Service Availability™ Forum Application Interface Specification

Cluster Membership Service SAI-AIS-CLM-B.04.01

SERVICE AVAILABILITY™ FORUM SPECIFICATION LICENSE AGREEMENT

The Service Availability™ Forum Application Interface Specification (the "Package") found at the URL http://www.saforum.org is generally made available by the Service Availability Forum (the "Copyright Holder") for use in developing products that are compatible with the standards provided in the Specification. The terms and conditions which govern the use of the Package are covered by the Artistic License 2.0 of the Perl Foundation, which is reproduced here.

The Artistic License 2.0

Copyright (c) 2000-2006, The Perl Foundation.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

This license establishes the terms under which a given free software Package may be copied, modified, distributed, and/or redistributed.

The intent is that the Copyright Holder maintains some artistic control over the development of that Package while still keeping the Package available as open source and free software.

You are always permitted to make arrangements wholly outside of this license directly with the Copyright Holder of a given Package. If the terms of this license do not permit the full use that you propose to make of the Package, you should contact the Copyright Holder and seek a different licensing arrangement.

Definitions

"Copyright Holder" means the individual(s) or organization(s) named in the copyright notice for the entire Package.

"Contributor" means any party that has contributed code or other material to the Package, in accordance with the Copyright Holder's procedures.

"You" and "your" means any person who would like to copy, distribute, or modify the Package.

"Package" means the collection of files distributed by the Copyright Holder, and derivatives of that collection and/or of those files. A given Package may consist of either the Standard Version, or a Modified Version.

"Distribute" means providing a copy of the Package or making it accessible to anyone else, or in the case of a company or organization, to others outside of your company or organization.

"Distributor Fee" means any fee that you charge for Distributing this Package or providing support for this Package to another party. It does not mean licensing fees.

"Standard Version" refers to the Package if it has not been modified, or has been modified only in ways explicitly requested by the Copyright Holder.

"Modified Version" means the Package, if it has been changed, and such changes were not explicitly requested by the Copyright Holder.

"Original License" means this Artistic License as Distributed with the Standard Version of the Package, in its current version or as it may be modified by The Perl Foundation in the future.

"Source" form means the source code, documentation source, and configuration files for the Package.

"Compiled" form means the compiled bytecode, object code, binary, or any other form resulting from mechanical transformation or translation of the Source form.

Permission for Use and Modification Without Distribution

(1) You are permitted to use the Standard Version and create and use Modified Versions for any purpose without restriction, provided that you do not Distribute the Modified Version.

Permissions for Redistribution of the Standard Version

(2) You may Distribute verbatim copies of the Source form of the Standard Version of this Package in any medium without restriction, either gratis or for a Distributor Fee, provided that you duplicate all of the original copyright notices and associated disclaimers. At your discretion, such verbatim copies may or may not include a Compiled form of the Package.

(3) You may apply any bug fixes, portability changes, and other modifications made available from the Copyright Holder. The resulting Package will still be considered the Standard Version, and as such will be subject to the Original License.

Distribution of Modified Versions of the Package as Source

(4) You may Distribute your Modified Version as Source (either gratis or for a Distributor Fee, and with or without a Compiled form of the Modified Version) provided that you clearly document how it differs from the Standard Version, including, but not limited to, documenting any non-standard features, executables, or modules, and provided that you do at least ONE of the following:

(a) make the Modified Version available to the Copyright Holder of the Standard Version, under the Original License, so that the Copyright Holder may include your modifications in the Standard Version.

(b) ensure that installation of your Modified Version does not prevent the user installing or running the Standard Version. In addition, the Modified Version must bear a name that is different from the name of the Standard Version.

(c) allow anyone who receives a copy of the Modified Version to make the Source form of the Modified Version available to others under

(i) the Original License or

(ii) a license that permits the licensee to freely copy, modify and redistribute the Modified Version using the same licensing terms that apply to the copy that the licensee received, and requires that the Source form of the Modified Version, and of any works derived from it, be made freely available in that license fees are prohibited but Distributor Fees are allowed.

Distribution of Compiled Forms of the Standard Version or Modified Versions without the Source

(5) You may Distribute Compiled forms of the Standard Version without the Source, provided that you include complete instructions on how to get the Source of the Standard Version. Such instructions must be valid at the time of your distribution. If these instructions, at any time while you are carrying out such distribution, become invalid, you must provide new instructions on demand or cease further distribution.

If you provide valid instructions or cease distribution within thirty days after you become aware that the instructions are invalid, then you do not forfeit any of your rights under this license.

(6) You may Distribute a Modified Version in Compiled form without the Source, provided that you comply with Section 4 with respect to the Source of the Modified Version.

Aggregating or Linking the Package

(7) You may aggregate the Package (either the Standard Version or Modified Version) with other packages and Distribute the resulting aggregation provided that you do not charge a licensing fee for the Package. Distributor Fees are permitted, and licensing fees for other components in the aggregation are permitted. The terms of this license apply to the use and Distribution of the Standard or Modified Versions as included in the aggregation.

(8) You are permitted to link Modified and Standard Versions with other works, to embed the Package in a larger work of your own, or to build stand-alone binary or bytecode versions of applications that include the Package, and Distribute the result without restriction, provided the result does not expose a direct interface to the Package.

Items That are Not Considered Part of a Modified Version

(9) Works (including, but not limited to, modules and scripts) that merely extend or make use of the Package, do not, by themselves, cause the Package to be a Modified Version. In addition, such works are not considered parts of the Package itself, and are not subject to the terms of this license.

General Provisions

(10) Any use, modification, and distribution of the Standard or Modified Versions is governed by this Artistic License. By using, modifying or distributing the Package, you accept this license. Do not use, modify, or distribute the Package, if you do not accept this license.

(11) If your Modified Version has been derived from a Modified Version made by someone other than you, you are nevertheless required to ensure that your Modified Version complies with the requirements of this license.

(12) This license does not grant you the right to use any trademark, service mark, tradename, or logo of the Copyright Holder.

(13) This license includes the non-exclusive, worldwide, free-of-charge patent license to make, have made, use, offer to sell, sell, import and otherwise transfer the Package with respect to any patent claims licensable by the Copyright Holder that are necessarily infringed by the Package. If you institute patent litigation (including a cross-claim or counterclaim) against any party alleging that the Package constitutes direct or contributory patent infringement, then this Artistic License to you shall terminate on the date that such litigation is filed.

(14) Disclaimer of Warranty:

THE PACKAGE IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS "AS IS' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES. THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT ARE DISCLAIMED TO THE EXTENT PERMITTED BY YOUR LOCAL LAW. UNLESS REQUIRED BY LAW, NO COPYRIGHT HOLDER OR CONTRIBUTOR WILL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING IN ANY WAY OUT OF THE USE OF THE PACKAGE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents — Cluster Membership Service

List of Figures

List of Tables

1 Document Introduction

1.1 Document Purpose

This document defines the Cluster Membership Service of the Application Interface Specification (AIS) of the Service Availability™ Forum (SA Forum). It is intended for use by implementers 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™ Forum Application Interface Specification will be used in conjunction with the Service Availability™ Forum Hardware Platform Interface Specification (HPI).

1.2 AIS Documents Organization

The Application Interface Specification is organized into several volumes. For a list of all Application Interface Specification documents, refer to the SA Forum Overview document ([1]).

1.3 History

Previous versions of the Cluster Membership Service specification:

(1) SAI-AIS-CLM-A.01.01
(2) SAI-AIS-CLM-B.01.01
(3) SAI-AIS-CLM-B.02.01
(4) SAI-AIS-CLM-B.03.01

This section presents the changes of the current version, SAI-AIS-CLM-B.04.01, with respect to the SAI-AIS-CLM-B.03.01 version. Editorial changes that do not change semantics or syntax of the described interfaces are not mentioned.

1.3.1 New Topics

This version of the Cluster Membership Service introduces the notion of correlation ids, adapts the Cluster Membership Service to the PLM Service, and enhances the track interface. The main changes are:

- Section 3.1 describes the interactions with the PLM Service, explains the principles of the track interface, and clarifies the behavior of the Cluster Membership Service for non-member nodes.
- Section 3.3.4 adds to the description of a cluster node the mapping of a CLM node to an EE. This modification led to the replacement of the `SaClmClusterNodeT` type with `SaClmClusterNodeT_4` and to the replacement of the `SaClmClusterNotificationT` type with `SaClmClusterNotificationT_4` (see Section 3.3.6). The latter modification caused the replacement of the `SaClmClusterNotificationBufferT` type with `SaClmClusterNotificationBufferT_4` (see Section 3.3.7).
  The preceding changes led in turn to the replacement of the functions `saClmClusterNodeGet()`, `SaClmClusterNodeGetCallbackT`, and `saClmClusterNotificationFree()` with `saClmClusterNodeGet_4()` (see Section 3.5.5), `SaClmClusterNodeGetCallbackT_4` (see Section 3.5.7), and `saClmClusterNotificationFree_4()` (see Section 3.5.4) respectively.
- Section 3.3.5 extends the `SaClmClusterChangesT` enumeration.
- In Section 3.3.8, the `SaClmCallbacksT_4` structure supersedes the `SaClmCallbacksT_3` due to the superseding callback functions. As a consequence, the `saClmInitialize_4()` function superseded `saClmInitialize_3()`.
- Section 3.3.14 adds a type to specify the `minorId` of the notifications produced by the Cluster Membership Service. The descriptions of the notifications in Section 6.2 refer to this type.
- Section 3.3.15 adds the `SA_CLM_ROOT_CAUSE_ENTITY` field to the `SaClmAdditionalInfoIdT` enumeration. This type was superseded due to a change explained in Section 1.4.
- Section 3.5.1 introduces the enhanced track API of the Cluster Membership Service, which is similar to the track API of the PLM Service. For the main design principles of the enhanced track API, see [2]. Due to this change and to the new type for the `notificationBuffer` parameter, the `saClmClusterTrack_4()` function superseded the `saClmClusterTrack()` function.

- Section 3.5.2 adds multiple steps to the extended track callback interface. This modification induced also the definition of the `SaClmChangeStepT` type (see Section 3.3.12). The track callback interface has also additional parameters providing the correlation identifiers (see [3]), the name of the entity directly targeted by the action or event that caused the membership change, and a parameter to specify a supervision time interval. Due to changes in the parameters of this function, the `SaClmClusterTrackCallbackT_4` function superseded `SaClmClusterTrackCallbackT`.
- The `SaClmResponse_4()` function in Section 3.5.8 supersedes the `saClmResponse()` function, as the type of the third parameter has changed to `SaClmResponseT` (see Section 3.3.13).
- Section 4.3 contains a UML diagram showing relationships between the Cluster Membership Service classes and classes of other SA Forum Services.
- Section 4.4 contains changes to the UML diagram for a cluster node. A cluster node is mapped now to an EE and not to an HPI entity path (see [6]). The corresponding attribute is `saClmNodeEE` in the `SaClmNode` object class, shown in FIGURE 2. Note also that the `saClmNodeEntityPaths` attribute has been removed from the `SaClmNode` object class.
- Chapter 5 on administrative operations was modified due to the changes in Chapter 3 described above.
- Chapter 5 was also changed to allow an unlock operation while a node is being administratively shut down.
- Section 6.1 adds the correlation id to common attributes.
- Section 6.2.2.1 and Section 6.2.2.2 add the cause to the additional information attribute of the state change notifications.
- The term “entity” was changed to “node” in all places where it referred to nodes only. The term “entity” is now used only when referring to entities outside the Cluster Membership Service.

1.3.2 Clarifications

None

1.3.3 Removed Topics

- The eviction callback function, `SaClmClusterNodeEvictionCallbackT`, is no longer provided in this version, as its functionality is covered to a great extent by the newly introduced track interface. The description of this callback and all references to it in other chapters have been removed. Note that the alarm “cluster node eviction callback failure” has also been removed. Due to this latter change, the `SA_CLM_NODE_NAME` value was removed from the `SaClmAdditionalInfoIdT` type; this modification, in turn, led to the replacement of the `SaClmAdditionalInfoIdT` type with `SaClmAdditionalInfoIdT_4`.
  Former users of the eviction callback should provide a track callback function in an invocation of the `saClmInitialize_4()` call and start cluster membership tracking by invoking `saClmClusterTrack_4()` with the track flags set to the logical ORing of `SA_TRACK_CHANGES`, `SA_TRACK_LOCAL`, and `SA_TRACK_START_STEP`. Doing this, the same events leading formerly to an invocation of the user's eviction callback will also lead to the invocation of the corresponding user’s track callback. Additionally, the user will receive the completed steps for other changes on its local cluster node as described in Section 3.1.4.
- Section 7 on “Cluster Membership Service Management Interface” was removed, as no MIBs are released with this specification.

