Service Availability™ Forum Application Interface Specification

Information Model Management Service SAI-AIS-IMM-A.02.01

.

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 Information Model Management Service

1 Document Introduction

1.1 Document Purpose

This document defines the Information Model Management 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 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

The first (and only previous release) of the IMM Service specification was:

SAI-AIS-IMM-A.01.01

This section presents the changes of the current release, SAI-AIS-IMM-A.02.01, with respect to the SAI-AIS-IMM-A.01.01 release. Editorial changes that do not change semantics or syntax of the described interfaces are not mentioned.

1.3.1 New Topics

- Section 3.1 describes rules to construct object names.
- Section 3.2 introduces the internal persistent repository.
- Section 3.3 explains the behavior of the IMM API functions on a non-member node. As a consequence, the `SA_AIS_ERR_UNAVAILABLE` return value has been added to various API functions (see Section 1.3.4).

- The SaImmAttrDefinitionT_2 structure in Section 4.2.7 replaced the SaImmAttrDefinitionT structure of version A.01.01 due to the removal of the attrNtfId member. As a consequence of this replacement, the functions saImmOmClassCreate_2(), saImmOmClassDescriptionGet_2(), and saImmOmClassDescriptionMemoryFree_2() have replaced the corresponding functions of version A.01.01 (those without the "_2" in the name).
- This version allows an initial value for persistent runtime attributes when an object is created. This enables, in particular, the configuration of the initial value of the administrative state of Availability Management Framework objects such as service units. To support this feature, the following changes were made:
  - the definition of the attrDefaultValue member of the SaImmAttrDefinitionT_2 structure (see Section 4.2.7) was extended;
  - the description and the SA_AIS_ERR_INVALID_PARAM return code of the functions saImmOmCcbObjectCreate_2() (see Section 4.8.2) and saImmOiRtObjectCreate_2() (see Section 5.5.1) were extended.
- The SaImmAttrValuesT_2 structure in Section 4.2.8 replaced the SaImmAttrValuesT structure of version A.01.01 due to the addition of the attrValueType member.
  As a consequence of this replacement, the SaImmAttrModificationT_2 structure (see Section 4.2.10) has replaced the SaImmAttrModificationT structure of version A.01.01.
  Due to these two preceding replacements, the following functions have replaced the corresponding functions of version A.01.01 (those without the "_2" in the name):
  SaImmOiCcbObjectCreateCallbackT_2,
  SaImmOiCcbObjectModifyCallbackT_2,
  saImmOiRtObjectCreate_2(), saImmOiRtObjectUpdate_2(),
  saImmOmAccessorGet_2(), saImmOmCcbObjectCreate_2(),
  saImmOmCcbObjectModify_2(), and saImmOmSearchNext_2().
- The SaImmSearchOneAttrT_2 structure in Section 4.2.13 replaced the SaImmSearchOneAttrT structure of version A.01.01 because the attrName member is no longer a pointer.
  As a consequence of this replacement, the SaImmSearchParametersT_2 union has replaced the SaImmSearchParametersT structure of version A.01.01.
  This last replacement in turn has led to the replacement of the saImmOmSearchInitialize() function of version A.01.01 with the saImmOmSearchInitialize_2() function.
- The SaImmAdminOperationParamsT_2 structure in Section 4.2.17 replaced the SaImmAdminOperationParamsT structure of version A.01.01 because the type of the paramBuffer member has changed, and the paramSize member

has been removed from the `SaImmAdminOperationParamsT` structure. As a consequence of this replacement, the functions `SaImmOiAdminOperationCallbackT_2`, `saImmOmAdminOperationInvoke_2()`, and `saImmOmAdminOperationInvokeAsync_2()` have replaced the corresponding functions of version A.01.01 (those without the "_2" in the name).

- Section 4.2.20 introduces the `SaImmRepositoryInitModeT` type.
- To allow the continuation of administrative operations, the API functions `saImmOmAdminOperationContinuationClear()`, `saImmOmAdminOperationContinue()`, and `saImmOmAdminOperationContinueAsync()` have been introduced (see Section 4.9 and subsections).
  Further changes due to this new feature:
  - ⇒ To support these functions, the `SaImmContinuationIdT` type has been introduced and the `continuationId` parameter has been added to the `saImmOmAdminOperationInvoke_2()` and `saImmOmAdminOperationInvokeAsync_2()` functions (see preceding item).
  - ⇒ Additionally, the SA_AIS_ERR_EXIST return has been added to the functions `SaImmOiAdminOperationCallbackT_2`, `saImmOmAdminOperationInvoke_2()`, and `saImmOmAdminOperationInvokeAsync_2()`.
  - ⇒ Furthermore, additional text has been added to the descriptions of the two functions in Section 4.7.3 and in Section 4.7.5 to explain that the continuation identifiers registered for the targeted objects are all cleared if these function calls succeed.
  - ⇒ The description of the `saImmOmAdminOwnerFinalize()` function in Section 4.7.3 explains under which conditions continuation identifiers are cleared.
  - ⇒ The description of the `saImmOmCcbObjectDelete()` function in Section 4.8.3 explains that this function also fails if one of the targeted objects has some registered continuation identifiers.
- To allow concurrent administrative operations on an IMM Service object, the definition of the SA_AIS_ERR_BUSY return value was changed in the `saImmOmAdminOperationInvoke_2()` and `saImmOmAdminOperationInvokeAsync_2()` function with respect to the superseded corresponding functions of version A.01.01. Section 4.8 was also updated accordingly.
- As superseding callback functions have been added, the `SaImmOiCallbacksT_2` in Section 5.2.4 replaced the `SaImmOiCallbacksT`

structure of version A.01.01. This change has led to replacement of the `saImmOiInitialize()` function of version A.01.01 with the `saImmOiInitialize_2()` function.

- Chapter 6 presents the IMM Service UML Information Model.
- Chapter 7 presents the IMM Service administrative functions.
- Chapter 8 states that the IMM Service does not contain any Alarms and Notification in this release.
- Chapter 9 states that no management interface is defined for the IMM Service in this release.

1.3.2 Clarifications

- A sentence has been added to the definition of the `SA_IMM_ATTR_CACHED` attribute in Section 4.2.5 to explain that persistent runtime attributes shall be cached. This section also explains that RDN values must be of type `SA_IMM_ATTR_SASTRINGT` or `SA_IMM_ATTR_SANAMET`.
- Section 4.2.8 clarifies that an attribute must have at least one value to be present in an object. As a consequence, optional attributes that have no value are not present in objects.
- The descriptions of the functions `saImmOmDispatch()` (see Section 4.3.3) and `saImmOiDispatch()` (see Section 5.3.3) clarify the meaning of the `SA_AIS_OK` return value.
- The descriptions of the functions `saImmOmFinalize()` (see Section 4.3.4) and `saImmOiFinalize()` (see Section 5.3.4) clarify that these functions free all resources allocated by the IMM Service for the process in the corresponding association between the process and the IMM Service.
- The notion of an “operation in progress” has been clarified in Section 4.7.3.
- Section 5.6 clarifies the scope in space and time of CCB identifiers, as seen by Object Implementers.

1.3.3 Superseded and Superseding Functions

The IMM Service defines for the version A.02.01 new functions and new type definitions to replace functions and type definitions of the version A.01.01. The list of replaced functions and type definitions in alphabetic order is presented in Table 1.

The superseded functions and type definitions are no longer supported in version A.02.01, and no description is provided for them in this document. The names of the superseding functions and type definitions are obtained by adding “_2” to the respective names of the previous version. Regarding the support of backward compatibility in SA Forum AIS, refer to the Overview document ([1]).

Table 1 Superseded Functions and Type Definitions in Version A.02.01

Functions and Type Definitions of A.01.01 no Longer Supported in A.02.01
SaImmAdminOperationParamsT
SaImmAttrDefinitionT
SaImmAttrModificationT
SaImmAttrValuesT
SaImmOiAdminOperationCallbackT
SaImmOiCallbacksT
SaImmOiCcbObjectCreateCallbackT
SaImmOiCcbObjectModifyCallbackT
saImmOiInitialize()
saImmOiRtObjectCreate()
saImmOiRtObjectUpdate()
saImmOmAccessorGet()
saImmOmAdminOperationInvoke()
saImmOmAdminOperationInvokeAsync()
saImmOmCcbObjectCreate()
saImmOmCcbObjectModify()
saImmOmClassCreate()
saImmOmClassDescriptionGet()
saImmOmClassDescriptionMemoryFree()
saImmOmSearchInitialize()
saImmOmSearchNext()
SaImmSearchOneAttrT
SaImmSearchParametersT

1.3.4 Changes in Return Values of API Functions

The first row in the following table applies to all functions of this release. The other rows apply only to functions that have not been superseded.

Table 2 Changes in Return Values of API Functions

API Function	Return Value	Change Type
All API functions except[1] `saImmOmFinalize()`, `saImmOiFinalize()`, and all callbacks listed in `SaImmOiCallbacksT_2`.	`SA_AIS_ERR_UNAVAILABLE`	new
`saImmOiImplementerClear()`	`SA_AIS_ERR_BAD_HANDLE`	extended
`saImmOiObjectImplementerRelease()`	`SA_AIS_ERR_NOT_EXIST`	corrected
`saImmOiRtObjectDelete()`	`SA_AIS_ERR_BAD_OPERATION`	extended
`saImmOiRtObjectDelete()`	`SA_AIS_ERR_EXIST`	deleted

`SaImmOmAdminOperationInvokeCallbackT`	`SA_AIS_ERR_BAD_OPERATION`
	`SA_AIS_ERR_BUSY`
	`SA_AIS_ERR_EXIST`
	`SA_AIS_ERR_NOT_EXIST`
		extended

`saImmOmCcbObjectDelete()`	`SA_AIS_ERR_BAD_OPERATION`	extended
`saImmOmCcbObjectDelete()`	`SA_AIS_ERR_EXIST`	deleted
`saImmOmDispatch()`	`SA_AIS_OK`	clarified

1. The `SaImmOmAdminOperationInvokeCallbackT` callback function has the `SA_AIS_ERR_UNAVAILABLE` return value in the `error` parameter.

1.3.5 Other Changes

- In the description of the functions `saImmOmInitialize()` (see Section 4.3.1) and `saImmOiInitialize_2()` (see Section 5.3.1), the sentence
  "If the implementation supports the required `releaseCode`, and a major version >= the required `majorVersion`, `SA_AIS_OK` is returned."
  has been replaced by the sentence
  "If the implementation supports the specified `releaseCode` and `majorVersion`, `SA_AIS_OK` is returned.".
- A sentence has been added to the `saImmOmInitialize()` function (see Section 4.3.1) to explain that the continuation identifier of the continuation functions is not cleared when the process exits.

- The `saImmOiInitialize_2()` function (see Section 5.3.1) was changed to clarify that If `immOiCallbacks` is set to NULL, no callback is registered.
- Section 5.4.4 on the `saImmOiClassImplementerRelease()` now states that this function removes all “non-persistent cached runtime attributes” from all objects of that class. In the preceding version, it stated that “cached runtime attributes” were removed. An analogous change was made for the `saImmOiObjectImplementerRelease()` function in Section 5.4.6.

1.4 References

The following document contains information that is relevant to the specification:

[1] Service Availability™ Forum, Service Availability Interface, Overview, SAI-Overview-B.04.01

[2] Service Availability™ Forum, Information Model in XML Metadata Interchange (XMI) v2.1 format, SAI-XMI-A.03.01

[3] Service Availability™ Forum, IMM XML Schema Definition, SAI-AIS-IMM-XSD.A.01.01

[4] Service Availability™ Forum, Application Interface Specification, Cluster Membership Service, SAI-AIS-CLM-B.03.01

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 website (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 Forum’s website (http://www.saforum.org).

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

1.7 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 Forum's website (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 Forum's website (http://www.saforum.org).

2 Overview

This specification defines the Information Model Management Service within the Application Interface Specification (AIS).

The IMM Service is a cluster-wide service that must be highly-available in the sense that no single failure should take the entire service down.

2.1 Information Model Management Service

The different entities of an SA Forum cluster, such as components provided by the Availability Management Framework, checkpoints provided by the Checkpoint Service, or message queues provided by the Message Service are represented by various objects of the SA Forum Information Model.

The SA Forum Information Model (IM) is specified in UML and managed by the Information Model Management (IMM) Service.

The objects in the Information Model are provided with their attributes and administrative operations (that is, operations that can be performed on the represented entities through system management interfaces). For management applications or Object Managers, the IMM provides the APIs to create, access, and manage these objects.

The IMM Service delivers the requested operations to the appropriate AIS Services or applications (referred to as **Object Implementers**) that implement these objects for execution.

Information Model objects and attributes can be classified into two categories:

- Configuration objects and attributes
- Runtime objects and attributes

The IMM Service defines two sets of APIs:

(1) An Object Management API (OM-API), which is typically exposed to system management applications (for example, SNMP agents).

(2) An Object Implementer API (OI-API) restricted to Object Implementers.

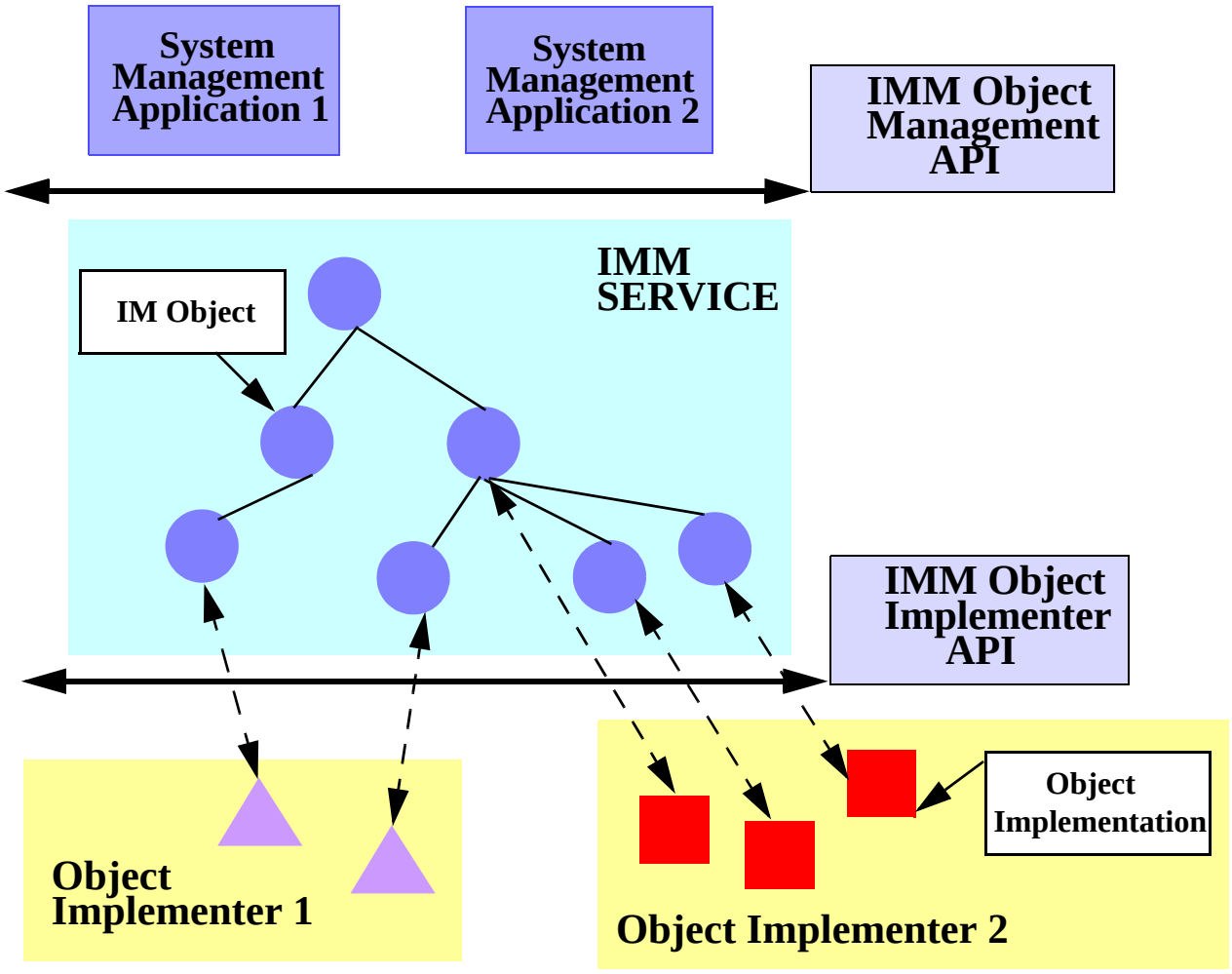
3 Information Model Management Service API

The Service Availability™ Forum (SA Forum) Information Model (IM) is specified in UML and represents the various objects that constitute an SA Forum system. The SA Forum IM also specifies the attributes of these objects and administrative operations that can be performed on the entities by using system management interfaces.

