JBoss Messaging Clustering Introduction

A brief guide to clustering in JBoss Messaging

Table of Contents

1

JBoss Messaging Clustering Introduction

This guide gives a brief overview of the features available in JBoss Messaging Clustering 1.2.0.CR1

It also gives a high level explanation of how clustering works and shows you how to set up a simple cluster of JBoss Messaging servers.

This guide is work in progress.

It will expand considerably for the 1.2.0 GA release.

Please send your suggestions or comments to the JBoss Messaging user forum [http://www.jboss.org/index.html?module=bb&op=viewforum&f=238].

2

Introduction

JBoss Messaging 1.2 GA will provide a highly sophisticated clustering implementation, far more sophisticated than you'll find in the vast majority of other messaging systems.

It will allow you to smoothly distribute your application load across your cluster, intelligently balancing and utilising each nodes CPU cycles, with no single point of failure, providing a highly scalable and performant clustering implementation.

2.1. JBoss Messaging 1.2 GA features:

JBoss Messaging 1.2 GA Clustering will provide the following features:

- Distributed queues. Messages sent to a distributed queue while attached to a particular node will be routed to a queue instance on a particular node according to a routing policy.
- Distributed topics. Messages sent to a distributed topic while attached at a particular node will be received by subscriptions on other nodes.
- Fully reliable message distribution. Once and only once delivery is fully guaranteed. When sending messages to a topic with multiple durable subscriptions across a cluster we guarantee that message reaches all the subscriptions (or none of them in case of failure).
- Persistent level reliability guarantee without persistence! By replicating persistent messages between nodes in memory, we can obtain comparable reliability levels to persistenting messages to disk, without actually storing them to disk.
- Pluggable routing implementation. The policy for routing messages to a queue is fully pluggable and easily replaceable. The default policy always chooses a queue at the local node if there is one, and if not, it round robins between queues on different nodes.
- Intelligent message redistribution policy. Messages are automatically distributed between nodes depending on how fast or slow consumers are on certain nodes. If there are no or slow consumers on a particular queue node, messages will be pulled from that queue to a queue with faster consumers on a different node. The policy is fully pluggable.
- Shared durable subscriptions. Consumers can connect to the same durable subscription while attached to different nodes. This allows processing load from durable subscriptions to be distributed across the cluster in a similar way to queues.
- High availability and seamless failover. If the node you are connected to fails, you will automatically fail over to another node and will not lose any persistent messages. You can carry on with your session seamlessly where

you left off. Once and only once delivery of persistent messages is respected at all times.

2.2. JBoss Messaging 1.2.0.CR1 features:

What's in the CR1 release?

JBoss Messaging 1.2.0.CR1 contains all features planned for the GA release, with the exception of the "unreliable link scenario" (http://jira.jboss.org/jira/browse/JBMESSAGING-676), whose development is still on-going on a parallel branch.

Please note that this is a candidate release and it possible to contain bugs! We are releasing it to the community, so you can try it out, get familiar with it and feedback your experiences to us so we can improve it and stabilise it.

Thanks for your support!

All the final features are available apart from:

- Persistent level reliability guarantee without persistence. There is no option for in memory replication of persistent messages in this release.

Bear in mind you will need to get a bit more "down and dirty" with the configuration in this release, than you would with a GA release.

3

Clustering overview

3.1. JBoss Messaging Clustering Overview

Here's a brief overview of how clustering works in JBoss Messaging 1.2.

As mentioned in the previous chapter, please note that not all this functionality is available in this release.

Clustered destinations (queues and topics) can be deployed at all or none of the nodes of the cluster.

A JMS client uses HA JNDI to lookup the connection factory. A client side load balancing policy will automatically chose a node to connect to (This is similar to how EJB clustering chooses a node).

The JMS client has now made a connection to a node where it can create sessions, message producers and message consumers and browsers and send or consume messages, using the standard JMS api.

When a distributed queue is deployed across the cluster, individual partial queues are deployed on each node.

When a message is sent from a message producer attached to a particular node to a distributed quueue, a routing policy determines which partial queue will receive the message.

By default the router will always pass the message to a local queue, if there is one, this is so we avoid unnecessary network traffic.

If there is no local queue then a partial queue on a different node will be chosen by the router, by default this will be round robin between remote partial queues.