1.3.4 Other Changes

- Section 3.1.5 on the Cluster Membership Service functionality in non-member cluster nodes has changed.
- In the previous version of the Cluster Membership Service, the `SaClmCluster` object class contained the `saClmClusterNodeDisableReboot` attribute. In the SAI-AIS-CLM-B.04.01 version, this attribute was moved to the `SaClmNode` class; it now has the name `saClmNodeDisableReboot` and a default value of `SA_FALSE` (see FIGURE 2).

1.3.5 Superseded and Superseding Functions

The Cluster Membership Service defines for the version B.04.01 new functions and new type definitions to replace functions and type definitions of the version B.03.01. The list of replaced functions and type definitions in alphabetic order is presented inTable 1.

The superseded functions and type definitions are no longer supported in version B.04.01, and no description is provided for them in this document. The names of the superseding functions and type definitions are obtained by adding "_4" to the respective names of the B.03.01 version or by replacing "_*n*" (where *n* is a number < =3) by "_4" if the superseded functions or type definitions had already "_*n*" at the end of their names. Exceptions to this rule are indicated by table footnotes.
Regarding the support of backward compatibility in SA Forum AIS, refer to [2].

Table 1 Superseded Functions and Type Definitions in Version B.04.01

Functions and Type Definitions of Version B.03.01 no Longer Supported in B.04.01
`SaClmAdditionalInfoIdT`
`SaClmCallbacksT_3`
`SaClmClusterNodeEvictionCallbackT`[1]
`saClmClusterNodeGet()`
`SaClmClusterNodeGetCallbackT`
`SaClmClusterNodeT`
`SaClmClusterNotificationBufferT`
`saClmClusterNotificationFree()`
`SaClmClusterNotificationT`
`saClmClusterTrack()`
`SaClmClusterTrackCallbackT`
`saClmInitialize_3()`
`SaClmResponse()`

1. The functionality that was provided by this function is now provided by a subset of the new `SaClmClusterTrackCallbackT` function.

1.3.6 Changes in Return Values of API and Administrative Functions

The following table applies only to functions that have not been superseded.

Table 2 Changes in Return Values of API and Administrative Functions

Function	Return Value	Change Type
All administrative operations described in Chapter 5	SA_AIS_ERR_TIMEOUT SA_AIS_ERR_NO_MEMORY	new
SA_CLM_ADMIN_LOCK and SA_CLM_ADMIN_SHUTDOWN administrative operations	SA_AIS_OK SA_AIS_ERR_REPAIR_PENDING	changed
SA_CLM_ADMIN_SHUTDOWN administrative operation	SA_AIS_ERR_INTERRUPT	new
SA_CLM_ADMIN_UNLOCK administrative operation	SA_AIS_ERR_BAD_OPERATION	removed
saClmClusterTrackStop() saClmDispatch() saClmSelectionObjectGet()	SA_AIS_ERR_UNAVAILABLE	new
saClmClusterNodeGetAsync()	SA_AIS_ERR_NOT_EXIST	new
saClmClusterNodeGetAsync()	SA_AIS_ERR_UNAVAILABLE	changed

1.4 References

The following documents contain information that is relevant to the specification:

[1] Service Availability™ Forum, Service Availability Interface,Overview, SAI-Overview-B.05.01

[2] Service Availability™ Forum, Service Availability Interface, C Programming Model, SAI-AIS-CPROG-B.05.01

[3] Service Availability™ Forum, Application Interface Specification, Notification Service, SAI-AIS-NTF-A.03.01

[4] Service Availability™ Forum, Application Interface Specification, Information Model Management Service, SAI-AIS-IMM-A.03.01

[5] Service Availability™ Forum, Application Interface Specification, Platform Management Service, SAI-AIS-PLM-A.01.01

[6] Service Availability™ Forum, Hardware Platform Interface Specification, SAI-HPI-B.03.01

[7] Service Availability™ Forum, Application Interface Specification, Availability Management Framework, SAI-AIS-AMF-B.04.01

[8] Service Availability™ Forum, SA Forum Information Model in XML Metadata Interchange (XMI) v2.1 format, SAI-IM-XMI-A.04.01.xml.zip

[9] CCITT Recommendation X.731 | ISO/IEC 10164-2, State Management Function

[10] CCITT Recommendation X.733 | ISO/IEC 10164-4, Alarm Reporting Function

References to these documents are made by inserting the number of the document in brackets.

1.5 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 Web site (http://www.saforum.org).

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

1.6 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 SA Forum Web site (http://www.saforum.org).

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

1.7 Additional Information

1.7.1 Member Companies

A list of the Service Availability™ Forum member companies can be viewed online by using the links provided on the SA Forum Web site (http://www.saforum.org).

1.7.2 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 SA Forum Web site (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 that have been administratively configured in the cluster configuration (these nodes are also called cluster nodes or configured nodes), and it is core to any clustered system. A cluster consists of this set of configured nodes, each with a unique node name.

A member node is a configured node that the Cluster Membership Service has recognized to be healthy and well-connected to be used for deploying HA applications and services. The set of member nodes at a point in time is referred to as the cluster membership or simply as membership. The Cluster Membership Service is the authority that decides whether a configured node is transitioned to be a member node of the cluster. The Cluster Membership Service must ensure that only a single cluster is formed from the set of configured nodes. It is implementation-specific whether the Cluster Membership Service can still satisfy this requirement in the presence of interconnect failures.

The Cluster Membership Service also allows application processes to register callback functions to receive notifications of impending and completed membership changes as those changes occur. For the administrative removal of a member from the cluster membership or in the case of an impending removal due to the sequenced extraction of a FRU, the application can provide a response to the Cluster Membership Service to indicate whether it is ready for the removal or not.

The SA Forum does not specify precisely the criteria for membership of nodes in a cluster. These criteria will vary with different application domains. For example, in high data integrity applications, membership may require that each node in the cluster membership can communicate with every other member of the cluster (democratic model). For applications requiring very fast consensus on the membership in large clusters, an approach where membership is determined only by a small group of control nodes may be more appropriate (dictator model). In conjunction with the connectivity criteria for membership, the criteria a node must meet to be considered healthy by the Cluster Membership Service must also be considered. The configuration that describes the set of hardware elements that can affect the state of health of a member node, (that is, the general hardware dependency graph) is not catered for in the current Cluster Membership Service Information Model (see Chapter 4). It is up to the different Cluster Membership Service implementations to ascertain which hard-

ware elements (HE objects provided by the PLM Service, see [5]) need to be monitored for determining the state of health of the node. A Cluster Membership Service implementation can use the PLM entity group creation and tracking interfaces for this purpose.

3 Cluster Membership Service API

3.1 Cluster Membership Service Model

3.1.1 Definitions

A **cluster node** is the logical representation of all software running on a single operating system instance. A **cluster** comprises a set of cluster nodes. The cluster nodes collaborate to provide in a location transparent manner a set of the services defined by the SA Forum Application Interface Specifications. In the remainder of this document, the single instance of an operating system on which this software is running is referred to as the execution environment of the cluster node, The execution environment is a logical entity of the PLM Service (see [5]).

The set of nodes that compose the cluster and are therefore eligible to participate in this collaboration is configured for the Cluster Membership Service. A cluster node belonging to this set is termed a **configured node**.

The cluster nodes and the cluster are the logical entities that the Cluster Membership Service manipulates. They are represented in the Cluster Membership Service Information Model by objects of the UML classes defined in Chapter 4. The Cluster Membership Service Information Model is used to configure the nodes that are eligible to become members in the cluster.

A **member node** is a configured node that the Cluster Membership Service has recognized to be healthy and well-connected to be used for deploying HA applications and services. The set of member nodes at a point in time is referred to as the **cluster membership** or simply as **membership**. The Cluster Membership Service is the authority that decides whether a configured node is transitioned to be a member node of the cluster. This is explained further down.

The term **unconfigured node** is used in this document to designate an execution environment that is not configured to host a CLM node.

A **membership transition** is a change in the cluster membership involving one or more configured nodes or a change to an attribute of one or more member nodes.

Each membership transition has an associated **view number**. The view number increases with each membership transition, although not necessarily by one.
All clients of the Cluster Membership Service having requested membership information with the same parameters and being informed about a particular membership transition see the same view number. In addition, they obtain the same information

for a particular membership transition.
A view number is not affected by the addition or removal of configured nodes that are not currently member nodes.
Some implementations of the Cluster Membership Service may choose not to use the view number. In this case, the view number must be set to zero.

3.1.2 Scope

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.

When the Cluster Membership Service realizes the existence of a configured node that is not yet a member node, the Cluster Membership Service transitions the configured node to a member node if all the following conditions are met:

(1) The execution environment of the node is sufficient for programs to be executed on it (typically, an operating system and a minimal set of daemons must be up on the node).

(2) The node must provide the Cluster Membership Service API in accordance with the Cluster Membership Service.

(3) The node must be reachable by the Cluster Membership Service.

(4) The node is not prohibited administratively from being a member.

Any time one or more of the preceding four conditions are not satisfied for a given member node, the Cluster Membership Service will remove the node from the cluster membership. The Cluster Membership Service will initiate a cluster node reboot as a fencing and repair action if the value of the `saClmNodeDisableReboot` attribute of the node (defined in the `SaClmNode` configuration object class, see Section 4.4) is SA_FALSE.

If a node is administratively removed from the cluster membership, the reboot action is only performed when a subscribed process responds with the `SA_CLM_CALLBACK_RESPONSE_ERROR` result to the invocation of its `saClmClusterTrackCallback()` callback function (see Section 3.5.2), or if a subscribed process does not respond to this callback within a certain specified time, irrespective of the current value of the `saClmNodeDisableReboot` attribute.

The Cluster Membership Service provides no service on an unconfigured node.

If the Cluster Membership Service realizes the existence of an unconfigured node, it will not allow such a node to join the cluster and will not provide any membership information about the given cluster.

3.1.3 Interaction with the Platform Management Service

As shown in the UML diagram in Section 4.3, a PLM execution environment (abbreviated as EE, see [5]) can host at most one CLM node. This one-to-one relationship between a CLM node and its hosting PLM execution environment is administratively configured.

The Cluster Membership Service uses the PLM Service (see [5]) to obtain state information about the execution environment hosting each configured node and to manage this execution environment. A node cannot be a member node if the readiness state of its hosting PLM execution environment is out-of-service. The Cluster Membership Service may use other private means (for example heartbeat mechanisms or quorum-based algorithms) to evaluate the eligibility of the node to join the cluster and remain in it. The Cluster Membership Service uses the PLM administrative API to perform a node reboot as required.

In some cases, the PLM Service loses management capability for some of its entities, for instance, if connectivity to an out-of-band management module is no longer possible. The PLM Service reports this fact to its users by setting the `SA_PLM_RF_MANAGEMENT_LOST` readiness flag of the entity. The Cluster Membership Service may have additional information about the state of health of a node running on an PLM entity having the `SA_PLM_RF_MANAGEMENT_LOST` flag set. The Cluster Membership Service may decide to remove such a node from the cluster membership and carry out appropriate actions.

3.1.4 Communication with Other AIS Services

The Cluster Membership Service provides the following types of interfaces to other AIS Services and applications:

- **Cluster Node Get Interface**

  This interface enables users to retrieve information about nodes in the cluster membership.

- **Track Interface**

  The track interface enables users to subscribe and be notified when nodes leave or join the cluster membership.

  The PLM Service allows its users to validate the service impact of certain administrative and manual operations (for instance, board extraction or administrative lock), because these operations can affect several services. So, the operation can be rejected by upper layers. To propagate this PLM Service feature to its users, the Cluster Membership Service track interface provides a stepwise approach.

  The track interface provides four options for the aforementioned steps:

  - **Validate**: subscribed users are asked to agree with the node leaving the cluster membership.
  - **Start**: subscribed users should now terminate services. The node will soon leave the cluster membership.
  - **Completed**: the affected node has left the cluster membership.
  - **Aborted**: the operation was rejected during the validate step.

  The validate step of the Cluster Membership Service track interface enables CLM users (such as the Availability Management Framework) to check whether the service impact is acceptable (for instance, if the remaining redundancy is sufficient), before the Cluster Membership Service allows the PLM Service to start to deactivate the entities in question, which will eventually result in the removal of the associated nodes from the cluster membership. If any of the CLM users rejects the operation, the Cluster Membership Service also rejects the PLM Service request, which, in turn, will lead to the abortion of the operation.

  The start of the deactivation at the PLM level is mapped onto the start step of the Cluster Membership Service track interface, which enables CLM users to orderly withdraw from the affected node. When the responses are received from all invoked CLM users, the Cluster Membership Service informs the PLM Service about the completion of the operation.

  The completion of the operation at the PLM level is mapped onto the completed step of the Cluster Membership Service track interface. Users shall not respond to this callback. A CLM user may select to track only completed operations.

The start and completed steps are used in similar manner in the case of Cluster Membership Service administrative operations. If a node unexpectedly leaves the cluster membership, only the completed step is called. The validate and aborted steps are only used as a consequence of a PLM operation.

Subscribers must use the track flags to indicate whether they request to be notified in the start or validate step.

All these steps are defined in the `SaClmChangeStepT` type in Section 3.3.12. An operational scenario illustrating the interaction with the PLM Service and applications is provided in [5].

