OpenClovis
Software Development Kit (SDK)
Service Description and API Reference for Group Management Service (GMS)

For OpenClovis SDK Release 2.3 V2.0

Document Revision Date: February 22, 2007

Contents

Chapter 1

Functional Overview

The **OpenClovis Group Membership Service (GMS)** is a high availability infrastructure component that allows a set of nodes to form a group and provide track notifications to registered applications. GMS is implemented in compliance with the Cluster Membership Service (CLM) as defined by SA Forum. It is the generalized form of CLM, where each member of a group is a process in a cluster interested in becoming a member of that group.

Apart from the services defined by SA Forum CLM specification, GMS also provides following services:

- Leader election among a group of nodes.
- Formation of process groups and their management along with track notification.

In this document, the Process Group functionality of GMS is indicated as "Process Group Service", and the SAF CLM implementation is indicated as "CLM Service". The following sections describe the functionalities, the usage scenarios, and the various service interfaces provided for each of these services.

Chapter 2

Service Model

2.1 Usage Model

TBD

2.2 Functional Description

1. **CLM Service**
   CLM Service manages the membership of the cluster with specified cluster configuration, for the nodes that are administratively configured to be part of it. It provides a consistent view of the cluster across all nodes of the cluster.

   A node must be member of the cluster to host high availability applications on it. CLM Service allows a node to be a member of a cluster only if it the node is healthy and is well-connected to the cluster.

   When ASP is started, Component Manager (CPM) sends a request to GMS for the membership of the node for the cluster. GMS processes the request and decides upon the membership of the node. It also elects a leader and a deputy node for the cluster based on the leadership credentials and the boot timestamp of the member nodes. If the process is executed successfully, GMS provides the current view of the cluster along with the leader and deputy information to CPM.

   An application can register for track notifications with GMS using the cluster track APIs. After registration, GMS notifies the application for any changes in the cluster membership such as when a node joins, leaves the cluster or a node is reconfigured. The application continues to receive such notifications until it invokes the track stop API.

   GMS provides the following functions for the clustering service:

   - Allows a node to join a cluster using clGmsClusterJoin API. item Allows a member node to leave the cluster using clGmsClusterLeave API.
   - Manages cluster membership of the nodes based on their health and communication with the cluster.

- Allows you to track a cluster by providing consistent view of the cluster across nodes and notifies if any changes are made using clGmsClusterTrack API.
- Allows you to stop receiving track notifications using clGmsClusterTrackStop API.
- Elects a leader and a deputy node automatically for a cluster whenever the cluster configuration is modified.
- Provides information about a member of a cluster using clGmsClusterMemberGet and clGmsClusterMemberGetAsync APIs.
- Initiates an explicit leader election process on the cluster nodes using clGmsClusterLeaderElect API.
- Removes a member from the cluster using clGmsClusterMemberEject.

Note:
In the ASP framework, joining or leaving a cluster is generally managed by CPM. The applications do not use these APIs as it can affect the other functionalities of ASP.

2. **Process Group Service**
GMS generalizes the CLM Service by allowing any set of applications or processes in the cluster to form a group and manage such groups. This service, referred to as the Process Group Service, also allows you to track the group by providing a consistent view of the group and notifies about the changes in the membership of the group.

The Process Group Service provides the following functionalities:

- Creates a group using clGmsGroupCreate API.
- Deletes or destroys a group using clGmsGroupDestroyAPI.
- Allows a process in the cluster to join a group using clGmsGroupJoin API.
- Allows a member to leave the group using clGmsGroupLeave API.
- Manages membership of the group based on the health of the member processes and the containing node.
- Allows you to track the membership by providing a consistent view of the group and notifies if any changes are made using clGmsGroupTrack API.
- Allows you to stop receiving track notifications for a given group using clGmsGroupTrackStop API.
- Provides a list of meta information on all the groups at any given time using clGmsGroupInfoListGet API.
- Provides the meta information on any given group using clGmsGetGroupInfo API.

Usage Scenario:

1. **CLM Service**
The main objective of CLM service is to provide high availability of services in co-ordination with AMF and CPM. These ASP services register with GMS for track notification on the cluster. So at any given time, AMF running on each node is aware of the leader (or master System Controller node) in the cluster.

If the master System Controller node is terminated, GMS instance informs all the nodes and applications through the track notifications. Based on this information, AMF makes the standby System Controller as Active. The other applications like Checkpointing Server can take necessary actions to move their Checkpoint Master Server on the newly active System Controller node.

2. **Process Group Service**
   The main objective of GMS is to allow a set of processes, implementing a distributed component, service, or application to form a group in order to share data or to co-ordinate access to shared resources. Generally these processes are multiple instances of the same code running on different nodes.

   For example, a Name Service (NS) implementation will depend on a local NS daemon running on every node in a system. An NS entry created by an application on a node is registered with the local NS daemon. The other nodes can view this entry by allowing the NS daemons to form a group and using group communication to disseminate information among them.

   This can be achieved in two possible ways. Both the solutions require that the current group view is available to all members.
   - By using a leader that is a dedicated member and holds the primary repository of the NS database, or
   - In a distributed fashion where each member owns its local data and multicasts the changes in the data to other members of the group.

Interaction with other components:

GMS depends on CPM for component management and uses RMD infrastructure between the GMS client and the server instances. Also it uses the event service to find out if any of the components that are member of the group goes down. This component death event is given by CMP service, and GMS would remove this member from all the groups of which it was member.