When a message is sent to a distributed topic while attached to a node, there may be multiple subscriptions on different nodes that need to receive the message. Depending on the number and location of subscriptions, the message may be multicast or unicast across the cluster so the other nodes can pick it up.

All group communication, unicast, multicast and group management is handled by JGroups.

In the case of shared durable subscriptions, if a durable subscription with the same name exists on more than node, then only one of the instances needs to receive the message.

Which one is determined by the same routing policy used to route messages to partial queues.

All of this is accomplished without losing the reliability guarantees required by JMS.

Subscriptions (both durable and non durable) can be created on all nodes and will receive messages sent via any node.

What happens if the consumers on one queue/subscription are faster/slower than consumers on another?

Normally this would result in messages building up on that queue and fast consumers being starved of work on another, thus wasting CPU cycles on the node that could be put to good use.

The most degenerate example is of a queue containing many messages then the consumers being closed on that queue. The messages might potentially remain stranded on the queue until another consumer attaches.

A routing policy is no use here, since the messages have already been routed to the queuee and the consumers closed / slowed down after they were routed there.

JBoss Messaging deals with this problem by intelligently pulling messages from other less busy nodes, if it detects idle consumers on the fast node and slow consumers on another node.

Another feature (not available in CR1) that will enable very fast, very scalable reliable messaging without using databases is in memory replication of reliable messages.

Normally, persistent messages are persisted in a shared database which is shared by all nodes in the cluster. JBoss Messaging 1.2.GA will contain an option where you can choose to not persist persistent messages in a database, but instead to replicate them between nodes of the cluster.

The idea here is the network IO on a fast network should be much faster than persisting to disk.

This solution should also be more scalable since different nodes replicate their messages onto different other nodes - there is no "master node".

If the messages are replicated onto suffficient nodes and the hardware is set-up with UPS, then we believe a comparable reliability guarantee to persisting messages to disk can be achieved. Of course, this won't be suitable for all situations, but you use the best tool for the job.

4

JBoss Messaging Clustering Installation

Note

You need at least ant 1.6.3 installed in order for the installation procedure to work correctly.

Use the `release-admin.xml` ant script shipped with the release to create individual cluster node configurations.

The typical usage is:

```
ant -f release-admin.xml [-Did=node-id] [-Dports=port-config-label] cluster-node
```

where:

- `node-id` is the unique node ID, an integer that must be unique per cluster. If not specified, it defaults to 0.
- `port-config-label` is a binding manager server configuration label. The short story behind this parameter is the following: multiple application servers running on the same physical machine need to use different service port ranges to avoid port conflicts. You can configure the whole port range used by a server instance by enabling a special service, the binding management service, and specifiying a "server" configuration in the binding manager's configuration file, which will determine specific port values to use when starting that instance. The Messaging installation script can enable the service binding manager and performs all configuration changes automatically. You only need to specify the "server" configuration you want to use, as 'port-config-label'. If you plan to run your clustering nodes on different physical machines, this parameter is irrelevant, and you should not use it. However, if you install two (or more) nodes of your cluster on the same physical machine, you need to give the value corresponding to a specific "server" configurations in the binding manager configuration file. JBoss AS ships "out-of-the-box" with several pre-configured port ranges: 'ports-default', 'ports-01', 'ports-02', 'ports-03'. Use one of these. If -Dports is not specified, the clustered instance created this way will fall over to the default port range for a JBoss instace. More details about the binding management service can be found in the Application Server documentation, at the following address http://docs.jboss.com/jbossas/guides/j2eeguide/r2/en/html_single/#ch10.bindingmanager

For example, in order to create the configuration for a four-node cluster intended to run on the same physical machine, use the following sequence:

```
ant -f release-admin.xml cluster-node
ant -f release-admin.xml -Did=1 -Dports=ports-01 cluster-node
ant -f release-admin.xml -Did=2 -Dports=ports-02 cluster-node
ant -f release-admin.xml -Did=3 -Dports=ports-03 cluster-node
```

The sequence will create four cluster node configurations ("messaging-node0", "messaging-node1", "messaging-

node2" and "messaging-node3").

The first command will create a cluster node with ID equals to '0' and using the default JBoss AS port assignments.