- **Notifications**

  The Cluster Membership Service issues notifications when nodes join or leave the cluster or get reconfigured.

- **Administrative Operations**

  The Cluster Membership Service provides administrative operations to manage the administrative state of nodes (see Section 5.2 on page 70).

3.1.5 Cluster Membership Functionality on Non-Member Cluster Nodes

3.1.5.1 Configured Non-Member Nodes

The Cluster Membership Service provides a limited set of functions on a configured non-member node. A client's valid handle of `SaClmHandleT` type remains valid after its node transitions from member to non-member node or after its node transitions from non-member to member node.

Track requests registered by a client also remain valid after its node transitions from member to non-member node, or after its node transitions from non-member to member node.

Library lifecycle functions provide the same functionality on member and non-member nodes.

The `saClmClusterTrack_4()`, `saClmClusterTrackStop()`, and `saClmClusterNotificationFree_4()` functions provide similar functionality when invoked on member nodes or on configured non-member nodes; however, the information returned by the tracking interface will be limited to the local node as long as the local node is not a part of the cluster membership (for details, see Section 3.5.1).

The functions `saClmClusterNodeGet_4()` and `saClmClusterNodeGetAsync()` will return with the `SA_AIS_ERR_UNAVAILABLE` error code if invoked by a client on a configured non-member node. The `SaClmClusterNodeGetCallbackT_4` function will set the `SA_AIS_ERR_UNAVAILABLE` error code in the `error` parameter if the node on which it runs is a configured non-member node.

3.1.5.2 Unconfigured Nodes

The Cluster Membership Service provides no functions on an unconfigured node.

When a client invokes `saClmInitialize_4()` on an unconfigured node, the `SA_AIS_ERR_UNAVAILABLE` error is returned.

For clients that already have a valid handle of `SaClmHandleT` type on a node that becomes unconfigured, all calls specifying that handle, except `saClmFinalize()`, will fail with `SA_AIS_ERR_UNAVAILABLE`, and no further callbacks will be invoked with that handle for the remainder of its lifecycle.

In some situations, such as interconnect failures, it may be impossible for the Cluster Membership Service to distinguish between a configured and an unconfigured non-member node. In such situations, all calls invoked by processes executing on such a node must behave as if the node is a configured non-member node.

If a management operation attempts to remove the node from the list of configured nodes, the Cluster Membership Service will reject this attempt, unless the node is administratively locked, and the SA Forum Information Model has no reference to that node in other objects. The Cluster Membership Service will not automatically finalize handles of users on that node, but it will behave as explained.

3.1.6 Effect of a Node Leaving the Cluster Membership on AIS Services

Certain AIS Services (such as the Availability Management Framework, the Checkpoint Service, and the Event Service) are offered only in a cluster-wide context. These services should not be available on a node after it has left the cluster membership. Certain other services (such as the Naming Service) support cluster-wide as well as node-local functionality. Such services should provide only node-local functionality after a node has left the cluster membership. Finally, certain services (such as the Timer Service) provide node-local functionality only; therefore, they are unaffected by a node leaving the cluster membership.

The Cluster Membership Service provides the track interface to alert client processes about a node leaving cluster membership: client processes can register with the Cluster Membership Service to track cluster membership changes and to take appropriate actions on receiving the information that a node is about to leave the cluster membership or has unexpectedly left the cluster membership.

The track interface should be used by AIS Service implementations to gracefully prepare for a node eviction from the cluster membership in response to an administrative request. The track interface with its start step provides also the possibility that the Cluster Membership Service waits for a response from all registered clients before it removes a node from membership. AIS Service implementations should also use the cluster membership tracking mechanism to determine when a node has left the cluster membership unexpectedly, so that they can carry out appropriate actions.

Note: Specific details of availability of AIS Services on a non-member cluster node is explained in the respective AIS Service specifications.

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 `SaClmHandleT` type is used for the handle to the Cluster Membership Service that a process acquires by invoking `saClmInitialize_4()` and uses in subsequent invocations of the functions of the Cluster Membership Service.

3.3.2 SaClmNodeIdT

```
typedef SaUint32T SaClmNodeIdT;
```

The `SaClmNodeIdT` type is used for the node identifier that is unique in the cluster.

```
#define SA_CLM_LOCAL_NODE_ID 0XFFFFFFFF
```

The `SA_CLM_LOCAL_NODE_ID` constant can be used in an API function of the Cluster Membership Service for the node identification (`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;
```

The `SaClmNodeAddressT` type provides a string representation of the communication address associated with a cluster 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_4

```
typedef struct {
     SaClmNodeIdT nodeId;
     SaClmNodeAddressT nodeAddress;
     SaNameT nodeName;
     SaNameT executionEnvironment;
     SaBoolT member;
     SaTimeT bootTimestamp;
     SaUint64T initialViewNumber;
} SaClmClusterNodeT_4;
```

The `SaClmClusterNodeT_4` type 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 `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 functionality.
- `executionEnvironment` - This is the DN of the PLM execution environment (EE) that hosts the node (see [5]).
- `member` - SA_TRUE if the node is a member, SA_FALSE if it is not.
- `bootTimestamp` - Timestamp at which the node was last booted.
- `initialViewNumber` - The **initial view number**, that is, the view number of the latest membership transition when this configured node joined the cluster and became a member. It is possible that two nodes have the same initial view numbers. However, it is required that if node `N1` is reported to service users as a member before node `N2`,
  `initialViewNumber` of `N1` < `initialViewNumber` of `N2`.
  Some implementations of the Cluster Membership Service may choose not to use the notion of view number. In this case, `initialViewNumber` must be set to zero.

When this structure is used to provide information about a node that has just left the cluster, the fields reflect the values last known to the Cluster Membership Service, except for the `member` field, which is set to SA_FALSE. If the structure is used to return the information about a local node that is not in the cluster membership, the following setting is provided:

- `nodeId` = SA_CLM_LOCAL_NODE_ID
- `member` = SA_FALSE
- `initialViewNumber` = 0

The other fields are set as appropriate.

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,
     SA_CLM_NODE_UNLOCK               = 5,
     SA_CLM_NODE_SHUTDOWN             = 6
} SaClmClusterChangesT;
```

The values of the `SaClmClusterChangesT` enumeration type refer to a configured cluster node. The cluster node may or may not be a member of the cluster. The values of the `SaClmClusterChangesT` enumeration type have the following interpretation:

- SA_CLM_NODE_NO_CHANGE - The membership has not changed for the configured node. This value is used when the `trackFlags` parameter of the `saClmClusterTrack_4()` function (as defined in Section 3.5.1 on page 42) is
  - either SA_TRACK_CURRENT or
  - SA_TRACK_CHANGES, and the configured node was already a member node that did not leave the membership, and none of its attributes have changed.
- SA_CLM_NODE_JOINED - The configured node joined the cluster membership.
- SA_CLM_NODE_LEFT - The member node left the cluster membership or is about to leave the cluster membership.

⇒ In the SA_CLM_CHANGE_VALIDATE, SA_CLM_CHANGE_START, or SA_CLM_CHANGE_ABORTED steps (see Section 3.3.12) of the saClmClusterTrack_4() function (see Section 3.5.1), the SA_CLM_NODE_LEFT value indicates that there was an eviction request on the node. Valid reasons for such an eviction request are for example:

- Administrative lock or shutdown operation on the node
- Request for physical extraction of hardware on which the node's execution environment is running
- Administrative lock or shutdown operation on the execution environment on which the node is running
- Administrative lock or shutdown operation on a hardware element on which the node's execution environment is running

The information provided for the node to be evicted is its current status as a member node.

⇒ In the SA_CLM_CHANGE_COMPLETED step, the SA_CLM_NODE_LEFT value indicates that the node is no longer in the cluster membership. The node may or may not still be a configured node.

- SA_CLM_NODE_RECONFIGURED - Some attributes associated with the member node have changed. Currently, only the nodeAddress attribute may change for a member node.
- SA_CLM_NODE_UNLOCK - An administrative unlock operation is issued on the node before a previous shut down administrative operation on the node has completed, that is, the unlock operation interrupted the shut-down operation before the node left the cluster membership.
- SA_CLM_NODE_SHUTDOWN - An administrative shutdown operation was issued on the node or on another entity in the system that affects the node. For this change, the SaClmClusterNodeGetCallbackT function is only invoked in the SA_CLM_CHANGE_START and SA_CLM_CHANGE_COMPLETED steps.

Note: Most values in the SaClmClusterChangesT enumeration use past tense in their names though they are also used in the SA_CLM_CHANGE_VALIDATE and SA_CLM_CHANGE_START steps of the track interface, that is, before the membership change actually occurs.

3.3.6 SaClmClusterNotificationT_4

```
typedef struct {
     SaClmClusterNodeT_4 clusterNode;
     SaClmClusterChangesT clusterChange;
} SaClmClusterNotificationT_4;
```

The fields of the `SaClmClusterNotificationT_4` type have the following interpretation:

- `clusterNode` - The information about a configured node that may or may not be a member of the cluster, as defined by the `SaClmClusterNodeT_4` structure in Section 3.3.4 on page 28.
- `clusterChange` - The changes in the cluster membership, as defined by the `SaClmClusterChangesT` enumeration type in Section 3.3.5 on page 29.

3.3.7 SaClmClusterNotificationBufferT_4

```
typedef struct {
     SaUint64T viewNumber;
     SaUint32T numberOfItems;
     SaClmClusterNotificationT_4 *notification;
} SaClmClusterNotificationBufferT_4;
```

The fields of the `SaClmClusterNotificationBufferT_4` type have the following interpretation:

- `viewNumber` - The current view number of the cluster membership. Some implementations may not use `viewNumber`. In this case, `viewNumber` must be set to zero.
- `numberOfItems` - The number of elements in the array to which `notification` points. Each element is of type `SaClmClusterNotificationT_4`.
- `notification` - A pointer to the notification array.

3.3.8 SaClmCallbacksT_4

A structure of the `SaClmCallbacksT_4` type (called a callbacks structure) is used to specify the callback functions that the Cluster Membership Service can invoke.

```
typedef struct {
     SaClmClusterNodeGetCallbackT_4
          saClmClusterNodeGetCallback;
     SaClmClusterTrackCallbackT_4
          saClmClusterTrackCallback;
} SaClmCallbacksT_4;
```

3.3.9 SaClmStateT

The `SaClmStateT` type holds Cluster Membership Service enumerations that are used in notifications to identify which state relating to a cluster node has changed (see also Chapter 6 and [3]).

```
typedef enum {
     SA_CLM_CLUSTER_CHANGE_STATUS          = 1,
     SA_CLM_ADMIN_STATE                    = 2
} SaClmStateT;
```

3.3.10 SaClmAdminOperationIdT

Enumeration of the administrative operations on a cluster node:

```
typedef enum {
     SA_CLM_ADMIN_UNLOCK              = 1,
     SA_CLM_ADMIN_LOCK                = 2,
     SA_CLM_ADMIN_SHUTDOWN            = 3
} SaClmAdminOperationIdT;
```

3.3.11 SaClmAdminStateT

Enumeration of the administrative states of a cluster node:

```
typedef enum {
     SA_CLM_ADMIN_UNLOCKED          = 1,
     SA_CLM_ADMIN_LOCKED            = 2,
     SA_CLM_ADMIN_SHUTTING_DOWN     = 3
} SaClmAdminStateT;
```

3.3.12 SaClmChangeStepT

```
typedef enum {
     SA_CLM_CHANGE_VALIDATE     = 1,
     SA_CLM_CHANGE_START        = 2,
     SA_CLM_CHANGE_ABORTED      = 3,
     SA_CLM_CHANGE_COMPLETED    = 4
} SaClmChangeStepT;
```

The `SaClmChangeStepT` type is used to indicate in which step of the cluster tracking process the track callback is invoked. The values of `SaClmChangeStepT` have the following interpretation:

- `SA_CLM_CHANGE_VALIDATE`
  The track callback is invoked to allow the called process to accept or reject the change.
- `SA_CLM_CHANGE_START`
  The change is occurring (it has not been rejected), and the track callback is invoked to let the client perform all necessary actions such that this change can occur with minimal service impact.
- `SA_CLM_CHANGE_ABORTED`
  A proposed change has been rejected.
- `SA_CLM_CHANGE_COMPLETED`
  A change in the cluster membership has occurred.

The `SA_CLM_CHANGE_VALIDATE`, `SA_CLM_CHANGE_START`, and `SA_CLM_CHANGE_ABORTED` steps are used in the `saClmClusterTrack_4()` function (as defined in Section 3.5.1 on page 42) only for nodes leaving the cluster membership, that is, in connection with a node being marked `SA_CLM_NODE_LEFT` in the array of type `SaClmClusterNotificationT_4`.

