Migrating JBoss Messaging 1.4 to HornetQ

This article is provided as a general guide for users to migrate their existing Jboss Messaging applications in JBoss Application Server 4 and 5 to HornetQ 2.0.

General Steps

1. Shutdown client and server

JBoss Messaging uses a database to store its persistent data (unless a null persistence is used). You don't need to shutdown the database for the purpose of migration. HornetQ doesn't need a Database at all. It has its own high performance journal system instead.

2. Back up data

Before starting the migration, it is important to back up all the data used in your application and JBoss Messaging server.

The following data are used by JBoss Messaging:

2.1. JBoss Messaging database tables

JBoss Messaging uses a few of database tables to store persistent data. Some of them keep some internal state information of JBoss Messaging. Some others contains persistent messages and security settings. Below is a list of tables that hold important data:

JBM_MSG_REF and JBM_MSG – The two tables are used to store persistent messages as well as their states.

JBM_TX and JBM_TX_EX (since 1.4.0.SP3.CP10 and 1.4.6.GA.SP1) – The two tables are used to keep transaction states.

JBM_USER and JBM_ROLE – The two tables are used to store users and roles information.

JBM_POSTOFFICE – It holds bindings information in the post office.

2.2 JBoss Messaging configuration files.

The location where most configuration files go is under {jboss-profile}/deploy/messaging in AS 5 or {jboss-profile}/deploy/messaging/jboss-messaging.sar in AS 4. For example if your JBOSS_HOME is /home/jboss/jboss-5.1.0.GA and your JBoss Messaging server profile is 'messaging', then the location would be /home/jboss/jboss-5.1.0.GA/messaging/deploy/messaging.

Applications may choose other places to deploy some configuration files than this location.

Below is a list of JBoss Messaging configuration files that you need to migrate to HornetQ

1. Connection Factory service configuration files – these files contain JMS connection factories deployed with JBoss Messaging server.

2. Destination service configuration files – these files contain JMS queues and topics deployed with JBoss Messaging server.
3. Bridge service configuration files – these files contains bridge services deployed with JBoss Messaging server.

Other configuration files such as messaging-service.xml and database persistence configuration file are JBoss Messaging mbean configurations. They are not targets of the migration – HornetQ implementations consists of only POJOs.

JBoss Messaging also rely on some other services to work. They are JBoss Remoting and Jgroups services. Their configuration files contains settings specific to applications. As HornetQ has a different transport layer and cluster design, you need to map the parameters in those configuration files to their HornetQ equivalents (if any).

3. Application Code

If you are using standard JMS in your application, there is no need to change your source code.

If you are using JBoss Messaging proprietary features, such as ordering groups, you need to adapt them to HornetQ equivalent features.

For convenience, the following table lists the JBoss Messaging JMS implementation classes and their corresponding HornetQ equivalents.

JBoss Messaging JMS implementation classes	**HornetQ equivalents**
org.jboss.jms.client.JBossConnectionFactory	org.hornetq.jms.client.HornetQConnectionFactory
org.jboss.jms.client.JBossConnection	org.hornetq.jms.client.HornetQConnection
org.jboss.jms.client.JBossSession	org.hornetq.jms.client.HornetQSession
org.jboss.jms.client.JBossMessageProducer	org.hornetq.jms.client.HornetQMessageProducer
org.jboss.jms.client.JBossMessageConsumer	org.hornetq.jms.client.HornetQMessageConsumer

Note: Unless you have used JBM specific APIs, you don't need to explicitly cast your JMS objects to its concrete implementations. Just use the standard JMS APIs whenever possible.

4. Installing HornetQ

To install HornetQ, use the scripts provided in HornetQ's binary distribution (under config/jboss-as-4 for AS 4 and config/jboss-as-5 for AS 5, please see HornetQ Quickstart Guide Chapter 5 for details).

The scripts automatically create two profiles (one non-clustered and clustered) with default configurations. You need to create more if your application requires more nodes. To do this just copy from the corresponding existing profiles.

