveTimeout`	Specifies the amount of time, in milliseconds, that the endpoint will wait for a response before it times out and issues an exception. The default value is 2000.
`messageTimeToLive`	Specifies the amount of time, in milliseconds, that a request can remain unrecieved before the JMS broker can delete it. The default value is 0 which specifies that the message can never be deleted.

The following example shows a configuration fragment that sets the consumer endpoint's request lifetime to 500 milliseconds and its timeout value to 500 milliseconds.

JMS Consumer Endpoint Runtime Configuration

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

Service Specific Runtime Configuration

The JMS service 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.

The `jms:serverConfig` element is used to specify the service runtime configuration. This element's attributes, listed below, specify the configuration values that control the service's runtime behavior.

Attribute	Description
`messageTimeToLive`	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.
`durableSubscriptionClientId`	Specifies the client identifier the endpoint uses to create and access durable subscriptions.

The following example shows a configuration fragment that sets the service endpoint's response lifetime to 500 milliseconds and its durable subscription client identifier to `jms-test-id`.

JMS Service Endpoint Runtime Configuration

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