3.3.13 SaClmResponseT

The `SaClmResponseT` type is used in the `SaClmResponse_4()` function to provide a response to an `saClmTrackCallback()` callback call that was previously invoked with a step parameter equal to either `SA_CLM_CHANGE_VALIDATE` or `SA_CLM_CHANGE_START`.

```
typedef enum {
      SA_CLM_CALLBACK_RESPONSE_OK = 1,
      SA_CLM_CALLBACK_RESPONSE_REJECTED= 2,
      SA_CLM_CALLBACK_RESPONSE_ERROR = 3
} SaClmResponseT;
```

3.3.14 SaClmNotificationMinorIdT

```
typedef enum {
      SA_CLM_NTFID_NODE_JOIN           = 0x065,
      SA_CLM_NTFID_NODE_LEAVE          = 0x066,
      SA_CLM_NTFID_NODE_RECONFIG       = 0x067,
      SA_CLM_NTFID_NODE_ADMIN_STATE    = 0x068,
} SaClmNotificationMinorIdT;
```

3.3.15 SaClmAdditionalInfoIdT_4

The `SaClmAdditionalInfoIdT_4` enumeration is used in Cluster Membership Service notifications:

```
typedef enum {
      SA_CLM_ROOT_CAUSE_ENTITY = 1
} SaClmAdditionalInfoIdT_4;
```

3.3.16 Limit Enumeration

The Cluster Membership Service has no enumeration containing values that identify limit for a specific implementation of this service at the time of publication of this specification.

3.4 Library Life Cycle

3.4.1 saClmInitialize_4()

Prototype

```
SaAisErrorT saClmInitialize_4(
     SaClmHandleT *clmHandle,
     const SaClmCallbacksT_4 *clmCallbacks,
     SaVersionT *version
);
```

Parameters

`clmHandle` - [`out`] A pointer to the handle which designates this particular initialization of the Cluster Membership Service, and which is to be returned by the Cluster Membership Service. The `SaClmHandleT` type is defined in Section 3.3.1 on page 26.

`clmCallbacks` - [`in`] If `clmCallbacks` is set to NULL, no callback is registered; if `clmCallbacks` is not set to NULL, it is a pointer to an `SaClmCallbacksT_4` structure which contains the callback functions of the process that the Cluster Membership Service may invoke. Only non-NULL callback functions in this structure will be registered. The type `SaClmCallbacksT_4` is defined in Section 3.3.8 on page 32.

`version` - [`in/out`] As an input parameter, `version` is a pointer to a structure containing the required Cluster Membership Service version. In this case, `minorVersion` is ignored and should be set to 0x00.
As an output parameter, `version` is a pointer to a structure containing the version actually supported by the Cluster Membership Service. The `SaVersionT` type is defined in [2].

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 pointed to by `clmHandle` is returned by the Cluster Membership Service 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 version of the Cluster Membership Service API specified by the `releaseCode` and `majorVersion` fields of the structure pointed to by the `version` parameter, `SA_AIS_OK` is returned. In this case, the structure pointed to by 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 preceding condition cannot be met, `SA_AIS_ERR_VERSION` is returned, and the structure pointed to by 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 lowest 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 lower 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. Note that this value does not indicate that the function is called on a node currently in the cluster membership.

`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 did not.

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 - This value is returned if the version requested by the invoking process in the structure pointed to by the `version` parameter is not compatible with the version of the Cluster Membership Service implementation.

SA_AIS_ERR_UNAVAILABLE - The invoking process is executing on an unconfigured node.

See Also

`saClmSelectionObjectGet()`, `saClmDispatch()`, `saClmFinalize()`

3.4.2 saClmSelectionObjectGet()

Prototype

```
SaAisErrorT saClmSelectionObjectGet(
      SaClmHandleT clmHandle,
      SaSelectionObjectT *selectionObject
);
```

Parameters

`clmHandle` - [in] The handle which was obtained by a previous invocation of `saClmInitialize_4()` and which designates this particular initialization of the Cluster Membership Service. The `SaClmHandleT` type is defined in Section 3.3.1 on page 26.

`selectionObject` - [out] A pointer to the operating system handle that the invoking process can use to detect pending callbacks. The `SaSelectionObjectT` type is defined in [2].

Description

This function returns the operating system handle associated with the handle `clmHandle`. The invoking process can use the operating system handle to detect pending callbacks, instead of repeatedly invoking the `saClmDispatch()` function 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 operating system handle returned by `saClmSelectionObjectGet()` is valid until the `saClmFinalize()` function 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 did not.

`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).

`SA_AIS_ERR_UNAVAILABLE` - The invoking process is executing on an unconfigured node.

See Also

`saClmInitialize_4()`, `saClmDispatch()`, `saClmFinalize()`

3.4.3 saClmDispatch()

Prototype

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

Parameters

`clmHandle` - [in] The handle which was obtained by a previous invocation of `saClmInitialize_4()` and which designates this particular initialization of the Cluster Membership Service. The `SaClmHandleT` type is defined in Section 3.3.1 on page 26.

`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`. These flags are values of the `SaDispatchFlagsT` enumeration type, which is described in [2].

Description

In the context of the calling thread, this function invokes pending callbacks for the handle `clmHandle` in the way specified by the `dispatchFlags` parameter.

Return Values

`SA_AIS_OK` - The function completed successfully. This value is also returned if this function is being invoked with `dispatchFlags` set to `SA_DISPATCH_ALL` or `SA_DISPATCH_BLOCKING`, and the handle `clmHandle` has been finalized.

`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 did not.

`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.

SA_AIS_ERR_UNAVAILABLE - The invoking process is executing on an unconfigured node.

See Also

saClmInitialize_4(), saClmSelectionObjectGet(), saClmFinalize()

3.4.4 saClmFinalize()

Prototype

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

Parameters

clmHandle - [in] The handle which was obtained by a previous invocation of saClmInitialize_4() and which designates this particular initialization of the Cluster Membership Service. The SaClmHandleT type is defined in Section 3.3.1 on page 26.

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_4() before it invokes this function. A process must invoke this function once for each handle it acquired by invoking saClmInitialize_4().

If the saClmFinalize() function completes successfully, it releases all resources acquired when saClmInitialize_4() was called. Moreover, it stops any tracking associated with the particular handle and frees all resources allocated for it, including the memory allocated for the process in the saClmClusterTrack_4() function, if it has not yet been freed by saClmClusterNotificationFree_4().
The saClmFinalize() function 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.

If a process terminates, the Cluster Membership Service implicitly finalizes all instances of the Cluster Membership Service that are associated with the process, as described in the preceding paragraph.

After saClmFinalize() completes successfully, the handle clmHandle and the selection object associated with it are 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 did not.

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_4()`, `saClmClusterTrack_4()`, `saClmClusterNotificationFree_4()`, `SaClmClusterNodeGetCallbackT_4`, `SaClmClusterTrackCallbackT_4`, `saClmSelectionObjectGet()`

3.5 Cluster Membership Operations

3.5.1 saClmClusterTrack_4()

Prototype

```
SaAisErrorT saClmClusterTrack_4(
      SaClmHandleT clmHandle,
      SaUint8T trackFlags,
      SaClmClusterNotificationBufferT_4 *notificationBuffer
);
```

Parameters

`clmHandle` - [in] The handle which was obtained by a previous invocation of `saClmInitialize_4()` and which designates this particular initialization of the Cluster Membership Service. The `SaClmHandleT` type is defined in Section 3.3.1 on page 26.

`trackFlags` - [in] The kind of tracking that is requested, which is the bitwise OR of one or more of the following flags (as defined in [2]), which have the following interpretation here:

- `SA_TRACK_CURRENT` - Request the current membership status.
  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 the structure to which `notificationBuffer` points when the `saClmClusterTrack_4()` call completes successfully.
  The `clusterChanges` field in each entry of the array pointed to by the `notification` pointer in the structure referred to by `notificationBuffer` is set to `SA_CLM_NODE_NO_CHANGE`.
- `SA_TRACK_CHANGES` - Start the cluster membership tracking, requesting each time a complete picture of all member nodes.
  The `saClmClusterTrackCallback()` callback function is invoked each time the cluster membership or an attribute of a member node changes. The structure to which `notificationBuffer` points in an invocation of the `saClmClusterTrackCallback()` function contains information about all cluster nodes that are currently members of the cluster, and also about cluster nodes that have left the cluster membership since the last invocation of `saClmClusterTrackCallback()`.

- SA_TRACK_CHANGES_ONLY - Start the cluster membership tracking, requesting each time only a list of nodes with changes.
  The saClmClusterTrackCallback() callback function is invoked each time the cluster membership or an attribute of a member node changes. The structure to which notificationBuffer points in an invocation of the saClmClusterTrackCallback() function contains information about new member nodes, member nodes whose attributes have changed, and cluster nodes that have left the cluster membership since the last invocation of saClmClusterTrackCallback().
- SA_TRACK_LOCAL - Request information only about the local node.
  If this flag is specified together with the SA_TRACK_CURRENT flag, only the membership status of the node on which the process is executing is returned. If notificationBuffer is NULL, the information is returned by a single subsequent invocation of the saClmClusterTrackCallback() notification callback; otherwise, this information is returned in the structure to which notificationBuffer points when the saClmClusterTrack_4() call completes successfully. The array pointed to by the notification pointer in the structure referred to by notificationBuffer always has one entry, and the clusterChanges field of this entry is set either to SA_CLM_NODE_NO_CHANGE, if the local node is in the cluster membership or to SA_CLM_NODE_LEFT, if the node is not in the cluster membership.
- SA_TRACK_START_STEP - Request additionally the SA_CLM_CHANGE_START step if a node is about to leave the cluster membership.
  This flag is ignored, if neither SA_TRACK_CHANGES nor SA_TRACK_CHANGES_ONLY is set.
- SA_TRACK_VALIDATE_STEP - Request additionally the SA_CLM_CHANGE_VALIDATE step if a node is about to leave the cluster membership.
  This flag is ignored, if neither SA_TRACK_CHANGES nor SA_TRACK_CHANGES_ONLY is set.

If both SA_TRACK_CHANGES and SA_TRACK_CHANGES_ONLY are set in an invocation of this function, the function returns SA_AIS_ERR_BAD_FLAGS, and tracking is not started. An invocation of this function is also invalid and returns SA_AIS_ERR_BAD_FLAGS if none of the flags SA_TRACK_CHANGES, SA_TRACK_CHANGES_ONLY, or SA_TRACK_CURRENT are set.
The SaUint8T type is defined in [2].

Note: If `saClmClusterTrack_4()` is invoked on non-member nodes, the following applies:

- if SA_TRACK_CURRENT is specified, only information about the local node is returned in the structure pointed to by `notificationBuffer` or in the subsequent callback;
- if SA_TRACK_CHANGES or SA_TRACK_CHANGES_ONLY is specified, callbacks will only be invoked when the node joins the cluster membership.

Note: If a process starts tracking with SA_TRACK_CHANGES_ONLY, it should also have a complete view of the cluster membership against which to compare the incremental information. Therefore, it is recommended to start the incremental tracking by ORing both flags SA_TRACK_CURRENT and SA_TRACK_CHANGES_ONLY.
If a process needs to be warned when a request to evict its own local node is issued, it is recommended to start tracking with the flags SA_TRACK_CHANGES_ONLY, SA_TRACK_LOCAL, and SA_TRACK_START_STEP set.

`notificationBuffer` - [in/out] - A pointer to a structure of type `SaClmClusterNotificationBufferT_4`, as defined in Section 3.3.7 on page 31. This parameter is ignored if SA_TRACK_CURRENT is not set in `trackFlags`; otherwise, if `notificationBuffer` is not NULL, this structure will contain information about all member nodes when `saClmClusterTrack_4()` returns. The meaning of the fields of the `SaClmClusterNotificationBufferT_4` structure is:

- `viewNumber` - [out] The current view number of the cluster membership. Some implementations may not use the `viewNumber`. In this case, `viewNumber` must be set to zero for all invocations of this callback.
  If an implementation supports `viewNumber`, the following applies:
  - if SA_TRACK_CURRENT is specified, the view number of the most recent transition is delivered.
  - If the callback is invoked on a non-member node, therefore providing information only about the local node, `viewNumber` must be set to zero.
- `numberOfItems` - [in/out] If `notification` is NULL, `numberOfItems` is ignored as input parameter; otherwise, it specifies that the array to which `notification` points provides memory for information about `numberOfItems` nodes in the cluster membership.
  When `saClmClusterTrack_4()` 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 library. The caller is responsible for freeing the allocated memory by calling the `saClmClusterNotificationFree_4()` function.

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 member node, depending on the value of the `trackFlags` parameter, as described above. These changes are notified by invoking the `saClmClusterTrackCallback()` callback function, which must have been supplied when the process invoked `saClmInitialize_4()`.

While a client process on a node that is not a cluster member may also use this function, the client process can receive information consistent with the semantics of the kind of tracking requested but limited to the node hosting the process as long as this node does not itself join the cluster membership.