Warning

The configuration that has just been created uses a generic mysql service descriptor. Check `$JBOSS_HOME/server/messaging-node<id>/deploy/mysql-ds.xml` and verify that that:

- 1. Your database is indeed mysql.
- 2. It is accessible from every physical node you installed Messaging on.
- 3. Contains a schema (database/tablespace) named 'messaging'.
- 4. The URL (hostname and port), username and password are correct.
- 5. The installed mysql-driver.jar's version maches your database.

To start the cluster, from four different terminals, run:

```
cd $JBOSS_HOME/bin
./run.sh -c messaging-node0

cd $JBOSS_HOME/bin
./run.sh -c messaging-node1

cd $JBOSS_HOME/bin
./run.sh -c messaging-node2

cd $JBOSS_HOME/bin
./run.sh -c messaging-node3
```

A successful two node cluster startup produces a log similar to:

Node 0:

```
...

00:24:04,796 WARN  [JDBCPersistenceManager]

JBoss Messaging Warning: DataSource connection transaction isolation should be READ_COMMITTED, but
                         Using an isolation level less strict than READ_COMMITTED may lead to data
                         Using an isolation level more strict than READ_COMMITTED may lead to deadl

00:24:05,718 INFO  [ServerPeer] JBoss Messaging 1.2.0.CR1 server [0] started
00:24:06,328 INFO  [STDOUT]
-------------------------------------------------------
GMS: address is 127.0.0.1:2452
-------------------------------------------------------
00:24:08,406 INFO  [DefaultClusteredPostOffice] ClusteredPostOffice[0:Clustered JMS:127.0.0.1:2452]
00:24:08,468 INFO  [STDOUT]
-------------------------------------------------------
GMS: address is 127.0.0.1:2455
-------------------------------------------------------
00:24:10,906 INFO  [ConnectionFactory] Connector socket://10.11.14.105:4457 has leasing enabled, le
```

```
00:24:10,921 INFO  [ConnectionFactory] [/ConnectionFactory, /XAConnectionFactory, java:/ConnectionF
00:24:10,953 INFO  [QueueService] Queue[/queue/DLQ] started, fullSize=75000, pageSize=2000, downCac
00:24:10,953 INFO  [QueueService] Queue[/queue/ExpiryQueue] started, fullSize=75000, pageSize=2000,
00:24:10,953 INFO  [TopicService] Topic[/topic/testTopic] started, fullSize=75000, pageSize=2000, d
00:24:10,953 INFO  [TopicService] Topic[/topic/securedTopic] started, fullSize=75000, pageSize=2000
00:24:10,968 INFO  [TopicService] Topic[/topic/testDurableTopic] started, fullSize=75000, pageSize=
00:24:10,968 INFO  [QueueService] Queue[/queue/testQueue] started, fullSize=75000, pageSize=2000, d
00:24:10,968 INFO  [QueueService] Queue[/queue/A] started, fullSize=75000, pageSize=2000, downCache
00:24:10,968 INFO  [QueueService] Queue[/queue/B] started, fullSize=75000, pageSize=2000, downCache
00:24:10,968 INFO  [QueueService] Queue[/queue/C] started, fullSize=75000, pageSize=2000, downCache
00:24:10,968 INFO  [QueueService] Queue[/queue/D] started, fullSize=75000, pageSize=2000, downCache
00:24:10,968 INFO  [QueueService] Queue[/queue/ex] started, fullSize=75000, pageSize=2000, downCach
00:24:10,984 INFO  [QueueService] Queue[/queue/PrivateDLQ] started, fullSize=75000, pageSize=2000,
00:24:10,984 INFO  [QueueService] Queue[/queue/PrivateExpiryQueue] started, fullSize=75000, pageSiz
00:24:10,984 INFO  [QueueService] Queue[/queue/QueueWithOwnDLQAndExpiryQueue] started, fullSize=750
00:24:10,984 INFO  [TopicService] Topic[/topic/TopicWithOwnDLQAndExpiryQueue] started, fullSize=750
00:24:10,984 INFO  [QueueService] Queue[/queue/QueueWithOwnRedeliveryDelay] started, fullSize=75000
00:24:10,984 INFO  [TopicService] Topic[/topic/TopicWithOwnRedeliveryDelay] started, fullSize=75000
00:24:11,000 INFO  [QueueService] Queue[/queue/testDistributedQueue] started, fullSize=75000, pageS
00:24:11,000 INFO  [TopicService] Topic[/topic/testDistributedTopic] started, fullSize=75000, pageS
00:24:11,093 INFO  [ConnectionFactoryBindingService] Bound ConnectionManager 'jboss.jca:name=JmsXA,
00:24:11,375 INFO  [TomcatDeployer] deploy, ctxPath=/jmx-console, warUrl=.../deploy/jmx-console.war
00:24:12,171 INFO  [Http11BaseProtocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-8080
00:24:12,421 INFO  [ChannelSocket] JK: ajp13 listening on /0.0.0.0:8009
00:24:12,453 INFO  [JkMain] Jk running ID=0 time=0/47  config=null
00:24:12,515 INFO  [Server] JBoss (MX MicroKernel) [4.0.5.GA (build: CVSTag=Branch_4_0 date=2006112

00:27:21,343 INFO  [DefaultClusteredPostOffice] ClusteredPostOffice[0:Clustered JMS:127.0.0.1:2452]
```