Configuration:

GMS provides a set of the configurable parameters. You can configure the values of these parameters through the `gmsconfig.xml` file located in the `$SISP_CONFIG` directory.

1. ***Clustername***: Name of the cluster being formed.
2. ***linkname***: Name of the IP network interface of the machine where GMS instance will bind.
3. ***MulticastAddress***: Multicast IP address where GMS Server binds and exchanges its information (such as handshake messages with other GMS instances in the cluster).
4. ***MulticastPort***: Port number to be used to bind for the multicast socket. This socket is used along with the MulticastAddress to exchange internal information with other GMS instances in the cluster.
5. ***MaxNoOfGroups***: Maximum number of process groups allowed to be formed in a cluster.
6. ***consolelog***: Specifies whether the log messages from GMS are printed on the console or not. Its values can be either *on* or *off*.
7. ***logLevel***: Controls the log messages printed in the log file. It can be debug, info, error, and so on.

Note:

The values of all the parameters must be identical across all nodes in the cluster, saving linkname.

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClGmsClusterManageCallbacksT

```
typedef struct {
          ClGmsClusterMemberEjectCallbackT clGmsMemberEjectCallback;
} ClGmsClusterManageCallbacksT;
```

ClGmsClusterManageCallbacksT This structure contains the cluster managing callbacks provided at the joining time by the member. The structure contains the ejection callback which is called when the member is ejected from the cluster. The callback is invoked after the member is ejected and the reason for ejection is passed as an argument to the callback.

clGmsMemberEjectCallback is a pointer to the Eject Callback function.

3.1.2 ClGmsGroupMemberT

```
typedef struct {
          ClGmsMemberIdT memberId ;
          ClIocAddressT memberAddress;
          ClGmsMemberNameT memberName ;
          ClBoolT memberActive ;
          ClTimeT joinTimestamp ;
          ClUint64T initialViewNumber ;
          ClGmsLeadershipCredentialsT credential ;
} ClGmsGroupMemberT;
```

The structure *ClGmsGroupMemberT* contains the member component descriptor. Its attributes have the following interpretation:

- *memberId* - Unique ID of a member of a given group.
- *memberAddress* - IOC physical address of the member.
- *memberName* - Textual name of the member.

- *memberActive* - True if the node is a member of group.
- *joinTimestamp* - The instant at which the member joined the group.
- *initialViewNumber* - The view of the cluster at the time the member joined.
- *credential* - Credentials for being the leader. The higher the credential, larger is the possibility of the node being elected as leader.

3.1.3 ClGmsLeadershipCredentialsT

typedef ClUint32T ClGmsLeadershipCredentialsT;

The type of an identifier for the credentials of leader election. Only members with non-zero value in the group are considered as candidates for leadership.

3.1.4 ClGmsClusterMemberEjectCallbackT

*typedef void (*ClGmsClusterMemberEjectCallbackT) (*
CL_IN ClGmsMemberEjectReasonT reasonCode);

The type of the callback function to indicate that a member has been expelled from the cluster. This functions takes *reasonCode* as the parameter and returns void. This type definition is later used to define member eject callback structure parameters.

- *reasonCode* - It can have the following two values:
  1. CL_GMS_MEMBER_EJECT_REASON_UNKNOWN = 0
  2. CL_GMS_MEMBER_EJECT_REASON_API_REQUEST =1

3.1.5 ClGmsClusterManageCallbacksT

```
typedef struct ClLogFileFullActionT{
        ClGmsClusterMemberEjectCallbackT clGmsMemberEjectCallback;
} ClGmsClusterManageCallbacksT;
```

The structure *ClGmsClusterManageCallbacksT* contains the cluster managing callbacks provided by the member when it joins the cluster. The structure contains the ejection callback function which is invoked when the member is ejected from the cluster. The reason for ejection is passed as argument for the callback.

- *clGmsMemberEjectCallback* - Pointer to the Eject Callback function.

3.1.6 ClGmsMemberIdT

typedef ClUint32T ClGmsMemberIdT;

The type of an identifier for the group-unique ID of a member.

3.1.7 ClGmsGroupIdT

typedef ClUint32T ClGmsGroupIdT;

ClGmsGroupIdT is a system-wide unique ID of the group. It is generated by GMS and you can use this ID for performing further operations on the group.

3.1.8 ClGmsClusterMemberT

```
typedef struct {
        ClGmsNodeIdT nodeId ;
        ClIocAddressT nodeAddress ;
        ClGmsNodeAddressT nodeIpAddress ;
        ClNameT nodeName ;
        ClBoolT memberActive ;
        ClTimeT bootTimestamp ;
        ClUint64T initialViewNumber ;
        ClGmsLeadershipCredentialsT credential ;
        ClVersionT gmsVersion;
} ClGmsClusterMemberT;
```

This structure *ClGmsClusterMemberT* describes one member (or node) of the cluster. Its attributes have the following interpretation:

- *nodeId* - Unique ID of node.
- *nodeAddress* - Physical address of node.
- *nodeIpAddress* - Node IP Address.
- *nodeName* - Textual name of node.
- *memberActive* - This is ˛TRUE if the node is a member of the cluster For tracking nodes it is not set.
- *bootTimestamp* - The time at which GMS was started on the node.
- *initialViewNumber* - The view number when the node joined.
- *credential* - This is an integer value specifying the leadership credibility of the node. Larger the value higher is the possibility of the node becoming a leader. Member with credentials CL_GMS_INELIGIBLE_CREDENTIALS cannot participate in the leader election
- *gmsVersion* - Version information of the GMS software running on the node, information is sent to the other peers in the cluster while joining the cluster. If there is a version mismatch the node is not allowed to join the Cluster.