The Information Model Management (IMM) Service is the SA Forum Service that manages all objects of the SA Forum Information Model and provides the APIs to access and manage these objects.

FIGURE 1 presents an overview of the interfaces provided by the IMM Service.

FIGURE 1 IMM Service Interfaces



The actual implementation of objects represented in the Information Model is not part of the IMM Service but is provided by user applications or other AIS Services such as the Checkpoint Service or the Availability Management Framework.

AIS Services and applications that implement the IMM objects are called **Object Implementers** in the remainder of this document.

IMM objects are organized in a tree hierarchy. The hierarchy follows the structure of the LDAP distinguished name of each object. For more information about LDAP object names, refer to the SA Forum Overview document ([1]).

IMM objects and attributes can be classified into two categories:

- Configuration Objects and Attributes
  - **Configuration objects** and **configuration attributes** are the means by which system management applications provide input to an Object Implementer on the desired sets of objects and on their handling. The set of configuration objects and attributes constitute the prescriptive part of the Information Model.
  - Configuration objects and attributes are typically under the control of system management applications. They are of a persistent nature and must survive a full cluster power-off.
  - Configuration attributes are read-write attributes from an Object Management perspective but read-only from an Object Implementer perspective.
- Runtime Objects and Attributes
  - **Runtime objects** and **runtime attributes** are the means by which Object Implementers reflect in the Information Model the current state of the objects they implement. The set of runtime objects and attributes constitute the descriptive part of the Information Model. Runtime objects and attributes are typically under the control of Object Implementers.
  - Runtime objects that contain persistent runtime attributes are persistent and must survive a full cluster power-off. Non-persistent runtime attributes do not survive a full cluster power-off.
  - Runtime attributes are read-only attributes from an Object Management perspective but read-write from an Object Implementer perspective.

As attributes cannot exist outside of an encapsulating object, configuration attributes can only belong to configuration objects, as opposed to runtime attributes that may belong to objects of either category. Runtime objects can only have runtime attributes.

Object Implementers cannot on their own initiative create and delete configuration objects or modify configuration attributes by using the Object Implementer interface. On the other hand, system management applications cannot directly create and delete runtime objects or modify runtime attributes. However, as a consequence of some administrative operations requested by these system management applications Object Implementers may create or delete runtime objects or modify runtime attributes to reflect the new system state after the completion of the administrative operation.

The IMM Service exposes two sets of APIs:

(1) An **Object Management API** (OM-API), which is typically exposed to system management applications (for example, SNMP agents).

(2) An **Object Implementer API** (OI-API), which is intended to be used by Object Implementers.

Chapter 4 describes the OM-API. The OI-API is found in Chapter 5.

3.1 Object Naming

The Distinguished Name (DN) of an object (also simply called the object name) is constructed by prefixing the DN of the object's parent in the IMM tree hierarchy with the Relative Distinguished Name (RDN) of the object. The ',' character is used as a separator between the RDN of the object and the DN of its parent as follows:

Object_DN = "Object_RDN,Parent_Object_DN"

Objects that are immediately under the root of the IMM hierarchy have a DN that is equal to their RDN.

Each object must have one and only one attribute which is used to build the object RDN as follows:

Object_RDN = "RDN_attribute_name=RDN_attribute_value"

3.2 Internal Persistent Repository

The IMM Service maintains a copy of all its persistent entities (class definitions and persistent objects with their persistent attributes) within an **internal persistent repository** kept on stable storage. The storage holding the IMM persistent repository must be highly available, which implies storage replication. The nature of this internal repository is implementation-specific.

During startup of the IMM Service, the contents of its internal repository may be over-written (or initialized if the internal repository was empty) from the contents of an XML file. It is implementation-specific how the XML file is provided to the IMM Service at startup. The XML file must conform to the **IMM XML Schema Definition** (see [3]). Such an XML file may be the result of the `SA_IMM_ADMIN_EXPORT` administrative operation (see Section 7.3.1 on page 146). If the XML file contains the description of non-persistent objects or attributes, these objects and attributes are ignored. The configuration parameter `saImmRepositoryInit` of the `SaImmMngt` object class (see Section 6.2 on page 143) specifies whether to overwrite or not the contents of the IMM internal repository at startup of the IMM Service.

When the IMM Service starts (for example, at the initial cluster startup or after a full cluster power-off), it contains only the class definitions and persistent objects with their persistent attributes that are present in its internal repository. Non-persistent runtime objects must be re-created by Object Implementers. The values of non-persistent runtime attributes (cached or not) will be obtained from the Object Implementers.

3.3 Unavailability of the IMM API on a Non-Member Node

The IMM Service does not provide service to processes on cluster nodes that are not in the cluster membership (see [4]).

The following subsection describes the behavior of the IMM Service under various conditions that cause the IMM Service to be unavailable on a cluster node. Section 3.3.2 on page 24 contains guidelines for IMM Service implementers for dealing with a temporary unavailability of the service.

3.3.1 A Member Node Leaves or Rejoins the Cluster Membership

If the cluster node has left the cluster membership (see [4]) or is being administratively evicted from the cluster membership, the IMM Service behaves as follows towards processes residing on that cluster node and using or attempting to use the service:

- Calls to `saImmOmInitialize()` and `saImmOiInitialize_2()` will fail with SA_AIS_ERR_UNAVAILABLE.
- All IMM Service APIs that are invoked by the process and that operate on handles already acquired by the process will fail with SA_AIS_ERR_UNAVAILABLE with the following exceptions, assuming that the handle `immHandle` or the handle `immOiHandle` has already been acquired:
  - ⇒ The `saImmOmAdminOperationInvokeAsync_2()` function may return SA_AIS_OK or SA_AIS_ERR_UNAVAILABLE, depending on the service implementation. If it returns SA_AIS_OK, the callback `SaImmOmAdminOperationInvokeCallbackT` will be called and will also return SA_AIS_ERR_UNAVAILABLE in the error parameter; otherwise, the callback will not be called.
  - ⇒ The `saImmOmFinalize()` and `saImmOiFinalize()` functions, which are used to free the Object Management or Object Implementer library handles and all resources associated with these handles.
- An outstanding callback `SaImmOmAdminOperationInvokeCallbackT` will return SA_AIS_ERR_UNAVAILABLE in the error parameter.

If the cluster node rejoins the cluster membership, processes executing on the cluster node will be able to reinitialize new library handles and use the entire set of IMM Service APIs that operate on these new handles. However, invocation of APIs that operate on handles acquired by any process before the cluster node left the membership will continue to fail with SA_AIS_ERR_UNAVAILABLE (or with the special treatment described above for asynchronous calls) with the exception of `saImmOmFinalize()` and `saImmOiFinalize()`, which are used to free the library handles and all resources associated with these handles. Hence, it is recommended for the pro-

cesses to finalize the library handles as soon as the processes detect that the cluster node left the membership.

When the cluster node leaves the membership, the IMM Service executing on the remaining nodes of the cluster behaves as if all processes that were using the IMM Service on the leaving cluster node had been terminated. In particular, if a process on the leaving cluster node was registered as an Object Implementer, the IMM Service will unregister it automatically (see Section 5.4.2 on page 113).

3.3.2 Guidelines for IMM Service Implementers

The implementation of the IMM Service must leverage the SA Forum Cluster Membership Service (see [4]) to determine the membership status of a cluster node for the case explained in Section 3.3.1 on page 23 before returning `SA_AIS_ERR_UNAVAILABLE`. If the Cluster Membership Service considers a cluster node as a member of the cluster but the IMM Service experiences difficulty in providing service to its clients because of transport, communication, or other issues, it must respond with `SA_AIS_ERR_TRY_AGAIN`.

4 IMM Service - Object Management API Specification

4.1 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 IMM Service Object Management API:

```
#include <saImmOm.h>
```

To use the IMM Service Object Management API, an application must be bound with the following library:

```
libSaImmOm.so
```

4.2 Type Definitions

The Information Model Management Service uses the types described in the following sections.

4.2.1 Handles Used by the IMM Service

```
typedef SaUint64T SaImmHandleT;
typedef SaUint64T SaImmAdminOwnerHandleT;
typedef SaUint64T SaImmCcbHandleT;
typedef SaUint64T SaImmSearchHandleT;
typedef SaUint64T SaImmAccessorHandleT;
```

The acronym CCB stands for **Configuration Changes Bundle**. For its usage, refer to Section 4.8 on page 72.

4.2.2 Various IMM Service Names

The following types represent object class names, administrative owner names, and object class attribute names. All these names are UTF-8 encoded character strings terminated by the NULL character.

```
typedef SaStringT SaImmClassNameT;
typedef SaStringT SaImmAttrNameT;
typedef SaStringT SaImmAdminOwnerNameT;
```

4.2.3 SaImmValueTypeT

The `SaImmValueTypeT` contains various data types used by the IMM Service for class attributes and administrative operation parameters.

```
typedef enum {
     SA_IMM_ATTR_SAINT32T       = 1,   /* SaInt32T */
     SA_IMM_ATTR_SAUINT32T      = 2,   /* SaUint32T */
     SA_IMM_ATTR_SAINT64T       = 3,   /* SaInt64T */
     SA_IMM_ATTR_SAUINT64T      = 4,   /* SaUint64T */
     SA_IMM_ATTR_SATIMET        = 5,   /* SaTimeT */
     SA_IMM_ATTR_SANAMET        = 6,   /* SaNameT */
     SA_IMM_ATTR_SAFLOATT       = 7,   /* SaFloatT */
     SA_IMM_ATTR_SADOUBLET      = 8,   /* SaDoubleT */
     SA_IMM_ATTR_SASTRINGT      = 9,   /* SaStringT */
     SA_IMM_ATTR_SAANYT         = 10   /* SaAnyT */
} SaImmValueTypeT;
```

4.2.4 SaImmClassCategoryT

The `SaImmClassCategoryT` type is used to distinguish among different categories of object classes.

```
typedef enum {
     SA_IMM_CLASS_CONFIG        = 1,
     SA_IMM_CLASS_RUNTIME       = 2
} SaImmClassCategoryT;
```

The values of `SaImmClassCategoryT` indicate whether the object class is a configuration object class or a runtime object class.

4.2.5 SaImmAttrFlagsT

The SaImmAttrFlagsT type used to specify the various characteristics of an attribute of an object class.

```
#define SA_IMM_ATTR_MULTI_VALUE       0x00000001
#define SA_IMM_ATTR_RDN               0x00000002
#define SA_IMM_ATTR_CONFIG            0x00000100
#define SA_IMM_ATTR_WRITABLE          0x00000200
#define SA_IMM_ATTR_INITIALIZED       0x00000400
#define SA_IMM_ATTR_RUNTIME           0x00010000
#define SA_IMM_ATTR_PERSISTENT        0x00020000
#define SA_IMM_ATTR_CACHED            0x00040000

typedef SaUint64T SaImmAttrFlagsT;
```

The meaning of the flags listed above is:

- SA_IMM_ATTR_MULTI_VALUE: if this flag is specified, the attribute is a multi-value attribute; otherwise, the attribute is a single-value attribute.
- SA_IMM_ATTR_RDN: the attribute is used as the Relative Distinguished Name (RDN) for the containing object. Each object class must have one and only one RDN attribute. This attribute must be a single-value attribute of type SA_IMM_ATTR_SASTRINGT or SA_IMM_ATTR_SANAMET and may not be modified after the object is created. The RDN attribute of a configuration object must be a configuration attribute.

The following two attributes are mutually exclusive, as an attribute is either a configuration or a runtime attribute.

- SA_IMM_ATTR_CONFIG: the attribute is a configuration attribute. Configuration attributes are only allowed within object classes of the SA_IMM_CLASS_CONFIG category.
- SA_IMM_ATTR_RUNTIME: the attribute is a runtime attribute. Runtime attributes can belong to all object class categories.

The following two attributes are only meaningful for configuration attributes. Setting them for runtime attributes is not allowed and generates an error.

- SA_IMM_ATTR_WRITABLE: setting this flag for a configuration attribute indicates that the attribute can be modified. If the flag is not present, the configuration

attribute can only be set when the object is created and cannot be modified or deleted later on.

- SA_IMM_ATTR_INITIALIZED: setting this flag for a configuration attribute indicates that a value must be specified for this attribute when the object is created. This flag may not be set in the definition of a configuration attribute that has a default value.

The following attributes are only meaningful for runtime attributes. Setting them for configuration attributes is not allowed and generates an error.

- SA_IMM_ATTR_PERSISTENT: setting this flag for runtime attributes indicates that the attribute must be stored in a persistent manner by the IMM Service. If a runtime object has persistent attributes, or if one of its children has persistent attributes, its RDN attribute must be persistent.
- SA_IMM_ATTR_CACHED: setting this flag for a runtime attribute indicates that the value of the attribute must be cached by the IMM Service. This flag is automatically set by the IMM Service when the SA_IMM_ATTR_PERSISTENT flag is set.

4.2.6 SaImmAttrValueT

The SaImmAttrValueT type is used to represent the values of object attributes.

```
typedef void *SaImmAttrValueT;
```

4.2.7 SaImmAttrDefinitionT_2

The SaImmAttrDefinitionT_2 type is used to specify the characteristics of an attribute belonging to a particular object class.

```
typedef struct {
     SaImmAttrNameT attrName;
     SaImmValueTypeT attrValueType;
     SaImmAttrFlagsT attrFlags;
     SaImmAttrValueT attrDefaultValue;
} SaImmAttrDefinitionT_2;
```

The various fields of the structure above have the following usage:

- attrName: contains the attribute name.
- attrValueType: indicates what type of values can be assigned to this attribute.
- attrFlags: contains additional characteristics of this attribute.

- `attrDefaultValue`: contains a value that will automatically be assigned by the IMM Service to this attribute if no value is specified when an object containing this attribute is created. A default value shall only be provided for configuration and persistent runtime attributes. Must be set to NULL if there is no default value for this attribute.

4.2.8 SaImmAttrValuesT_2

The `SaImmAttrValuesT_2` type is used to specify the values of one attribute of an object.

```
typedef struct {
     SaImmAttrNameT attrName;
     SaImmValueTypeT attrValueType;
     SaUint32T attrValuesNumber;
     SaImmAttrValueT *attrValues;
} SaImmAttrValuesT_2;
```

The `attrName` field indicates the attribute name, the `attrValueType` field the type of the attribute, and the `attrValuesNumber` field the number of attribute values contained in the array of value descriptors to which `attrValues` points.

In order to be present within an object, an attribute must have at least one value. Optional attributes that have no value are not present in objects.

4.2.9 SaImmAttrModificationTypeT

The `SaImmAttrModificationTypeT` type specifies the type of modification to apply on the values of an attribute.

```
typedef enum {
     SA_IMM_ATTR_VALUES_ADD          = 1,
     SA_IMM_ATTR_VALUES_DELETE       = 2,
     SA_IMM_ATTR_VALUES_REPLACE      = 3
} SaImmAttrModificationTypeT;
```

- `SA_IMM_ATTR_VALUES_ADD` is used to add one or several values to an attribute in an object. If the attribute did not already have a value, the attribute is added.
- `SA_IMM_ATTR_DELETE` is used to remove one or several specified values from an attribute of an object. If all values of the attribute are removed, the attribute is also removed from the object. If the intention is to remove an attribute without specifying all its values, the `SA_IMM_ATTR_REPLACE` enum can be used.