A process may call `saClmClusterTrack_4()` 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 a process has enabled tracking by calling `saClmClusterTrack_4()` with either `SA_TRACK_CHANGES` or `SA_TRACK_CHANGES_ONLY` set and then calls `saClmClusterTrack_4()` again with the same value of `clmHandle`, the following applies, depending on the flags in the second call:

- If the second call has either `SA_TRACK_CHANGES` or `SA_TRACK_CHANGES_ONLY` set, the new combination of flags is used to change the settings for the tracking. For example, if the first call had `SA_TRACK_START_STEP` set, and the second call does not have this flag set, the process will not receive further callbacks for the `SA_TRACK_START_STEP`.
- If the second call has neither `SA_TRACK_CHANGES` nor `SA_TRACK_CHANGES_ONLY` set, but rather only `SA_TRACK_CURRENT` or `SA_TRACK_CURRENT` and `SA_TRACK_LOCAL`, the tracking started by the first call will proceed unchanged, and the process will additionally receive the current membership.
  `SA_TRACK_START_STEP` and `SA_TRACK_VALIDATE_STEP` are ignored in this case.

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 did not.

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 invocation of `saClmInitialize_4()` to initialize the Cluster Membership Service was incomplete, since the `saClmClusterTrackCallback()` callback function is missing. This value is not returned if only the SA_TRACK_CURRENT flag is set in `trackFlags` and the `notificationBuffer` is not NULL.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly. In particular, this applies if in the structure to which `notificationBuffer` points the `notification` pointer 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` pointer in the structure referred to by `notificationBuffer` is not NULL, but the value of `numberOfItems` in this structure is smaller than the number of entries to be provided in the array referred to by the `notification` pointer.

SA_AIS_ERR_BAD_FLAGS – The `trackFlags` parameter is invalid. In particular, this applies if

- the SA_TRACK_CHANGES and SA_TRACK_CHANGES_ONLY flags are both specified or if
- none of the flags SA_TRACK_CHANGES, SA_TRACK_CHANGES_ONLY, or SA_TRACK_CURRENT are set.

SA_AIS_ERR_VERSION - The invoked function is not supported in the version specified in the call to initialize this instance of the Cluster Membership Service library.

SA_AIS_ERR_UNAVAILABLE - The invoking process is executing on an unconfigured node.

See Also

saClmClusterTrackStop(), SaClmClusterTrackCallbackT_4, saClmClusterNotificationFree_4(), saClmInitialize_4(), saClmFinalize()

3.5.2 SaClmClusterTrackCallbackT_4

Prototype

```
typedef void (*SaClmClusterTrackCallbackT_4) (
      const SaClmClusterNotificationBufferT_4 *notificationBuffer,
      SaUint32T numberOfMembers,
      SaInvocationT invocation,
      const SaNameT *rootCauseEntity,
      const SaNtfCorrelationIdsT *correlationIds,
      SaClmChangeStepT step,
      SaTimeT timeSupervision,
      SaAisErrorT error
);
```

Parameters

notificationBuffer - [in] A pointer to a structure of type SaClmClusterNotificationBufferT_4, as defined in Section 3.3.7 on page 31. The array pointed to by the notification pointer in this structure contains entries for current member nodes or changes in the cluster membership. It is possible that there is a change in the view number and no change in the cluster membership. In this case, the most recent view number of the most recent transition is returned. The provided information is as follows:

⇒ If the callback provides the information requested by the SA_TRACK_CURRENT flag (the `notificationBuffer` pointer was NULL in the corresponding `saClmClusterTrack_4()` call), the `clusterChanges` field in the entries of the array to which the `notification` pointer refers is set to SA_CLM_NODE_NO_CHANGE.
Only if the local node is not in the cluster membership, is the `clusterChanges` field set differently, namely to SA_CLM_NODE_LEFT. In this case, the array to which the `notification` pointer refers contains information only for the local node.

⇒ In other cases, the value of the `clusterChanges` field in the entries of the array pointed to by the `notification` pointer is as follows:

- SA_CLM_NODE_NO_CHANGE - There is no change reported on this node. This value is only possible if SA_TRACK_CHANGES was specified.
- SA_CLM_NODE_JOINED - The node has newly joined the cluster membership. This is only reported with the SA_CLM_CHANGE_COMPLETED step.
- SA_CLM_NODE_LEFT - The member node left the cluster membership or is about to leave the cluster membership.

  ⇒ In the SA_CLM_CHANGE_VALIDATE, SA_CLM_CHANGE_START, or SA_CLM_CHANGE_ABORTED steps (see Section 3.3.12), this value indicates that there was an eviction request on the node. Valid reasons for such an eviction request are for example:

  - Administrative lock or shutdown operation on the node
  - Request for physical extraction of hardware on which the node's execution environment is running
  - Administrative lock or shutdown operation on the execution environment on which the node is running
  - Administrative lock or shutdown operation on a hardware element on which the node's execution environment is running

  The information provided for the node to be evicted is its current status as a member node.

  ⇒ In the SA_CLM_CHANGE_COMPLETED step, this value indicates that the node is no longer in the cluster membership. The node may or may not still be a configured node.
- SA_CLM_NODE_RECONFIGURED - There was a configuration change on the node (an attribute of the `SaClmNode` configuration object class, see Section 4.4, has changed). This is only reported with the SA_CLM_CHANGE_COMPLETED step.

- SA_CLM_NODE_UNLOCK - An administrative unlock operation is issued on the node before a previous shut down administrative operation on the node has completed, that is, the unlock operation interrupted the shutdown operation before the node had left the cluster membership.
- SA_CLM_NODE_SHUTDOWN - An administrative shutdown operation was issued on the node or on another entity in the system that affects the node. For this change, the SaClmClusterNodeGetCallbackT function is only invoked in the SA_CLM_CHANGE_START and SA_CLM_CHANGE_COMPLETED steps.

numberOfMembers - [in] The current number of members in the cluster membership. The SaUint32T type is defined in [2].

invocation – [in] This parameter identifies a particular invocation of the callback function. The invoked process returns invocation when it responds to the Cluster Membership Service by calling the saClmResponse_4( ) function. The SaInvocationT type is defined in [2].

rootCauseEntity – [in] A pointer to the DN of the PLM entity or CLM node directly targeted by the action or event that caused the membership change. The SaNameT type is defined in [2].

correlationIds – [in] A pointer to the correlation identifiers associated with the root cause. For the meaning of correlation identifiers, refer to [3]. The user of the track interface should include these correlation identifiers in subsequent notifications to allow root cause analysis and notification correlation. The SaNtfCorrelationIdsT type is defined in [3].

step – [in] Indicates the tracking step in which the callback is invoked. The SaClmChangeStepT type is defined in Section 3.3.12 on page 33.

timeSupervision - [in] This parameter specifies the time for how long the Cluster Membership Service waits for the process to provide the response for the callback by invoking the saClmResponse_4( ) function. The time is specified in nanoseconds. If this parameter is set to SA_TIME_UNKNOWN, no time supervision is initiated. The Cluster Membership Service sets the timeSupervision parameter as follows:

- if the SaClmClusterTrackCallbackT function is called in the SA_CLM_CHANGE_START step and indicates an administrative lock command on a cluster node, the timeSupervision parameter is set to the saClmNodeLockCallbackTimeout attribute of the node affected by the lock operation; in all other cases for the SA_CLM_CHANGE_START step, the timeSupervision parameter is set to SA_TIME_UNKNOWN;

- if the SaClmClusterTrackCallbackT function is called in the SA_CLM_CHANGE_VALIDATE step, the timeSupervision parameter is set to SA_TIME_UNKNOWN;
- if the SaClmClusterTrackCallbackT function is called in the SA_CLM_CHANGE_COMPLETED or SA_CLM_CHANGE_ABORTED steps, the timeSupervision parameter is set to zero, as no response is expected before the Cluster Membership Service continues its processing.

The SaTimeT type is defined in [2].

error - [in] This parameter indicates whether the Cluster Membership Service was able to perform the requested operation. The SaAisErrorT type is defined in [2]. 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 did not.
- SA_AIS_ERR_BAD_HANDLE - The handle clmHandle that was passed to the corresponding saClmClusterTrack_4() function has become invalid, as it is now 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. The process that invoked saClmClusterTrack_4() 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_4() might have missed one or more notifications.
- SA_AIS_ERR_UNAVAILABLE - The invoking process is executing on an unconfigured node.

If the error returned is SA_AIS_ERR_NO_MEMORY or SA_AIS_ERR_NO_RESOURCES, the process that invoked saClmClusterTrack_4() should invoke saClmClusterTrackStop(). It may then invoke saClmClusterTrack_4() again.

Description

This callback is invoked in the context of a thread calling `saClmDispatch()` with the handle `clmHandle` that was specified when the process requested tracking of the cluster membership by invoking the `saClmClusterTrack_4()` call.

The `saClmClusterTrackCallback()` function returns information about cluster membership changes in the structure to which the `notificationBuffer` parameter points. The kind of information returned depends on the setting of the `trackFlags` parameter of the `saClmClusterTrack_4()` function.

If the callback is invoked on a process hosted on a node that is currently not in the cluster membership, the callback can return information only about that node.

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

The value of the `numberOfItems` attribute in the structure to which the `notificationBuffer` parameter points might be greater than the value of the `numberOfMembers` parameter if either the `SA_TRACK_CHANGES` flag or the `SA_TRACK_CHANGES_ONLY` flags is set, and one or more member nodes have left the cluster membership. In this case, the structure to which the `notificationBuffer` parameter points 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_4()`, `saClmClusterTrackStop()`, `saClmDispatch()`

3.5.3 saClmClusterTrackStop()

Prototype

```
SaAisErrorT saClmClusterTrackStop(
      SaClmHandleT clmHandle
);
```

Parameters

`clmHandle` - [in] The handle which was obtained by a previous invocation of `saClmInitialize_4()` and which designates this particular initialization of the Cluster Membership Service. The `SaClmHandleT` type is defined in Section 3.3.1 on page 26.

Description

This function stops any further notifications of cluster membership changes which were requested by specifying the handle `clmHandle` in an invocation of the `saClmClusterTrack_4()` function, and this request is still in effect. 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 did not.

`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 by invoking the `saClmClusterTrack_4()` function with track flags `SA_TRACK_CHANGES` or `SA_TRACK_CHANGES_ONLY` that would still be in effect.

SA_AIS_ERR_UNAVAILABLE - The invoking process is executing on an unconfigured node.

See Also

saClmClusterTrack_4(), SaClmClusterTrackCallbackT_4, saClmInitialize_4()

3.5.4 saClmClusterNotificationFree_4()

Prototype

```
SaAisErrorT saClmClusterNotificationFree_4(
      SaClmHandleT clmHandle,
      SaClmClusterNotificationT_4 *notification
);
```

Parameters

clmHandle - [in] The handle which was obtained by a previous invocation of saClmInitialize_4() and which designates this particular initialization of the Cluster Membership Service. The SaClmHandleT type is defined in Section 3.3.1 on page 26.

notification - [in] A pointer to the memory that was allocated by the Cluster Membership Service library in the saClmClusterTrack_4() function and is to be deallocated. The SaClmClusterNotificationT_4 type is defined in Section 3.3.6 on page 31.

Description

This function frees the memory to which notification points and which was allocated by the Cluster Membership Service library in a previous call to the saClmClusterTrack_4() function.

For details, refer to the description of the notification pointer in the structure referred to by the notificationBuffer parameter in the corresponding invocation of the saClmClusterTrack_4() function.

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_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_VERSION - The invoked function is not supported in the version specified in the call to initialize this instance of the Cluster Membership Service library.

SA_AIS_ERR_UNAVAILABLE - The invoking process is executing on an unconfigured node.

See Also

saClmClusterTrack_4()

3.5.5 saClmClusterNodeGet_4()

Prototype

```
SaAisErrorT saClmClusterNodeGet_4(
      SaClmHandleT clmHandle,
      SaClmNodeIdT nodeId,
      SaTimeT timeout,
      SaClmClusterNodeT_4 *clusterNode
);
```

Parameters

clmHandle - [in] The handle which was obtained by a previous invocation of saClmInitialize_4() and which designates this particular initialization of the Cluster Membership Service. The SaClmHandleT type is defined in Section 3.3.1 on page 26.

nodeId - [in] The node identifier of the member node for which the clusterNode information structure is to be retrieved. The SaClmNodeIdT type is defined in Section 3.3.2 on page 26.

timeout - [in] The saClmClusterNodeGet_4() invocation is considered to have failed if it does not complete by the time specified. The SaTimeT type is defined in [2].

`clusterNode` - [`out`] A pointer to a cluster node structure that contains information about a member node. The invoking process provides space for this structure, and the Cluster Membership Service fills in the fields of this structure. The `SaClmClusterNodeT_4` type is defined in Section 3.3.4 on page 28.

Description

This function provides the means for synchronously retrieving information about a member node identified by the `nodeId` parameter. The member node information is returned in the structure to which `clusterNode` points.

By invoking this function, a process can obtain the member node information for the node designated by `nodeId`.

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. The node specified in the `nodeId` parameter is in the cluster membership.

`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 did not.

`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).