3.1.9 ClGmsGroupInfo

```
typedef struct ClGmsGroupInfo {
        ClGmsGroupNameT groupName;
        ClGmsGroupIdT groupId;
        ClGmsGroupParamsT groupParams;
        ClUint32T noOfMembers;
```

```
            ClBoolT setForDelete;
            ClIocMulticastAddressT iocMulticastAddr;
            ClTimeT creationTimestamp;
            ClTimeT lastChangeTimestamp;
} ClGmsGroupInfoT;
```

The structure *ClGmsGroupInfoT* contains the values of a group. Its attributes are:

- *groupName* - Name of the group.
- *groupId* - Group ID.
- *groupParams* - Requested group parameters.
- *noOfMembers* - Number of members in the group.
- *setForDelete* - No more joins are allowed.
- *iocMulticastAddr* - IOC multicast address created by GMS.
- *creationTimestamp* - Time at which group was created.
- *lastChangeTimestamp* - Time at which the last view changed.

3.1.10 clGmsGroupInfoList

```
typedef struct clGmsGroupInfoList {
            ClUint32T noOfGroups;
            ClGmsGroupInfoT *groupInfoList;
} ClGmsGroupInfoListT;
```

The structure *ClGmsGroupInfoT* contains the information on all the existing groups. Its attributes are:

- *noOfGroups* - Number of groups.
- *groupInfoList* - Array of *ClGmsGroupT* data.

3.1.11 ClGmsHandleT

typedef ClHandleT ClGmsHandleT;

The type of the handle for the GMS API. This handle is assigned during the initialization of the Group Membership Service. It must be passed as first parameter for all operations pertaining to the GMS library.

3.1.12 ClGmsNodeIdT

typedef ClUint32T ClGmsNodeIdT;

The type of a unique and consistent identifier for a Node - Node ID.

3.1.13 ClGmsCallbacksT

```
typedef struct {
        ClGmsClusterMemberGetCallbackT clGmsClusterMemberGetCallback;
        ClGmsClusterTrackCallbackT clGmsClusterTrackCallback;
        ClGmsGroupTrackCallbackT clGmsGroupTrackCallback;
        ClGmsGroupMemberGetCallbackT clGmsGroupMemberGetCallback;
} ClGmsCallbacksT;
```

The type of the callback structure provided to the GMS library during initialization. Its attributes have the following interpretation:

- *clGmsClusterMemberGetCallback* - This callback is called when the response is received from the server side for the asynchronous request made by the *clGmsClusterMemberGetAsync()*. The callback is invoked with the invocation ID and the requested information of the member.
- *clGmsClusterTrackCallback* - This callback is used to register for receiving any change notifications in the cluster. This registration with the server is performed by the clGmsClusterTrack function. The *trackFlags* should be *CL_GMS_TRACK_CHANGES or CL_GMS_TRACK_CHANGES_ONLY*. The callback is invoked when there is a change in the cluster and with the notification buffer containing the information of all members in the cluster. The callback is also called when clGmsClusterTrack API is called with *C*L_TRACK_CURRENT flag and notificationBuffer parameter is NULL.
- *clGmsGroupTrackCallback* - This callback is used to register for receiving any change notifications in the This registration with the server is performed by the clGmsGroupTrack function. The *trackFlags* should be *CL_GMS_TRACK_CHANGES or CL_GMS_TRACK_CHANGES_ONLY*. The callback is invoked when there is a change in the group and with the notification buffer containing the information of all members in the group. The callback is also called when clGmsClusterTrack API is called with *C*L_TRACK_CURRENT flag and notificationBuffer parameter is NULL.
- *clGmsGroupMemberGetCallback* - This callback is invoked when the response is received from the server side for the asynchronous request made by *clGmsGroupMemberGetAsync()*. The callback is invoked with the invocation ID and the requested information of the member.

3.1.14 ClGmsTrackFlagsT

```
typedef enum ClGmsTrackFlags {
CL_GMS_TRACK_CURRENT = 0x01,
CL_GMS_TRACK_CHANGES = 0x02,
CL_GMS_TRACK_CHANGES_ONLY = 0x04
} ClGmsTrackFlagsT;
```

The ClGmsTrackFlagsT enumeration type contains flags for tracking request flag.

- *CL_GMS_TRACK_CURRENT* - Returns current view.
- *CL_GMS_TRACK_CHANGES* - To subscribe for complete view notifications.
- *CL_GMS_TRACK_CHANGES_ONLY* - To subscribe for delta notifications.

3.1.15 ClGmsClusterNotificationBufferT

```
typedef struct {
        ClUint64T viewNumber;
        ClUint32T numberOfItems;
        ClGmsClusterNotificationT *notification;
        ClGmsNodeIdT leader;
        ClGmsNodeIdT deputy;
        ClBoolT leadershipChanged;
} ClGmsClusterNotificationBufferT;
```

The structure *ClGmsClusterNotificationBufferT* contains a buffer to communicate the view. The view is the list of nodes and their status. deputy Node marked as deputy. Its attributes have the following interpretation:

- *viewNumber* - Current view number.
- *numberOfItems* - Length of the notification array.
- *notification* - Array of nodes.
- *leader* - Node ID of current leader.
- *deputy* - Node marked as deputy.
- *leadershipChanged* - Check if the leader has changed since the last view.

