Service Availability™ Forum Application Interface Specification

Volume 3: Cluster Membership Service SAI-AIS-CLM-B.01.01

SERVICE AVAILABILITY™ FORUM

Table of Contents Volume 3, Cluster Membership Service

1 Document Introduction 7

1.1 Document Purpose 7
1.2 AIS Documents Organization 7
1.3 How to Provide Feedback on the Specification 8
1.4 How to Join the Service Availability™ Forum 8
1.5 Additional Information 8
1.5.1 Member Companies 8
1.5.2 Press Materials 8

2 Overview 11

2.1 Cluster Membership Service 11

3 SA Cluster Membership Service API 13

3.1 Cluster Membership Model 13
3.2 Include File and Library Name 14
3.3 Type Definitions 14
3.3.1 SaClmHandleT 14
3.3.2 SaClmNodeIdT 14
3.3.3 SaClmNodeAddressT 15
3.3.4 SaClmClusterNodeT 16
3.3.5 SaClmClusterChangesT 17
3.3.6 SaClmClusterNotificationT 17
3.3.7 SaClmClusterNotificationBufferT 18
3.3.8 SaClmCallbacksT 18
3.4 Library Life Cycle 19
3.4.1 saClmInitialize() 19
3.4.2 saClmSelectionObjectGet() 21
3.4.3 saClmDispatch() 22
3.4.4 saClmFinalize() 23
3.5 Cluster Membership Operations 25
3.5.1 saClmClusterTrack() 25
3.5.2 SaClmClusterTrackCallbackT 28
3.5.3 saClmClusterTrackStop() 30
3.5.4 saClmClusterNodeGet() 31
3.5.5 saClmClusterNodeGetAsync() 32
3.5.6 SaClmClusterNodeGetCallbackT 34

1 Document Introduction

1.1 Document Purpose

This document defines the Cluster Membership Service of the Application Interface Specification (AIS) of the Service Availability^{TM} Forum. It is intended for use by implementors of the Application Interface Specification and by application developers who would use the Application Interface Specification to develop applications that must be highly available. The AIS is defined in the C programming language, and requires substantial knowledge of the C programming language.

Typically, the Service Availability^{TM} Forum Application Interface Specification will be used in conjunction with the Service Availability^{TM} Forum Hardware Interface Specification (HPI) and with the Service Availability^{TM} Forum System Management Specification, which is still under development.

1.2 AIS Documents Organization

The Application Interface Specification is organized into the following volumes:

Volume 1, the Overview document, provides a brief guide to the remainder of the Application Interface Specification. It describes the objectives of the AIS specification as well as programming models and definitions that are common to all specifications. Additionally, it contains an overview of the Availability Management Framework and of the other services as well as the system description and conceptual models, including the physical and logical entities that make up the system. Volume 1 also contains a chapter that describes the main abbreviations, concepts and terms used in the AIS documents.
The name of the pdf file containing this document for the AIS version B.01.01 is:
aisOverview.B0101.pdf

Volume 2 describes the Availability Management Framework API.
The name of the pdf file containing this document for the AIS version B.01.01 is:
aisAmf.B0101.pdf

Volume 3 (this volume) describes the Cluster Membership Service API.
The name of the pdf file containing this document for the AIS version B.01.01 is:
aisClm.B0101.pdf

Volume 4 describes the Checkpoint Service API.
The name of the pdf file containing this document for the AIS version B.01.01 is:
aisCkpt.B0101.pdf

Volume 5 describes the Event Service API.
The name of the pdf file containing this document for the AIS version B.01.01 is: **aisEvt.B0101.pdf**

Volume 6 describes the Message Service API.
The name of the pdf file containing this document for the AIS version B.01.01 is: **aisMsg.B0101.pdf**

Volume 7 describes the Lock Service API.
The name of the pdf file containing this document for the AIS version B.01.01 is: **aisLck.B0101.pdf**

1.3 How to Provide Feedback on the Specification

If you have a question or comment about this specification, you may submit feedback online by following the links provided for this purpose on the Service Availability™ Forum website (**http://www.saforum.org**).

You can also sign up to receive information updates on the Forum or the Specification.

1.4 How to Join the Service Availability™ Forum

The Promoter Members of the Forum require that all organizations wishing to participate in the Forum complete a membership application. Once completed, a representative of the Service Availability™ Forum will contact you to discuss your membership in the Forum. The Service Availability™ Forum Membership Application can be completed online by following the pertinent links provided on the Forum's website (**http://www.saforum.org**).