Node 1:

```
...

00:33:54,468 WARN  [JDBCPersistenceManager]

JBoss Messaging Warning: DataSource connection transaction isolation should be READ_COMMITTED, but
                         Using an isolation level less strict than READ_COMMITTED may lead to data
                         Using an isolation level more strict than READ_COMMITTED may lead to deadl

00:33:55,062 INFO  [ServerPeer] JBoss Messaging 1.2.0.CR1 server [1] started
00:33:55,609 INFO  [STDOUT]
-------------------------------------------------------
GMS: address is 127.0.0.1:2514
-------------------------------------------------------
00:33:57,734 INFO  [DefaultClusteredPostOffice] ClusteredPostOffice[1:Clustered JMS:127.0.0.1:2514]
00:33:57,765 INFO  [STDOUT]
-------------------------------------------------------
GMS: address is 127.0.0.1:2519
-------------------------------------------------------
00:34:00,203 INFO  [ConnectionFactory] Connector socket://10.11.14.105:4557 has leasing enabled, le
00:34:00,203 INFO  [ConnectionFactory] [/ConnectionFactory, /XAConnectionFactory, java:/ConnectionF
00:34:00,234 INFO  [QueueService] Queue[/queue/DLQ] started, fullSize=75000, pageSize=2000, downCac
00:34:00,234 INFO  [QueueService] Queue[/queue/ExpiryQueue] started, fullSize=75000, pageSize=2000,
00:34:00,234 INFO  [TopicService] Topic[/topic/testTopic] started, fullSize=75000, pageSize=2000, d
00:34:00,250 INFO  [TopicService] Topic[/topic/securedTopic] started, fullSize=75000, pageSize=2000
00:34:00,250 INFO  [TopicService] Topic[/topic/testDurableTopic] started, fullSize=75000, pageSize=
00:34:00,250 INFO  [QueueService] Queue[/queue/testQueue] started, fullSize=75000, pageSize=2000, d
00:34:00,250 INFO  [QueueService] Queue[/queue/A] started, fullSize=75000, pageSize=2000, downCache
00:34:00,250 INFO  [QueueService] Queue[/queue/B] started, fullSize=75000, pageSize=2000, downCache
00:34:00,250 INFO  [QueueService] Queue[/queue/C] started, fullSize=75000, pageSize=2000, downCache
00:34:00,250 INFO  [QueueService] Queue[/queue/D] started, fullSize=75000, pageSize=2000, downCache
00:34:00,250 INFO  [QueueService] Queue[/queue/ex] started, fullSize=75000, pageSize=2000, downCach
```

```
      00:34:00,265 INFO  [QueueService] Queue[/queue/PrivateDLQ] started, fullSize=75000, pageSize=2000,
      00:34:00,265 INFO  [QueueService] Queue[/queue/PrivateExpiryQueue] started, fullSize=75000, pageSiz
      00:34:00,265 INFO  [QueueService] Queue[/queue/QueueWithOwnDLQAndExpiryQueue] started, fullSize=750
      00:34:00,265 INFO  [TopicService] Topic[/topic/TopicWithOwnDLQAndExpiryQueue] started, fullSize=750
      00:34:00,265 INFO  [QueueService] Queue[/queue/QueueWithOwnRedeliveryDelay] started, fullSize=75000
      00:34:00,265 INFO  [TopicService] Topic[/topic/TopicWithOwnRedeliveryDelay] started, fullSize=75000
      00:34:00,296 INFO  [QueueService] Queue[/queue/testDistributedQueue] started, fullSize=75000, pageS
      00:34:00,296 INFO  [TopicService] Topic[/topic/testDistributedTopic] started, fullSize=75000, pageS
      00:34:00,343 INFO  [ConnectionFactoryBindingService] Bound ConnectionManager 'jboss.jca:name=JmsXA,
      00:34:00,453 INFO  [TomcatDeployer] deploy, ctxPath=/jmx-console, warUrl=.../deploy/jmx-console.war
      00:34:00,796 INFO  [Http11BaseProtocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-8180
      00:34:01,078 INFO  [ChannelSocket] JK: ajp13 listening on /0.0.0.0:8109
      00:34:01,125 INFO  [JkMain] Jk running ID=0 time=0/125  config=null
      00:34:01,125 INFO  [Server] JBoss (MX MicroKernel) [4.0.5.GA (build: CVSTag=Branch_4_0 date=2006112
```