`SA_AIS_ERR_NOT_EXIST` - There is no node in the cluster membership with the identifier specified in the `nodeId` parameter.

`SA_AIS_ERR_VERSION` - The invoked function is not supported in the version specified in the call to initialize this instance of the Cluster Membership Service library.

SA_AIS_ERR_UNAVAILABLE - The invoking process is not executing on a member node.

See Also

saClmClusterNodeGetAsync(), saClmInitialize_4()

3.5.6 saClmClusterNodeGetAsync()

Prototype

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

Parameters

clmHandle - [in] The handle which was obtained by a previous invocation of saClmInitialize_4() and which designates this particular initialization of the Cluster Membership Service. The SaClmHandleT type is defined in Section 3.3.1 on page 26.

invocation - [in] This parameter allows the invoking process to match this invocation of saClmClusterNodeGetAsync() with the corresponding saClmClusterNodeGetCallback(). The SaInvocationT type is defined in [2].

nodeId - [in] The node identifier of the member node for which information is to be retrieved. The SaClmNodeIdT type is defined in Section 3.3.2 on page 26.

Description

This function requests information to be provided asynchronously about the particular member node identified by the nodeId parameter. If SA_CLM_LOCAL_NODE_ID is used as nodeId, the function returns information about the member 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 saClmInitialize_4().

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 did not.

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).

SA_AIS_ERR_NOT_EXIST - There is no node in the cluster membership with the identifier specified in the `nodeId` parameter.

SA_AIS_ERR_INIT - The previous invocation of `saClmInitialize_4()` to initialize the Cluster Membership Service was incomplete, since the `saClmClusterNodeGetCallback()` callback function is missing.

SA_AIS_ERR_UNAVAILABLE - The invoking process is not executing on a member node.

See Also

`saClmClusterNodeGet_4()`, `SaClmClusterNodeGetCallbackT_4`, `saClmInitialize_4()`

3.5.7 SaClmClusterNodeGetCallbackT_4

Prototype

```
typedef void (*SaClmClusterNodeGetCallbackT_4)(
      SaInvocationT invocation,
      const SaClmClusterNodeT_4 *clusterNode,
      SaAisErrorT error
);
```

Parameters

`invocation` - [in] This parameter makes the association between an invocation of the `saClmClusterNodeGetCallback()` function by the Cluster Membership Service and an invocation of the `saClmClusterNodeGetAsync()` function by a process. The `SaInvocationT` type is defined in [2].

`clusterNode` - [in] A pointer to a structure that contains information about the member node identified by `nodeId` in the invocation of the `saClmClusterNodeGetAsync()` function. The `SaClmClusterNodeT_4` type is defined in Section 3.3.4 on page 28.

`error` - [in] This parameter indicates whether the `saClmClusterNodeGetAsync()` function was successful. The `SaAisErrorT` type is defined in [2]. The possible return values are:

- `SA_AIS_OK` - The function completed successfully. The node specified in the `nodeId` parameter of the associated `saClmClusterNodeGetAsync()` invocation is in the cluster membership.
- `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 did not.
- `SA_AIS_ERR_BAD_HANDLE` - The handle `clmHandle` that was passed to the corresponding `saClmClusterNodeGetAsync()` function is invalid, since it is corrupted, uninitialized, or has already been finalized.
- `SA_AIS_ERR_INVALID_PARAM` - A parameter was not set correctly in the corresponding invocation of the `saClmClusterNodeGetAsync()` function.
- `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 - There is no node in the cluster membership with the identifier specified in the `nodeId` parameter in the corresponding invocation of the `saClmClusterNodeGetAsync()` function.
- SA_AIS_ERR_UNAVAILABLE - The invoking process is not executing on a member node.

Description

The Cluster Membership Service invokes this callback function when the operation requested by the invocation of `saClmClusterNodeGetAsync()` completes.

This callback is invoked in the context of a thread calling `saClmDispatch()` with the handle `clmHandle` that was specified in the `saClmClusterNodeGetAsync()` call.

If successful, this callback function returns in the structure to which `clusterNode` points information about a particular member node identified by the `nodeId` parameter in the corresponding 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.

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

Return Values

None

See Also

`saClmClusterNodeGetAsync()`, `saClmDispatch()`

3.5.8 saClmResponse_4()

Prototype

```
SaAisErrorT saClmResponse_4(
      SaClmHandleT clmHandle,
      SaInvocationT invocation,
      SaClmResponseT response
);
```

Parameters

`clmHandle` - [in] The handle which was obtained by a previous invocation of `saClmInitialize_4()` and which designates this particular initialization of the Cluster Membership Service. The `SaClmHandleT` type is defined in Section 3.3.1 on page 26.

`invocation` - [in] Used to match this invocation of `saClmResponse_4()` with the previous corresponding invocation of the `saClmTrackCallback()` callback. The `SaInvocationT` type is defined in [2].

`response` - [in] Result of the execution of the callback function (the `SaClmResponseT` type is defined in Section 3.3.13).

Description

This function is used by a process to provide a response to the `saClmTrackCallback()` callback call that was previously invoked with a `step` parameter equal to either `SA_CLM_CHANGE_VALIDATE` or `SA_CLM_CHANGE_START`.

The `invocation` parameter must be set to the value passed in the invocation parameter of the track callback. The `response` parameter holds the response of the process.

The response parameter can be set to `SA_CLM_CALLBACK_RESPONSE_REJECTED` only for a response to the `saClmTrackCallback()` callback in the `SA_CLM_CHANGE_VALIDATE` step.

This function can only be called by the process for which its `saClmTrackCallback()` callback has been invoked.

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 time-out occurred before the call could complete. It is unspecified whether the call succeeded or whether it did not.

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. In particular, this value is returned if the `invocation` parameter is invalid or identifies an invocation of the `saClmTrackCallback()` function with a `step` parameter that does not require a response.

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 invoked function is not supported in the version specified in the call to initialize this instance of the Cluster Membership Service library.

SA_AIS_ERR_UNAVAILABLE - The invoking process is not executing on a member node.

See Also

`saClmInitialize_4(),SaClmTrackCallbackT_4`

4 Cluster Membership Service UML Information Model

The Cluster Membership Service Information Model is described in UML and has been organized in UML class diagrams.

The Cluster Membership Service Information Model is implemented by the SA Forum IMM Service ([4]). For further details on this implementation, refer to the SA Forum Overview document ([1]).

The classes in the Cluster Membership Service UML class diagrams show the contained attributes and their type, multiplicity, default values, and constraints. The description of each attribute is provided in the SA Forum XMI document (see [8]). The class diagrams additionally show the administrative operations (if any) applicable on these classes.

The UML diagrams defined for the Cluster Membership Service are:

- Cluster View
- Cluster Membership Service UML Classes

These diagrams will be described starting with Section 4.3.

4.1 Notes on the Conventions Used in UML Diagrams

A general explanation of the conventions used in the UML diagrams, such as the use of constraints, default values, and the like is presented in [1].

4.2 DN for Cluster Membership Service Classes

Table 3 provides the format of the various DNs used to name Cluster Membership Service objects of the SA Forum Information Model. One format is defined for each object class.

Table 3 DN Formats for Objects of Cluster Membership Service Classes

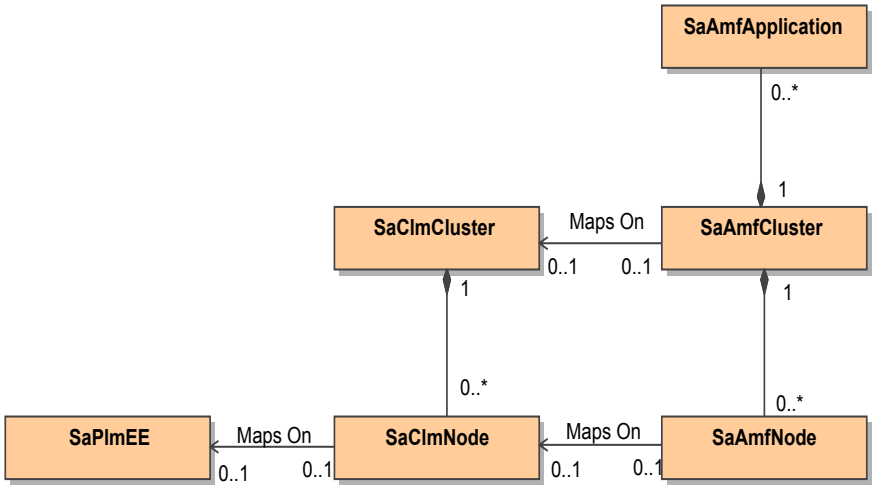
Object Class	DN Formats for Objects of the Class
`SaClmCluster`	`"safCluster=..."`
`SaClmNode`	`"safNode=..., safCluster=..."`

4.3 Cluster Membership Service Classes and Other Services' Classes

The overview diagram presented in FIGURE 1 shows the relationships among the classes of the Cluster Membership Service and some classes of the Platform Management Service and of the Availability Management Framework.

Attributes and operations of the Cluster Membership Service classes `SaClmCluster` and `SaClmNode` are shown in Section 4.4.
The `SaPlmEE` class is described in [5], the classes `SaAmfCluster`, `SaAmfNode`, and `SaAmfApplication` are described in [7].

FIGURE 1 Cluster View



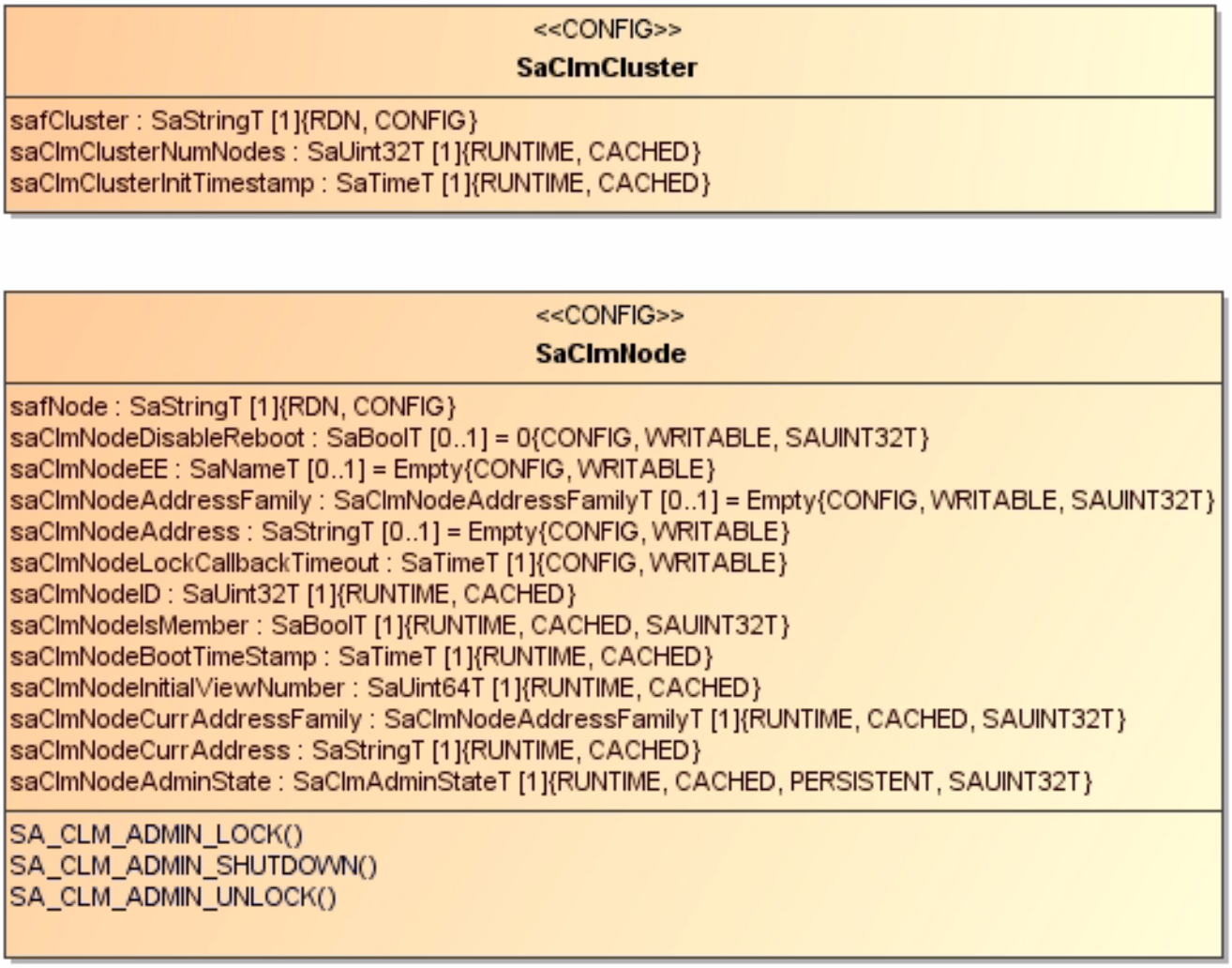
4.4 Cluster Membership Service Classes

The classes of the Cluster Membership Service UML model are:

- `SaClmCluster` —This is a configuration object class that enables the configuration of various attributes of a cluster. In addition, this class also exposes certain runtime attributes that are applicable for the cluster.
- `SaClmNode` — This is a configuration object class that enables the configuration of various attributes of a cluster node. In addition, this class also exposes certain runtime attributes that are applicable for cluster nodes.