3.1.16 ClGmsGroupInfoT

```
typedef struct ClGmsGroupInfo{
        ClGmsGroupNameT groupName;
        ClGmsGroupIdT groupId;
        ClGmsGroupParamsT groupParams;
        ClUint32T noOfMembers;
        ClBoolT setForDelete;
        ClIocMulticastAddressT iocMulticastAddr;
        ClTimeT creationTimestamp;
        ClTimeT lastChangeTimestamp;
} ClGmsGroupInfoT;
```

The structure *ClGmsGroupInfoT* contains the values of a group. Its attributes have the following interpretation:

- *groupName* - Name of the group.
- *groupId* - Id of the group.
- *groupParams* - Parameters of the group.
- *noOfMembers* - Number of members in the group.
- *setForDelete* - NO more joins are allowed.
- *iocMulticastAddr* - IOC multicast address created by GMS.
- *creationTimestamp* - Time when the group was created.
- *lastChangeTimestamp* - Time when the last view changed.

3.1.17 ClGmsGroupInfoListT

```
typedef struct clGmsGroupInfoList{
            ClUint32T noOfGroups;
            ClGmsGroupInfoT *groupInfoList;
} ClGmsGroupInfoListT;
```

The structure *ClGmsGroupInfoListT* is used to return the information on all the existing groups. Its attributes have the following interpretation:

- *noOfGroups* - Number of existing groups.
- *groupInfoList* - Array of ClGmsGroupT data.

3.2 Library Life Cycle APIs

3.2.1 clGmsInitialize

clGmsInitialize

Synopsis:
Initializes the GMS library and registers the callback functions.

Header File:
clGmsViewApi.h

Syntax:

```
        ClRcT clGmsInitialize(
                               CL_OUT   ClGmsHandleT           *gmsHandle,
                               CL_IN    const ClGmsCallbacksT *gmsCallbacks,
                               CL_INOUT ClVersionT             *version);
```

Parameters:

gmsHandle: (in/out) GMS service handle created by the library. This is used in subsequent use of the library in this session.

gmsCallbacks: (in) This is an optional parameter. This is a pointer to the array of callback functions you can provide. If *gmsCallbacks* is NULL, the callback is not registered. If *gmsCallbacks* is not NULL, it acts a pointer to clGmsCallbacksT structure, containing the callback functions of the process that the Group Membership Service invokes. Only non-NULL callback functions in this structure are registered. If any callback is NULL, the corresponding asynchronous operation returns error.

version: (in/out) It can have the following values:

- On input, this is the version required by you.
- On return, the library returns the version it supports.

Return values:

CL_OK: The API executed successfully.

CL_ERR_NOT_INITIALIZED: If library was not initialized.

CL_ERR_NULL_POINTER: On passing a NULL pointer.

CL_ERR_VERSION_MISMATCH: If the requested version is not compatible with the library version supported by OpenClovis ASP.

CL_ERR_NO_RESOURCE: If an instance or a new handle cannot be created.

Description:
This function initializes the Group Membership Service (GMS) for the invoking process, registers the various callback functions and negotiates the version of GMS library. This function must be invoked prior to the invocation of any other Group Membership Service functionality. The handle *gmsHandle* is returned as the reference to this association between the process and the Group Membership Service. The process uses this handle in subsequent communication with the GMS.

Library File:
ClGms

Related Function(s):
clGmsFinalize

3.2.2 clGmsFinalize

clGmsFinalize

Synopsis:

Finalizes the handle associated with a prior initialization of the GMS client library.

Header File:

clGmsViewApi.h

Syntax:

```
            ClRcT clGmsFinalize(
                                   CL_IN ClGmsHandleT gmsHandle);
```

Parameters:

gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

Description:

The *clGmsFinalize()* function closes the association, represented by the gmsHandle parameter, between the invoking process and the Group Membership Service. A process must invoke this function once for each handle it acquires by invoking *clGmsInitialize()*.
If the clGmsFinalize() function executes successfully, it releases all the resources acquired when clGmsInitialize() was invoked. It stops any tracking associated with the handle and cancels all pending callbacks related to the handle. As the callback invocation is asynchronous, some callbacks are processed after this call returns successfully. After *clGmsFinalize()* is invoked, the selection object is no longer valid.

Note:

On successful execution of this function, it releases all the resources allocated during the initialization of the library.

Library File:

ClGms

Related Function(s):

clGmsInitialize

3.3 Functional APIs

3.3.1 clGmsClusterJoin

clGmsClusterJoin

Synopsis:

Allows a node to join a cluster as a member.

Header File:

clGmsClusterManageApi.h

Syntax:

```
ClRcT clGmsClusterJoin(
                        CL_IN ClGmsHandleT                         gmsHandle,
                        CL_IN const ClGmsClusterManageCallbacksT *clusterManageCallbacks,
                        CL_IN ClGmsLeadershipCredentialsT          credentials,
                        CL_IN ClTimeT                              timeout,
                        CL_IN ClGmsNodeIdT                         nodeId,
                        CL_IN ClNameT                             *nodeName);
```

Parameters:

gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service.

clusterManageCallbacks: (in) Callbacks for managing the cluster.

credentials: (in) This is an integer value specifying the leadership credibility of the node. Larger the value, higher is the possibility of the node becoming a leader. Member with credentials `CL_GMS_INELIGIBLE_CREDENTIALS` cannot participate in the leader election.

timeout: (in) If the cluster join is not completed within this time, then the join request is timed out.

nodeId: (in) Node ID of the member that will join the cluster.

nodeName: (in) Name of the node that will join the cluster.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: If the handle passed to the function is not valid. The handle passed should have been obtained from the *clGmsInitialize()* function.

