012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
1. Introduction
    1.1. Project/Component Working Name:
         Group Management Service

    1.2. Name(s) and e-mail address of Document Author(s)/Supplier:
         Shreedhar Ganapathy
         shreed@sun.com

    1.3. Date of This Document:
         10/26/2006

2. Project Summary
    2.1. Project Description:
        GMS was previously approved under case number:
        WSARC/2005/619/

        Refer to http://sac.sfbay.sun.com/Archives/CaseLog/arc/WSARC/2005/619/

        This proposal presents GMS as an open source project using
        Jxta Peer to Peer technology as a group communication and membership
        library.

        Client Facing APIs of GMS remain the same as in the previous submission.
        Changes in GMS are internal to allow for pluggability of group
        communication providers.

        For context, project description of GMS is provided below.

        Group Management Service (GMS) is an independent software
        module, from Project Shoal (https://shoal.dev.java.net), that
        may be embedded and started by processes that require
        runtime cluster communications and group management services
        such as :
            Static Group Membership Composition change notifications :
                   MemberAddedNotification
                   MemberRemovedNotification
            Dynamic Group Membership Composition change notifications :
                    Join Notifications
                    Failure Suspicion Notifications
                    Failure Notifications
                    Planned Shutdown Notifications
            Recovery Oriented Support Services:
                    Delegate Recovery Instance Selection and notification
                    Protecting recovery operations through failure fencing
            Messaging Service API for Group and Member-to-Member messaging
            Distributed Caching of lightweight state and recovery states

        GMS is an in-process component that can be accessed by other components
        within the process to receive events occuring in a group of
        distributed processes.

    2.2. Risks and Assumptions:


3. Problem Summary
    3.1. Problem Area:
         A number of problems are addressed by GMS as described below:

Cluster membership and state during the runtime lifecycle of the application server (or any other product) is required for administrators and other components in order to react to cluster group events and take appropriate steps to remedy negative group events.

Discovery of appserver entities such as the domain admin server, the node agent, and clustered instances happens today through preconfigured locational documents that are then used to execute a hand shake and then perform communications. If one or more entities move their location, such a move has to be administratively documented in other entities' configuration. If a solution can provide dynamic discovery without having to preconfigure locational details of each entity, this saves significant hours of code maintenance while reducing the chances for serious issues in those areas.

Many value added features such as automated transaction recovery, in-memory state replication, IIOP load balancer, read-mostly enterprise bean's cache change detection, self management, etc. require knowledge of the presence and/or absence of cluster members in enterprise environments.

GMS provides a solution for these problems by providing a uniform infrastructure for cluster membership discovery and state management, recovery oriented support and group messaging. This backbone enables application server components to provide value-added features/components in the areas of In-memory state replication, delegated transaction recovery, cluster health monitoring and domain management.

3.2. Justification:
Common reliable backbone service for value added features and competitive product differentiation.

4. Technical Description:

4.1. Details:
The project is in advanced state of completion and sources for the same have been released under the open source project Shoal at java.net. (https://shoal.dev.java.net)

GMS provides a simple, easy-to-use API to its clients for accessing and consuming its functionalities. GMS provides a Group Communication Service Provider Interface for group communications provider technologies to be integrated. In our implementation, we use a Service Provider implementation based on JXTA peer-to-peer technology to construct the desired group communications infrastructure.

The functionalities of GMS are primarily aimed at providing an infrastructure to monitor the availability state of a group of processes so that surviving members in a group can be notified of other member failures. GMS also provides members the ability to send and receive messages to and from other group members. Such functionalities augment capabilites of a process group to better manage information and state that are distributed among various group members. At this time, we are not offering high throughput messaging capabilities but supporting lightweight messaging although the underlying infrastructure would fully support a

high throughput requirement.

Examples of GMS clients in the application server include the Timer Service, the Transaction Service, the EJB Container for Read-Only or Read-Mostly beans' cache update notifications, the IIOP Failover Loadbalancer, the In-Memory replication module and the instance that serves as the administration server for reporting cluster health.

GMS provides the following features for release 9.1 timeframe:

a. Failure Notifications
b. Recovery member selection and corresponding notifications
c. Failure Fencing
d. Member Joins and Planned Shutdown Notifications
e. Support for administrative configurations
f. Group, One-to-Many and One-To-One Messaging
g. a Distributed State Cache implementation to to store data in a shared cache that lives in each instance's GMS module.