- SA_IMM_ATTR_REPLACE is used to replace all current values of an attribute with a new set of values. If the new set of values is empty, the attribute is removed. If one or several values are specified and the attribute does not exist in the object, the attribute is added to the object with the new set of values.

The SaImmAttrModificationTypeT type is used to specify the modification to apply on an object attribute.

4.2.10 SaImmAttrModificationT_2

```
typedef struct {
     SaImmAttrModificationTypeT modType;
     SaImmAttrValuesT_2 modAttr;
} SaImmAttrModificationT_2;
```

The modType field indicates the type of modification to perform. The modAttr field specifies the attribute name and the values to be added to the attribute, or to be removed from the attribute, or that will replace the existing values. An empty set of values can be specified by setting attrValuesNumber to 0 and attrValues to NULL in the modAttr field. It is an error to use such an empty set of values with the SA_IMM_ATTR_VALUES_ADD or SA_IMM_ATTR_VALUES_DELETE modification types.

4.2.11 SaImmScopeT

The SaImmScopeT type is used to specify the scope of some IMM Service operations.

```
typedef enum {
     SA_IMM_ONE      = 1,
     SA_IMM_SUBLEVEL = 2,
     SA_IMM_SUBTREE  = 3
} SaImmScopeT;
```

- SA_IMM_ONE indicates that the scope of the operation is targeted to a single object.
- SA_IMM_SUBLEVEL indicates that the scope of the operation is targeted to one object and its direct children.
- SA_IMM_SUBTREE indicates that the scope of the operation is targeted to one object and the entire subtree rooted at that object.

4.2.12 SaImmSearchOptionsT

The `SaImmSearchOptionsT` is used to specify various options when performing searches amongst IMM Service objects.

```
typedef SaUint64T SaImmSearchOptionsT;
```

Two kinds of options can be specified by `SaImmSearchOptionsT`:

- Options related to the search criteria. Currently, only one such option is supported by the IMM Service. It must be specified for all search operations:

  ```
  #define SA_IMM_SEARCH_ONE_ATTR 0x0001
  ```

  `SA_IMM_SEARCH_ONE_ATTR` enables the retrieval of objects containing an attribute of a particular name and assigned to a particular value.

- Options used to specify which attributes of the objects matching the search criteria must be returned to the process performing the search. One and only one of these three options must be specified for each search operation:

  ```
  #define SA_IMM_SEARCH_GET_ALL_ATTR  0x0100

  #define SA_IMM_SEARCH_GET_NO_ATTR   0x0200

  #define SA_IMM_SEARCH_GET_SOME_ATTR 0x0400
  ```

  `SA_IMM_SEARCH_GET_ALL_ATTR` indicates that for each object matching the search criteria, all its attributes along with their values must be returned to the process performing the search.

  `SA_IMM_SEARCH_GET_NO_ATTR` indicates that no attributes of the objects matching the search criteria must be returned to the process performing the search. In this case, only the names of the objects matching the search criteria are returned.

  `SA_IMM_SEARCH_GET_SOME_ATTR` indicates that for each object matching the search criteria, only a subset of its attributes along with their values must be returned to the process performing the search. The list of attribute names to be returned is specified by another parameter of the search operation.

4.2.13 SaImmSearchParametersT_2

The `SaImmSearchParametersT_2` type is used to provide the criteria parameters used for search operations.

```
typedef struct {
     SaImmAttrNameT attrName;
     SaImmValueTypeT attrValueType;
     SaImmAttrValueT attrValue;
} SaImmSearchOneAttrT_2;
```

The `SaImmSearchOneAttrT_2` type contains the attribute description for `SA_IMM_SEARCH_ONE_ATTR` search operations. The fields `attrName` and `attrValue` specify the attribute name and value being searched for. The `attrValueType` field indicates the type of value that is assigned to the attribute.

If `attrValue` is not set to NULL, an object matches the search criteria if one of its attributes has a name identical to the name to which `attrName` points, the values for this attribute are of type `attrValueType`, and the value of the attribute (or one of its values for multi-valued attributes) is identical to the value to which `attrValue` points.

If `attrValue` is set to NULL, only the attribute name is used as a search criteria, and all objects having an attribute with such a name will be retrieved by the search operation, regardless of their attribute values.

If `attrName` is set to NULL, `attrValue` must also be set to NULL. Such an empty criterion will match all IMM Service objects. This empty criterion can be used to browse through all IMM Service objects.

```
typedef union {
     SaImmSearchOneAttrT_2 searchOneAttr;
} SaImmSearchParametersT_2;
```

Note: Searching for a particular value of a non-cached runtime attribute should be used with care, as it forces the IMM Service to fetch all values from the Object Implementers, which creates extra load on the system.

4.2.14 SaImmCcbFlagsT

The `SaImmCcbFlagsT` type is used to specify the various characteristics of a CCB. Currently, only one value is provided.

```
#define SA_IMM_CCB_REGISTERED_OI      0x00000001

typedef SaUint64T SaImmCcbFlagsT;
```

`SA_IMM_CCB_REGISTERED_OI`: if this flag is specified, the CCB can only hold changes for objects that have a registered Object Implementer. This flag must be set by applications that expect Object Implementers to validate the changes made using the CCB. If this flag is not set, the IMM Service accepts changes on objects with no registered implementer.

4.2.15 SaImmContinuationIdT

```
typedef SaUint64T SaImmContinuationIdT;
```

The type `SaImmContinuationIdT` is used to identify a particular invocation of an administrative operation on an IMM object. Its scope is cluster-wide, and it must be unique on a per-IMM object basis.

For more details, refer to Section 4.9 on page 85.

4.2.16 SaImmAdminOperationIdT

The `SaImmAdminOperationIdT` type is used to hold an identifier designating a particular administrative operation to perform on an object. The identifiers for all administrative operations of a given object class must have different integer values. However, the same values can be used for administrative operations of different object classes. In other words, the scope of an operation identifier is the object class.

```
typedef SaUint64T SaImmAdminOperationIdT;
```

4.2.17 SaImmAdminOperationParamsT_2

The SaImmAdminOperationParamsT_2 type is used to specify the parameters of an administrative operation performed on an object.

```
typedef struct {
     SaStringT paramName;
     SaImmValueTypeT paramType;
     SaImmAttrValueT paramBuffer;
} SaImmAdminOperationParamsT_2;
```

The paramName field indicates the name of the parameter. The paramType field indicates the type of the parameter. The paramBuffer field contains the parameter value.

4.2.18 SaImmCallbacksT

The SaImmCallbacksT structure defines the set of callbacks a process can provide to the IMM Service at initialization time.

```
typedef struct {
     SaImmOmAdminOperationInvokeCallbackT
          saImmOmAdminOperationInvokeCallback;
} SaImmCallbacksT;
```

4.2.19 IMM Service Object Attributes

The following #define directives are used to refer to the name of attributes of objects in the SA Forum Information Model.

```
#define SA_IMM_ATTR_CLASS_NAME "SaImmAttrClassName"
```

The IMM Service adds an attribute to each object holding the name of the class of the object. The name of this attribute is specified by the constant SA_IMM_ATTR_CLASS_NAME.

```
#define SA_IMM_ATTR_ADMIN_OWNER_NAME "SaImmAttrAdminOwnerName"
```

When an object has been assigned an administrative owner, the IMM Service stores the name of the object administrative owner in one attribute of the object. The name of this attribute is specified by the constant SA_IMM_ATTR_ADMIN_OWNER_NAME. This attribute does not exist in objects having no administrative owners.

```
#define SA_IMM_ATTR_IMPLEMENTER_NAME
     "SaImmAttrImplementerName"
```

When an object has an implementer, the IMM Service stores the name of the Object Implementer in one attribute of the object. The name of this attribute is specified by the constant SA_IMM_ATTR_IMPLEMENTER_NAME. This attribute does not exist in objects having no implementers.

The above attributes are single-value attributes and their value is of type SA_IMM_ATTR_SASTRINGT. For configuration objects, these attributes are configuration attributes, and for runtime objects, these attributes are runtime attributes. If the runtime object is persistent, these attributes are also persistent.

4.2.20 SaImmRepositoryInitModeT

```
typedef enum {
     SA_IMM_KEEP_REPOSITORY     = 1,
     SA_IMM_INIT_FROM_FILE      = 2
} SaImmRepositoryInitModeT;
```

The values of SaImmRepositoryInitModeT specify how the IMM Service initializes its internal repository when the IMM Service starts up.

- SA_IMM_KEEP_REPOSITORY: at startup, the IMM Service keeps the contents of its internal repository.
- SA_IMM_INIT_FROM_FILE: at startup, the IMM Service must overwrite the contents of its internal repository with the contents of an XML file. The location of this initial XML file is implementation-dependent.

4.3 Library Life Cycle

4.3.1 saImmOmInitialize()

Prototype

```
SaAisErrorT saImmOmInitialize(
      SaImmHandleT *immHandle,
      const SaImmCallbacksT *immCallbacks,
      SaVersionT *version
);
```

Parameters

`immHandle` - [`out`] A pointer to the handle which identifies this particular initialization of the IMM Service, and which is to be returned by the IMM Service. This handle provides access to the Object Management APIs of the IMM Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`immCallbacks` - [`in`] If `immCallbacks` is set to NULL, no callback is registered; If `immCallbacks` is not set to NULL, it is a pointer to an `SaImmCallbacksT` structure which contains the callback functions of the process that the IMM Service may invoke. Only non-NULL callback functions in this structure will be registered. The `SaImmCallbacksT` type is defined in Section 4.2.18 on page 34.

`version` - [`in/out`] As an input parameter, `version` is a pointer to a structure containing the required IMM 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 IMM Service. The `SaVersionT` type is defined in [1].

Description

This function initializes the Object Management functions of the Information Model Management Service for the invoking process and registers the various callback functions. This function must be invoked prior to the invocation of any other Object Management functions of the Information Model Management Service functionality. The handle pointed to by `immHandle` is returned by the IMM Service as the reference to this association between the process and the Object Management of the IMM Service. The process uses this handle in subsequent communication with the Object Management of the IMM Service.

If the invoking process exits after successfully returning from the `saImmOmInitialize()` function and before invoking `saImmOmFinalize()` to finalize the handle `immHandle` (see Section 4.3.4 on page 41), the IMM Service automatically finalizes this handle and any other handles that have been acquired using the handle `immHandle` when the IMM Service detects the death of the process.

If the implementation supports the specified `releaseCode` and major version, `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.

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 Information Model Management Service library or the provider of the service is out of memory and cannot provide the service.

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

SA_AIS_ERR_VERSION - The version provided in the structure to which the `version` parameter points is not compatible with the version of the Information Model Management Service implementation.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node because it is not a member node.

See Also

`saImmOmSelectionObjectGet()`, `saImmOmDispatch()`, `saImmOmFinalize()`

4.3.2 saImmOmSelectionObjectGet()

Prototype

```
SaAisErrorT saImmOmSelectionObjectGet(
      SaImmHandleT immHandle,
      SaSelectionObjectT *selectionObject
);
```

Parameters

`immHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`selectionObject` - [`out`] A pointer to the operating system handle that the process can use to detect pending callbacks. The `SaSelectionObjectT` type is defined in [1].

Description

This function returns the operating system handle associated with the handle `immHandle`. The invoking process can use the operating system handle to detect pending callbacks, instead of repeatedly invoking `saImmOmDispatch()` 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 `saImmOmSelectionObjectGet()` is valid until `saImmOmFinalize()` is successfully invoked on the same handle `immHandle`.

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 `immHandle` 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 Information Model Management 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 operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immmHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmInitialize(), saImmOmDispatch(), saImmOmFinalize()

4.3.3 saImmOmDispatch()

Prototype

```
SaAisErrorT saImmOmDispatch(
      SaImmHandleT immHandle,
      SaDispatchFlagsT dispatchFlags
);
```

Parameters

immHandle - [in] The handle which was obtained by a previous invocation of the saImmOmInitialize() function and which identifies this particular initialization of the Information Model Management Service. The SaImmHandleT type is defined in Section 4.2.1 on page 25.

dispatchFlags - [in] Flags that specify the callback execution behavior of the saImmOmDispatch() 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 [1].

Description

In the context of the calling thread, this function invokes pending callbacks for the handle immmHandle in a way that is 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 immmHandle 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 `immHandle` 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_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmInitialize()`, `saImmOmSelectionObjectGet()`, `saImmOmFinalize()`

4.3.4 saImmOmFinalize()

Prototype

```
SaAisErrorT saImmOmFinalize(
      SaImmHandleT immHandle
);
```

Parameters

`immHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

Description

The `saImmOmFinalize()` function closes the association represented by the `immHandle` parameter between the invoking process and the IMM Service. The process must have invoked `saImmOmInitialize()` before it invokes this function. A process must invoke this function once for each handle it acquired by invoking `saImmOmInitialize()`.

If the `saImmOmFinalize()` function completes successfully, it releases all resources acquired when `saImmOmInitialize()` was called. Moreover, it implicitly invokes:

- `saImmOmSearchFinalize()` on all search handles initialized with `immHandle` and not yet finalized.
- `saImmOmAccessorFinalize()` on all accessor handles initialized with `immHandle` and not yet finalized.
- `saImmOmAdminOwnerFinalize()` on all administrative owner handles initialized with `immHandle` and not yet finalized.

Furthermore, `saImmOmFinalize()` cancels all pending callbacks related to asynchronous operations performed with `immHandle`. Note that because the callback invocation is asynchronous, it is still possible that some callback calls are processed after this call returns successfully.

After `saImmOmFinalize()` returns successfully, the handle `immHandle` 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 `immHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

See Also

`saImmOmInitialize()`

4.4 Object Class Management

The following APIs are used to create and delete object classes. A caller can also use them to query the definition of an existing object class.

4.4.1 saImmOmClassCreate_2()

Prototype

```
SaAisErrorT saImmOmClassCreate_2(
      SaImmHandleT immHandle,
      const SaImmClassNameT className,
      SaImmClassCategoryT classCategory,
      const SaImmAttrDefinitionT_2 **attrDefinitions
);
```

Parameters

`immHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`className` - [in] The name of the object class to create. The `SaImmClassNameT` type is defined in Section 4.2.2 on page 26.

`classCategory` - [in] Category of the object class. The `SaImmClassCategoryT` type is defined in Section 4.2.4 on page 26.

`attrDefinitions` - [in] Pointer to a NULL-terminated array of pointers to definitions of the class attributes. The `SaImmAttrDefinitionT_2` type is defined in Section 4.2.7 on page 28.

Description

This function creates a new object class with the name `className`. The new object class can be a configuration or runtime object class, depending on the `classCategory` parameter setting.

Object class definitions are stored in a persistent manner by the IMM Service.

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 `immHandle` 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, the `attrDefinitions` parameter refers to a NULL or zero length attribute name, an invalid value type, an invalid default attribute value, or a set of attribute flags that are inconsistent with the class category specified by the `classCategory` parameter.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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_EXIST - An object class with a name identical to `className` already exists.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmInitialize()`

4.4.2 saImmOmClassDescriptionGet_2()

Prototype

```
SaAisErrorT saImmOmClassDescriptionGet_2(
      SaImmHandleT immHandle,
      const SaImmClassNameT className,
      SaImmClassCategoryT *classCategory,
      SaImmAttrDefinitionT_2 ***attrDefinitions
);
```

Parameters

`immHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`className` - [in] The name of the object class for which a description is requested. The `SaImmClassNameT` type is defined in Section 4.2.2 on page 26.

`classCategory` - [out] Pointer to an `SaImmClassCategoryT` structure to contain the category of the object class. The `SaImmClassCategoryT` type is defined in Section 4.2.4 on page 26.