5. Server Configuration Migration

As HornetQ's configuration are quite different from JBoss Messaging, it is not possible to have a one to one mapping between the two. However some of the JBoss Messaging server attributes can find their equivalents in HornetQ configuration. Below gives those attributes for your reference when doing the migration. For more details please consult the user's manuals. (Unless explicitly indicated, attributes mentioned in HornetQ server column are from hornetq-configuration.xml)

JBoss Messaging Server Attributes (ServerPeer xmbean)	**HornetQ Server Attributes**
ServerPeerID	N/A HornetQ doesn't need a server ID specified.
DefaultQueueJNDIContext DefaultTopicJNDIContext	N/A
PostOffice	N/A
DefaultDLQ	N/A In stead of DLQ, HornetQ defines dead letter address at core level, and there is no default dead letter address for an address unless you specify one.
DefaultMaxDeliveryAttempts	N/A. In HornetQ, it's always 10.
DefaultExpiryQueue	N/A In stead of ExpiryQueue, HornetQ defines expiry address at core level, and there is no default expiry address for an address unless you specify one
DefaultRedeliveryDelay	N/A HornetQ's default redelivery delay is always 0, meaning no delay.
MessageCounterSamplePeriod	message-counter-sample-period
FailoverStartTimeout	N/A
FailoverCompleteTimeout	N/A
DefaultMessageCounterHistoryDayLimit	N/A
ClusterPullConnectionFactory	N/A
DefaultPreserveOrdering	N/A
RecoverDeliveriesTimeout	N/A
EnableMessageCounters	Message-counter-enabled
SuckerPassword	Cluster-password
SuckerConnectionRetryTimes	bridges.reconnect- attempts
SuckerConnectionRetryInterval	bridges.retry-interval
StrictTCK	N/A
Destinations MessageCounters MessageStatistics	N/A. These are part of HornetQ management functionalities. Please refer to HornetQ user's manual for details.
SupportsFailover	N/A
PersistenceManager	N/A HornetQ uses its built-in high-performance journal as its persistent utility

JMSUserManager	N/A
SecurityStore	N/A The security manager is configured in hornetq-beans.xml or hornet-jboss-beans.xml (in JBoss AS).

6. Migrating JMS Administered Objects and Bridges

The ways used by HornetQ to create and deploy JMS connection factories, destinations and bridges are different from those used by JBoss Messaging.

With JBoss Messaging, a JMS object (a connection factory or a JMS queue/topic) or a bridge is configured as a Mbean service within a JBoss Application server, whereas with HornetQ it is implemented as a POJO. Therefore, to migrate the configuration of such an object from JBoss Messaging to HornetQ, you need to know which configuration parameter in JBM maps to which one in HornetQ.

The following gives the mapping of those configurations between JBoss Messaging and HornetQ.

6.1 JMS Connection Factories

JBoss Messaging Connection Factories Attributes	**HornetQ JMS Connection Factories Attributes**
ClientID	connection-factory.client-id
JNDIBindings	connection-factory.entries
PrefetchSize	connection-factory.consumer-window-size
SlowConsumers	N/A equivalent to consumer-window-size = 0
StrictTck	N/A
SendAcksAsync	connection-factory.block-on-acknowledge
DefaultTempQueueFullSize DefaultTempQueuePageSize DefaultTempQueueDownCacheSize	N/A
DupsOKBatchSize	connection-factory.dups-ok-batch-size
SupportsLoadBalancing	N/A
SupportsFailover	N/A
DisableRemotingChecks	N/A
LoadBalancingFactory	connection-factory.connection-load-balancing-policy-class-name
Connector	connection-factory.connectors
EnableOrderingGroup DefaultOrderingGroupName	N/A

Note: Unless otherwise described, the HornetQ attributes in the table are in hornetq-jms.xml.

6.2 JMS Queues and Topics

6.2.1 Queue configurations

JBoss Messaging Queue Attributes	**HornetQ Attributes**
Name	queue.name (hornetq-jms.xml)
JNDIName	queue.entry (hornetq-jms.xml)
DLQ	address-settings.dead-letter-address
ExpiryQueue	address-settings.expiry-address
RedeliveryDelay	address- settings.redelivery-delay
MaxDeliveryAttempts	address-settings.max- delivery-attempts
SecurityConfig	security-settings
FullSize	address-settings.max- size-bytes Note: due to different paging mechanism in HornetQ, the paging attributes do not have exactly the same meaning as JBM's. Refer to user's manual for details.
PageSize	address-settings.page- size-bytes Note: due to different paging mechanism in HornetQ, the paging attributes do not have exactly the same meaning as JBM's. Refer to user's manual for details.
DownCacheSize	N/A
CreatedProgrammatically MessageCount ScheduledMessageCount MessageCounter MessageCounterStatistics ConsumerCount	Please refer to org.hornetq.api.jms.management.JMSQueueControl for retrieving those attributes.
DropOldMessageOnRedeploy	N/A
MaxSize	N/A
Clustered	N/A

Note: Unless otherwise described, the HornetQ attributes in the table are in hornetq-configuration.xml.

