URI Protocols

ActiveMQ is designed to support mutliple different topologies and protocols. Which one you use depends on your messaging requirements, quality of service and network topology.

The following table describes the different network protocols available for JMS clients along with showing the connection URL string you use to enable this communication protocol. On the broker side there are additional transports supported. You can specify the connection URL on an ActiveMQConnectionFactory (in a constructor or via the brokerURL property).

e.g. if you don't want to bother setting up JNDI and so forth and just wanna create a JMS connection you can do something like

```
ConnectionFactory factory = new ActiveMQConnectionFactory("tcp://somehost:61616");
Connection connection = factory.createConnection();
```

Protocol Summary

Protocol	Example	Description	Server?
VM	vm://host:port	Client connect to each other within the same JVM. This does use an asynchronous channel and a separate worker thread. You can enable sync sending using a query parameter, such as `vm://localhost?async=false`	Yes
TCP	tcp://host:port	Client connects to the broker at the given URL	Yes
SSL	ssl://host:port	Client connects to the broker at the given URL	Yes
Failover	failover:(Uri1,Uri2,Uri3,...,UriN)	Provides a list of possible URIs to connect to and one is randomly chosen. If the connection fails then the transport auto-reconnects to a different one	
Peer	peer://serviceName	Creates a pure peer to peer network of nodes of a given service name. In peer mode there is no server, nodes just automatically connect and make a peer network. The serviceName allows you to keep networks apart from each other, such as development, testing, UAT and production.	
Discovery	discovery://host:port	Uses Discovery to connect to an available broker of the correct channel name. If multiple brokers can be found then one is chosen at random. If the connection fails then another broker is chosen, if available	
Zeroconf	zeroconf:_activemq.broker.development.	Uses Zeroconf to connect to an available broker of the correct Zeroconf service name. If multiple brokers can be found then one is chosen at random. If the connection fails then another broker is chosen, if available	
HTTP	http://host:port	Client connects to the broker using HTTP tunnelling, with XML payloads suitable for going through firewalls	Yes
UDP	udp://host:port	Client connects to the broker at the given URL	
multicast	multicast://host:port	No server, though only works for pub/sub. A pure peer based network where all traffic is multicasted around and filtering is performed on the client.	

The *Server?* column above indiciates whether or not a protocol can be used in an ActiveMQ broker transport connector. All of the above protocols can be used in a JMS client to connect to the messaging fabric; only those protocols indicated can be used in a broker-side transport connector.

When connecting to an ActiveMQ broker, this could reside locally inside your JVM or be remote on another machine somewhere. If you want to enable the deployment of the ActiveMQ inside your JVM you can enable the useEmbeddedBroker property on the ActiveMQConnectionFactory.

Please refer to the topologies overview to see how we can use ActiveMQ in many different topologies to suit your messaging needs.

Specifying multiple URLs to connect to

If you want full HA to provide failover and auto-reconnection you can use the *failover:* prefix

```
failover:(tcp://foo:61699,tcp://bar:61617,tcp://whatnot:61698)
```

see Configuring Transports for more details