GF 3.1 JMS Integration one-pager

Table of Contents

1. Introduction

1.1. Project/Component Working Name

GlassFish JMS integration changes for GF 3.1

1.2. Name(s) and e-mail address of Document Author(s)/Supplier

Satish Kumar

Satish.Kumar@SUN.COM

1.3. Date of This Document

19 May 2010

2. Project Summary

2.1. Project Description

The key focus of this release is to reinstate support for clustering of MQ brokers in GlassFish. This is essentially a continuation of a feature that has been available in GF 2.x release. However, the following enhancements are planned with this release:

- Support for auto-clustering of MQ brokers in Embedded mode. The previous releases only supported auto-clustering in the LOCAL mode.
- Remove dependence on master broker for non-HA clusters
- Support for dynamic cluster changes

Other important features include:

- Integrating existing MQ broker metrics with GF monitoring framework
- JMS support for embedded GF API

This document primarily focuses on the GlassFish JMS module related changes. The MQ changes are documented as separate one-pagers. The list of features and links to the respective one-pagers are given in the table below.

Feature	One-pager link
Dynamically Synchronize Broker List in MQ Conventional Cluster with GlassFish Cluster	https://mq.dev.java.net/4.5-content/sync-brokerlist-in-glassfish.txt
Support for conventional clustering of MQ brokers in Embedded mode (Broker Embedded)	https://mq.dev.java.net/4.5-content/embeddedBrokersInConventionalClusters-one-pager.html
Improve MQ conventional clustering with master broker	https://mq.dev.java.net/4.5-content/cluster-improvemaster.txt
MQ conventional clustering without master broker in GlassFish EMBEDED/LOCAL Mode (Broker Embedded or Broker Local)	https://mq.dev.java.net/4.5-content/cluster-nomaster-db.txt

2.2. Risks and Assumptions

The JMS clustering feature depends on the timely delivery of the GF clustering changes and MQ 4.5.

3. Problem Summary

3.1. Problem Area

Support for MQ clusters:

In order to maintain feature parity with v2.1 and re-enabling clustering, this is a key release driver for 3.1. Earlier releases had supported clustering of MQ broker (in HA and non-HA clusters) for Local and Remote modes of MQ integration. However, the following limitations existed:

- Auto-clustering of MQ brokers was only supported in LOCAL mode
- Dynamic changes to the cluster topology were not recognized without restarting the entire cluster

3.2. Justification

- Clustering is a key release driver for the 3.1 release
- Clustering of embedded brokers – Both the GF instance and broker instance run on the same VM
- Dynamic clustering - the MQ cluster would reshape itself based on changes to the GF cluster hence ensures a closer synergy with GF.
- Removing the dependence on master broker – to have a closer synergy with GF's symmetric clusters.

4. Technical Description

4.1 Support for MQ clusters

GlassFish Message Queue has in-built clustering capacities. It supports two modes of clustering:

- Conventional Clusters with service HA
- Enhanced Clusters with data HA

The most common use case is to have GF and MQ clusters to interoperate with each other seamlessly. However, there are differences in the clustering architectures of GF and MQ. A key difference is that while GF clusters are homogenous, MQ conventional clusters require one instance to be designated as a master broker. The master broker is required for certain admin related operations like creation/update/deletion of durable subscription and physical destination and is required to be running all the time. MQ broker instances need to rendezvous with the master broker at start-up for them to operate correctly. However, it is important to note that designating a master broker is only required for conventional clusters. Enhanced clusters use a shared database to store configuration information.

There are three possible modes for integrating GF clusters with MQ clusters. The modes are:

- Embedded – The MQ broker instance runs in the same process as the GF instance.
- Local – GF instance and MQ broker instance are co-located on the same machine but run as independent processes
- Remote – GF and MQ are configured independently but interact through a configured JMS hosts list.