6.2.2 Topic configurations

JBoss Messaging Queue Attributes	**HornetQ Attributes**
Name	topic.name (hornetq-jms.xml)
JNDIName	topic.entry (hornetq-jms.xml)
DLQ	address-settings.dead-letter-address
ExpiryQueue	address-settings.expiry-address

RedeliveryDelay	address- settings.redelivery-delay
MaxDeliveryAttempts	address-settings.max-delivery-attempts
SecurityConfig	security-settings
FullSize	address-settings.max- size-bytes Note: due to different paging mechanism in HornetQ, the paging attributes do not have exactly the same meaning as JBM's. Refer to user's manual for details.
PageSize	address-settings.page- size-bytes Note: due to different paging mechanism in HornetQ, the paging attributes do not have exactly the same meaning as JBM's. Refer to user's manual for details.
DownCacheSize	N/A
CreatedProgrammatically MessageCounterHistoryDayLimit MessageCounters AllMessageCount DurableMessageCount NonDurableMessageCount AllSubscriptionsCount DurableSubscriptionsCount NonDurableSubscriptionsCount	Please refer to org.hornetq.api.jms.management.TopicControl for retrieving those attributes.
MaxSize	N/A
Clustered	N/A
DropOldMessageOnRedeploy	N/A

Note: Unless otherwise described, the HornetQ attributes in the table are in hornetq-configuration.xml.

6.3 JMS Bridges

HornetQ's JMS Bridge has its attributes defined in its bean configuration files – as parameters of the Bridge bean's constructor. Please refer to Chapter 33 of the HornetQ User's Manual for details.

JBoss Messaging Bridge Attributes	**HornetQ JMS Bridge bean constructor parameters**
SourceProviderLoader	SourceCFF
TargetProviderLoader	TargetCFF
SourceDestinationLookup	SourceDestinationFactory
TargetDestinationLookup	TargetDestinationFactory
SourceUsername	Source user name parameter
SourcePassword	Source user password parameter
TargetUsername	Target user name parameter
TargetPassword	Target password parameter
QualityOfServiceMode	Quality of Service parameter
Selector	Selector parameter
MaxBatchSize	Max batch size parameter

MaxBatchTime	Max batch time parameter
SubName	Subscription name parameter
ClientID	Client ID parameter
FailureRetryInterval	Failure retry interval parameter
MaxRetries	Max retry times parameter
AddMessageIDInHeader	Add Message ID in Header parameter

7. Other configurations in JBoss Messaging

There are two kinds of configurations for JBoss Messaging's dependent services, one for JBoss Remoting and the other for Jgroups. The two services are independent projects and their configurations belongs to their respective documentations.

HornetQ doesn't use them at all. It has its own pluggable transportation architecture and clustering implementation. HornetQ currently uses Netty as its transport.

You need to consult corresponding documents for a complete description of the parameters in Remoting and Jgroups in order to get a proper settings for HornetQ.

8. Migrating Existing Messages

After you have migrated all the JMS destinations on to HornetQ, you can
It is recommended that user use Bridge services to move existing messages in JBoss Messaging to HornetQ.

For any prepared transactions, it is recommended that those transactions be finished with JBoss Messaging.

9. Applications that uses management APIs

JBoss Messaging exposes the management API through mbean interfaces. HornetQ has different ways to expose their management API. Please refer to HornetQ's user's manual (Chapter 30).

While JBoss Messaging management API's can be accessed by way of JMX, HornetQ has as many as three different ways of doing so, i.e.

- Using JMX -- JMX is the standard way to manage Java applications
- Using the core API -- management operations are sent to HornetQ server using core messages
- Using the JMS API -- management operations are sent to HornetQ server using JMS messages

The three ways of management provided in HornetQ have same set of functionalities. That means a functionality achieved by one way can also be achieved by another.

Below we listed the JBoss Messaging manageable objects and their HornetQ JMX counterparts. For other management APIs such as core API and JMS messages please see HornetQ's user's manual.

JBoss Messaging management Objects (xmbeans)	**HornetQ management Objects**
org.jboss.jms.server.ServerPeer	org.hornetq.api.jms.management.JMSServerControl
org.jboss.jms.server.connectionfactory.ConnectionFactory	org.hornetq.api.jms.management.ConnectionFactoryControl
org.jboss.jms.server.destination.QueueService	org.hornetq.api.jms.management.JMSQueueControl
org.jboss.jms.server.destination.TopicService	org.hornetq.api.jms.management.TopicControl

Note: Some of the JBoss Messaging mbeans have no peers in HornetQ. For example there is no equivalent JDBCPersistenceManagerService in HornetQ because in HornetQ there is no longer a data source needed as JBoss Messaging does.

10. Removing JBM and restart your application.

To remove JBM, simply delete the JBoss profile that containing the JBM.

After JBM has been removed, you can restart client and server.