`attrDefinitions` - [out] Pointer to a pointer to a NULL-terminated array of pointers to definitions of the class attributes. The `SaImmAttrDefinitionT_2` type is defined in Section 4.2.7 on page 28.

Description

This function returns a description of the object class identified by the name `className`.

The Information Model Management Service library allocates the memory to return the attribute definitions. When the calling process no longer needs to access the attribute definitions, the memory must be freed by calling the `saImmOmClassDescriptionMemoryFree_2()` 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_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 `immHandle` 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 Information Model Management Service library or the provider of the service is out of memory and cannot provide the service.

SA_AIS_ERR_NOT_EXIST - No object class exists with a name identical to `className`.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmInitialize()`, `saImmOmClassCreate_2()`, `saImmOmClassDescriptionMemoryFree_2()`

4.4.3 saImmOmClassDescriptionMemoryFree_2()

Prototype

```
SaAisErrorT saImmOmClassDescriptionMemoryFree_2(
      SaImmHandleT immHandle,
      SaImmAttrDefinitionT_2 **attrDefinitions
);
```

Parameters

`immHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`attrDefinitions` - [in] Pointer to a NULL-terminated array of pointers to attribute definitions to be freed. The `SaImmAttrDefinitionT_2` type is defined in Section 4.2.7 on page 28.

Description

This function deallocates the memory that was allocated by a previous call to the `saImmOmClassDescriptionGet_2()` function; this deallocation includes

- the memory areas containing the attribute definitions which are referred to by the pointers held in the NULL-terminated array referred to by `attrDefinitions` and
- the memory of the NULL-terminated array of pointers referred to by `attrDefinitions`.

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 `immHandle` 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_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmInitialize()`, `saImmOmClassCreate_2()`, `saImmOmClassDescriptionGet_2()`

4.4.4 saImmOmClassDelete()

Prototype

```
SaAisErrorT saImmOmClassDelete(
     SaImmHandleT immHandle,
     const SaImmClassNameT className
     );
```

Parameters

`immHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`className` - [in] Name of the object class to be deleted. The `SaImmClassNameT` type is defined in Section 4.2.2 on page 26.

Description

This function deletes the object class whose name is `className`, provided no objects of this class exist.

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 immHandle 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 Information Model Management 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 object class exists with a name identical to className.

SA_AIS_ERR_BUSY - The object class cannot be deleted as objects of this class still exist, or a request to create an object of this class has been added to a CCB.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmInitialize(), saImmOmClassCreate_2()

4.5 Object Search

The API functions in this section are used to perform **object search**, that is, to search for particular objects in the IMM Service object tree and also to obtain the values of some of their attributes.

In order to facilitate the management of the memory allocated by the IMM Service library to return the results of the search, the search is performed by using a **search iterator**.

The **search criteria** is specified when the search iterator is initialized. At initialization time, the attributes to be retrieved are also specified for each object that matches the search criteria. Then, each invocation of the iterator returns the object name and the specified attributes of the next object satisfying the search criteria.

The iteration is terminated by invoking the finalize API.

Every object which was created before the invocation of the `saImmOmSearchInitialize_2()` function and which matches the search criteria and has not been modified or deleted before the invocation of `saImmOmSearchFinalize()`, will be returned exactly once by the `saImmOmSearchNext_2()` search iterator. No other guarantees are made: objects that are created after the iteration is initialized, or modified, or deleted before the iteration is finalized, may or may not be returned by the search iterator.

4.5.1 saImmOmSearchInitialize_2()

Prototype

```
SaAisErrorT saImmOmSearchInitialize_2(
      SaImmHandleT immHandle,
      const SaNameT *rootName,
      SaImmScopeT scope,
      SaImmSearchOptionsT searchOptions,
      const SaImmSearchParametersT_2 *searchParam,
      const SaImmAttrNameT *attributeNames,
      SaImmSearchHandleT *searchHandle
);
```

Parameters

`immHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`rootName` - [`in`] Pointer to the name of the root object for the search. If set to NULL, the search starts at the root of the IMM Service tree. The `SaNameT` type is defined in [1].

`scope` - [`in`] Scope of the search. The `SaImmScopeT` type is defined in Section 4.2.11 on page 30.

`searchOptions` - [`in`] Specifies the type of criteria being used as well as which attribute values must be returned for each object matching the search criteria. The `SaImmSearchOptionsT` type is defined in Section 4.2.12 on page 31.

`searchParam` - [`in`] A pointer to the search parameters according to the search criteria specified in `searchOption`. The `SaImmSearchParametersT_2` type is defined in Section 4.2.13 on page 32.

`attributeNames` - [`in`] Pointer to a NULL-terminated array of attribute names for which values must be returned while iterating through all objects matching the search criteria.
Only used if the `SA_IMM_SEARCH_GET_SOME_ATTR` option has been set in the `searchOptions` parameter. The `attributeNames` pointer must be set to NULL otherwise.
The `SaImmAttrNameT` type is defined in Section 4.2.2 on page 26.

`searchHandle` - [`out`] Search handle used later to iterate through all objects that match the search criteria. The `SaImmSearchHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function initializes a search operation limited to a set of targeted objects identified by the `scope` parameter and the name to which the `rootName` parameter points.

The targeted set of objects is determined as follows:

- If `scope` is `SA_IMM_SUBLEVEL`, the scope of the operation is the object having the name to which `rootName` points and its direct children.
- If `scope` is `SA_IMM_SUBTREE`, the scope of the operation is the object having the name to which `rootName` points and the entire subtree rooted at that object.
- `SA_IMM_ONE` is not a valid value for the `scope` parameter.

If the `SA_IMM_SEARCH_ONE_ATTR` option is not set in the `searchOptions` parameter, the `searchOptions` parameter must be set to NULL. In this case, no selection criteria is applied for the search, and all objects in the defined scope will be retrieved by the search operation.

One and only one of the following three options must be set in the `searchOptions` parameter:

- `SA_IMM_SEARCH_GET_ALL_ATTR`,
- `SA_IMM_SEARCH_GET_NO_ATTR`, or
- `SA_IMM_SEARCH_GET_SOME_ATTR`.

This parameter specifies which attributes must be returned for each object matching the search criteria. If `SA_IMM_SEARCH_GET_SOME_ATTR` is set, the `attributeNames` parameter specifies the names of the attributes to be returned. If `SA_IMM_SEARCH_GET_SOME_ATTR` is not set, the `attributeNames` parameter must be set to NULL.

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 `immHandle` 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 Information Model Management 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 - The name to which `rootName` points is not the name of an existing object.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmInitialize()`

4.5.2 saImmOmSearchNext_2()

Prototype

```
SaAisErrorT saImmOmSearchNext_2(
      SaImmSearchHandleT searchHandle,
      SaNameT *objectName,
      SaImmAttrValuesT_2 ***attributes
);
```

Parameters

`searchHandle` - [in] Handle returned by `saImmOmSearchInitialize_2()`. The `SaImmSearchHandleT` type is defined in Section 4.2.1 on page 25.

`objectName` - [out] Pointer to the name of the next object matching the search criteria. The `SaNameT` type is defined in [1].

`attributes` - [out] Pointer to a pointer to a NULL-terminated array of pointers to data structures holding the names and values of the attributes (of the object whose name is pointed to by `objectName`) that were selected when the search was initialized. The `SaImmAttrValuesT_2` type is defined in Section 4.2.8 on page 29.

Description

This function is used to obtain the next object matching the search criteria.

The memory used to return the selected object attribute names and values is allocated by the library and will be deallocated at the next invocation of `saImmOmSearchNext_2()` or `saImmOmSearchFinalize()` for the same search handle.

If the handle `searchHandle` was not obtained by specifying `SA_IMM_SEARCH_GET_ALL_ATTR` or `SA_IMM_SEARCH_GET_SOME_ATTR` in the `searchOptions` parameter of the corresponding `saImmOmSearchInitialize_2()` call, no attribute names and values will be returned by this call, and the pointer to which the `attributes` parameter refers is set to NULL.

If one of the attributes requested by the search has no value or is a non-persistent runtime attribute, and no Object Implementer is registered for the object, only the attribute name is returned (`attrValuesNumber` is set to 0 and `attrValues` is set to NULL in the `SaImmAttrValuesT_2` data structure referred to by the corresponding entry in the array whose address is referred to by the `attributes` parameter).

Return Values

`SA_AIS_OK` - The function completed successfully.

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

`SA_AIS_ERR_TIMEOUT` - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it 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 `searchHandle` 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 Information Model Management 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` - All objects matching the search criteria have already been returned to the calling process. The caller can now invoke the

`saImmOmSearchFinalize()` function. Note that if no object matches the search criteria, this value is returned at the first invocation of `saImmOmSearchNext_2()`.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `searchHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmInitialize()`, `saImmOmSearchInitialize_2()`, `saImmOmSearchFinalize()`

4.5.3 saImmOmSearchFinalize()

Prototype

```
SaAisErrorT saImmOmSearchFinalize(
     SaImmSearchHandleT searchHandle
);
```

Parameters

`searchHandle` - [in] Handle returned by `saImmOmSearchInitialize_2()`. The `SaImmSearchHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function finalizes the search initialized by a previous call to `saImmOmSearchInitialize_2()`. It frees all memory previously allocated by that search, in particular, the memory used to return attribute names and values in the previous `saImmOmSearchNext_2()` invocation.

Returned 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 `searchHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `searchHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmInitialize()`, `saImmOmSearchInitialize_2()`, `saImmOmSearchNext_2()`

4.6 Object Access

The API functions in this section are used to perform **object access**, that is, to access the values of some attributes of an object already known by its name. Once an application has discovered the object hierarchy, it can use this interface to fetch some particular attribute values.

The **object accessor** is a way to facilitate the management of the memory allocated by the IMM Service library to return attribute names and values.

4.6.1 saImmOmAccessorInitialize()

Prototype

```
SaAisErrorT saImmOmAccessorInitialize(
      SaImmHandleT immHandle,
      SaImmAccessorHandleT *accessorHandle
);
```

Parameters

`immHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`accessorHandle` - [`out`] Pointer to the object accessor handle. The `SaImmAccessorHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function initializes an object accessor.

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 immHandle 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 Information Model Management 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 operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmInitialize()

4.6.2 saImmOmAccessorGet_2()

Prototype

```
SaAisErrorT saImmOmAccessorGet_2(
      SaImmAccessorHandleT accessorHandle,
      const SaNameT *objectName,
      const SaImmAttrNameT *attributeNames,
      SaImmAttrValuesT_2 ***attributes
);
```

Parameters

accessorHandle - [in] Object accessor handle. The SaImmAccessorHandleT type is defined in Section 4.2.1 on page 25.

objectName - [in] Pointer to the name of the object being accessed. The SaNameT type is defined in [1].

attributeNames - [in] Pointer to a NULL-terminated array of attribute names for which values must be returned. The SaImmAttrNameT type is defined in Section 4.2.2 on page 26.

`attributes` - [out] Pointer to a pointer to a NULL-terminated array of pointers to data structures containing the name and values of the attributes being accessed. The `SaImmAttrValuesT_2` type is defined in Section 4.2.8 on page 29.

Description

This function uses an object accessor to obtain the values assigned to some attributes of an object. If `attributeNames` is set to NULL, the values of all attributes of the object are returned.

If one of the requested attributes has no value or is a non-persistent runtime attribute, and there is no registered Object Implementer for the object, only the attribute name is returned (`attrValuesNumber` is set to 0 and `attrValues` is set to NULL in the `SaImmAttrValuesT_2` data structure specified by the `attributes` parameter).

The memory used to return the object attribute names and values is allocated by the library and will be deallocated at the next invocation of `saImmOmAccessorGet_2()` or `saImmOmAccessorFinalize()`.

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 `accessorHandle` 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 Information Model Management 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` - The name to which `objectName` points is not the name of an existing object, or any of the names specified by `attributeNames` does not exist for the object identified by the name to which `objectName` points.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle accessorHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmAccessorInitialize()

4.6.3 saImmOmAccessorFinalize()

Prototype

```
SaAisErrorT saImmOmAccessorFinalize(
     SaImmAccessorHandleT accessorHandle
);
```

Parameters

accessorHandle - [in] Object accessor handle. The SaImmAccessorHandleT type is defined in Section 4.2.1 on page 25.

Description

This function finalizes the object accessor and deallocates all memory previously allocated for this object accessor. In particular, this function frees the memory used to return the object attribute names and values during the previous invocation of saImmOmAccessorGet_2().

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 `accessorHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `accessorHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAccessorInitialize()`

4.7 Object Administration Ownership

Each object of the IMM Service may have at any time one and only one **administrative owner**, which has the ability to modify the object or invoke administrative operations on the object. The administrative owner is usually distinct from the Object Implementer. Establishing the **administrative ownership** of an object or a set of objects guarantees that a process unrelated with this administrative owner will not modify the objects concurrently.

As management operations may be performed by a set of cooperating processes, an administrative owner is identified by its name, and several processes may perform sequentially or concurrently administrative operations under the same **administrative owner name** (by initializing several administrative owner handles with the same name).

A process acting under that administrative owner name will typically release the administrative ownership on the objects. Note that this process need not necessarily be any of the one or more processes that set the administrative owner name of the objects. For recovery purposes, a process with appropriate privileges can also release the administrative ownership of a set of objects (by invoking the `saImmOmAdminOwnerClear()` function) without acting under the name of their current administrative owner.

Management applications are responsible for releasing the administrative ownership on objects when their management activities are completed.

4.7.1 saImmOmAdminOwnerInitialize()

Prototype

```
SaAisErrorT saImmOmAdminOwnerInitialize(
      SaImmHandleT immHandle,
      const SaImmAdminOwnerNameT adminOwnerName,
      SaBoolT releaseOwnershipOnFinalize,
      SaImmAdminOwnerHandleT *ownerHandle
);
```

Parameters

`immHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`adminOwnerName` - [`in`] Name of the administrative owner. The `SaImmAdminOwnerNameT` type is defined in Section 4.2.2 on page 26.

`releaseOwnershipOnFinalize` - [`in`] This parameter specifies how to release administrative ownerships that were acquired with the newly initialized handle `ownerHandle` when this handle is finalized. The `SaBoolT` type is defined in [1].

`ownerHandle` - [`out`] Pointer to the handle for the administrative owner. The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function initializes a handle for an administrative owner whose name is specified by `adminOwnerName`. All objects owned by an administrative owner have the attribute whose name is defined by the constant `SA_IMM_ATTR_ADMIN_OWNER_NAME` set to the name of the administrative owner. For objects without an administrative owner, that attribute does not exist.

If `releaseOwnershipOnFinalize` is set to `SA_TRUE`, the IMM Service automatically releases all administrative ownerships that were acquired with the newly initialized handle `ownerHandle` when this handle is finalized.
If `releaseOwnershipOnFinalize` is set to `SA_FALSE`, the IMM Service does not automatically release the ownership when the handle is finalized. In this case, if a management application fails while holding the administrative ownership on some objects, it is the responsibility of the recovery procedure of the failed application to release the administrative ownership on these objects.

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 immHandle 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 Information Model Management 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 operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmInitialize(), saImmOmAdminOwnerSet(), saImmOmAdminOwnerFinalize()

4.7.2 saImmOmAdminOwnerSet()

Prototype

```
SaAisErrorT saImmOmAdminOwnerSet(
      SaImmAdminOwnerHandleT ownerHandle,
      const SaNameT **objectNames,
      SaImmScopeT scope
);
```

Parameters

ownerHandle - [in] Administrative owner handle. The SaImmAdminOwnerHandleT type is defined in Section 4.2.1 on page 25.

objectNames - [in] Pointer to a NULL-terminated array of pointers to object names. The SaNameT type is defined in [1].

`scope` - [`in`] Scope of the operation. The `SaImmScopeT` type is defined in Section 4.2.11 on page 30.

Description

This function sets the administrative owner identified by `ownerHandle` as the owner of the set of objects identified by the `scope` and the `objectNames` parameters. This function can be used to acquire the administrative ownership of either configuration or runtime objects.