Note

The installation script may fail while installing Messaging with source-generated JBoss 4.0.5.GA-ejb3 instance. This is because release-admin.xml relies on finding `$JBOSS_HOME/docs/examples/binding-manager/sample-bindings.xml`. 4.0.5.GA-ejb3 installations seem not to have a "docs" sub-directory. A very simple work-around for this situation is to recursively copy the "docs" sub-directory available under a regular (non-EJB3) source-generated 4.0.5.GA instance and retry the installation process.

5

JBoss Messaging Clustering Configuration

In order to understand JBoss Messaging clustering configuration, we will start with a short clustering architectural overview, where we will identify "configuration areas", meaning architectural elements that have corresponding configuration files, and whose behavior can be changed through configuration.

5.1. Clustering Architectural Overview

One of the fundamental Messaging Core building blocks is the "Post Office". A JBoss Messaging Post Office is message routing component, which accepts messages for delivery and synchronously forwards them to their destination queues or topic subscriptions.

There is a single Post Office instance per JBoss Messaging server (cluster node). Both queues and topics deployed on a JBoss Messaging node are "plugged" into that Post Office instance. Internally JBoss Messaging only deals with the concepts of queues, and considers a topic to just be a set of queues (one for each subscription). Depending on the type of subscription - durable or non-durable - the corresponding queue saves messages to persistent storage or it just holds messages in memory and discards them when the non-durable subscription is closed.

Therefore, for a JMS queue, the Post Office routes messages to one and only one core queue, depending on the queue name, whereas for a JMS topic, the Post Office routes a message to a set of core queues, one for each topic subscription, depending on the topic name.

Clustering across multiple address spaces is achieved by clustering Post Office instances. Each JBoss Messaging cluster node runs a Clustered Post Office instance, which is aware of the presence of all other clustered Post Offices in the cluster. There is an one-to-one relationship between cluster nodes and clustered Post Office instances. So, naturally, the most important piece of clustering configuration is the *clustered Post Office service configuration*, covered in detail below.

Clustered Post Office instances connect to each other via JGroups and they heavily rely on JGroups group management and notification mechanisms. *JGroups stack configuration* is an essential part of JBoss Messaging clustering configuration. JGroups configuration is only briefly addressed in this guide. Detailed information on JGroups can be found in JGroups release documentation or on-line at http://www.jgroups.org or http://wiki.jboss.org/wiki/Wiki.jsp?page=JGroups

When routing messages, a clustered Post Office has a choice of forwarding the message to local queues or remote queues, plugged into remote Post Office instances that are part of the same cluster. Local queues are usually preferred, but if a local queue is part of a distributed queue, has no consumers, and other local queues part of the same distributed queue have consumers, messages can be automatically redistributed, subject of the message redistribution policy in effect. This allows us to create distributed queues and distributed topics. *Message redistribution configuration* is another subject that we will insist on.