You can also submit information requests online. Information requests are generally responded to within three business days.

1.5 Additional Information

1.5.1 Member Companies

A list of the Service Availability™ Forum member companies can also be viewed online by using the links provided on the Forum's website (**http://www.saforum.org**).

1.5.2 Press Materials

The Service Availability™ Forum has available a variety of downloadable resource materials, including the Forum Press Kit, graphics, and press contact information.

Visit this area often for the latest press releases from the Service Availability™ Forum and its member companies by following the pertinent links provided on the Forum’s website (**http://www.saforum.org**).

2 Overview

This specification defines the Cluster Membership Service within the Application Interface Specification (AIS).

2.1 Cluster Membership Service

The Cluster Membership Service provides applications with membership information about the nodes in a cluster, and it is core to any clustered system. A cluster consists of a set of nodes, each with a unique node name. As nodes join and leave the cluster, the cluster membership is modified and maintained.

The Cluster Membership Service also allows application processes to register a callback function to receive membership change notifications as those changes occur. All of these functions are available on all nodes in the cluster.

To be eligible as a cluster member, a node must run the Cluster Membership Service but does not need to run the Availability Management Framework or any of the other SA AIS services.

If the Cluster Membership Service detects serious communication problems between a node and the remaining cluster, such as intermittent communication or total lack of communication, and the Cluster Membership Service indicates that this node left the cluster through its API functions, the Cluster Membership Service must trigger a failfast mechanism on that node, such as a reboot, to isolate the leaving node from the remaining cluster, in the sense that no process using AIS interfaces is left over on this node. The Cluster Membership Service should make all possible attempts to ensure that the failfast is triggered before the membership change is indicated via Cluster Membership Service API functions.

If a node leaves the cluster via administrative actions directed to the Cluster Membership Service (these administrative actions are not covered by this specification), the Cluster Membership Service must inform its client processes of this membership change through its API functions. Before such administrative action is taken, it must be ensured that the leaving node has been isolated from the remaining of the cluster, in the sense that no process using AIS interfaces is left over on this node. The mechanism for doing this is implementation-specific and may or may not be part of the Cluster Membership Service.

3 SA Cluster Membership Service API

3.1 Cluster Membership Model

A *cluster* comprises a set of cluster nodes. A *cluster node,* or simply *node,* is a logical representation of a physical node. The set of cluster nodes that are in the cluster at a given point in time is referred to as the *cluster membership*, or simply *membership*.

A membership transition is a change in the cluster membership. Each membership transition has an associated *view number*. The view number increases with each membership transition, although not necessarily by one. All processes see the same view number for a particular membership transition.

When processes invoke the Cluster Membership Service APIs, they obtain the same information, about the cluster membership or about a cluster node, for a particular view number. Some implementations of the Cluster Membership Service may choose not to use the view number and may set the view number to zero.

The cluster membership model does not cover implementation-dependent features such as the existence of a master node responsible for maintaining the cluster membership status.

3.2 Include File and Library Name

The following statement containing declarations of data types and function prototypes must be included in the source of an application using the Cluster Membership Service API:

#include <saClm.h>

To use the Cluster Membership Service API, an application must be bound with the following library:

libSaClm.so

3.3 Type Definitions

The Cluster Membership Service uses the types described in the following sections.

3.3.1 SaClmHandleT

typedef SaUint64T SaClmHandleT;

The handle to the Cluster Membership Service that a process acquires through the *saClmInitialize()* function and uses in subsequent invocations of the functions of the Cluster Membership Service.

3.3.2 SaClmNodeIdT

typedef SaUint32T SaClmNodeIdT;

Node identifier that is unique in the cluster

#define SA_CLM_LOCAL_NODE_ID 0XFFFFFFFF

The constant SA_CLM_LOCAL_NODE_ID, used as *nodeId* in an API function of the Cluster Membership Service, identifies the *nodeId* of the cluster node that hosts the invoking process.

3.3.3 SaClmNodeAddressT

```
#define SA_CLM_MAX_ADDRESS_LENGTH 64

typedef enum {
        SA_CLM_AF_INET = 1,
        SA_CLM_AF_INET6 = 2
} SaClmNodeAddressFamilyT;

typedef struct {
        SaClmNodeAddressFamilyT family;
        SaUint16T length;
        SaUint8T value[SA_CLM_MAX_ADDRESS_LENGTH];
} SaClmNodeAddressT;
```

SaClmNodeAddressT provides a string representation of the communication address associated with a node.