The targeted set of objects is determined as follows:

- If `scope` is `SA_IMM_ONE`, the scope of the operation are the objects having names specified by `objectNames`.
- If `scope` is `SA_IMM_SUBLEVEL`, the scope of the operation are the objects having names specified by `objectNames` and their direct children.
- If `scope` is `SA_IMM_SUBTREE`, the scope of the operation are the objects having names specified by `objectNames` and the entire subtrees rooted at these objects.

The operation fails if one of the targeted objects has already an administrative owner whose name is different from the name used to initialize `ownerHandle`. If the operation fails, the administrative owner of the targeted objects is not changed.

If the operation succeeds, the `SA_IMM_ATTR_ADMIN_OWNER_NAME` attribute of all targeted objects is set to the administrative owner name that was specified when `ownerHandle` was initialized.

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 `ownerHandle` 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 Information Model Management 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 - At least one of the names specified by `objectNames` is not the name of an existing object.

SA_AIS_ERR_EXIST - At least one of the objects targeted by this operation already has an administrative owner having a name different from the name used to initialize `ownerHandle`.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ownerHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAdminOwnerInitialize()`, `saImmOmAdminOwnerRelease()`, `saImmOmAdminOwnerClear()`

4.7.3 saImmOmAdminOwnerRelease()

Prototype

```
SaAisErrorT saImmOmAdminOwnerRelease(
      SaImmAdminOwnerHandleT ownerHandle,
      const SaNameT **objectNames,
      SaImmScopeT scope
);
```

Parameters

`ownerHandle` - [`in`] Administrative owner handle. The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

`objectNames` - [`in`] Pointer to a NULL-terminated array of pointers to object names. The `SaNameT` type is defined in [1].

`scope` - [`in`] Scope of the operation. The `SaImmScopeT` type is defined in Section 4.2.11 on page 30.

Description

This function releases the administrative owner of the set of objects identified by the `scope` and `objectNames` parameters.

The targeted set of objects is determined as follows:

- If `scope` is `SA_IMM_ONE`, the scope of the operation are the objects having names specified by `objectNames`.
- If `scope` is `SA_IMM_SUBLEVEL`, the scope of the operation are the objects having names specified by `objectNames` and their direct children.
- If `scope` is `SA_IMM_SUBTREE`, the scope of the operation are the objects having names specified by `objectNames` and the entire subtrees rooted at these objects.

If the operation succeeds, the `SA_IMM_ATTR_ADMIN_OWNER_NAME` attribute of all targeted objects is removed from the objects, and the continuation identifiers registered for these objects are all cleared.

The operation fails if an administrative operation is currently in progress on one of the targeted objects. An administrative operation is considered to be **in progress** on an object if the `SaImmOiAdminOperationCallbackT_2` Object Implementer's callback has been invoked for that operation and the Object Implementer is still registered but has not yet called `saImmOiAdminOperationResult()` to provide the operation results. The operation also fails if a change request for one of the targeted objects is included in a CCB that has not been finalized.

If the operation fails, the administrative owner of all objects in the given scope is not changed.

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 `ownerHandle` 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 Information Model Management 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 - At least one of the names specified by `objectNames` is not the name of an existing object, or at least one of the objects targeted by this operation is not owned by the administrative owner whose name was used to initialize `ownerHandle`.

SA_AIS_ERR_BUSY - An administrative operation is currently in progress on one of the targeted objects, or a change request for one of the targeted objects is included in a CCB that has not been applied or finalized.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ownerHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAdminOwnerInitialize()`, `saImmOmAdminOwnerSet()`

4.7.4 saImmOmAdminOwnerFinalize()

Prototype

```
SaAisErrorT saImmOmAdminOwnerFinalize(
      SaImmAdminOwnerHandleT ownerHandle
);
```

Parameters

`ownerHandle` - [in] Administrative owner handle. The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function releases `ownerHandle`. If `ownerHandle` has been initialized with the `releaseOwnershipOnFinalize` option set to SA_FALSE, this function neither affects registered continuation identifiers of any object nor releases the administrative ownership set on objects by using this handle.

If `ownerHandle` has been initialized with the `releaseOwnershipOnFinalize` option set to `SA_TRUE`, this operation also releases the administrative ownership that has been set on objects by using this handle and clears all continuation identifiers registered for these objects.

This function implicitly invokes `saImmOmCcbFinalize()` on all CCB handles initialized with `ownerHandle` and not yet finalized.

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 `ownerHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

`SA_AIS_ERR_UNAVAILABLE` - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ownerHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAdminOwnerInitialize()`, `saImmOmCcbInitialize()`

4.7.5 saImmOmAdminOwnerClear()

Prototype

```
SaAisErrorT saImmOmAdminOwnerClear(
      SaImmHandleT immHandle,
      const SaNameT **objectNames,
      SaImmScopeT scope
);
```

Parameters

`immHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOmInitialize()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmHandleT` type is defined in Section 4.2.1 on page 25.

`objectNames` - [`in`] Pointer to a NULL-terminated array of pointers to object names. The `SaNameT` type is defined in [1].

`scope` - [`in`] Scope of the operation. The `SaImmScopeT` type is defined in Section 4.2.11 on page 30.

Description

This function clears the administrative owner of the set of objects identified by the `scope` and `objectNames` parameters.
The targeted set of objects is determined as follows:

- If `scope` is `SA_IMM_ONE`, the scope of the operation are the objects having names specified by `objectNames`.
- If `scope` is `SA_IMM_SUBLEVEL`, the scope of the operation are the objects having names specified by `objectNames` and their direct children.
- If `scope` is `SA_IMM_SUBTREE`, the scope of the operation are the objects having names specified by `objectNames` and the entire subtrees rooted at these objects.

The operation succeeds even if some targeted objects do not have an administrative owner, or if the set of targeted objects have different administrative owners.

If the operation succeeds, the `SA_IMM_ATTR_ADMIN_OWNER_NAME` attribute of all targeted objects is removed from the objects, and the continuation identifiers registered for these objects are all cleared.

The operation fails if an administrative operation is currently in progress on one of the targeted objects (for the term “in progress”, see Section 4.7.3 on page 66), or if a change request for one of the targeted objects is included in a CCB that has not been applied or finalized.
If the operation fails, the administrative owner of all objects in the given scope is not changed.

This function is intended to be used only when recovering from situations where some management applications took ownership of some objects and did not release them.

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 `immHandle` 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_NOT_EXIST` - At least one of the names specified by `objectNames` is not the name of an existing object.

`SA_AIS_ERR_BUSY` - An administrative operation is currently in progress on one of the targeted objects, or a change request for one of the targeted objects is included in a CCB that has not been applied or finalized.

`SA_AIS_ERR_UNAVAILABLE` - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAdminOwnerInitialize()`, `saImmOmAdminOwnerSet()`, `saImmOmAdminOwnerRelease()`

4.8 Configuration Changes

All changes of IMM Service configuration objects are performed in the context of **configuration change bundles** (**CCB**). Once a CCB has been initialized, change requests can be added to a CCB. A **change request** can be a creation, a deletion, or a modification. Later on, when the CCB is applied, all **pending change requests** included in the CCB are applied with all-or-nothing semantics (either all change requests are applied or none are applied). The change requests are applied in the order they have been added to the CCB.

A CCB is associated with a single administrative owner, and all objects modified by change requests included in one CCB must have the same administrative owner as the CCB.

The IMM Service does not prevent applications from reading (by invoking `saImmOmSearchNext_2()` or `saImmOmAccessorGet_2()`) the attribute values of the objects modified by a CCB while a CCB is being applied. Therefore, it may happen, for example, that a search operation returns for some matching objects the values that their attributes had before the CCB was applied and for other objects the values that their attributes had after the CCB was applied. However, the IMM Service must guarantee that all CCB changes are applied atomically for each particular object. The attribute values returned by `saImmOmSearchNext_2()` or `saImmOmAccessorGet_2()` for a particular object must all be the values before the CCB was applied or all be the values after the CCB was applied (in other words, mixing old and new values is not allowed).

The IMM Service enforces the following limitation regarding concurrent management tasks for a particular object: at a given time, an object can be the target of either a single CCB or one or several administrative operations.

4.8.1 saImmOmCcbInitialize()

Prototype

```
SaAisErrorT saImmOmCcbInitialize(
      SaImmAdminOwnerHandleT ownerHandle,
      SaImmCcbFlagsT ccbFlags,
      SaImmCcbHandleT *ccbHandle
);
```

Parameters

`ownerHandle` - [`in`] Administrative owner handle. The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

`ccbFlags` - [`in`] CCB flags. The `SaImmCcbFlagsT` type is defined in Section 4.2.14 on page 33.

`ccbHandle` - [`out`] Pointer to the CCB handle. The `SaImmCcbHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function initializes a new CCB and returns a handle for it. The CCB is initialized as empty (it contains no change requests).

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 `ownerHandle` 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 Information Model Management 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 operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ownerHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAdminOwnerInitialize()`

4.8.2 saImmOmCcbObjectCreate_2()

Prototype

```
SaAisErrorT saImmOmCcbObjectCreate_2(
     SaImmCcbHandleT ccbHandle,
     const SaImmClassNameT className,
     const SaNameT *parentName,
     const SaImmAttrValuesT_2 **attrValues
);
```

Parameters

`ccbHandle` - [`in`] CCB handle. The `SaImmCcbHandleT` type is defined in Section 4.2.1 on page 25.

`className` - [`in`] Object name class. The `SaImmClassNameT` type is defined in Section 4.2.2 on page 26.

`parentName` - [`in`] Pointer to the name of the parent of the new object. The `SaNameT` type is defined in [1].

`attrValues` - [`in`] Pointer to a NULL-terminated array of pointers to attribute descriptors. The `SaImmAttrValuesT_2` type is defined in Section 4.2.8 on page 29.

Description

This function adds to the CCB identified by its handle `ccbHandle` a request to create a new IMM Service object. Once this new object is successfully created, it will be automatically owned by the administrative owner of the CCB. The new object is created as a child of the object designated by the name to which `parentName` points. If `parentName` is set to NULL, the new object is created as a top level object.

This function can be used only to create configuration objects. The attributes specified by the array to which `attrValues` refers must match the object class definition. Only configuration and persistent runtime attributes can be specified by this array.

Attributes named `SA_IMM_ATTR_CLASS_NAME`, `SA_IMM_ATTR_ADMIN_OWNER_NAME`, and `SA_IMM_ATTR_IMPLEMENTER_NAME` cannot be specified by the `attrValues` descriptors, as these attributes are automatically set by the IMM Service.

The creation will only be performed when the CCB is applied. However, the IMM Service invokes any existing Object Implementer synchronously to validate the creation request and may return an error if this creation is not a valid operation.

The IMM Service adds an `SA_IMM_ATTR_CLASS_NAME` attribute to the new object; the value of this attribute contains the name of the object class as specified by the `className` parameter.

If the parent object is not administratively owned by the administrative owner of the CCB, this function fails and returns `SA_AIS_ERR_BAD_OPERATION`.

If this function returns an error, the creation request has not been added to the CCB.

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 `ccbHandle` 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:

- the `className` parameter specifies a runtime object class,
- there is no valid RDN attribute specified for the new object,
- all of the configuration attributes required at object creation are not provided by the caller,
- the `attrValues` parameter includes:
  - non-persistent runtime attributes,
  - attributes that are not defined for the specified class,
  - attributes with values that do not match the defined value type for the attribute, and
  - multiple values for a single-valued attribute.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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_BAD_OPERATION - The parent object is not administratively owned by the administrative owner of the CCB, or the creation of the object has been rejected by its Object Implementer.

SA_AIS_ERR_NOT_EXIST - This value is returned due to one or more of the following reasons:

- The name to which the `parentName` parameter points is not the name of an existing object.
- The `className` parameter is not the name of an existing object class.
- One or more of the attributes specified by `attrValues` are not valid attribute names for `className`.
- There is no registered Object Implementer for the object to be created, and the CCB has been initialized with the SA_IMM_CCB_REGISTERED_OI flag set.

SA_AIS_ERR_EXIST - An object with the same name already exists.

SA_AIS_ERR_FAILED_OPERATION - The operation failed because the CCB has been aborted due to the registration of an Object Implementer or a problem with one of the registered Object Implementers. The CCB is now empty.

SA_AIS_ERR_NAME_TOO_LONG - The size of the new object's DN is greater than SA_MAX_NAME_LENGTH.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle ccbHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmCcbInitialize(), saImmOmCcbApply()

4.8.3 saImmOmCcbObjectDelete()

Prototype

```
SaAisErrorT saImmOmCcbObjectDelete(
      SaImmCcbHandleT ccbHandle,
      const SaNameT *objectName
);
```

Parameters

ccbHandle - [in] CCB handle. The SaImmCcbHandleT type is defined in Section 4.2.1 on page 25.

objectName - [in] Pointer to the object name. The SaNameT type is defined in [1].

Description

This function adds to the CCB identified by its handle ccbHandle a request to delete the configuration object designated by the name to which the objectName parameter points and the entire subtree of configuration objects rooted at that object.

This operation fails if one of the targeted objects is not a configuration object that is administratively owned by the administrative owner of the CCB. It also fails if one of the targeted objects has some registered continuation identifiers.

The deletion will only be performed when the CCB is applied. However, the IMM Service invokes any existing Object Implementer synchronously to validate the deletion request and may return an error if the deletion is not a valid operation.

If this function returns an error, the deletion request has not been added to the CCB.

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 ccbHandle 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 Information Model Management 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_BAD_OPERATION - This value is returned due to one or more of the following reasons:

- at least one of the targeted objects is not a configuration object that is owned by the administrative owner of the CCB;
- at least one of the targeted objects has some registered continuation identifiers;
- the Object Implementer has rejected the deletion of at least one of the targeted objects.

SA_AIS_ERR_NOT_EXIST - This value is returned due to one or both of the following reasons:

- The name to which the objectName parameter points is not the name of an existing object.
- There is no registered Object Implementer for at least one of the objects targeted by this operation, and the CCB has been initialized with the SA_IMM_CCB_REGISTERED_OI flag set.

SA_AIS_ERR_BUSY - At least one of the targeted objects is already the target of an administrative operation or of a change request in another CCB.

SA_AIS_ERR_FAILED_OPERATION - The operation failed because the CCB has been aborted due to the registration of an Object Implementer or a problem with one of the registered Object Implementers. The CCB is now empty.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle ccbHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmCcbInitialize(), saImmOmCcbApply()

4.8.4 saImmOmCcbObjectModify_2()

Prototype

```
SaAisErrorT saImmOmCcbObjectModify_2(
      SaImmCcbHandleT ccbHandle,
      const SaNameT *objectName,
      const SaImmAttrModificationT_2 **attrMods
);
```

Parameters

ccbHandle - [in] CCB handle. The SaImmCcbHandleT type is defined in Section 4.2.1 on page 25.

objectName - [in] Pointer to the name of the object to be modified. The SaNameT type is defined in [1].

attrMods - [in] Pointer to a NULL-terminated array of pointers to descriptors of the modifications to perform. The SaImmAttrModificationT_2 type is defined in Section 4.2.10 on page 30.

Description

This function adds to the CCB identified by its handle ccbHandle a request to modify configuration attributes of an IMM Service object. Only writable configuration attributes can be modified (SA_IMM_ATTR_WRITABLE).

This operation fails if the targeted object is not administratively owned by the administrative owner of the CCB.

The modify request will only be performed when the CCB is applied. However, the IMM Service invokes any existing Object Implementer synchronously to validate the

modify request and may return an error if the requested modifications are not allowed.

Attributes named SA_IMM_ATTR_CLASS_NAME, SA_IMM_ATTR_ADMIN_OWNER_NAME, and SA_IMM_ATTR_IMPLEMENTER_NAME cannot be modified.

If this function returns an error, the modify request has not been added to the CCB.