CL_ERR_TIMEOUT: If the join request timed out.

CL_ERR_ALREADY_EXIST: If the node is already part of the cluster.

CL_ERR_INVALID_PARAMETER: If any of the input parameters are invalid.

CL_ERR_NULL_POINTER: If either of the parameters *clusterManageCallbacks* or *nodeName* are NULL.

CL_ERR_TRY_AGAIN: GMS server is not ready to process the request.

Description:

This function is used to include a node to the cluster as a member of the cluster. Success or failure is reported through the return value. Members who have registered for tracking, get notified by the tracking callback function.

Library File:

ClGms

Note:

This API is used internally by the CPM service. The user is not required to use this API.

Related Function(s):

clGmsClusterLeave

3.3.2 clGmsClusterLeave

clGmsClusterLeave

Synopsis:
Allows a node to leave a cluster.

Header File:
clGmsClusterManageApi.h

Syntax:

```
ClRcT clGmsClusterLeave(
                        CL_IN ClGmsHandleT                          gmsHandle,
                        CL_IN ClTimeT                               timeout,
                        CL_IN ClGmsNodeIdT                          nodeId);
```

Parameters:

gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service

timeout: (in) If the cluster leave operation is not completed within this time, then the leave request is timed out.

nodeId: (in) Node ID of the member that is leaving the cluster.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: If the handle passed to the function is not valid. The handle passed should have been obtained from the *clGmsInitialize()* function.

CL_ERR_INVALID_PARAMETER: If any of the input parameters are invalid.

Description:
This function can be used by a node to leave a cluster. After the node leaves the cluster and the groups/components are expelled, a reason for expulsion is returned through their callback functions.

Library File:
ClGms

Note:
This API is used internally by the CPM service. The user is not required to use this API.

Related Function(s):
clGmsClusterJoin

3.3.3 clGmsClusterLeaderElect

clGmsClusterLeaderElect

Synopsis:
Initiates leader election synchronously.

Header File:
clGmsClusterManageApi.h

Syntax:

```
ClRcT clGmsClusterLeaderElect(
                        CL_IN ClGmsHandleT     gmsHandle);
```

Parameters:
gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service

Return values:
CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: If the handle passed to the function is not valid. The handle passed should have been obtained from the *clGmsInitialize()* function.

Description:
This function is used to initiate leader election synchronously. The elected leader is announced through the tracking callback. This function is invoked when a node leaves or joins, or on any event which would alter the leadership of the cluster. The algorithm is then run by the GMS server engine, and a leader and deputy leader are elected.

Library File:
ClGms

Related Function(s):
None

3.3.4 clGmsClusterMemberEject

clGmsClusterMemberEject

Synopsis:
Forcibly removes a member from the cluster.

Header File:
clGmsClusterManageApi.h

Syntax:

```
ClRcT clGmsClusterMemberEject(
                        CL_IN ClGmsHandleT                  gmsHandle,
                        CL_IN ClGmsNodeIdT                  nodeId,
                        CL_IN ClGmsMemberEjectReasonT       reason);
```

Parameters:

gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service

nodeId: (in) Node ID of the member to be ejected out.

reason: (in) Reason for ejecting the member out of the cluster. A member can be ejected from the cluster either upon request, or for an unknown reason. *reasonCode* can have 2 values:

- CL_GMS_MEMBER_EJECT_REASON_UNKNOWN = 0
- CL_GMS_MEMBER_EJECT_REASON_API_REQUEST =1.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: If the handle passed to the function is not valid. The handle passed should have been obtained from the *clGmsInitialize()* function.

CL_ERR_INVALID_PARAMETER: If the node ID is not a valid node ID or if the reason is not valid.

Description:
This function is used to remove a member forcibly from the cluster. A reason is given when a member is removed. The tracking members of the cluster are notified through the tracking callback.

Library File:
ClGms

Related Function(s):
clGmsClusterLeaderElect

3.3.5 clGmsClusterMemberGet

clGmsClusterMemberGet

Synopsis:
Returns cluster member information.

Header File:
clGmsViewApi.h

Syntax:

```
ClRcT  clGmsClusterMemberGet(
                           CL_IN  ClGmsHandleT              gmsHandle,
                           CL_IN  ClGmsNodeIdT              nodeId,
                           CL_IN  ClTimeT                   timeout,
                           CL_OUT ClGmsClusterMemberT      *clusterMember);
```

Parameters:

gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service.

nodeId: (in) The identifier of the cluster node for which the *clusterNode* information structure is to be retrieved.

timeout: (in) The *clGmsClusterMemberGet()* invocation is considered to have failed if it does not get complete during the time specified through this parameter.

clusterMember: (out) A pointer to a cluster node structure that contains information about a cluster node. The invoking process provides space for this structure, and the Group Membership Service fills in the fields of this structure.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

CL_ERR_NULL_POINTER: If the parameter *clusterMember* passed is a NULL pointer.

CL_ERR_INVALID_PARAM: The requested node does not exist.

CL_ERR_TIMEOUT: Communication request timed out.

Description:
This API is used to retrieve the information about a given cluster member and check if the node is a member of the cluster. You should allocate the space for the node. The information about a cluster member is retrieved synchronously. The node is identified by *nodeId* parameter. The cluster node information is returned in the *clusterNode* parameter. By invoking this function, a process can obtain the cluster node information for the node, designated by *nodeId*, and can then check the member field to determine whether this node is a member of the cluster. If the constant CL_GMS_LOCAL_NODE_ID is used as *nodeId*, the function returns information about the cluster node that hosts the invoking process.