If *family* is set to SA_CLM_AF_INET, *value* contains a text representation of an IPv4 address using the d.d.d.d notation (where the 'd's are the decimal values of the four 8-bit pieces of the address).

If *family* is set to SA_CLM_AF_INET6, *value* contains a string representation of an IPv6 address using the x:x:x:x:x:x:x:x representation (where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address).

The *length* field indicates the number of bytes in *value* being used to represent the address.

3.3.4 SaClmClusterNodeT

```
typedef struct {
        SaClmNodeIdT nodeId;
        SaClmNodeAddressT nodeAddress;
        SaNameT nodeName;
        SaBoolT member;
        SaTimeT bootTimestamp;
        SaUint64T initialViewNumber;
} SaClmClusterNodeT;
```

The *SaClmClusterNodeT* structure contains information about a cluster node. The fields of this structure have the following interpretation:

- *nodeId* - Node identifier that is unique in the cluster. A given *nodeId* identifies a node only for as long as the node is a member of the cluster membership. A node, identified by its name, can join the cluster again with the same *nodeId* that it had when it left the cluster membership or with a different one.
- *nodeAddress* - Node network communication address.
- *nodeName* - Node name that is unique in the cluster. Node names are not attached to a particular piece of hardware or a particular physical location of that hardware inside the cluster but rather to the ability (from a provisioning and I/O connectivity point of view) to host a particular set of components. Hence, replacing a board by another similar board, moving a board to another location in a chassis, booting an upward compatible version of the OS on the board, and so on, can be performed without changing the name of the node. For best practices, on operating systems providing the notion of node name, it is recommended that *nodeName* be derived from the operating system node name (Node DN: *safNode*=os_node_name; refer to the volume 1 of the AIS specification for details on naming issues).
- *member* - Node is, or is not, in the cluster membership.
- *bootTimestamp* - Timestamp at which the node was last booted.
- *initialViewNumber* - The initial view number, that is, the view number when this node joined the cluster. It is OK for two nodes joining at the same time to have the same initial view numbers. The only requirement is that if node N1 joins the cluster before node N2, then *initialViewNumber* of N1 < *initialViewNumber* of N2.

3.3.5 SaClmClusterChangesT

```
typedef enum {
        SA_CLM_NODE_NO_CHANGE = 1,
        SA_CLM_NODE_JOINED = 2,
        SA_CLM_NODE_LEFT = 3,
        SA_CLM_NODE_RECONFIGURED = 4
} SaClmClusterChangesT;
```

The values of the *SaClmClusterChangesT* enumeration type refer to a cluster node that is a member of the cluster, or that was a member, and has just left the cluster. These values have the following interpretation:

- SA_CLM_NODE_NO_CHANGE - The membership has not changed for the node. This value is used when the *trackFlags* parameter of the *saClmClusterTrack()* function, defined in Section 3.5.1 on page 25, is either SA_TRACK_CHANGES (and the node was already in the membership, it did not leave the membership, and none of its attributes have changed) or SA_TRACK_CURRENT.
- SA_CLM_NODE_JOINED - The node joined the cluster membership.
- SA_CLM_NODE_LEFT - The node left the cluster membership.
- SA_CLM_NODE_RECONFIGURED - Some attributes associated with the node have changed, for instance, the node name.

3.3.6 SaClmClusterNotificationT

```
typedef struct {
        SaClmClusterNodeT clusterNode;
        SaClmClusterChangesT clusterChange;
} SaClmClusterNotificationT;
```

The fields of the *SaClmClusterNotificationT* structure have the following interpretation:

- *clusterNode* - The information about a node that is a member of the cluster, or that was a member and has just left the cluster, as defined by the *SaClmClusterNodeT* structure described in Section 3.3.4 on page 16.
- *clusterChange* - The changes in the cluster membership, as defined by the *SaClmClusterChangesT* enumeration type in Section 3.3.5 on page 17.