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 ccbHandle 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, the attrMods parameter includes:

- runtime attributes,
- attributes that are not defined for the specified class,
- attributes with values that do not match the defined value type for the attribute,
- a new value for the RDN attribute,
- attributes that cannot be modified,
- multiple values or additional values for a single-valued attribute.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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_BAD_OPERATION - The modified object is not a configuration object owned by the administrative owner of the CCB, or its Object Implementer has rejected the modification.

SA_AIS_ERR_NOT_EXIST - This value is returned due to one or more of the following reasons:

- The name to which the objectName parameter points is not the name of an existing object.
- One or more attribute names specified by the attrMods parameter are not valid for the object class.
- There is no registered Object Implementer for the object designated by the name to which the objectName parameter points, and the CCB has been initialized with the SA_IMM_CCB_REGISTERED_OI flag set.

SA_AIS_ERR_NOT_EXIST - The name to which the objectName parameter points is not the name of an existing object, or one or more attribute names specified by the attrMods parameter are not valid for the object class.

SA_AIS_ERR_BUSY - The object designated by the name to which objectName points is already the target of an administrative operation or of a change request in another CCB.

SA_AIS_ERR_FAILED_OPERATION - The operation failed because the CCB has been aborted due to the registration of an Object Implementer or a problem with one of the registered Object Implementers. The CCB is now empty.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle ccbHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmCcbInitialize(), saImmOmCcbApply()

4.8.5 saImmOmCcbApply()

Prototype

```
SaAisErrorT saImmOmCcbApply(
      SaImmCcbHandleT ccbHandle
);
```

Parameters

`ccbHandle` - [`in`] CCB handle. The `SaImmCcbHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function applies all requests included in the configuration change bundle identified by its handle `ccbHandle`. The requests are applied with all-or-nothing semantics, that is, either all requests are applied or none are applied. All requests are applied in the order in which they have been added to the CCB.

Any existing Object Implementer involved by the change requests contained in the CCB is invoked to apply the changes. The Object Implementers are responsible for checking that the set of requested changes is valid.

This operation fails if the administrative ownership of an object targeted by this CCB has changed since the change was added to the CCB, and the new administrative owner of the object is not anymore the administrative owner of the CCB.

When this call returns with success or failure, all requests included in the CCB when the call was issued have been removed. The CCB is empty and can be populated again with change requests belonging to the same administrative owner.

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 `ccbHandle` 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 Information Model Management 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_BAD_OPERATION - The changes requested do not constitute a valid set of changes.

SA_AIS_ERR_FAILED_OPERATION - The operation failed because the CCB has been aborted due to the registration of an Object Implementer or a problem with one of the registered Object Implementers. The CCB is now empty.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ccbHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmCcbInitialize()`, `saImmOmCcbObjectCreate_2()`, `saImmOmCcbObjectDelete()`, `saImmOmCcbObjectModify_2()`

4.8.6 saImmOmCcbFinalize()

Prototype

```
SaAisErrorT saImmOmCcbFinalize(
     SaImmCcbHandleT ccbHandle
);
```

Parameters

`ccbHandle` - [`in`] CCB handle. The `SaImmCcbHandleT` type is defined in Section 4.2.1 on page 25.

Description

This function finalizes the CCB identified by `ccbHandle`.

All change requests contained in the CCB are removed without being applied.

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 `ccbHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

`SA_AIS_ERR_UNAVAILABLE` - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ccbHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmCcbInitialize()`

4.9 Administrative Operations Invocation

Processes can invoke **administrative operations** on IMM objects by using the `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()` API functions.

The IMM Service transfers the administrative operation to the Object Implementer by invoking its `SaImmOiAdminOperationCallbackT_2` registered callback, passing along all parameters provided to the `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()` API functions.

If the invoking process exits (due to a failure, for example) before the administrative operation completes, the IMM allows another process to carry over the invocation and wait for its result by invoking the `saImmOmAdminOperationContinue()` or `saImmOmAdminOperationContinueAsync()` API functions. These functions are called **continuation functions**.
The administrative operation may have completed when a continuation function is called. In this case, the continuation function can just fetch the result of the administrative operation that has been buffered by the IMM Service.
An Object Implementer is not aware of the continuation functions, the support of which is entirely handled by the IMM Service.

In order for an administrative operation to be carried over (or continued), the original invoker of the administrative operation must provide a nonzero **continuation identifier**. The continuation identifier must be unique on a per-object basis. It is the responsibility of the process that initiates the administrative operation to store the continuation identifier in a location where a process that may need to continue the operation can access it. The location where a continuation identifier is stored is not specified by the IMM Service and is application-specific; checkpoints or files may be used to store continuation identifiers.
The IMM registers a particular continuation identifier with an object when an administrative operation is invoked on the object by a call to `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()`. The continuation identifier will stay registered with the object until explicitly cleared with `saImmOmAdminOperationContinuationClear()`, or until the administrative ownership on the object that was in effect at the time of the invocation of `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()`is released.
As long as a continuation identifier stays registered with the object, it is said to be a **registered continuation identifier**.
Continuation identifiers are not persistent, and they are all cleared when the IMM Service is terminated.

The IMM Service does not allow concurrent continuation operations for the same continuation identifier. As a consequence, `saImmOmAdminOperationContinue()` and `saImmOmAdminOperationContinueAsync()` will fail and return an SA_AIS_ERR_EXIST error if

- the administrative owner handle that was used when the continuation identifier for an object was first provided in an invocation of `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()` is still valid, or if
- the administrative owner handle that was used when the continuation identifier for an object was last provided in an invocation of any of the two continuation functions is still valid.

4.9.1 saImmOmAdminOperationInvoke_2(), saImmOmAdminOperationInvokeAsync_2()

Prototype

```
SaAisErrorT saImmOmAdminOperationInvoke_2(
      SaImmAdminOwnerHandleT ownerHandle,
      const SaNameT *objectName,
      SaImmContinuationIdT continuationId,
      SaImmAdminOperationIdT operationId,
      const SaImmAdminOperationParamsT_2 **params,
      SaAisErrorT *operationReturnValue,
      SaTimeT timeout
);
```

Parameters

`ownerHandle` - [in] Administrative owner handle.
The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

`objectName` - [in] Pointer to the object name. The `SaNameT` type is defined in [1].

`continuationId` - [in] Continuation identifier for this particular invocation of the administrative operation. In case `ownerHandle` is finalized before the process retrieved the result of the operation, the result of the operation may be obtained by specifying another valid administrative owner handle in an invocation of one of the `saImmOmAdminOperationContinue()` or `saImmOmAdminOperationContinueAsync()` functions.

The `continuationId` parameter must be set to 0 if the invocation shall not be continued. The `SaImmContinuationIdT` type is defined in Section 4.2.15 on page 33.

`operationId` - [`in`] Identifier of the administrative operation.
The `SaImmAdminOperationIdT` type is defined in Section 4.2.16 on page 33.

`params` - [`in`] Pointer to a NULL-terminated array of pointers to parameter descriptors. The `SaImmAdminOperationParamsT_2` type is defined in Section 4.2.17 on page 34.

`operationReturnValue` - [`out`] Pointer to the value returned by the Object Implementer for the invoked operation. This value is specific to the administrative operation being performed, and it is valid only if the `saImmOmAdminOperationInvoke_2()` function returns `SA_AIS_OK`. For more details about this value, refer to the Object Implementer administrative operation description. The `SaAisErrorT` type is defined in [1].

`timeout` - [`in`] The `saImmOmAdminOperationInvoke_2()` invocation is considered to have failed if it does not complete by the time specified.
The `SaTimeT` type is defined in [1].

Prototype

```
SaAisErrorT saImmOmAdminOperationInvokeAsync_2(
      SaImmAdminOwnerHandleT ownerHandle,
      SaInvocationT invocation,
      const SaNameT *objectName,
      SaImmContinuationIdT continuationId,
      SaImmAdminOperationIdT operationId,
      const SaImmAdminOperationParamsT_2 **params
);
```

Parameters

`ownerHandle` - [`in`] Administrative owner handle.
The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

`invocation` - [`in`] Used to match this invocation of `saImmOmAdminOperationInvokeAsync_2()` with the corresponding invocation of the `SaImmOmAdminOperationInvokeCallbackT` callback.
The `SaInvocationT` type is defined in [1].

`objectName` - [`in`] Pointer to the object name. The `SaNameT` type is defined in [1].

`continuationId` - [`in`] Continuation identifier for this particular invocation of the administrative operation. In case `ownerHandle` is finalized before the process retrieved the result of the operation, the result of the operation may be obtained by specifying another valid administrative owner handle in an invocation of one of the `saImmOmAdminOperationContinue()` or `saImmOmAdminOperationContinueAsync()` functions.
The `continuationId` parameter must be set to 0 if the invocation shall not be continued. The `SaImmContinuationIdT` type is defined in Section 4.2.15 on page 33.

`operationId` - [`in`] Identifier of the administrative operation.
The `SaImmAdminOperationIdT` type is defined in Section 4.2.16 on page 33.

`params` - [`in`] Pointer to a NULL-terminated array of pointers to parameter descriptors. The `SaImmAdminOperationParamsT_2` type is defined in Section 4.2.17 on page 34.

Description

Using the IMM Service as an intermediary, these two functions request the implementer of the object designated by the name to which `objectName` points to perform an administrative operation characterized by `operationId` on that object. Administrative operations can be performed on configuration or runtime objects.

Each descriptor pointed to by an element of the array of pointers to which the `params` parameter points represents an input parameter of the administrative operation to execute.

The function `saImmOmAdminOperationInvoke_2()` is the synchronous variant and returns only when the Object Implementer has successfully completed the execution of the administrative operation, or when an error has been detected by the IMM Service or the Object Implementer.

The function `saImmOmAdminOperationInvokeAsync_2()` is the asynchronous variant; it returns as soon as the IMM Service has registered the request to be transmitted to the Object Implementer. If the IMM Service detects an error while registering the request, an error is immediately returned, and no further invocation of the `SaImmOmAdminOperationInvokeCallbackT` callback must be expected for this invocation of `saImmOmAdminOperationInvokeAsync_2()`. If no error is detected by the IMM Service while registering the request, the invocation of `saImmOmAdminOperationInvokeAsync_2()` completes successfully, and a later invocation of the `SaImmOmAdminOperationInvokeCallbackT` callback will occur to indicate the success or failure of the administrative operation on the target object.

If the administrative owner handle `ownerHandle` becomes finalized before the process could retrieve the result of the administrative operation (returned by `saImmOmAdminOperationInvoke_2()` or passed to `SaImmOmAdminOperationInvokeCallbackT`), the current process or another process may invoke one of the functions `saImmOmAdminOperationContinue()` or `saImmOmAdminOperationContinueAsync()` on a valid administrative owner handle to continue the operation, if necessary, and retrieve its result.

Return Values

`SA_AIS_OK` - The function completed successfully.

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

`SA_AIS_ERR_TIMEOUT` - An implementation-dependent timeout occurred, or the timeout, specified by the `timeout` parameter, occurred before the call could complete. It is unspecified whether the call succeeded or whether it 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 `ownerHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

`SA_AIS_ERR_INIT` - The corresponding previous invocation of `saImmOmInitialize()` to initialize the IMM Service and obtain the IMM Service handle (with which the handle `ownerHandle` was obtained by invoking `saImmOmAdminOwnerInitialize()`) was incomplete, since the `SaImmOmAdminOperationInvokeCallbackT` callback function was missing. This return value applies only to the `saImmOmAdminOperationInvokeAsync_2()` function.

`SA_AIS_ERR_INVALID_PARAM` - A parameter is not set correctly.

`SA_AIS_ERR_NO_MEMORY` - Either the Information Model Management Service library or its library 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_BAD_OPERATION` - The object designated by the name to which `objectName` points is not owned by the administrative owner associated with `ownerHandle`.

`SA_AIS_ERR_NOT_EXIST` - The name to which the `objectName` parameter points is not the name of an existing object, or there is no registered Object Implementer for this object.

SA_AIS_ERR_EXIST - The object designated by the name to which `objectName` points has already a registered continuation identifier identical to `continuationId`.

SA_AIS_ERR_BUSY - The object designated by the name to which `objectName` points is already the target of a change request in a CCB.

SA_AIS_ERR_FAILED_OPERATION - The operation failed due to a problem with the Object Implementer.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ownerHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAdminOwnerInitialize()`,
`SaImmOmAdminOperationInvokeCallbackT`,
`saImmOmAdminOperationContinue()`,
`saImmOmAdminOperationContinueAsync()`,
`saImmOmAdminOperationContinueClear()`

4.9.2 SaImmOmAdminOperationInvokeCallbackT

Prototype

```
typedef void (*SaImmOmAdminOperationInvokeCallbackT) (
      SaInvocationT invocation,
      SaAisErrorT operationReturnValue,
      SaAisErrorT error
);
```

Parameters

`invocation` - [in] Used to match this callback invocation to the corresponding previous invocation of either `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()`, depending on which of these functions was called last. The `SaInvocationT` type is defined in [1].

`operationReturnValue` - [`in`] Value returned by the Object Implementer for the administrative operation requested in the corresponding previous invocation of either `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()`, depending on which of these functions was called last.
This value is specific to the administrative operation being performed, and it is valid only if the `error` parameter is set to `SA_AIS_OK`. For more details about this value, refer to the Object Implementer administrative operation description.
The `SaAisErrorT` type is defined in [1].

`error` - [`in`] Indicates whether the IMM Service succeeded or not to invoke the Object Implementer.
The `SaAisErrorT` type is defined in [1].
The returned values are:

- `SA_AIS_OK` - The function completed successfully.
- `SA_AIS_ERR_LIBRARY` - An unexpected problem occurred in the library (such as corruption). The library cannot be used anymore.
- `SA_AIS_ERR_TIMEOUT` - An implementation-dependent timeout occurred before the call could complete. It is unspecified whether the call succeeded or whether it 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 `ownerHandle` in the corresponding invocation of either `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()` (depending on which of these functions was called last) 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 IMM 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_BAD_OPERATION` - The object designated by the name to which the `objectName` parameter points in the corresponding invocation of either `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()` (depending on which of these functions was called last) is not owned by the administrative owner associated with `ownerHandle`.

- SA_AIS_ERR_NOT_EXIST - The name to which the `objectName` parameter points in the corresponding invocation of either `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()` (depending on which of these functions was called last) is not the name of an existing object, or there is no registered Object Implementer for this object.
- SA_AIS_ERR_EXIST - Two cases must be distinguished:
  - This callback has been requested by the `saImmOmAdminOperationInvokeAsync_2()` call: the object designated by the name to which the `objectName` parameter points in the `saImmOmAdminOperationInvokeAsync_2()` call has already a registered continuation identifier identical to `continuationId`.
  - This callback has been requested by the `saImmOmAdminOperationContinueAsync()` call: the object designated by the name to which the `objectName` parameter points in the `saImmOmAdminOperationContinueAsync()` call has already a registered continuation identifier identical to `continuationId`, and the administrative owner handle specified for this object in a preceding call to one of the following functions (depending on which of these four functions was called last) has not yet been finalized:
    - either `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()`, or
    - either `saImmOmAdminOperationContinue()` or `saImmOmAdminOperationContinueAsync()`
- SA_AIS_ERR_BUSY - The object designated by the name to which the `objectName` parameter points in the corresponding invocation of either `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()` (depending on which of these functions was called last) is already the target of a change request in a CCB.
- SA_AIS_ERR_FAILED_OPERATION - The operation failed due to a problem with the Object Implementer.
- SA_AIS_ERR_UNAVAILABLE - The operation requested by either the corresponding `saImmOmAdminOperationContinueAsync()` call or the corresponding `saImmOmAdminOperationInvokeAsync_2()` call is unavailable on this cluster node due to one of the two reasons:
  - the cluster node has left the cluster membership;
  - the cluster node has rejoined the cluster membership, but the handle `ownerHandle` specified in either the corresponding `saImmOmAdminOperationContinueAsync()` call or the corresponding

`saImmOmAdminOperationInvokeAsync_2()` call was acquired before the cluster node left the cluster membership.