Please note that some elements of clustering configuration are likely to change before the 1.2 final release. In par-

ticular we will try to add the ability for JBoss Messaging to use a pre-configured JGroups multiplex channel when used inside JBoss Application Server, but this is subject to availability of a AS clustering service supporting a multiplexed JGroups channel; such a service is currently being worked on by the AS Clustering team.

5.2. Clustered Post Office Configuration

In JBoss Messaging 1.2.0, the JGroups configuration for each clustered Post Office instance is specified in the Post Office MBean service configuration element. The Post Office configuration is present in the persistence configuration file corresponding to the specific database you're using for the cluster instance.

So, if you are using a MySQL database instance as shared persistent storage for your cluster, the Post Office configuration for a specific node instance is available in `$JBOSS_HOME/server/messaging-nodeX/deploy/jboss-messaging.sar/clustered-mysql-persistence-service.xml`

An example of a Clustered Post Office configuration is shown below:

```
        <mbean code="org.jboss.messaging.core.plugin.ClusteredPostOfficeService"
        name="jboss.messaging:service=PostOffice"
        xmbean-dd="xmdesc/ClusteredPostOffice-xmbean.xml">
        <depends optional-attribute-name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
        <depends>jboss.jca:service=DataSourceBinding,name=DefaultDS</depends>
        <depends optional-attribute-name="TransactionManager">jboss:service=TransactionManager</depends>
        <attribute name="PostOfficeName">Clustered JMS</attribute>
        <attribute name="DataSource">java:/DefaultDS</attribute>
        <attribute name="CreateTablesOnStartup">true</attribute>
        <attribute name="SqlProperties"><![CDATA[
        CREATE_POSTOFFICE_TABLE=CREATE TABLE JBM_POSTOFFICE (POSTOFFICE_NAME VARCHAR(255), NODE_ID INTEG
        INSERT_BINDING=INSERT INTO JBM_POSTOFFICE (POSTOFFICE_NAME, NODE_ID, QUEUE_NAME, COND, SELECTOR,
        DELETE_BINDING=DELETE FROM JBM_POSTOFFICE WHERE POSTOFFICE_NAME=? AND NODE_ID=? AND QUEUE_NAME=?
        LOAD_BINDINGS=SELECT NODE_ID, QUEUE_NAME, COND, SELECTOR, CHANNEL_ID, IS_FAILED_OVER, FAILED_NOD
        ]]></attribute>
        <attribute name="GroupName">DefaultPostOffice</attribute>
        <attribute name="StateTimeout">5000</attribute>
        <attribute name="CastTimeout">5000</attribute>
        <attribute name="StatsSendPeriod">10000</attribute>
        <attribute name="MessagePullPolicy">org.jboss.messaging.core.plugin.postoffice.cluster.NullMessa
        <attribute name="ClusterRouterFactory">org.jboss.messaging.core.plugin.postoffice.cluster.Defaul

        <attribute name="AsyncChannelConfig">
        <config>
        <UDP mcast_recv_buf_size="500000" down_thread="false" ip_mcast="true" mcast_send_buf_size="32000
        mcast_port="45567" ucast_recv_buf_size="500000" use_incoming_packet_handler="false"
        mcast_addr="228.8.8.8" use_outgoing_packet_handler="true" loopback="true" ucast_send_buf_size="3
        <AUTOCONF down_thread="false" up_thread="false"/>
        <PING timeout="2000" down_thread="false" num_initial_members="3" up_thread="false"/>
        <MERGE2 max_interval="10000" down_thread="false" min_interval="5000" up_thread="false"/>
        <FD timeout="2000" max_tries="3" down_thread="false" up_thread="false" shun="true"/>
        <VERIFY_SUSPECT timeout="1500" down_thread="false" up_thread="false"/>
        <pbcast.NAKACK max_xmit_size="8192" down_thread="false" use_mcast_xmit="true" gc_lag="50" up_thr
        retransmit_timeout="100,200,600,1200,2400,4800"/>
        <UNICAST timeout="1200,2400,3600" down_thread="false" up_thread="false"/>
        <pbcast.STABLE stability_delay="1000" desired_avg_gossip="20000" down_thread="false" max_bytes="
        <FRAG frag_size="8192" down_thread="false" up_thread="false"/>
        <VIEW_SYNC avg_send_interval="60000" down_thread="false" up_thread="false" />
        <pbcast.GMS print_local_addr="true" join_timeout="3000" down_thread="false" join_retry_timeout="
        </config>
        </attribute>
```

```
        <attribute name="SyncChannelConfig">
        <config>
        <UDP mcast_recv_buf_size="500000" down_thread="false" ip_mcast="true" mcast_send_buf_size="32000
        mcast_port="45568" ucast_recv_buf_size="500000" use_incoming_packet_handler="false"
        mcast_addr="228.8.8.8" use_outgoing_packet_handler="true" loopback="true" ucast_send_buf_size="3
        <AUTOCONF down_thread="false" up_thread="false"/>
        <PING timeout="2000" down_thread="false" num_initial_members="3" up_thread="false"/>
        <MERGE2 max_interval="10000" down_thread="false" min_interval="5000" up_thread="false"/>
        <FD timeout="2000" max_tries="3" down_thread="false" up_thread="false" shun="true"/>
        <VERIFY_SUSPECT timeout="1500" down_thread="false" up_thread="false"/>
        <pbcast.NAKACK max_xmit_size="8192" down_thread="false" use_mcast_xmit="true" gc_lag="50" up_thr
        retransmit_timeout="100,200,600,1200,2400,4800"/>
        <UNICAST timeout="1200,2400,3600" down_thread="false" up_thread="false"/>
        <pbcast.STABLE stability_delay="1000" desired_avg_gossip="20000" down_thread="false" max_bytes="
        <FRAG frag_size="8192" down_thread="false" up_thread="false"/>
        <VIEW_SYNC avg_send_interval="60000" down_thread="false" up_thread="false" />
        <pbcast.GMS print_local_addr="true" join_timeout="3000" down_thread="false" join_retry_timeout="
        <pbcast.STATE_TRANSFER down_thread="false" up_thread="false"/>
        </config>
        </attribute>
        </mbean>
```