3.3.7 SaClmClusterNotificationBufferT

typedef struct {

SaUint64T viewNumber;

SaUint32T numberOfItems;

*SaClmClusterNotificationT *notification;*

} SaClmClusterNotificationBufferT;

The fields of the *SaClmClusterNotificationBufferT* structure have the following interpretation:

- *viewNumber* - The current view number of the cluster membership. Some implementations may not use *viewNumber* and set it to zero.
- *numberOfItems* - number of elements of type *SaClmClusterNotificationT* in the notification buffer
- *notification* - start address of the notification buffer

3.3.8 SaClmCallbacksT

typedef struct {

SaClmClusterNodeGetCallbackT saClmClusterNodeGetCallback;

SaClmClusterTrackCallbackT saClmClusterTrackCallback;

} SaClmCallbacksT;

The callbacks structure supplied by a process to the Cluster Membership Service, containing the callback functions that the Cluster Membership Service can invoke.

3.4 Library Life Cycle

3.4.1 saClmInitialize()

Prototype

```
SaAisErrorT saClmInitialize(
        SaClmHandleT *clmHandle,
        const SaClmCallbacksT *clmCallbacks,
        SaVersionT *version
);
```

Parameters

clmHandle - [out] A pointer to the handle designating this particular initialization of the Cluster Membership Service that is to be returned by the Cluster Membership Service.

clmCallbacks - [in] If *clmCallbacks* is set to NULL, no callback is registered; otherwise, it is a pointer to a *SaClmCallbacksT* structure, containing the callback functions of the process that the Cluster Membership Service may invoke. Only non-NULL callback functions in this structure will be registered.

version - [in/out] As an input parameter, *version* is a pointer to the required Cluster Membership Service version. In this case, *minorVersion* is ignored and should be set to 0x00.
As an output parameter, the version actually supported by the Cluster Membership Service is delivered.

Description

This function initializes the Cluster Membership Service for the invoking process and registers the various callback functions. This function must be invoked prior to the invocation of any other Cluster Membership Service functionality. The handle *clmHandle* is returned as the reference to this association between the process and the Cluster Membership Service. The process uses this handle in subsequent communication with the Cluster Membership Service.

If the implementation supports the required *releaseCode,* and a major version >= the required *majorVersion*, SA_AIS_OK is returned. In this case, the *version* parameter is set by this function to:

- *releaseCode* = required release code

- *majorVersion* = highest value of the major version that this implementation can support for the required *releaseCode*
- *minorVersion* = highest value of the minor version that this implementation can support for the required value of *releaseCode* and the returned value of *majorVersion*

If the above mentioned condition cannot be met, SA_AIS_ERR_VERSION is returned, and the *version* parameter is set to:

```
if (implementation supports the required releaseCode)
    releaseCode = required releaseCode
else {
    if (implementation supports releaseCode higher than the required
    releaseCode)
        releaseCode = the least value of the supported release codes that is
        higher than the required releaseCode
    else
        releaseCode = the highest value of the supported release codes that is
        less than the required releaseCode
}
```

majorVersion = highest value of the major versions that this implementation can support for the returned *releaseCode*

minorVersion = highest value of the minor versions that this implementation can support for the returned values of *releaseCode* and *majorVersion*

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn’t.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly.

SA_AIS_ERR_NO_MEMORY - Either the Cluster Membership Service library or the provider of the service is out of memory and cannot provide the service.

SA_AIS_ERR_NO_RESOURCES -The system is out of required resources (other than memory).

SA_AIS_ERR_VERSION - The *version* parameter is not compatible with the version of the Cluster Membership Service implementation.

See Also

saClmSelectionObjectGet(), saClmDispatch(), saClmFinalize()

3.4.2 saClmSelectionObjectGet()

Prototype

SaAisErrorT saClmSelectionObjectGet(

SaClmHandleT clmHandle,

*SaSelectionObjectT *selectionObject*

);

Parameters

clmHandle - [in] The handle, obtained through the *saClmInitialize()* function, designating this particular initialization of the Cluster Membership Service.

selectionObject - [out] A pointer to the operating system handle that the invoking process can use to detect pending callbacks.

Description

This function returns the operating system handle, *selectionObject,* associated with the handle *clmHandle*. The invoking process can use this handle to detect pending callbacks, instead of repeatedly invoking *saClmDispatch()* for this purpose.

In a POSIX environment, the operating system handle is a file descriptor that is used with the *poll()* or *select()* system calls to detect pending callbacks.

The *selectionObject* returned by *saClmSelectionObjectGet()* is valid until *saClmFinalize()* is invoked on the same handle *clmHandle*.