4.1.1 Auto-clustering

Typically, setting up a joint GF and MQ cluster would require the user to configure a GF cluster and MQ cluster independently and link the two with additional configuration in GF through an addresslist that the RA can then use to communicate with the MQ cluster. Auto-clustering is a feature by which a MQ cluster is auto-created when a GF cluster is configured by the user. The process is entirely transparent to the user since there is no additional configuration that will be required to get this to work. This feature was also supported in v2.1 but the default mode for auto-clustering was LOCAL. The key highlights of auto-clustering are:

- One MQ instance is configured per GF instance
- The host and port information of the various GF nodes are accessed through the 'node' element of domain.xml.
- The addresslist for each instance is created in such a way as to ensure that each GF instance uses its local MQ instance by default.
- In case of conventional clusters, the MQ instance associated with the first GF instance as configured in the domain.xml is designated as the master broker.
- An important enhancement in GF 3.1 is the support for auto-clustering of EMBEDDED brokers where the GF process and the broker process run on the same Java VM.
- Embedded now becomes the default mode for auto-clustering.
- While the Embedded mode for stand-alone (non-clustered) GF servers with MQ use a direct in-process communication with the MQ broker, clustered instances will need to use TCP. The communication mode will be selected automatically and will not need to be configured by the user. Direct mode communication uses API calls and completely bypasses the network stack. As a consequence, there is a significant speed-up. This cannot be used in the case of clustered instances since when running in a cluster these need to be able to handle broker or connection failure by failing over to another broker. For full details see the one-pager https://mq.dev.java.net/4.5-content/embeddedBrokersInConventionalClusters-one-pager.html
- The MQ broker lifecycles in both the EMBEDDED and LOCAL modes are managed by GF.
- The MQ changes to support Embedded broker in conventional (non-HA) clusters is covered in the following one-pager: https://mq.dev.java.net/4.5-content/embeddedBrokersInConventionalClusters-one-pager.html

4.1.2 Support for MQ clusters in REMOTE mode

GF will continue to support this feature by which it can connect to an existing, user-created MQ cluster without the auto-clustering and lifecycle management capabilities. The JMS Host information will need to be manually configured by the user.

4.1.3 Support for HA clusters

MQ's Enhanced Clusters (HA clusters) provides a peer-to-peer broker topology with a shared persistent data store offering data-availability. This mode does away with the master broker requirement. In v2.1, this mode was only supported in the 'Enterprise profile' with HADB. MQ would share the HADB data source that is configured under availability-services of domain.xml with GF.

GF 3.1 no longer bundles HADB and the asadmin command (configure-ha-cluster) to configure a HA service has been discontinued. Hence, the user will need to manually configure a HA database. The corresponding HA data source information will need to be configured manually in MQ's cluster.properties file.

The default support will continue to be for non-HA clusters. Auto-clustering of HA clusters will be supported but only in the LOCAL mode.

There will be no support for MQ in EMBEDDED mode for HA clusters due to limitations around restarting of the MQ broker (required in certain failure scenarios) without stopping GF when they share the same JVM.

REMOTE mode will be supported for HA clusters but without auto-clustering and life-cycle management capabilities.

4.1.4 Dependence on Master broker

Non-HA clusters in MQ have traditionally required configuring a master broker for certain admin related operations like create/update/delete of durable subscription and physical destination. MQ broker instances also require to rendezvous with the master broker at start-up. This imposes the requirement for the master broker to be started before the remaining broker instances can start and function correctly. They have been several complaints from users running into "master broker not started" errors if there are delays in the start-up of the master broker. To address issue, the following changes are proposed.

MQ is proposing to introduce a new broker property 'imq.cluster. nowaitForMasterBrokerTimeoutInSeconds that can be configured through GF (as a property in the jms-host element of domain.xml) and passed on to the MQ broker through the RA at start-up. This property will define the timeout interval before the exceptions are reported. The MQ changes are documented here -
https://mq.dev.java.net/4.5-content/cluster-improvemaster.txt.

MQ 4.5 also proposes to provide an option to entirely switch off the need for a master broker and instead use a database. The one pager for this feature is available here –
https://mq.dev.java.net/4.5-content/cluster-nomaster-db.txt
However, by default, GF will continue to configure a master broker for non-HA clusters. As in v2.1, the first broker in the server-list will be designated as the master broker. To configure the DB data source, the user will need to manually configure the properties in MQ's cluster.properties file.

4.1.5 Support for Dynamic Cluster changes

In v2.1, the MQ broker address-list was populated only during start-up. As a consequence, any changes to the cluster at run-time were not reflected. Changes to the cluster required a restart of the entire cluster. As an enhancement in v3.1, it is proposed to support dynamic changes in cluster topologies. GF JMS module will now listen for cluster change events. These changes will be propagated to the MQ broker instance through the RA. Every MQ broker instance in the cluster will receive these notifications.