Library File:
ClGms

Related Function(s):
clGmsClusterTrack,clGmsClusterTrackStop, clGmsClusterMemberGetAsync

3.3.6 clGmsClusterMemberGetAsync

clGmsClusterMemberGetAsync

Synopsis:
Returns information on the cluster node asynchronously through *clGmsClusterMemberGetCallback()*.

Header File:
clGmsViewApi.h

Syntax:

```
ClRcT clGmsClusterMemberGetAsync(
                        CL_IN ClGmsHandleT              gmsHandle,
                        CL_IN ClInvocationT             invocation,
                        CL_IN ClGmsNodeIdT              nodeId);
```

Parameters:

gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service.

invocation: (in) Correlates the invocation with the corresponding callback. This parameter allows the invoking process to match this invocation of *clGmsClusterMemberGetAsync()* with the corresponding clGmsClusterMemberGetCallback().

nodeId: (in) The identifier of the cluster node for which the information is to be retrieved.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

CL_ERR_INVALID_PARAM: The requested node does not exist.

Description:
This API is used to query the information on a given cluster node. This function requests information about a particular cluster node, identified by the *nodeId* parameter. The information about a cluster is provided asynchronously.
If `CL_GMS_LOCAL_NODE_ID` is used as *nodeId*, the function returns information about the cluster node that hosts the invoking process.
The process matches the corresponding callback, clGmsClusterMemberGetCallback(), with this particular invocation. The clGmsClusterMemberGetCallback() callback function is provided when the process invokes the *clGmsInitialize()*.

Library File:
ClGms

Related Function(s):
clGmsClusterTrack, clGmsClusterTrackStop, clGmsClusterMemberGet

It makes an asynchronous call to *ClGmsClusterMemberGetCallbackT()*.

3.3.7 clGmsClusterTrack

clGmsClusterTrack

Synopsis:
Configures the cluster tracking mode.

Header File:
clGmsViewApi.h

Syntax:

```
        ClRcT clGmsClusterTrack(
                                CL_IN    ClGmsHandleT             gmsHandle,
                                CL_IN    ClUint8T                 trackFlags,
                                CL_INOUT ClGmsClusterNotificationBufferT *notificationBuffer);
```

Parameters:

gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service.

trackFlags: (in) Requested tracking mode.

notificationBuffer: (in/out) This is an optional parameter and is used when *trackFlags* is set to *CL_TRACK_CURRENT*. It provides a buffer that contains the current cluster information when the API returns successfully. If it is NULL, the information is returned through an invocation to *clGmsClusterTrackCallback()*.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

CL_ERR_INVALID_PARAMETER: If `CL_GMS_TRACK_CURRENT` flag is set and notification buffer is provided, but size of allocated array is not set (0).

CL_GMS_ERR_INVALID_TRACKFLAGS: If both `CHANGES` and `CHANGES_ONLY` flags are set.

CL_ERR_NO_CALLBACK: If request was asynchronous, but no callback was registered.

CL_ERR_TRY_AGAIN: Communication error, try again.

CL_ERR_TIMEOUT: Communication request timed out.

Description:
This API is used to configure the cluster tracking mode for the caller. It can be called subsequently to modify the requested tracking mode. This function is used to obtain the current cluster membership and request notification of changes in the cluster membership or of changes in an attribute of a cluster node, depending on the value of the *trackFlags* parameter.
These changes are notified through invocation of the clGmsClusterTrackCallback() callback function, which must have been supplied when the process invoked the *clGmsInitialize()* call. An application may call *clGmsClusterTrack()* repeatedly for the same values of *gmsHandle*, regardless of whether the call initiates a one-time status request or a series of callback notifications.

Library File:
ClGms

Related Function(s):
clGmsClusterTrackStop , clGmsClusterMemberGet, clGmsClusterMemberGetAsync

3.3.8 clGmsClusterTrackStop

clGmsClusterTrackStop

Synopsis:
Stops all the clusters tracking.

Header File:
clGmsViewApi.h

Syntax:

```
        ClRcT clGmsClusterTrackStop(
                                CL_IN ClGmsHandleT              gmsHandle);
```

Parameters:
gmsHandle: (in) The handle, obtained through the *clGmsInitialize()* function, designating this particular initialization of the Group Membership Service.

Return values:
CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

Description:
This API is used to immediately stop the tracking of all the clusters for a given client. This function stops any further notifications through the handle gmsHandle. Pending callbacks are removed. This is usually invoked during shut-down of the application.

Library File:
ClGms

Related Function(s):
clGmsClusterTrack,clGmsClusterMemberGet, clGmsClusterMemberGetAsync

3.3.9 clGmsGroupCreate

clGmsGroupCreate

Synopsis:
Creates a group.

Header File:
clGmsGroupManageApi.h

Syntax:

```
ClRcT clGmsGroupCreate(
                    CL_IN    ClGmsHandleT                      gmsHandle,
                    CL_IN    ClGmsGroupNameT                  *groupName,
                    CL_OUT   ClGmsGroupIdT                    *groupId,
                    CL_IN    const ClGmsGroupManageCallbacksT *groupManageCallbacks,
                    CL_INOUT ClGmsGroupParamsT                *groupParams);
```

Parameters:

gmsHandle: (in) gmsHandle provided during clGmsInitialize.

groupName: (in) Name of the group. Specify the value and length.

groupId: (out) Pointer to the memory to store groupId generated by GMS.

groupManageCallbacks: (in) It can be NULL as currently no functionality is provided.

groupParams: (in/out)Includes parameters such as isIocGroup etc. By default all the groups are IOC Groups.