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn't.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_BAD_HANDLE - The handle *clmHandle* is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly.

SA_AIS_ERR_NO_MEMORY - Either the Cluster Membership Service library or the provider of the service is out of memory and cannot provide the service.

SA_AIS_ERR_NO_RESOURCES -The system is out of required resources (other than memory).

See Also

saClmInitialize(), saClmDispatch(), saClmFinalize()

3.4.3 saClmDispatch()

Prototype

```
SaAisErrorT saClmDispatch(
        SaClmHandleT clmHandle,
        SaDispatchFlagsT dispatchFlags
);
```

Parameters

clmHandle - [in] The handle, obtained through the *saClmInitialize()* function, designating this particular initialization of the Cluster Membership Service.

dispatchFlags - [in] Flags that specify the callback execution behavior of the *saClmDispatch()* function, which have the values SA_DISPATCH_ONE, SA_DISPATCH_ALL, or SA_DISPATCH_BLOCKING, as defined in volume 1 of the AIS specification.

Description

This function invokes, in the context of the calling thread, pending callbacks for the handle *clmHandle* in a way that is specified by the *dispatchFlags* parameter.

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn’t.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_BAD_HANDLE - The handle *clmHandle* is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_INVALID_PARAM - The *dispatchFlags* parameter is invalid.

See Also

saClmInitialize(), saClmSelectionObjectGet(), saClmFinalize()

3.4.4 saClmFinalize()

Prototype

```
SaAisErrorT saClmFinalize(
        SaClmHandleT clmHandle
);
```

Parameters

clmHandle - [in] The handle, obtained through the *saClmInitialize()* function, designating this particular initialization of the Cluster Membership Service.

Description

The *saClmFinalize()* function closes the association, represented by the *clmHandle* parameter, between the invoking process and the Cluster Membership Service. The process must have invoked *saClmInitialize()* before it invokes this function. A process must invoke this function once for each handle it acquired by invoking *saClmInitialize()*.

If the *saClmFinalize()* function returns successfully, the *saClmFinalize()* function releases all resources acquired when *saClmInitialize()* was called. Moreover, it stops any tracking associated with the particular handle. Furthermore, it cancels all pending callbacks related to the particular handle. Note that because the callback invocation is asynchronous, it is still possible that some callback calls are processed after this call returns successfully.

After *saClmFinalize()* is invoked, the selection object is no longer valid.

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn't.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_BAD_HANDLE - The handle *clmHandle* is invalid, since it is corrupted, uninitialized, or has already been finalized.

See Also

saClmInitialize()

3.5 Cluster Membership Operations

3.5.1 saClmClusterTrack()

Prototype

SaAisErrorT saClmClusterTrack(

SaClmHandleT clmHandle,

SaUint8T trackFlags,

*SaClmClusterNotificationBufferT *notificationBuffer*

);

Parameters

clmHandle - [in] The handle, obtained through the *saClmInitialize()* function, designating this particular initialization of the Cluster Membership Service.

trackFlags - [in] The kind of tracking that is requested, which is the bitwise OR of one or more of the flags SA_TRACK_CURRENT, SA_TRACK_CHANGES or SA_TRACK_CHANGES_ONLY, defined in volume 1 of the AIS specification, which have the following interpretation here:

- SA_TRACK_CURRENT - If *notificationBuffer* is NULL, information about all nodes that are currently members in the cluster is returned by a single subsequent invocation of the *saClmClusterTrackCallback()* notification callback; otherwise, this information is returned in *notificationBuffer* when the *saClmClusterTrack()* call completes.
- SA_TRACK_CHANGES - The callback function *saClmClusterTrackCallback()* is invoked each time a change in the cluster membership or in an attribute of a cluster node occurs. The *notificationBuffer,* returned by *saClmClusterTrackCallback(),* contains information about all nodes that are currently members of the cluster, and also about nodes that have left the cluster membership since the last invocation of *saClmClusterTrackCallback()*.
- SA_TRACK_CHANGES_ONLY - The callback function *saClmClusterTrackCallback()* is invoked each time a change in the cluster membership or in an attribute of a cluster node occurs. The *notificationBuffer,* returned by *saClmClusterTrackCallback(),* contains information about new nodes in the cluster membership, nodes in the cluster membership whose attributes have changed, and nodes that have left the cluster membership since the last invocation of *saClmClusterTrackCallback()*.