FIGURE 2 shows these classes.

FIGURE 2 Cluster Membership Service UML Classes



5 Cluster Membership Service Administration API

This section describes the administrative states and API functions that the IMM Service exposes on behalf of the Cluster Membership Service. The main clients of the administrative API are system management applications such as SNMP agents or CIM providers, which will typically convert system administration commands (invoked from a management station) to the correct administrative API sequence.

5.1 Cluster Node Administrative States and Operations

5.1.1 Administrative States

The administrative state of a cluster node follows the ITU X.731 state management model ([9]). It is set by the system administrator and is persistent over cluster reboots. Valid values for the administrative state of a cluster node are:

- **unlocked**: The cluster node has not been administratively prohibited from being a cluster member.
- **locked**: The cluster node has been administratively prohibited from being a cluster member.
- **shutting-down**: The cluster node has been administratively requested to gracefully transition to the locked state. Client processes of the Cluster Membership Service on a cluster node in the shutting-down state are expected to follow ITU X.731 (see [9]) shutdown semantics.

The following table shows possible cluster membership status for different administrative states of a cluster node.

Table 4 Membership Status of a Cluster Node for Different Administrative States

Administrative State	Cluster Membership Status
unlocked	member or not a member
locked	not a member
shutting-down	member

5.1.2 Administrative Operations

The administrative operations defined on a cluster node to manipulate the administrative state are:

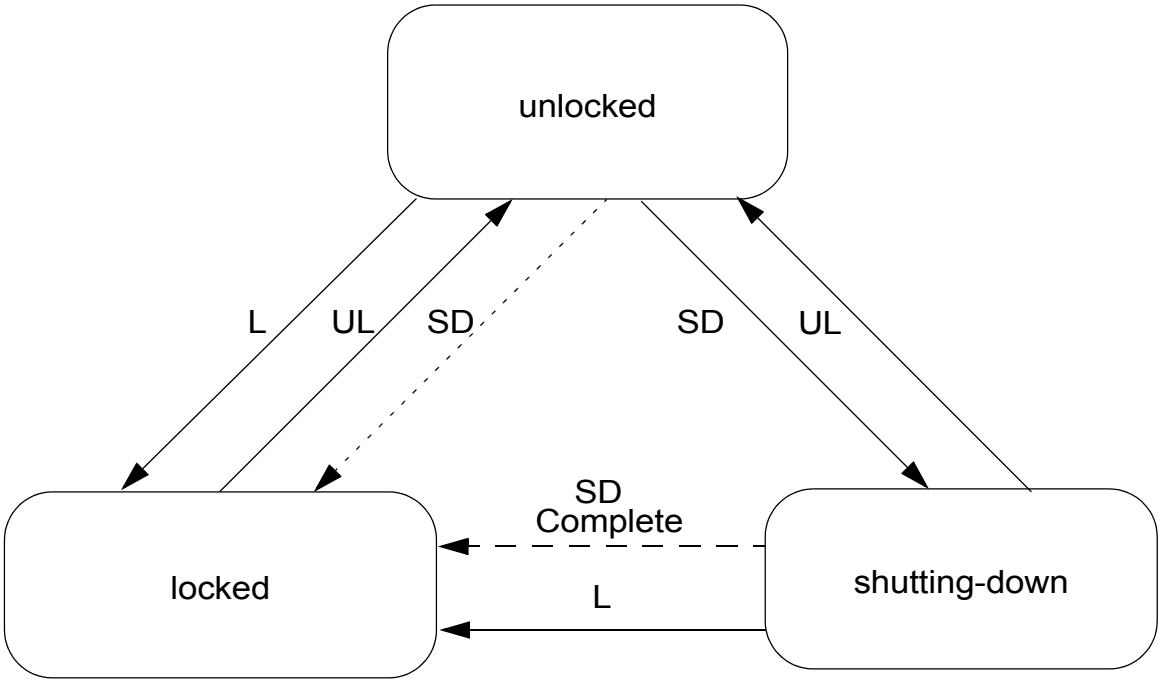
- SA_CLM_ADMIN_LOCK
- SA_CLM_ADMIN_UNLOCK
- SA_CLM_ADMIN_SHUTDOWN

The figure below shows the effect of these administrative operations on the administrative state and uses the following abbreviated form for the administrative operations:

- UL = SA_CLM_ADMIN_UNLOCK
- L = SA_CLM_ADMIN_LOCK
- SD = SA_CLM_ADMIN_SHUTDOWN

The dashed line in the figure represents the internal (spontaneous) transition corresponding to the completion of the shutting down operation, which transitions the node into locked state without further external intervention. The dotted line applies only to a non-member cluster node and shows that the SA_CLM_ADMIN_SHUTDOWN administrative operation directly transitions the administrative state of the cluster node from unlocked to locked.

FIGURE 3 Administrative States and Related Operations for Cluster Membership Service Nodes



5.1.3 Implications for AIS Services

After the administrative state of a cluster node is set to locked, it is permissible to reboot the execution environment of the cluster node abruptly. Some of the triggers for a reboot can be:

- diagnostics running on the locked node
- direct administrator interaction with the execution environment

Rebooting the execution environment means that all processes on the locked cluster node, including those processes implementing AIS Services, can be abruptly terminated.

AIS Services need to take all necessary actions on a node as part of `SA_CLM_ADMIN_LOCK` and `SA_CLM_ADMIN_SHUTDOWN` cluster node administrative operations to ensure that an abrupt reboot of the locked node does not adversely affect the service that these AIS Services provide or manage on the remaining member nodes. More details specific to each AIS Service may be found in the respective specifications.

5.2 Administrative Operations on a Cluster Node

Administrative operations on cluster nodes can be carried out using the IMM Service API functions `saImmOmAdminOperationInvoke_3()` or `saImmOmAdminOperationInvokeAsync_3()` on Cluster Membership Service node objects for which the Cluster Membership Service is the object implementer (runtime owner). The appropriate `operationId` and `objectName` parameters need to be specified to execute a particular administrative operation on a specified Cluster Membership Service node.
Return values are passed in the `operationReturnValue` parameter.
These IMM Service functions are described in the SA Forum IMM Service Specification (see [4]).

The administrative operations on Cluster Membership Service nodes are described referring to the values `operationId` and `operationReturnValue` (for more details, see [4]) for the Cluster Membership Service. In all cases, the `objectName` parameter is a pointer to the name of the Cluster Membership Service node on which the administrative operation is to be carried out. The name is expressed as an LDAP DN.

5.2.1 Include File and Library Name

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

```
#include <saImm.h>
```

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

```
libSaImm.so
```

5.2.2 SA_CLM_ADMIN_LOCK

Parameters

`operationId` = `SA_CLM_ADMIN_LOCK`

Description

When this operation is applied to a cluster node that is not a member of the cluster, the Cluster Membership Service sets the administrative state of the cluster node to `SA_CLM_ADMIN_LOCKED`.

When this operation is applied to a cluster node that is a member of the cluster, the Cluster Membership Service invokes for the `SA_CLM_CHANGE_START` step the `saClmClusterTrackCallback()` callback function of all processes that subscribed to the callback with the `SA_TRACK_START_STEP` flag set. The Cluster Membership Service then waits for these processes to respond (by invoking the `saClmResponse_4()` function) for the timeout period specified in the `saClmNodeLockCallbackTimeout` attribute for the node being locked. The processing continues when one of the following conditions is met. For each condition, the complete further processing is explained.

- The responses to all callbacks were `SA_CLM_CALLBACK_RESPONSE_OK`: in this case, the Cluster Membership Service removes the node from the cluster membership, sets the administrative state of the node to `SA_CLM_ADMIN_LOCKED`, and notifies all subscribed processes in the cluster about this fact by invoking the `saClmClusterTrackCallback()` callback function of each subscribed process for the `SA_CLM_CHANGE_COMPLETED` step.
- The response to a callback was `SA_CLM_CALLBACK_RESPONSE_ERROR`, or a process did not respond to the callback in the aforementioned timeout period: in this case, the Cluster Membership Service sets the administrative state of the node to `SA_CLM_ADMIN_LOCKED` and initiates a cluster node reboot as a preventive repair action. This way, it is guaranteed that all services are terminated and the node leaves the cluster membership. After the reboot, the node will not join the cluster due of its administrative state.
  The Cluster Membership Service then notifies all subscribed processes that the node left the membership. This notification is made by invoking the `saClmClusterTrackCallback()` callback function of all subscribed processes for the `SA_CLM_CHANGE_COMPLETED` step.
  In this case, the `operationReturnValue` is set to `SA_AIS_ERR_REPAIR_PENDING`, which indicates that the Cluster Membership Service had to reboot the node.

Note that for this administrative command the cluster node may be rebooted irrespective of the current value of the `saClmNodeDisableReboot` attribute of the cluster node (defined in the `SaClmNode` configuration object class, see Section 4.4).

operationReturnValue

`SA_AIS_OK` - The operation is successful; the administrative state of the node was set to `SA_CLM_ADMIN_LOCKED`, and the node is no longer in the cluster membership.

`SA_AIS_ERR_TIMEOUT` - An implementation-specific— other than the `saClmNodeLockCallbackTimeout`— timeout occurred. It is unspecified whether the call succeeded or whether it did not.

`SA_AIS_ERR_REPAIR_PENDING` - The node was successfully locked. However, at least one process subscribed to the `SaClmClusterTrackCallbackT_4` callback in the `SA_CLM_CHANGE_START` step did not respond within the `saClmNodeLockCallbackTimeout` period or responded with the `SA_CLM_CALLBACK_RESPONSE_ERROR` result. A reboot has already been initiated by the Cluster Membership Service.

`SA_AIS_ERR_TRY_AGAIN` - The operation cannot be provided at this time. The client may retry later. This error generally should be returned when the requested administrative operation is valid but not currently possible, probably because another administrative operation is currently acting upon the cluster node. The administrative state of the node and its membership did not change.

`SA_AIS_ERR_NO_MEMORY` - The Cluster Membership Service or a library is out of memory and cannot provide the service. The administrative state of the node and its membership did not change.

`SA_AIS_ERR_NO_RESOURCES` - There are insufficient resources to carry out the operation. The administrative state of the node and its membership did not change.

`SA_AIS_ERR_NO_OP` - The invocation of the operation has no effect on the current state of the cluster node, as its administrative state is already `SA_CLM_ADMIN_LOCKED`.

See Also

`SA_CLM_ADMIN_UNLOCK`, `SA_CLM_ADMIN_SHUTDOWN`

5.2.3 SA_CLM_ADMIN_UNLOCK

Parameters

`operationId` = `SA_CLM_ADMIN_UNLOCK`

Description

When this operation is applied to a cluster node, the Cluster Membership Service sets the administrative state of the cluster node to `SA_CLM_ADMIN_UNLOCKED`, thereby allowing the cluster node to join the cluster membership or to remain in the cluster membership (this latter case applies if the target node is in the `SA_CLM_ADMIN_SHUTTING_DOWN` administrative state), provided all the conditions for cluster membership are met.

For a successful operation, the `operationReturnValue` is `SA_AIS_OK`.

For other `operationReturnValues`, the administrative state of the cluster node remains unchanged and the operation is unsuccessful.

operationReturnValue

`SA_AIS_OK` - The operation is successful.

`SA_AIS_ERR_TIMEOUT` - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it did not.

`SA_AIS_ERR_TRY_AGAIN` - The operation cannot be provided at this time. The client may retry later. This error generally should be returned when the requested administrative operation is valid but not currently possible, probably because another operation is currently acting upon the cluster node.

`SA_AIS_ERR_NO_MEMORY` - The Cluster Membership Service or a library is out of memory and cannot provide the service.

`SA_AIS_ERR_NO_RESOURCES` - There are insufficient resources to carry out the operation.

`SA_AIS_ERR_NO_OP` - The invocation of this administrative operation has no effect on the current state of the node, as its administrative state is already `SA_CLM_ADMIN_UNLOCKED`.

See Also

`SA_CLM_ADMIN_LOCK`, `SA_CLM_ADMIN_SHUTDOWN`

5.2.4 SA_CLM_ADMIN_SHUTDOWN

Parameters

`operationId` = `SA_CLM_ADMIN_SHUTDOWN`

Description

When this operation is applied to a cluster node that is not a member of the cluster, the Cluster Membership Service sets the administrative state of the cluster node to `SA_CLM_ADMIN_LOCKED`.