Return values:

CL_OK: The API executed successfully.

CL_ERR_TRY_AGAIN: Server is not ready to serve group functionality.

CL_ERR_VERSION_MISMATCH: Client version not supported.

CL_ERR_INVALID_HANDLE: gmsHandle is invalid.

CL_ERR_NULL_POINTER: groupName or groupId params are NULL.

CL_ERR_NO_MEMORY: Could not allocate memory to create groups.

CL_ERR_NO_RESOURCE: Could not allocate resource to create groups.

CL_ERR_ALREADY_EXIST: Group is already created. At this point the created groupId is returned in groupId parameter.

CL_ERR_TIMEOUT: Group Creation timeout.

CL_ERR_UNSPECIFIED: Group creation failed with unknown error.

Description:
User needs to pass groupName and groupId pointer. GMS will generate a groupId which will be unique across the cluster, and it will be returned through groupId pointer.

Library File:
ClGms

Related Function(s):
clGmsGroupDestroy.

3.3.10 clGmsGroupDestroy

clGmsGroupDestroy

Synopsis:
Destroys a group.

Header File:
clGmsGroupManageApi.h

Syntax:

```
ClRcT clGmsGroupDestroy(
                     CL_IN   ClGmsHandleT                  gmsHandle,
                     CL_IN   ClGmsGroupIdT                 groupId);
```

Parameters:

gmsHandle: (in) gmsHandle provided during clGmsInitialize

groupId: (in) groupId provided during GroupCreate

Return values:

CL_OK: The API executed successfully.

CL_ERR_TRY_AGAIN: Server is not ready to serve group functionality.

CL_ERR_VERSION_MISMATCH: Client version not supported.

CL_ERR_INVALID_HANDLE: gmsHandle is invalid.

CL_ERR_DOESNT_EXIST: Requested group with groupId doesn't exist.

CL_ERR_INUSE: Group is not empty. However in this case, the group will be set inActive and hence no further joins can happen.

Description:
The group will be destroyed on all nodes across the cluster.

Library File:
ClGms

Related Function(s):
clGmsGroupCreate

3.3.11 clGmsGroupJoin

clGmsGroupJoin

Synopsis:
Allows a member to join a group.

Header File:
clGmsGroupManageApi.h

Syntax:

```
ClRcT clGmsGroupJoin(
   CL_IN ClGmsHandleT                        gmsHandle,
   CL_IN ClGmsGroupIdT                       groupId,
   CL_IN ClGmsMemberIdT                      memberId,
   CL_IN ClGmsMemberNameT                   *memberName,
   CL_IN ClGmsLeadershipCredentialsT         credentials,
   CL_IN ClTimeT                             timeout);
```

Parameters:

gmsHandle: (in) gmsHandle provided during clGmsInitialize.

groupId: (in) groupId provided during GroupCreate.

memberId: (in) Id of the member joining the group.

memberName: (in) Name of the member joining the group. This is optional.

credentials: Leadership credentials of the group member. This parameter is currently not used by GMS. It is meant for future enhancements.

timeout: (in) Join timeout.

Return values:

CL_OK: The API executed successfully.

CL_ERR_TRY_AGAIN: Server is not ready to serve group functionality.

CL_ERR_VERSION_MISMATCH: Client version not supported.

CL_ERR_INVALID_HANDLE: gmsHandle is invalid.

CL_ERR_DOESNT_EXIST: Requested group with groupId doesn't exist.

CL_ERR_TIMEOUT: Groupjoin timeout.

CL_ERR_ALREADY_EXIST: The application is an existing member of the group.

CL_ERR_INVALID_OPERATION: Join is denied as the group is marked to be destroyed.

Description:
This API can be used by any application to join an existing group as a member. Applications which have registered for track notifications will be notified by invoking *clGmsGroupTrackCallback()*.

Library File:
ClGms

Related Function(s):
clGmsGroupLeave

3.3.12 clGmsGroupLeave

clGmsGroupLeave

Synopsis:

Allows a member to leave a group.

Header File:

clGmsGroupManageApi.h

Syntax:

```
        ClRcT clGmsGroupLeave(
                CL_IN ClGmsHandleT                        gmsHandle,
                CL_IN ClGmsGroupIdT                       groupId,
                CL_IN ClGmsMemberIdT                      memberId,
                CL_IN ClTimeT                             timeout);
```

Parameters:

gmsHandle: (in) gmsHandle provided during clGmsInitialize.

groupId: (in) groupId provided during GroupCreate.

memberId: (in) Id of the member joining the group.

timeout: (in) Join timeout.

Return values:

CL_OK: The API executed successfully.

CL_ERR_TRY_AGAIN: Server is not ready to serve group functionality.

CL_ERR_VERSION_MISMATCH: Client version not supported.

CL_ERR_INVALID_HANDLE: gmsHandle is invalid.

CL_ERR_DOESNT_EXIST: The member with given *memberId* does not exist in the given group with *groupId*.

CL_GMS_ERR_GROUP_DOESNT_EXIST: requested group does not exist.

CL_ERR_TIMEOUT: Groupjoin timeout.

Description:

This API can be used by a member of the group to leave the group. Applications registered for track notifications on the group, will get notified through clGmsGroupTrackCallback().