If the SA_TRACK_CHANGES_ONLY flag is set, a process can receive incremental cluster membership change information for a view number j, where j > i+1 and it last received complete cluster membership information for view number i, which is not an appropriate complete membership against which to compare the increment. To avert such a situation, a process might invoke this function with both flags SA_TRACK_CURRENT and SA_TRACK_CHANGES_ONLY set.

It is not permitted to set both SA_TRACK_CHANGES and SA_TRACK_CHANGES_ONLY in an invocation of this function.

notificationBuffer - [in/out] - A pointer to a buffer of type *SaClmClusterNotificationBufferT*. This parameter is ignored if SA_TRACK_CURRENT is not set in *trackFlags*; otherwise, if *notificationBuffer* is not NULL, the buffer will contain information about all nodes in the cluster membership when *saClmClusterTrack()* returns. The meaning of the fields of the *SaClmClusterNotificationBufferT* buffer is:

- *viewNumber* - [out] The current view number of the cluster membership. Some implementations may not use the *viewNumber* and may set it to zero. If SA_TRACK_CURRENT is specified, the view number of the most recent transition is delivered.
- *numberOfItems* - [in/out] If *notification* is NULL, *numberOfItems* is ignored as input parameter; otherwise, it specifies that the buffer pointed to by *notification* provides memory for information about *numberOfItems* nodes in the cluster membership.
  When *saClmClusterTrack()* returns with SA_AIS_OK or with SA_AIS_ERR_NO_SPACE, *numberOfItems* contains the number of nodes in the cluster membership.
- *notification* - [in/out] If *notification* is NULL, memory for the cluster membership information is allocated by the Cluster Membership Service. The caller is responsible for freeing the allocated memory.

Description

This function is used to obtain the current cluster membership as well as to 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, as described above. These changes are notified via the invocation of the *saClmClusterTrackCallback()* callback function, which must have been supplied when the process invoked the *saClmInitialize()* call.

An application may call *saClmClusterTrack()* repeatedly for the same values of *clmHandle,* regardless of whether the call initiates a one-time status request or a series of callback notifications.

If *saClmClusterTrack()* is called with *trackFlags* containing SA_TRACK_CHANGES_ONLY when cluster changes are currently being tracked with SA_TRACK_CHANGES, the Cluster Membership Service will invoke further notification callbacks according to the new value of *trackFlags*. The same is true vice versa.
Once *saClmClusterTrack()* has been called with *trackFlags* containing either SA_TRACK_CHANGES or SA_TRACK_CHANGES_ONLY, notifications can only be stopped by an invocation of *saClmClusterTrackStop()* or *saClmFinalize()*.

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn't.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_BAD_HANDLE - The handle *clmHandle* is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_INIT - The previous initialization with *saClmInitialize()* was incomplete, since the *saClmClusterTrackCallback()* callback function is missing.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly. In particular this applies if, in *notificationBuffer, notification* is not NULL and *numberOfItems* is 0*.*

SA_AIS_ERR_NO_MEMORY - Either the Cluster Membership Service library or the provider of the service is out of memory and cannot provide the service.

SA_AIS_ERR_NO_RESOURCES -The system is out of required resources (other than memory).

SA_AIS_ERR_NO_SPACE - The SA_TRACK_CURRENT flag is set and the notification field in *notificationBuffer* is not NULL, but the *numberOfItems* field in *notificationBuffer* indicates that the provided buffer is too small to hold information about all nodes in the cluster membership.

SA_AIS_ERR_BAD_FLAGS - The *trackFlags* parameter is invalid.

See Also

saClmClusterTrackStop(), SaClmClusterTrackCallbackT, saClmInitialize()

3.5.2 SaClmClusterTrackCallbackT

Prototype

```
typedef void (*SaClmClusterTrackCallbackT) (
        const SaClmClusterNotificationBufferT *notificationBuffer,
        SaUint32T numberOfMembers,
        SaAisErrorT error
);
```

Parameters

notificationBuffer - [in] A pointer to a buffer of cluster node structures. It is possible that there is a change in the view number and no change in the cluster membership. In this case, if the flag SA_TRACK_CHANGES was set in the associated *saClmClusterTrack()* call, then the *clusterChanges* field in each entry of *notificationBuffer* is set to SA_CLM_NODE_NO_CHANGE.
If the flag SA_TRACK_CHANGES_ONLY was set in the associated *saClmClusterTrack()* call, then *notificationBuffer* does not contain any items.
If SA_TRACK_CURRENT was specified, the view number of the most recent transition is delivered.