Description

The IMM Service invokes this callback function when the operation requested by the corresponding invocation of either `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()` (depending on which of these functions was called last) completes successfully, or an error is detected.

This callback is invoked in the context of a thread calling `saImmOmDispatch()` on the handle `immHandle` that was used to initialize the `ownerHandle` specified in one of the corresponding functions `saImmOmAdminOperationInvokeAsync_2()` or `saImmOmAdminOperationContinueAsync()`, depending on which of these functions was called last.

Return Values

None

See Also

`saImmOmAdminOwnerInitialize()`, `saImmOmDispatch()`, `saImmOmAdminOperationInvokeAsync_2()`, `saImmOmAdminOperationContinue()`, `saImmOmAdminOperationContinueAsync()`, `saImmOmAdminOperationContinueClear()`

4.9.3 saImmOmAdminOperationContinue(), saImmOmAdminOperationContinueAsync()

Prototype

```
SaAisErrorT saImmOmAdminOperationContinue(
      SaImmAdminOwnerHandleT ownerHandle,
      const SaNameT *objectName,
      SaImmContinuationIdT continuationId,
      SaAisErrorT *operationReturnValue
);
```

Parameters

`ownerHandle` - [`in`] Administrative owner handle.
The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

`objectName` - [`in`] Pointer to the object name. The `SaNameT` type is defined in [1].

`continuationId` - [`in`] Identifies the corresponding previous invocation of `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()`.
The `SaImmContinuationIdT` type is defined in Section 4.2.15 on page 33.

`operationReturnValue` - [`out`] Pointer to the value returned by the Object Implementer for the operation requested by the corresponding previous call to `saImmOmAdminOperationInvoke_2()` or to `saImmOmAdminOperationInvokeAsync_2()`. The value returned by the Object Implementer is specific to the administrative operation being performed, and it is valid only if the `saImmOmAdminOperationContinue()` function returns `SA_AIS_OK`. For more details about this value, refer to the Object Implementer administrative operation description. The `SaAisErrorT` type is defined in [1].

Prototype

```
SaAisErrorT saImmOmAdminOperationContinueAsync(
      SaImmAdminOwnerHandleT ownerHandle,
      SaInvocationT invocation,
      const SaNameT *objectName,
      SaImmContinuationIdT continuationId
);
```

Parameters

`ownerHandle` - [in] Administrative owner handle.
The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

`invocation` - [in] Used to match this invocation of `saImmOmAdminOperationContinueAsync()` with the corresponding invocation of the `SaImmOmAdminOperationInvokeCallbackT` callback.
The `SaInvocationT` type is defined in [1].

`objectName` - [in] Pointer to the object name. The `SaNameT` type is defined in [1].

`continuationId` - [in] Identifies the corresponding previous invocation of `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()`.
The `SaImmContinuationIdT` type is defined in Section 4.2.15 on page 33.

Description

These two functions allow a process to take over the continuation of an administrative operation that had been initiated with a particular administrative handle but did not complete before the handle was finalized (explicitly or as a side effect of the process termination).

The process taking over the operation may use a synchronous or asynchronous continue operation regardless of whether the respective administrative operation was initiated by invoking `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()`.

The function `saImmOmAdminOperationContinue()` is the synchronous variant and returns only when the Object Implementer has successfully completed the execution of the administrative operation, or when an error has been detected by the IMM Service or the Object Implementer.

The function saImmOmAdminOperationContinueAsync() is the asynchronous variant; it returns as soon as the IMM Service has registered the request. If the IMM Service detects an error while registering the request, an error is immediately returned, and no further invocation of the SaImmOmAdminOperationInvokeCallbackT callback must be expected for this invocation of saImmOmAdminOperationContinueAsync(). If no error is detected by the IMM Service while registering the request, the invocation of saImmOmAdminOperationInvokeAsync_2() completes successfully, and the SaImmOmAdminOperationInvokeCallbackT callback will be invoked later to indicate the success or failure of the administrative operation on the target object.

The object name pointed to by objectName and the continuation identifier continuationId must be the same that were supplied in a corresponding previous invocation of saImmOmAdminOperationInvoke_2(), saImmOmAdminOperationInvokeAsync_2().

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 ownerHandle is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_INIT - The corresponding previous invocation of saImmOmInitialize() to initialize the IMM Service and obtain the IMM Service handle (with which the handle ownerHandle was obtained by invoking saImmOmAdminOwnerInitialize()) was incomplete, since the SaImmOmAdminOperationInvokeCallbackT callback function was missing. This return value only applies to the saImmOmAdminOperationContinueAsync() function.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management Service library or its library 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_BAD_OPERATION - The object designated by the name to which the objectName parameter points is not owned by the administrative owner associated with ownerHandle.

SA_AIS_ERR_NOT_EXIST - This error is returned if one of the following conditions apply:

- The name to which the objectName parameter points is not the name of an existing object, or there is no registered Object Implementer for this object.
- The continuationId parameter is not a valid continuation identifier (that is, it is not a registered continuation identifier) for the object whose name is pointed to by the objectName parameter.

SA_AIS_ERR_EXIST - The object designated by the name to which the objectName parameter points has already a registered continuation identifier identical to continuationId, and the administrative owner handle specified for this object in the last call to one of the following functions (depending on which of these four functions was called last) has not yet been finalized:

- either saImmOmAdminOperationInvoke_2() or saImmOmAdminOperationInvokeAsync_2(), or
- either saImmOmAdminOperationContinue() or saImmOmAdminOperationContinueAsync()

SA_AIS_ERR_BUSY - The object designated by the name to which the objectName parameter points is already the target of a change request in a CCB.

SA_AIS_ERR_FAILED_OPERATION - The operation failed due to a problem with the Object Implementer.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle ownerHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOmAdminOwnerInitialize(), saImmOmAdminOperationInvoke_2(), saImmOmAdminOperationInvokeAsync_2(), SaImmOmAdminOperationInvokeCallbackT, saImmOmAdminOperationContinueClear()

4.9.4 saImmOmAdminOperationContinueClear()

Prototype

```
SaAisErrorT saImmOmAdminOperationContinuationClear(
      SaImmAdminOwnerHandleT ownerHandle,
      const SaNameT *objectName,
      SaImmContinuationIdT continuationId
);
```

Parameters

`ownerHandle` - [in] Administrative owner handle.
The `SaImmAdminOwnerHandleT` type is defined in Section 4.2.1 on page 25.

`objectName` - [in] Pointer to the object name. The `SaNameT` type is defined in [1].

`continuationId` - [in] The continuation identifier that was supplied in the corresponding previous invocation of `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()`.
The `SaImmContinuationIdT` type is defined in Section 4.2.15 on page 33.

Parameters

Description

This function instructs the IMM Service to clear all information kept to allow the continuation of the administrative operation identified by `continuationId` for the object whose name is pointed to by `objectName` and the administrative owner identified by `ownerHandle`. After successful completion of this function, the `continuationId` identifier is cleared, that is, it is no longer a registered continuation identifier.

The object name pointed to by `objectName` and the continuation identifier `continuationId` must be the same that were supplied in the corresponding previous invocation of `saImmOmAdminOperationInvoke_2()`, `saImmOmAdminOperationInvokeAsync_2()`.

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 `ownerHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management Service library or its library 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_BAD_OPERATION - The object designated by the name to which the `objectName` parameter points is not owned by the administrative owner associated with `ownerHandle`.

SA_AIS_ERR_NOT_EXIST - This error is returned if one of the following conditions apply:

- The name to which the `objectName` parameter points is not the name of an existing object, or there is no registered Object Implementer for this object.
- The `continuationId` parameter is not a valid continuation identifier (that is, it is not a registered continuation identifier) for the object whose name is pointed to by the `objectName` parameter.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `ownerHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOmAdminOperationInvoke_2()`,
`saImmOmAdminOperationInvokeAsync_2()`,
`SaImmOmAdminOperationInvokeCallbackT`,
`saImmOmAdminOperationContinue()`,
`saImmOmAdminOperationContinueAsync()`

5 IMM Service - Object Implementer API Specification

5.1 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 IMM Service Object Implementer API:

```
#include <saImmOi.h>
```

To use the IMM Service Object Implementer API, an application must be bound with the following library:

```
libSaImmOi.so
```

5.2 Type Definitions

5.2.1 IMM Service Handle

The following handle is used by IMM Service Object Implementer API functions:

```
typedef SaUint64T SaImmOiHandleT;
```

5.2.2 SaImmOiImplementerNameT

The `SaImmOiImplementerNameT` type represents an Object Implementer name; it points to an UTF-8 encoded character string, terminated by the NULL character.

```
typedef SaStringT SaImmOiImplementerNameT;
```

5.2.3 SaImmOiCcbIdT

```
typedef SaUint64T SaImmOiCcbIdT;
```

This type is used in the IMM Service Object Implementer APIs to identify a particular configuration change bundle (CCB).

5.2.4 SaImmOiCallbacksT_2

The SaImmOiCallbacksT_2 structure defines the set of callbacks a process implementing IMM Service objects can provide to the IMM Service at initialization time.

```
typedef struct {
     SaImmOiAdminOperationCallbackT_2
          saImmOiAdminOperationCallback;
     SaImmOiCcbAbortCallbackT
          saImmOiCcbAbortCallback;
     SaImmOiCcbApplyCallbackT
          saImmOiCcbApplyCallback;
     SaImmOiCcbCompletedCallbackT
          saImmOiCcbCompletedCallback;
     SaImmOiCcbObjectCreateCallbackT_2
          saImmOiCcbObjectCreateCallback;
     SaImmOiCcbObjectDeleteCallbackT
          saImmOiCcbObjectDeleteCallback;
     SaImmOiCcbObjectModifyCallbackT_2
          saImmOiCcbObjectModifyCallback;
     SaImmOiRtAttrUpdateCallbackT
          saImmOiRtAttrUpdateCallback;
} SaImmOiCallbacksT_2;
```

5.3 Library Life Cycle

5.3.1 saImmOiInitialize_2()

Prototype

```
SaAisErrorT saImmOiInitialize_2(
      SaImmOiHandleT *immOiHandle,
      const SaImmOiCallbacksT_2 *immOiCallbacks,
      SaVersionT *version
);
```

Parameters

`immOiHandle` - [`out`] A pointer to the handle which identifies this particular initialization of the IMM Service, and which is to be returned by the IMM Service. This handle provides access to the Object Implementer APIs of the IMM Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`immOiCallbacks` - [`in`] If `immOiCallbacks` is set to NULL, no callback is registered; If `immOiCallbacks` is not set to NULL, it is pointer to an `SaImmOiCallbacksT_2` structure which contains the callback functions of the process that the IMM Service may invoke. Only non-NULL callback functions in this structure will be registered. The `SaImmOiCallbacksT_2` type is defined in Section 5.2.4 on page 102.

`version` - [`in/out`] As an input parameter, `version` is a pointer to a structure containing the required Information Model Management 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 Information Model Management Service. The `SaVersionT` type is defined in [1].

Description

This function initializes the Object Implementer functions of the Information Model Management Service for the invoking process and registers the various callback functions. This function must be invoked prior to the invocation of any other IMM Service Object Implementer functionality. The handle pointed to by `immOiHandle` is returned by the IMM Service as the reference to this association between the process and the IMM Service. The process uses this handle in subsequent communication with the IMM Service.

The returned handle `immOiHandle` is not associated with any implementer name. The association of the handle with an implementer name is performed by the invocation of the `saImmOiImplementerSet()` function.

If the invoking process exits after successfully returning from the `saImmOiInitialize_2()` function and before invoking `saImmOiFinalize()` to finalize the handle `immOiHandle` (see Section 5.3.4 on page 109), the IMM Service automatically finalizes this handle when the death of the process is detected.

If the implementation supports the required `releaseCode` and `majorVersion`, `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.

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 Information Model Management Service library or the provider of the service is out of memory and cannot provide the service.

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

SA_AIS_ERR_VERSION - The version provided in the structure to which the `version` parameter points is not compatible with the version of the Information Model Management Service implementation.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node because it is not a member node.

See Also

`saImmOiSelectionObjectGet()`, `saImmOiDispatch()`, `saImmOiFinalize()`, `saImmOiImplementerSet()`

5.3.2 saImmOiSelectionObjectGet()

Prototype

```
SaAisErrorT saImmOiSelectionObjectGet(
      SaImmOiHandleT immOiHandle,
      SaSelectionObjectT *selectionObject
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`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 [1].

Description

This function returns the operating system handle associated with the handle `immOiHandle`. The invoking process can use the operating system handle to detect pending callbacks, instead of repeatedly invoking `saImmOiDispatch()` 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 `saImmOiSelectionObjectGet()` is valid until `saImmOiFinalize()` is successfully invoked on the same handle `immOiHandle`.

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 immOiHandle 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 Information Model Management 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 operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immOiHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOiInitialize_2(), saImmOiDispatch(), saImmOiFinalize()

5.3.3 saImmOiDispatch()

Prototype

```
SaAisErrorT saImmOiDispatch(
     SaImmOiHandleT immOiHandle,
     SaDispatchFlagsT dispatchFlags
);
```

Parameters

immOiHandle - [in] The handle which was obtained by a previous invocation of the saImmOiInitialize_2() function and which identifies this particular initialization of the Information Model Management Service. The SaImmOiHandleT type is defined in Section 5.2.1 on page 101.

dispatchFlags - [in] Flags that specify the callback execution behavior of the saImmOiDispatch() 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 [1].

Description

In the context of the calling thread, this function invokes pending callbacks for the handle immOiHandle in a way that is 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 immOiHandle 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 immOiHandle 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_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immOiHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOiInitialize_2(), saImmOiSelectionObjectGet(), saImmOiFinalize()

5.3.4 saImmOiFinalize()

Prototype

```
SaAisErrorT saImmOiFinalize(
      SaImmOiHandleT immOiHandle
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

Description

The `saImmOiFinalize()` function closes the association represented by the `immOiHandle` parameter between the invoking process and the Information Model Management Service. The process must have invoked `saImmOiInitialize_2()` before it invokes this function. A process must invoke this function once for each handle it acquired by invoking `saImmOiInitialize_2()`.

This function does not release the associations established between object classes or objects and the implementer name that may still be associated with the handle `immOiHandle`.
The next time a process associates the same implementer name with an Object Implementer handle, that process automatically becomes the implementer of all objects having the same implementer name.

If the `saImmOiFinalize()` function completes successfully, it releases all resources acquired when `saImmOiInitialize_2()` was called. Furthermore, `saImmOiFinalize()` cancels all pending callbacks related to asynchronous operations performed with the handle `immOiHandle`. 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 Information Model Management Service implicitly finalizes all instances of the Information Model Management Service that are associated with the process, as described in the preceding paragraph.

After `saImmOiFinalize()` returns successfully, the handle `immOiHandle` 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 `immOiHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

See Also

`saImmOiInitialize_2()`

5.4 Object Implementer

As a runtime object is created by its Object Implementer**,** the IMM Service can automatically set the name of the implementer of a runtime object when the object is created.

On the other hand, configuration objects are typically created by management applications that are not the Object Implementers. Configuration Object Implementers must explicitly indicate to the IMM Service which configuration objects they implement. This can be done for all objects of a given class or by targeting a particular set of objects.

The implementer of an object is identified by an **implementer name**. Once the implementer name is set, it remains associated with the object until explicitly released. This association applies even if the process that was registered as the Object Implementer (called the **registered Object Implementer**) clears the implementer name associated with its Object Implementer handle. This feature enables faster recovery of Object Implementers failures, as the new Object Implementer does not have to explicitly re-register all objects it implements. Simply registering itself with the same implementer name allows the IMM Service to associate all objects with the same implementer name with that process.

5.4.1 saImmOiImplementerSet()

Prototype

```
SaAisErrorT saImmOiImplementerSet(
      SaImmOiHandleT immOiHandle,
      const SaImmOiImplementerNameT implementerName
);
```

Parameters

`immOiHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`implementerName` - [`in`] Name of the Object Implementer. The `SaImmOiImplementerNameT` type is defined in Section 5.2.2 on page 101.