Library File:

ClGms

Related Function(s):

clGmsGroupJoin

3.3.13 clGmsGroupTrack

clGmsGroupTrack

Synopsis:
Configures the group tracking mode.

Header File:
clGmsViewApi.h

Syntax:

```
ClRcT clGmsGroupTrack(
                          CL_IN    ClGmsHandleT              gmsHandle,
                          CL_IN    ClGmsGroupIdT             groupId,
                          CL_IN    ClUint8T                  trackFlags,
                          CL_INOUT ClGmsGroupNotificationBufferT *notificationBuffer);
```

Parameters:

gmsHandle: (in) Handle of the GMS service session.

groupId: (in) Id of the group.

trackFlags: (in) Requested tracking mode.

notificationBuffer: (in/out) his is an optional parameter and is used when *trackFlags* is set to *CL_TRACK_CURRENT*. It provides a buffer that contains the current cluster information when the API returns successfully. If it is NULL, the information is returned through an invocation to *clGmsGroupTrackCallback()*

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

CL_GMS_ERR_GROUP_DOES_NOT_EXIST: If the requested group does not exist.

CL_ERR_INVALID_PARAMETER: On passing an invalid parameter.

CL_ERR_NULL_POINTER: On passing a NULL pointer.

Description:
This API is used to configure the group tracking mode for the caller. It can be called subsequently to modify the requested tracking mode. This function is used to obtain the current group membership and request notification of changes in the cluster membership or of changes in an attribute of a cluster node, depending on the value of the *trackFlags* parameter.
These changes are notified through invocation of the clGmsClusterTrackCallback() callback function, which must have been supplied when the process invoked the *clGmsInitialize()* call. An application may call *clGmsClusterTrack()* repeatedly for the same values of *gmsHandle*, regardless of whether the call initiates a one-time status request or a series of callback notifications.

Library File:
ClGms

Related Function(s):
clGmsGroupTrackStop

3.3.14 clGmsGroupTrackStop

clGmsGroupTrackStop

Synopsis:
Stops all the group tracking.

Header File:
clGmsViewApi.h

Syntax:

```
        ClRcT clGmsGroupTrackStop(
                                    CL_IN ClGmsHandleT              gmsHandle,
                                    CL_IN ClGmsGroupIdT             groupId);
```

Parameters:

gmsHandle: (in) Handle of the GMS service session.

groupId: (in) Id of the group.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

CL_GMS_ERR_GROUP_DOESNT_EXIST: If the requested group does not exist.

Description:
This API is used to immediately stop all the group tracking for a group.

Library File:
ClGms

Related Function(s):
clGmsGroupTrack

3.3.15 clGmsGetGroupInfo

clGmsGetGroupInfo

Synopsis:
Returns the information of a group specified by the groupName

Header File:
clGmsViewApi.h

Syntax:

```
ClRcT clGmsGetGroupInfo(
                    CL_IN     ClGmsHandleT            gmsHandle,
                    CL_IN     ClGmsGroupNameT        *groupName,
                    CL_IN     ClTimeT                 timeout,
                    CL_INOUT  ClGmsGroupInfoT        *groupInfo);
```

Parameters:

gmsHandle: (in) Handle of the GMS service session.

groupName: (in) Pointer to ClGmsGroupNameT structure holding the value and length for the name of the group for which info is requested.

timeout: (in) Max timeout value for the API.

groupInfo: (in/out) Pointer to ClGmsGroupInfoT structure. User has to allocate memory for the ClGmsGroupInfoT structure and pass the address of the memory location through groupInfo pointer. GMS will fill-in the values in the memory pointed by groupInfo

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

CL_ERR_NULL_POINTER: groupName or groupInfo pointers are NULL

CL_ERR_TIMEOUT: Operation Timed out.

CL_ERR_DOESNT_EXIST: Group indicated by the groupName doesn't exist.

CL_ERR_TRY_AGAIN: Server is not ready to serve the request.

Description:
This API is used to retrieve the information on a given group member. The space for the member node must be allocated by you.

Library File:
ClGms

Related Function(s):
clGmsGroupsInfoListGet

3.3.16 clGmsGroupsInfoListGet

clGmsGroupsInfoListGet

Synopsis:
Returns the information of all the groups.

Header File:
clGmsViewApi.h

Syntax:

```
ClRcT clGmsGroupsInfoListGet(
                                CL_IN     ClGmsHandleT            gmsHandle,
                                CL_IN     ClTimeT                 timeout,
                                CL_INOUT  clGmsGroupInfoListT         *groups);
```

Parameters:

gmsHandle: (in) Handle of the GMS service session.

timeout: (in) If the operation is not completed within this time, then the request is timed out.

groups: (in/out) Pointer to the structure holding noOfGroups param and a pointer. The user should specify the pointer to the structure of type clGmsGroupInfoT, in which GMS will allocate the memory for groupInfo pointer and fills the noOfGroups value. The user should deallocate the memory given with groupInfo pointer.

Return values:

CL_OK: The API executed successfully.

CL_ERR_INVALID_HANDLE: On passing an invalid handle.

CL_ERR_INVALID_PARAMETER: On passing an invalid parameter.

CL_ERR_NULL_POINTER: On passing a NULL pointer.

Description:
This API is used to retrieve the information on all the groups existing on the node. The user should pass the pointer to the ClGmsGroupInfoListT data structure. GMS allocates the memory for all the groups.

Library File:
ClGms

Related Function(s):
clGmsGetGroupInfo

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Debug CLI

TBD

Index