numberOfMembers - [in] The current number of members in the cluster membership.

error - [in] This parameter indicates whether the Cluster Membership Service was able to perform the operation. The parameter *error* has one of the values:

- SA_AIS_OK - The function completed successfully.
- SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.
- SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn't.
- SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry the *saClmClusterTrack()* later.
- SA_AIS_ERR_NO_MEMORY - Either the Cluster Membership Service library or the provider of the service is out of memory and cannot provide the service. The process that invoked *saClmClusterTrack()* might have missed one or more notifications.
- SA_AIS_ERR_NO_RESOURCES -Either the Cluster Membership Service library or the provider of the service is out of required resources (other than

memory), and cannot provide the service. The process that invoked *saClmClusterTrack()* might have missed one or more notifications.

If the error returned is SA_AIS_ERR_NO_MEMORY or SA_AIS_ERR_NO_RESOURCES, the process that invoked *saClmClusterTrack()* should invoke *saClmClusterTrackStop()* and then invoke *saClmClusterTrack()* again to resynchronize with the current state.

Description

This callback is invoked in the context of a thread issuing an *saClmDispatch()* call on the handle *clmHandle,* which was specified when the process requested tracking of the cluster membership using the *saClmClusterTrack()* call*.* The *saClmClusterTrackCallback()* function returns information about the cluster nodes in the *notificationBuffer* parameter. The kind of information returned depends on the setting of the *trackFlags* parameter of the *saClmClusterTrack()* function.

If two processes, on the same or different nodes, request tracking with the same flags set then, for a particular view number, the Cluster Membership Service shall provide the same information about the cluster membership when it invokes the *saClmClusterTrackCallback()* function of those processes.

The value of the *numberOfItems* attribute in the *notificationBuffer* parameter might be greater than the value of the *numberOfMembers* parameter, because nodes have left the cluster membership: If the SA_TRACK_CHANGES flag or the SA_TRACK_CHANGES_ONLY flag is set, the *notificationBuffer* might contain information about the current members of the cluster and also about nodes that have recently left the cluster membership.

Return Values

None.

See Also

saClmClusterTrack(), saClmClusterTrackStop(), saClmDispatch()

3.5.3 saClmClusterTrackStop()

Prototype

SaAisErrorT saClmClusterTrackStop(

SaClmHandleT clmHandle

);

Parameters

clmHandle - [in] The handle, obtained through the *saClmInitialize()* function, designating this particular initialization of the Cluster Membership Service.

Description

This function stops any further notifications through the handle *clmHandle*. Pending callbacks are removed.

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn't.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_BAD_HANDLE - The handle *clmHandle* is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_NO_MEMORY - Either the Cluster Membership Service library or the provider of the service is out of memory and cannot provide the service.

SA_AIS_ERR_NO_RESOURCES -The system is out of required resources (other than memory).

SA_AIS_ERR_NOT_EXIST - No track of changes in the cluster membership was previously started via the *saClmClusterTrack()* call with track flags SA_TRACK_CHANGES or SA_TRACK_CHANGES_ONLY, and which is still in effect.

See Also

saClmClusterTrack(), SaClmClusterTrackCallbackT, saClmInitialize()

3.5.4 saClmClusterNodeGet()

Prototype

```
SaAisErrorT saClmClusterNodeGet(
        SaClmHandleT clmHandle
        SaClmNodeIdT nodeId,
        SaTimeT timeout,
        SaClmClusterNodeT *clusterNode
);
```

Parameters

clmHandle - [in] The handle, obtained through the *saClmInitialize()* function, designating this particular initialization of the Cluster Membership Service.

nodeId - [in] The identifier of the cluster node for which the *clusterNode* information structure is to be retrieved.

timeout - [in] The *saClmClusterNodeGet()* invocation is considered to have failed if it does not complete by the time specified.

clusterNode - [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 Cluster Membership Service fills in the fields of this structure.

Description

This function provides the means for synchronously retrieving information about a cluster member, identified by the *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 SA_CLM_LOCAL_NODE_ID is used as *nodeId,* the function returns information about the cluster node that hosts the invoking process.

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred, or the timeout, specified by the *timeout* parameter, occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn't.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_BAD_HANDLE - The handle *clmHandle* is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_INVALID_PARAM - The *nodeId* parameter is invalid.

