ActiveMQ

ActiveMQ Component

The ActiveMQ component allows messages to be sent to a JMS Queue or Topic or messages to be consumed from a JMS Queue or Topic using Apache ActiveMQ. This component is based on JMS Component and uses Spring's JMS support for declarative transactions, using Spring's **`Jms Template`** for sending and a **`MessageListenerContainer`** for consuming. All the options from the JMS component also applies for this component.

To use this component make sure you have the **`activemq.jar`** or **`activemq-core.jar`** on your classpath along with any Camel dependencies such as **`camel-core.jar`**, **`camel-spring.jar`** and **`camel-jms.jar`**.
Transacted and caching
See section *Transactions and Cache Levels* below on JMS page if you are using transactions with JMS as it can impact performance.

URI format

Where **`destinationName`** is an ActiveMQ queue or topic name. By default, the **`destinationName`** is interpreted as a queue name. For example, to connect to the queue, **`FOO.BAR`**, use:

You can include the optional **`queue:`** prefix, if you prefer:

To connect to a topic, you must include the **`topic:`** prefix. For example, to connect to the topic, **`Stocks.Prices`**, use:

Options

See Options on the JMS component as all these options also apply for this component.

Configuring the Connection Factory

This test case shows how to add an **`ActiveMQComponent`** to the CamelContext using the `activeMQComponent()` method while specifying the brokerURL used to connect to ActiveMQ.

Configuring the Connection Factory using Spring XML

You can configure the ActiveMQ broker URL on the **`ActiveMQComponent`** as follows
xml

Using Connection Pooling

When sending to an ActiveMQ broker using Camel it's recommended to use a pooled connection factory to efficiently handle pooling of JMS connections, sessions and producers. This is documented on the ActiveMQ Spring Support page.

You can grab ActiveMQ's **`org.apache.activemq.pool.PooledConnectionFactory`** with Maven:
xml

And then setup the **`activemq`** Camel component as follows:
xml
Notice the **`init`** and **`destroy`** methods on the pooled connection factory. This is important to ensure the connection pool is properly started and shutdown.
Important information about when using transactions
If you are using transactions then see more details at JMS. And remember to set **`cacheLevelName`** to **`CACHE_CONSUMER`** if you are not using XA transactions. This can dramatically improve performance.

The **`PooledConnectionFactory`** will then create a connection pool with up to 8 connections in use at the same time. Each connection can be shared by many sessions. There is an option named **`maximumActive`** you can use to configure the maximum number of sessions per connection; the default value is **`500`**. From **ActiveMQ 5.7**: the option has been renamed to better reflect its purpose, being named as **`maximumActiveSessionPerConnection`**. Notice the **`concurrentConsumers`** is set to a higher value than **`maxConnections`** is. This is okay, as each consumer is using a session, and as a session can share the same connection, we are in the safe. In this example we can have **`8 * 500 = 4000`** active sessions at the same time.

Invoking MessageListener POJOs in a Camel route

The ActiveMQ component also provides a helper Type Converter from a JMS **`MessageListener`** to a Processor. This means that the Bean com ponent is capable of invoking any JMS **`MessageListener`** bean directly inside any route.

So for example you can create a **`MessageListener`** in JMS like this:

Then use it in your Camel route as follows

That is, you can reuse any of the Camel Components and easily integrate them into your JMS **`MessageListener`** POJO!

Using ActiveMQ Destination Options

Available as of ActiveMQ 5.6

You can configure the Destination Options in the endpoint URI, using the **`destination.`** prefix. For example to mark a consumer as exclusive, and set its prefetch size to **`50`**, you can do as follows:

Consuming Advisory Messages

ActiveMQ can generate Advisory messages which are put in topics that you can consume. Such messages can help you send alerts in case you detect slow consumers or to build statistics (number of messages/produced per day, etc.) The following Spring DSL example shows you how to read messages from a topic.

The below route starts by reading the topic *ActiveMQ.Advisory.Connection*. To watch another topic, simply change the name according to the name provided in ActiveMQ Advisory Messages documentation. The parameter **`mapJmsMessage=false`** allows for converting the **`org.apache.activemq.command.ActiveMqMessage`** object from the JMS queue. Next, the body received is converted into a String for the purposes of this example and a carriage return is added. Finally, the string is added to a file

If you consume a message on a queue, you should see the following files under the **`data/activemq`** folder :

`advisoryConnection-20100312.txt`
`advisoryProducer-20100312.txt`

containing the following string:

Getting Component JAR

You will need this dependency

- **`activemq-camel`**

ActiveMQ is an extension of the JMS component released with the ActiveMQ project.

Endpoint See Also