Please refer to the following document for detailed discussion on these features:
https://shoal.dev.java.net/ShoalOverview.html

For the 9.1 release of the appserver, the GMS component is a binary dependency in that a shoal-gms.jar and jxta.jar is placed in installation's lib directory at the time of installation.

4.2. Bug/RFE Number(s):

4.3. In Scope:

4.4. Out of Scope:

In this release, we have not addressed consequences of a network split that divides a clustered network into a sub groups. We have placed some checks and balances through masternode collision detection, view histories, view incarnation numbers in order to provide for group coalescence after a network split has healed.

4.5. Interfaces:

4.5.1 Exported Interfaces

GMS interfaces have not changed since last submission. These are project private although the jar is packaged in the lib directory and hence available to applications. We do not for this release, support applications that directly use GMS functionality.

4.5.2 Imported interfaces

The GMS API and its implementation project does not directly import any external third party interfaces but relies on a Service Provider Interface to interact with the implementation.

For the 9.1 release of the application server, the default service provider is based on JXTA platform

| Interface | Stability | Exporting Project | Comments |
|---|---|---|---|
| JXTA Technology<br>net.jxta.endpoint<br>net.jxta.peergroup<br>net.jxta.pipe<br>net.jxta.protocol<br>net.jxta.util | Volatile | JXTA 2.5 | Integrated as jxta.jar |

4.5.3 Other interfaces (Optional)

4.6. Doc Impact:
Appserver Documentation should cover how to configure GMS to be turned on, how to set various configuration options for GMS and its log level.

We can also include documentation specific to GMS if need be but since GMS is not exposed to end user applications, this may not be necessary.

4.7. Admin/Config Impact:
Admin GUI screen provides a Group Management Service configuration page (this is already in place)

Admin CLI continues to take advantage of the set, get and list commands for managing attribute values for the group management service elements.

4.8. HA Impact:
The In-memory replication component is dependent on GMS being turned on. The in-memory replication component relies on GMS to provide it with information regarding members being added to the cluster, members joining the cluster at runtime, members suspected to have failed, members confirmed to have failed, members shutting down as a result of an administrative actions to shutdown instances or entire cluster. Additionally, the in-memory component calls an API provided by GMS to query the current state of any cluster member.

4.9. I18N/L10N Impact:
Localization of GMS log messages will be required.

4.10. Packaging & Delivery:
shoal-gms.jar and jxta.jar will be included in the appserver packages and installer will install these libraries in the installation's lib directory.

4.11. Security Impact:
JXTA provides for pluggable keystores so that the appserver keystores can be plugged in to encrypt all communications between peers.
In the case of use of multicast, Jxta provides for shared key encryption. In the case of TCP/Http based point-to-point communications, JXTA provides for PKI based TLS support.
These are achieved through simply passing in the keystore location and cert nickname to the Jxta layer.

For release 9.1, GMS will use the above security features for encryption in both the TCP and multicast transports.

4.12. Compatibility Impact

This is a brand new feature and hence there are compatibility concerns.

4.13. Dependencies:
Various Application server components rely on GMS's availability in each cluster member and DAS to be able to use its functionality for their reliability infrastructure. These components are :

1. The Timer Service for timer migrations
2. The Ejb Container for read-only cache update notifications
3. The IIOP Failover LoadBalancer for dynamic cluster shape change notifications
4. The in-memory replication component for dynamic cluster shape change notifications and current member and group state.
5. The DAS for cluster health functionality provided to both the Admin GUI and CLI's get-health command
6. The transaction service for notifying cluster shape and for automated delegated transaction recovery feature. GMS picks a surviving member to perform transaction recovery automatically when another member has failed.
7. The self-management service for notifications related to members joining, leaving, and failing.

GMS depends on JXTA Technology for inter-process messaging.

5. Reference Documents:
The following documents are available:
1. Shoal Overview : https://shoal.dev.java.net/ShoalOverview.html contains an overall explanation of the project's goals.
2. Shoal Design Document: https://shoal.dev.java.net/ShoalDesignDocument.html explains the design specification for GMS
3. Shoal Group Event Notifications are covered in https://shoal.dev.java.net/ShoalGroupEventNotifications.html
4. Shoal Automated Delegated Recovery Initiation and failure fencing is covered in https://shoal.dev.java.net/ShoalAutomatedDelegatedRecoveryInitiation.html
5. Shoal's Messaging APIs are covered under https://shoal.dev.java.net/ShoalMessaging.html

6. Schedule:
6.1. Projected Availability:
Shoal has been integrated into the appserver codebase and is ready for Quality Engineering's testing program.