SA_AIS_ERR_NO_MEMORY - Either the Cluster Membership Service library or the provider of the service is out of memory and cannot provide the service.

SA_AIS_ERR_NO_RESOURCES -The system is out of required resources (other than memory).

See Also

saClmClusterNodeGetAsync(), saClmInitialize()

3.5.5 saClmClusterNodeGetAsync()

Prototype

```
SaAisErrorT saClmClusterNodeGetAsync(
        SaClmHandleT clmHandle,
        SaInvocationT invocation,
        SaClmNodeIdT nodeId
);
```

Parameters

clmHandle - [in] The handle, obtained through the *saClmInitialize()* function, designating this particular initialization of the Cluster Membership Service.

invocation - [in] This parameter allows the invoking process to match this invocation of *saClmClusterNodeGetAsync()* with the corresponding *saClmClusterNodeGetCallback()*.

nodeId - [in] The identifier of the cluster node for which the information is to be retrieved.

Description

This function requests information, to be provided asynchronously, about the particular cluster node, identified by the *nodeId* parameter. If SA_CLM_LOCAL_NODE_ID is used as *nodeId,* the function returns information about the cluster node that hosts the invoking process. The process sets *invocation,* which it uses subsequently to match the corresponding callback, *saClmClusterNodeGetCallback(),* with this particular invocation. The *saClmClusterNodeGetCallback()* callback function must have been supplied when the process invoked the *saClmInitialize()* call.

Return Values

SA_AIS_OK - The function completed successfully.

SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.

SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn’t.

SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.

SA_AIS_ERR_BAD_HANDLE - The handle *clmHandle* is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_INIT - The previous initialization with *saClmInitialize()* was incomplete, since the *saClmClusterNodeGetCallback()* callback function is missing.

SA_AIS_ERR_INVALID_PARAM - The *nodeId* parameter is invalid.

See Also

saClmClusterNodeGet(), SaClmClusterNodeGetCallbackT, saClmInitialize()

3.5.6 SaClmClusterNodeGetCallbackT

Prototype

```
typedef void (*SaClmClusterNodeGetCallbackT)(
        SaInvocationT invocation,
        const SaClmClusterNodeT *clusterNode,
        SaAisErrorT error
);
```

Parameters

invocation - [in] This parameter makes the association between an invocation of the s*aClmClusterNodeGetCallback*() function by the Cluster Membership Service and an invocation of the *saClmClusterNodeGetAsync()* function by a process.

clusterNode - [in] A pointer to a structure containing information about the node of the cluster, identified by *nodeId,* in the invocation of the *saClmClusterNodeGetAsync()* function.

error - [in] This parameter indicates whether the *saClmClusterNodeGetAsync()* function was successful. The possible return values are:

- SA_AIS_OK - The function completed successfully.
- SA_AIS_ERR_LIBRARY - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.
- SA_AIS_ERR_TIMEOUT - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it didn't.
- SA_AIS_ERR_TRY_AGAIN - The service cannot be provided at this time. The process may retry later.
- SA_AIS_ERR_INVALID_PARAM - The *nodeId* parameter provided in the *saClmClusterNodeGetAsync()* function is invalid.
- SA_AIS_ERR_NO_MEMORY - Either the Cluster Membership Service library or the provider of the service is out of memory and cannot provide the service.
- SA_AIS_ERR_NO_RESOURCES -The system is out of required resources (other than memory).

Description

The Cluster Membership Service invokes this callback function when the operation requested by the invocation of *saClmClusterNodeGetAsync()* completes. This call-back is invoked in the context of a thread issuing an *saClmDispatch()* call on the han-dle *clmHandle,* which was specified in the *saClmClusterNodeGetAsync()* call. If successful, this callback function returns information in *clusterNode* about a particular cluster node, identified by the *nodeId* field of the *clusterNode* parameter, provided by the process in the invocation of *saClmClusterNodeGetAsync()*. The process sets the value of *invocation* when it invokes the *saClmClusterNodeGetAsync()* function. The Cluster Membership Service uses *invocation* when it invokes this callback function, so that the process can associate its request with this callback.

Upon receipt of this function, the *member* field of the *clusterNode* parameter must be inspected to determine whether this node is a member of the cluster.

If an error occurs, it is returned in the error parameter.

Return Values

None.

See Also

saClmClusterNodeGetAsync(), saClmDispatch()