The master broker or DB need not be running when these dynamic cluster change requests come-in. However, when running with the master broker option, any changes to master broker (except changing the port-mapper port) will result in an exception. The master broker can only be deleted and replaced with a new master broker by following the MQ backup/restore procedures documented in the MQ admin guide -
http://docs.sun.com/app/docs/doc/821-0027/aeoih?l=en&a=view. This requires a shutdown of all the MQ brokers instances in the cluster. Then, the master broker change records need to be backed-up and restored to the new master broker. The imq.cluster.masterbroker property for all the broker instances needs to be updated before restarting the cluster.

MQ changes to support this feature are covered in the following one-pager:
https://mq.dev.java.net/4.5-content/sync-brokerlist-in-glassfish.txt

4.1.6 Integrating MQ monitoring statistics with GF monitoring framework

The MQ broker provides a rich set of monitoring statistics that are currently not accessible through GF. A majority of these stats are implemented as JMX MBeans while a smaller number of them are only accessible through the MQ command line. In this release, we plan to provide access to these MQ monitoring statistics through the GF monitoring framework. The GF JMS module will implement the StatsProvider interface and will act as a proxy for these JMX MBeans. The primary focus will be on providing accessibility to the JMX enabled stats. Access to the non-JMX metrics will require code changes from MQ by either making these available through JMX or providing an alternative interface.

This approach offers the following advantages:
- Leveraging existing metrics without any code changes to MQ
- No requirement for MQ to be repackaged as an OSGi module
- Since this approach is based on JMX, this approach will work in all three integration modes – Embedded, Local and Remote.

4.1.7 JMS support for Embedded GF

In this release, we plan to introduce JMS support for Embedded GF. EMBEDDED and REMOTE modes of MQ integration will be supported. However, local mode of integration will not be supported. The JMSRA and MQ jars need to be bundled with glassfish-embedded-all.jar in order to support EMBEDDED mode.

4.2. Bug/RFE Number(s)

There are no existing bugs or RFEs for these proposed changes.

4.3. In Scope

This document covers only the proposed GF JMS module related changes.

4.4. Out of Scope

The MQ related changes are covered in other one-pagers. Please see the table above for details and links.

4.5. Interfaces

4.5.1 Public Interfaces

This work does not modify or introduce new public interfaces.

Interface	Comments

4.5.2 Private interfaces

This work does not modify or introduce new private interfaces.

4.5.3 Deprecated/Removed Interfaces

This work does not deprecate or remove any existing public interfaces.

4.6. Doc Impact

Application Server Developer's Guide, Administration Guide and HA guide. There will be no changes to the core JMS functionality but the guides will need to document the propos

4.7. Admin/Config Impact

The JMS related admin command will need to change to support clusters. The following CLI commands will be impacted:

create-jms-host
delete-jms-host
list-jms-hosts
create-jms-resource
delete-jms-resource
list-jms-resources
create-jmsdest
delete-jmsdest

list-jmsdest
flush-jmsdest
jms-ping

4.8. HA Impact

Discussed in section 4.1.5

4.9. I18N/L10N Impacts

None.

4.10. Packaging, Delivery & Upgrade

4.10.1 Packaging

JMS support for Embedded GF introduces the need for the MQ binaries to be included in the Embedded GF jar. Hence, jmara.rar will need to be added to glassfish-embedded-all.jar.

4.10.2 Delivery

No installer changes proposed.

4.10.3 Upgrade and Migration

It should be possible to upgrade an existing GF 2.1 cluster installation to GF 3.1. The important assumptions here are:

- An existing installation of MQ clusters in LOCAL mode will be upgraded to LOCAL mode only
- Non-HA clusters with a master broker will be upgraded to a cluster with a master broker.

4.11. Security Impact

No security impact

4.12. Compatibility Impact

It is proposed to change the default mode for non-HA clusters from LOCAL to EMBEDDED.

4.13. Dependencies

4.13.1 Internal Dependencies

The work described in this document depends on the GF clustering infrastructure, the related admin changes and changes in MQ 4.5 as described in the MQ 4.5 one-pagers.

4.13.2 External Dependencies

No external dependencies.

4.14. Testing Impact

Existing system tests should suffice for testing non-HA and HA clusters. However, new tests are required to cover the enhancements to the cluster architecture as described above.

5. Reference Documents

The list of MQ one-pagers listed on the table above.

6. Schedule

6.1. Projected Availability

The detailed milestone schedule is available here.