Relevant clustered Post Office configuration attributes are discussed below:

5.2.1. `GroupName`

This identifies the JGroups group the clustered Post Office will connect to. All clustered Post Office instances with the same group name will connect to that group.

5.2.2. `StateTimeout`

The maximum amount of time in milliseconds to wait when making a request to get the group state and waiting for the result. You will not normally need to change this value. Default is 5000 ms.

5.2.3. `CastTimeout`

The maximum amount of time in milliseconds to wait when casting a message and waiting for a result. You will not normally need to change this value. Default is 5000 ms.

5.2.4. `StatsSendPeriod`

The period in milliseconds between much statistics for each queue will be cast across the cluster.

5.2.5. `ClusterRouterFactory`

The fully qualified class name of the class that implements a factory for creating message routers. For different message routing policies this can be changed. Default is `org.jboss.messaging.core.plugin.postoffice.cluster.DefaultRouterFactory`. This factory creates routers that always favor local queues.

5.2.6. `MessagePullPolicy`

The fully qualified class name of the class that implements the MessagePullPolicy. The message pull policy specifies how messages are redistributed after they leave the Post Office and are delivered to queues. You can find more on message redistribution policy in the dedicated section below. By default, the message redistribution policy is `org.jboss.messaging.core.plugin.postoffice.cluster.DefaultMessagePullPolicy`.

5.3. JGroups Stack Configuration

The details of the JGroups configuration won't be discussed here since it is standard JGroups configuration. Detailed information on JGroups can be found in JGroups release documentation or on-line at http://www.jgroups.org or http://wiki.jboss.org/wiki/Wiki.jsp?page=JGroups

5.4. Message Redistribution Configuration

Each clustered Post Office instance can be configured to use a customizable *message redistribution policy*. The message redistribution policy dictates what happens with messages that are delivered to a local queue that is part of a distributed queue or a distributed subscription. In order to use a specific message redistribution policy, use the fully qualified class name of the class that implements the MessagePullPolicy.

By default, the message redistribution policy is `org.jboss.messaging.core.plugin.postoffice.cluster.DefaultMessagePullPolicy` which tries to redistribute messages depending on receivers's consumption speed.

To disable message redistribution completely, specify `org.jboss.messaging.core.plugin.postoffice.cluster.NullMessagePullPolicy` as the value of `MessagePullPolicy` attribute. In this case, a message is not redistributed, even if the local queue that has been delivered to has no consumers.