When this operation is applied to a cluster node that is a member of the cluster, the Cluster Membership Service sets the administrative state of the cluster node to `SA_CLM_ADMIN_SHUTTING_DOWN`. Then, the Cluster Membership Service invokes for the `SA_CLM_CHANGE_START` step the `saClmClusterTrackCallback()` callback function of all processes that subscribed to the callback with the `SA_TRACK_START_STEP` flag set and waits for the processes to respond with the `saClmResponse_4()` function until one of the four following conditions is met. For each condition, the complete further processing is explained.

- The responses to all callbacks were `SA_CLM_CALLBACK_RESPONSE_OK`:
  in this case, the Cluster Membership Service removes the node from the cluster membership, sets the administrative state of the node to `SA_CLM_ADMIN_LOCKED`, and notifies all subscribed processes in the cluster about this fact by invoking the `saClmClusterTrackCallback()` callback function of each subscribed process for the `SA_CLM_CHANGE_COMPLETED` step.
- The response to a callback was `SA_CLM_CALLBACK_RESPONSE_ERROR`:
  in this case, the Cluster Membership Service sets the administrative state of the node to `SA_CLM_ADMIN_LOCKED` and initiates a cluster node reboot as a preventive repair action. This way, it is guaranteed that all services are terminated and the node leaves the cluster membership. After the reboot, the node will not join the cluster due of its administrative state.
  The Cluster Membership Service then notifies all subscribed processes that the node left the membership. This notification is made by invoking the `saClmClusterTrackCallback()` callback function of all subscribed processes for the `SA_CLM_CHANGE_COMPLETED` step.
  In this case, the `operationReturnValue` is set to `SA_AIS_ERR_REPAIR_PENDING`, which indicates that the Cluster Membership Service had to reboot the node.
  In this case, the `operationReturnValue` is set to `SA_AIS_ERR_REPAIR_PENDING`, which indicates that the Cluster Membership Service had to reboot the node.

- An SA_CLM_ADMIN_LOCK administrative operation is executed on the cluster node that is in the SA_CLM_ADMIN_SHUTTING_DOWN state to force it into the locked state within the configured time interval (see Section 5.2.2): in this case, no further invocations of the SaClmClusterTrackCallbackT callback function are issued for the SA_CLM_ADMIN_SHUTDOWN administrative operation, as it has been superseded by the SA_CLM_ADMIN_LOCK administrative operation.
- An SA_CLM_ADMIN_UNLOCK operation is executed on the cluster node that is in the SA_CLM_ADMIN_SHUTTING_DOWN state to abort the shutdown operation: in this case, no further invocations of the SaClmClusterTrackCallbackT callback function are issued for the SA_CLM_ADMIN_SHUTDOWN administrative operation, as it has been superseded by the SA_CLM_ADMIN_UNLOCK administrative operation.

Note that for this administrative command the cluster node may be rebooted irrespective of the current value of the saClmNodeDisableReboot attribute of the cluster node (defined in the SaClmNode configuration object class, see Section 4.4).

operationReturnValue

SA_AIS_OK - The operation is successful; all subscribers of the SaClmClusterTrackCallbackT_4 function in the SA_CLM_CHANGE_START step have responded to the callback with the SA_CLM_CALLBACK_RESPONSE_OK result.

SA_AIS_ERR_TIMEOUT - An implementation-specific timeout occurred. It is unspecified whether the call succeeded or whether it did not.

SA_AIS_ERR_REPAIR_PENDING - The node was successfully locked. However, at least one process subscribed to the SaClmClusterTrackCallbackT_4 callback in the SA_CLM_CHANGE_START step responded with the SA_CLM_CALLBACK_RESPONSE_ERROR result. A reboot has already been initiated by the Cluster Membership Service.

SA_AIS_ERR_TRY_AGAIN - The operation cannot be provided at this time. The client may retry later. This error generally should be returned when the requested action is valid but not currently possible, probably because another administrative operation is acting upon the cluster node.

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

SA_AIS_ERR_NO_RESOURCES - There are insufficient resources to carry out this operation.

SA_AIS_ERR_INTERRUPT - The SA_CLM_ADMIN_SHUTDOWN administrative operation was superseded by a SA_CLM_ADMIN_LOCK or by a SA_CLM_ADMIN_UNLOCK operation.

SA_AIS_ERR_NO_OP - The invocation of the operation has no effect on the current state of the cluster node, as its administrative state is already SA_CLM_ADMIN_LOCKED or SA_CLM_ADMIN_SHUTTING_DOWN.

See Also

SA_CLM_ADMIN_LOCK, SA_CLM_ADMIN_UNLOCK

6 Alarms and Notifications

The Cluster Membership Service produces alarms and notifications to convey important information regarding the operational and functional state of the objects under its control to an administrator or a management system.

These reports vary in perceived severity and include alarms, which potentially require an operator intervention, and notifications which signify important state or object changes. A management entity should regard notifications, but they do not necessarily require an operator intervention.

The vehicle to be used for producing alarms and notifications is the Notification Service of the Service Availability™ Forum (abbreviated as NTF, see [3]), and hence the various notifications are partitioned into categories as described in this service.

In some cases, this specification uses the word “Unspecified” for values of attributes that the vendor is at liberty to set to whatever makes sense in the vendor’s context, and the SA Forum has no specific recommendation regarding such values. Such values are generally optional from the CCITT Recommendation X.733 perspective (see [10]).

6.1 Setting Common Attributes

The following attributes of the notifications presented in Section 6.2 are not shown in their description, as the generic description presented here applies to all of them:

- Notification Id - Depending on the Notification Service function used to send the notification, this attribute is either implicitly set by the Notification Service or provided by the caller.
- Notifying Object - DN of the entity generating the notification. This name must conform to the SA Forum AIS naming convention and must contain at least the `safApp` RDN value portion of the DN set to the specified standard RDN value of the SA Forum AIS Service generating the notification, that is, `safClmService`. For details on the AIS naming convention, refer to [2].
- Event Time - This attribute contains the time when the Notification Service detected the event leading to the notification.

The following note applies to all Cluster Membership Service notifications presented in Section 6.2:

- Correlated Notifications - Correlation ids are supplied to correlate notifications that have been generated because of a related cause. The correlated notifications attribute should include

- in the first position the root notification identifier of the related tree of notifications as described in the Notification Service specification (see [3]);
- in the second position the parent notification identifier of the same tree;
- in the third position the notification identifier of the sibling notification, if any. This sibling notification is the opening pair of the current notification such as the alarm that is being cleared or the start of an administrative operation or a configuration change that has ended.

If any of these notifications is unknown, the `SA_NTF_IDENTIFIER_UNUSED` value must be used. This value may be omitted in trailing positions.

- Notification Class Identifier - The `vendorId` field of the `SaNtfClassIdT` data structure must be set to `SA_NTF_VENDOR_ID_SAF`, and the `majorId` field must be set to `SA_SVC_CLM` (as defined in the `SaServicesT` enumeration in [2]) for all notifications that follow the standard formats described in this specification. The `minorId` field will vary based on the specific notification.

6.2 Cluster Membership Service Notifications

This following subsections describe the notifications that a Cluster Membership Service implementation shall produce.

6.2.1 Cluster Membership Service Alarms

The Cluster Membership Service does not issue any alarms at the time of publication of this specification.

6.2.2 Cluster Membership Service State Change Notifications

6.2.2.1 Member Node Entry

Description

A particular configured node joined the cluster membership.

Table 5 Member Node Entry Notification

NTF Attribute Name	Mandatory/ Optional	Specified Value
Event Type	Mandatory	`SA_NTF_OBJECT_STATE_CHANGE`
Notification Object	Mandatory	LDAP DN of the new member node, as specified in Section 4.2
Notification Class Identifier	NTF-Internal	`minorId` = `SA_CLM_NTFID_NODE_JOIN`, see Section 3.3.14 on page 34
Additional Text	Optional	Unspecified

Additional Information	Optional	`infoId` = `SA_CLM_ROOT_CAUSE_ENTITY`
		`infoType` = `SA_NTF_VALUE_LDAP_NAME`
		`infoValue` = LDAP DN of root state change entity
		This attribute is provided only if there are correlated notifications, and the DN is different from the notification object.

Number of Correlated Notifications	Mandatory	0, 1, or more, depending on the value of correlated notifications
Correlated Notifications	Optional	correlated notification ids as available. In the case of a PLM root cause entity, `rootCorrelationId` and `parentCorrelationId` passed to CLM by PLM. In the case of CLM administrative operations as passed to CLM by IMM.
Source Indicator	Mandatory	`SA_NTF_OBJECT_OPERATION` or `SA_NTF_UNKNOWN_OPERATION`
Changed State Attribute ID	Optional	`SA_CLM_CLUSTER_CHANGE_STATUS`
Old Attribute Value	Optional	Unspecified
New Attribute Value	Mandatory	`SA_CLM_NODE_JOINED`

6.2.2.2 Member Node Exit

Description

A particular member node left the cluster membership.

Table 6 Member Node Exit Notification

NTF Attribute Name	Mandatory/ Optional	Specified Value
Event Type	Mandatory	`SA_NTF_OBJECT_STATE_CHANGE`
Notification Object	Mandatory	LDAP DN of the member node (as specified in Section 4.2) that left the cluster membership
Notification Class Identifier	NTF-Internal	`minorId` = `SA_CLM_NTFID_NODE_LEAVE`, see Section 3.3.14 on page 34
Additional Text	Optional	Unspecified

Additional Information	Optional	`infoId` = `SA_CLM_ROOT_CAUSE_ENTITY`
		`infoType` = `SA_NTF_VALUE_LDAP_NAME`
		`infoValue` = LDAP DN of root state change entity
		This attribute is provided only if there are correlated notifications, and the DN is different from the notification object.

Number of Correlated Notifications	Mandatory	0, 1, or more, depending on the value of correlated notifications
Correlated Notifications	Optional	correlated notification ids as available. In the case of a PLM root cause entity, `rootCorrelationId` and `parentCorrelationId` passed to CLM by PLM. In the case of CLM administrative operations as passed to CLM by IMM.
Source Indicator	Mandatory	`SA_NTF_OBJECT_OPERATION` or `SA_NTF_UNKNOWN_OPERATION`
Changed State Attribute ID	Optional	`SA_CLM_CLUSTER_CHANGE_STATUS`
Old Attribute Value	Optional	Unspecified
New Attribute Value	Mandatory	`SA_CLM_NODE_LEFT`

6.2.2.3 Member Node Reconfigured

Description

Some attributes associated with the member node have changed. Currently, only the `nodeAddress` attribute may change for a member node.

Table 7 Member Node Reconfigured Notification

NTF Attribute Name	Mandatory/ Optional	Specified Value
Event Type	Mandatory	`SA_NTF_OBJECT_STATE_CHANGE`
Notification Object	Mandatory	LDAP DN of the member node (as specified in Section 4.2) whose attributes have changed
Notifying Object	NTF-Internal	As specified above
Notification Class Identifier	NTF-Internal	`minorId` = `SA_CLM_NTFID_NODE_RECONFIG`, see Section 3.3.14 on page 34
Additional Text	Optional	Unspecified
Additional Information	Optional	Unspecified
Number of Correlated Notifications	Mandatory	0, 1, or 2, depending on the value of the `rootCorrelationId` and `parentCorrelationId` fields passed to CLM by IMM
Correlated Notifica-tions	Optional	`rootCorrelationId` and `parentCorrelationId` passed to CLM by IMM
Source Indicator	Mandatory	`SA_NTF_OBJECT_OPERATION` or `SA_NTF_UNKNOWN_OPERATION`
Changed State Attribute ID	Optional	`SA_CLM_CLUSTER_CHANGE_STATUS`
Old Attribute Value	Optional	Unspecified
New Attribute Value	Mandatory	`SA_CLM_NODE_RECONFIGURED`

6.2.2.4 Cluster Node Administrative State Change

Description

The administrative state of a cluster node has changed.

Table 8 Cluster Node Administrative State Change Notification

NTF Attribute Name	Mandatory/ Optional	Specified Value
Event Type	Mandatory	`SA_NTF_OBJECT_STATE_CHANGE`
Notification Object	Mandatory	LDAP DN of the cluster node (as specified in Section 4.2) whose administrative state changed
Notifying Object	NTF-Internal	As specified above
Notification Class Identifier	NTF-Internal	`minorId` = `SA_CLM_NTFID_NODE_ADMIN_STATE`, see Section 3.3.14 on page 34
Additional Text	Optional	Unspecified
Additional Information	Optional	Unspecified
Number of Correlated Notifications	Mandatory	0, 1, or 2, depending on the value of the `rootCorrelationId` and `parentCorrelationId` fields passed to CLM by IMM
Correlated Notifications	Optional	`rootCorrelationId` and `parentCorrelationId` passed to CLM by IMM
Source Indicator	Mandatory	`SA_NTF_MANAGEMENT_OPERATION`
Changed State Attribute ID	Optional	`SA_CLM_ADMIN_STATE`
Old Attribute Value	Optional	Applicable value from `SaClmAdminStateT` enumeration
New Attribute Value	Mandatory	Applicable value from `SaClmAdminStateT` enumeration

Index of Definitions