Description

This function sets the implementer name specified in the `implementerName` parameter for the handle `immOiHandle`. In order to be a valid parameter to all Object Implementer APIs except for `saImmOiSelectionObjectGet()`, `saImmOiDispatch()`, `saImmOiImplementerSet()`, and `saImmOiFinalize()`, an Object Implementer handle must be successfully associated with an implementer name.

This function also registers the invoking process as an Object Implementer having the name which is specified in the `implementerName` parameter. At any given time, only a single process in the entire cluster can be registered under a particular Object Implementer name.

The invoking process becomes the implementer of all existing IMM Service objects that have an implementer name identical to `implementerName`.

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 `immOiHandle` is invalid, since it is corrupted, uninitialized, or has already been finalized.

`SA_AIS_ERR_EXIST` - An Object Implementer with the same name is already registered with the IMM Service.

`SA_AIS_ERR_UNAVAILABLE` - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immOiHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOiInitialize_2()`, `saImmOiImplementerClear()`

5.4.2 saImmOiImplementerClear()

Prototype

```
SaAisErrorT saImmOiImplementerClear(
      SaImmOiHandleT immOiHandle
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

Description

This function clears the implementer name associated with the `immOiHandle` handle and unregisters the invoking process as an Object Implementer for the name previously associated with `immOiHandle`.

With no associated implementer name, `immOiHandle` is only a valid parameter for the following APIs: `saImmOiSelectionObjectGet()`, `saImmOiDispatch()`, `saImmOiImplementerSet()`, and `saImmOiFinalize()`.

IMM object classes and objects that have an implementer name equal to the name previously associated with `immOiHandle` keep the same implementer name, but stay without any registered Object Implementer until a process invokes `saImmOiImplementerSet()` again with the same implementer name.

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 `immOiHandle` is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immOiHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOiInitialize_2(), saImmOiImplementerSet()

5.4.3 saImmOiClassImplementerSet()

Prototype

```
SaAisErrorT saImmOiClassImplementerSet(
      SaImmOiHandleT immOiHandle,
      const SaImmClassNameT className
);
```

Parameters

immOiHandle - [in] The handle which was obtained by a previous invocation of the saImmOiInitialize_2() function and which identifies this particular initialization of the Information Model Management Service. The SaImmOiHandleT type is defined in Section 5.2.1 on page 101.

className - [in] Object class name. The SaImmClassNameT type is defined in Section 4.2.2 on page 26.

Description

This function informs the IMM Service that all the objects that are instances of the object class whose name is specified by the className parameter are implemented by the Object Implementer whose name has been associated with the handle immOiHandle.

This operation fails if the object class whose name is specified by the className parameter has already an Object Implementer whose name is different from the implementer name associated with the handle immOiHandle.

If this operation succeeds, the current process becomes the current implementer of all objects of the object class whose name is specified by className (existing

objects as well as objects that will be created in the future), and the IMM Service adds to these objects an SA_IMM_ATTR_IMPLEMENTER_NAME attribute with a value equal to the implementer name associated with the handle immOiHandle.

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 immOiHandle is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

SA_AIS_ERR_BAD_OPERATION - The className parameter specifies the name of a runtime object class.

SA_AIS_ERR_NOT_EXIST - The className parameter does not specify the name of an existing class.

SA_AIS_ERR_EXIST - The object class whose name is specified by the className parameter has already an Object Implementer whose name is different from the implementer name associated with the handle immOiHandle.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immOiHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOiInitialize_2()

5.4.4 saImmOiClassImplementerRelease()

Prototype

```
SaAisErrorT saImmOiClassImplementerRelease(
      SaImmOiHandleT immOiHandle,
      const SaImmClassNameT className
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`className` - [in] Object class name. The `SaImmClassNameT` type is defined in Section 4.2.2 on page 26.

Description

This function informs the IMM Service that the implementer whose name is associated with the handle `immOiHandle` must not be considered anymore as the implementer of the objects that are instances of the object class whose name is specified by `className`.

If the operation succeeds, the IMM Service removes the `SA_IMM_ATTR_IMPLEMENTER_NAME` attribute as well as all non-persistent cached runtime attributes from all objects of that class.

This operation fails if the invoking process is not the current implementer of the class whose name is specified by `className`, or if one or more objects affected by the operation are currently taking part in an in-progress CCB and/or administrative operations.

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 `immOiHandle` is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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_BAD_OPERATION - The `className` parameter specifies the name of a runtime object class.

SA_AIS_ERR_NOT_EXIST - The name specified by the `className` parameter is not the name of an existing object class, or the implementer of object instances from the object class whose name is specified by `className` is different from the implementer name associated with the handle `immOiHandle`.

SA_AIS_ERR_BUSY - One or more objects affected by this operation are taking part in an in-progress CCB and/or an administrative operation.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immOiHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOiInitialize_2()`, `saImmOiClassImplementerSet()`

5.4.5 saImmOiObjectImplementerSet()

Prototype

```
SaAisErrorT saImmOiObjectImplementerSet(
      SaImmOiHandleT immOiHandle,
      const SaNameT *objectName,
      SaImmScopeT scope
);
```

Parameters

`immOiHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`objectName` - [`in`] Pointer to the object name. The `SaNameT` type is defined in [1].

`scope` - [`in`] Scope of the operation. The `SaImmScopeT` type is defined in Section 4.2.11 on page 30.

Description

This function informs the IMM Service that the objects identified by the `scope` and `objectName` parameters are implemented by the Object Implementer whose name has been associated with the handle `immOiHandle`.

The current process becomes the current implementer of all targeted objects.

The targeted set of objects is determined as follows:

- If `scope` is `SA_IMM_ONE`, the scope of the operation is the object designated by the name to which `objectName` points.
- If `scope` is `SA_IMM_SUBLEVEL`, the scope of the operation is the object designated by the name to which `objectName` points and its direct children.
- If `scope` is `SA_IMM_SUBTREE`, the scope of the operation is the object designated by the name to which `objectName` points and the entire subtree rooted at that object.

The operation fails if one of the targeted objects has already an implementer whose name is different from the name associated with the handle `immOiHandle`. If the operation fails, the implementer of the targeted objects is not changed.

If the operation succeeds, the SA_IMM_ATTR_IMPLEMENTER_NAME attribute of all targeted objects is set to the implementer name associated with the handle immOiHandle.

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 immOiHandle is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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_BAD_OPERATION - One or more targeted objects are runtime objects.

SA_AIS_ERR_NOT_EXIST - The name to which the objectName parameter points is not the name of an existing object.

SA_AIS_ERR_EXIST - At least one of the objects targeted by this operation already has an implementer having a name different from the name associated with the handle immOiHandle.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle immOiHandle was acquired before the cluster node left the cluster membership.

See Also

saImmOiInitialize_2(), saImmOiObjectImplementerRelease()

5.4.6 saImmOiObjectImplementerRelease()

Prototype

```
SaAisErrorT saImmOiObjectImplementerRelease(
      SaImmOiHandleT immOiHandle,
      const SaNameT *objectName,
      SaImmScopeT scope
);
```

Parameters

`immOiHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`objectName` - [`in`] Pointer to the object name. The `SaNameT` type is defined in [1].

`scope` - [`in`] Scope of the operation. The `SaImmScopeT` type is defined in Section 4.2.11 on page 30.

Description

This function informs the IMM Service that the implementer whose name is associated with the handle `immOiHandle` must no longer be considered as the implementer of the set of objects identified by `scope` and the name to which `objectName` points.

The targeted set of objects is determined as follows:

- If `scope` is `SA_IMM_ONE`, the scope of the operation is the object designated by the name to which `objectName` points.
- If `scope` is `SA_IMM_SUBLEVEL`, the scope of the operation is the object designated by the name to which `objectName` points and its direct children.
- If `scope` is `SA_IMM_SUBTREE`, the scope of the operation is the object designated by the name to which `objectName` points and the entire subtree rooted at that object.

The operation fails if one of the targeted objects is not implemented by the current process, or if one or more objects affected by the operation are taking part in an in-progress CCB and/or an administrative operation. If the operation fails, the implementer of the targeted objects is not changed.

If the operation succeeds, the SA_IMM_ATTR_IMPLEMENTER_NAME attribute and all non-persistent cached runtime attributes of all targeted objects are removed from the objects.

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 `immOiHandle` is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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_BAD_OPERATION - One or more targeted objects are runtime objects.

SA_AIS_ERR_NOT_EXIST - The name to which the `objectName` parameter points is not the name of an existing object, or at least one of the objects targeted by this operation does not have the same implementer name as the one associated with the handle `immOiHandle`.

SA_AIS_ERR_BUSY - One or more objects affected by this operation are taking part in an in-progress CCB and/or an administrative operation.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immOiHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOiInitialize_2()`, `saImmOiClassImplementerSet()`

5.5 Runtime Objects Management

The set of functions contained in this section are used by an Object Implementer to create or delete runtime objects and update the runtime attributes of either configuration or runtime objects. They are similar to the functions provided in the IMM Service Object Management interface, the difference being that they are not part of a configuration change bundle (CCB).

The values of non-cached runtime attributes are not accessible when an implementer is not registered for the objects to which these attributes belong.

Runtime attributes whose values are cached by the IMM Service must be updated by its Object Implementer whenever their value changes. The value of non-cached attributes must be updated by the Object Implementer only when the IMM Service requests such an update by invoking the `SaImmOiRtAttrUpdateCallbackT` callback function.

Updating cached runtime attribute values in the IMM Service generates some load on the system each time the values change. Attributes whose values change frequently, but are rarely read by using the Object Management API should typically not be cached.

5.5.1 saImmOiRtObjectCreate_2()

Prototype

```
SaAisErrorT saImmOiRtObjectCreate_2(
      SaImmOiHandleT immOiHandle,
      const SaImmClassNameT className,
      const SaNameT *parentName,
      const SaImmAttrValuesT_2 **attrValues
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`className` - [in] Object class name. The `SaImmClassNameT` type is defined in Section 4.2.2 on page 26.

`parentName` - [`in`] Pointer to the name of the parent of the new object. The `SaNameT` type is defined in [1].

`attrValues`- [`in`] Pointer to a NULL-terminated array of pointers to attribute descriptors. The `SaImmAttrValuesT_2` type is defined in Section 4.2.8 on page 29.

Description

This function creates a new IMM Service runtime object.

The new object is created as a child of the object designated by the name to which `parentName` points. If `parentName` is set to NULL, the new object is created as a top level object.

The attributes referred to by the pointers in the array of pointers to which the `attrValues` parameter points must match the object class definition. These attributes can only be cached runtime attributes. One and only one of these attributes must have the `SA_IMM_ATTR_RDN` flag set; this attribute is used as the Relative Distinguished Name of the new object.

Attributes named `SA_IMM_ATTR_CLASS_NAME`, `SA_IMM_ATTR_ADMIN_OWNER_NAME`, and `SA_IMM_ATTR_IMPLEMENTER_NAME` cannot be specified by the `attrValues` descriptors, as these attributes are automatically set by the IMM Service.

The IMM Service adds an `SA_IMM_ATTR_CLASS_NAME` attribute to the new object; the value of this attribute contains the name of the object class as specified by the `className` parameter.

The invoking process becomes the implementer of the new object, and the IMM Service adds an `SA_IMM_ATTR_IMPLEMENTER_NAME` attribute to the new object with a value equal to the implementer name associated with the handle `immOiHandle`.

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 `immOiHandle` is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

SA_AIS_ERR_INVALID_PARAM - A parameter is not set correctly. In particular:

- the `className` parameter specifies the name of a configuration object class,
- there is no valid RDN attribute specified for the new object,
- some cached attributes do not have values,
- the `attrValues` parameter includes:
  - attributes that are not defined for the specified class,
  - attributes with values that do not match the defined value type for the attribute,
  - multiple values for a single-valued attribute, and
  - non-cached runtime attributes.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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 - This value is returned due to one or more of the following reasons:

- The name to which the `parentName` parameter points is not the name of an existing object.
- The `className` parameter is not the name of an existing object class.
- One or more of the attributes specified by `attrValues` are not valid attribute names for the object class designated by the name `className`.

SA_AIS_ERR_EXIST - An object with the same name already exists.

SA_AIS_ERR_NAME_TOO_LONG - The size of the new object's DN is greater than SA_MAX_NAME_LENGTH.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immOiHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOiInitialize_2()`

5.5.2 saImmOiRtObjectDelete()

Prototype

```
SaAisErrorT saImmOiRtObjectDelete(
      SaImmOiHandleT immOiHandle,
      const SaNameT *objectName
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`objectName` - [in] Pointer to the object name. The `SaNameT` type is defined in [1].

Description

This function deletes the object designated by the name to which the `objectName` parameter points and the entire subtree of objects rooted at that object.

This operation fails if one of the targeted objects is not a runtime object implemented by the invoking process.

Return Values

`SA_AIS_OK` - The function completed successfully.

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

`SA_AIS_ERR_TIMEOUT` - An implementation-dependent timeout occurred 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 `immOiHandle` is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

`SA_AIS_ERR_INVALID_PARAM` - A parameter is not set correctly.

SA_AIS_ERR_NO_MEMORY - Either the Information Model Management 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_BAD_OPERATION - This value is returned due to one or more of the following reasons:

- at least one of the targeted objects is a configuration object;
- at least one of the targeted object is a runtime object not implemented by the invoking process.

SA_AIS_ERR_NOT_EXIST - The name to which the `objectName` parameter points is not the name of an existing object.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immOiHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOiInitialize_2()`

5.5.3 saImmOiRtObjectUpdate_2()

Prototype

```
SaAisErrorT saImmOiRtObjectUpdate_2(
      SaImmOiHandleT immOiHandle,
      const SaNameT *objectName,
      const SaImmAttrModificationT_2 **attrMods
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`objectName` - [`in`] Pointer to the name of the updated object. The `SaNameT` type is defined in [1].

`attrMods` - [`in`] Pointer to a NULL-terminated array of pointers to descriptors of the modifications to perform. The `SaImmAttrModificationT_2` type is defined in Section 4.2.10 on page 30.

Description

This function updates runtime attributes of a configuration or runtime object.

Attributes named `SA_IMM_ATTR_CLASS_NAME`, `SA_IMM_ATTR_ADMIN_OWNER_NAME`, and `SA_IMM_ATTR_IMPLEMENTER_NAME` cannot be modified.

This operation fails and returns the `SA_AIS_ERR_BAD_OPERATION` error code if the targeted object is not implemented by the invoking process.

Return Values

`SA_AIS_OK` - The function completed successfully.

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

`SA_AIS_ERR_TIMEOUT` - An implementation-dependent timeout occurred 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 `immOiHandle` is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

`SA_AIS_ERR_INVALID_PARAM` - A parameter is not set correctly. In particular, the `attrMods` parameter includes:

- configuration attributes,
- a new value for the RDN attribute,
- attributes that are not defined for the specified class,
- attributes with values that do not match the defined value type for the attribute,
- multiple values or additional values for a single-valued attribute.

`SA_AIS_ERR_NO_MEMORY` - Either the Information Model Management 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_BAD_OPERATION - The targeted object is not implemented by the invoking process.

SA_AIS_ERR_NOT_EXIST - The name to which the `objectName` parameter points is not the name of an existing object, or one or more attribute names specified by the `attrMods` parameter are not valid for the object class.

SA_AIS_ERR_FAILED_OPERATION - The targeted object is not implemented by the invoking process.

SA_AIS_ERR_UNAVAILABLE - The operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immOiHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOiInitialize_2()`

5.5.4 SaImmOiRtAttrUpdateCallbackT

Prototype

```
typedef SaAisErrorT (*SaImmOiRtAttrUpdateCallbackT)(
     SaImmOiHandleT immOiHandle,
     const SaNameT *objectName,
     const SaImmAttrNameT *attributeNames
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`objectName` - [in] Pointer to the name of the object for which the update is requested. The `SaNameT` type is defined in [1].

`attributeNames` - [`in`] Pointer to a NULL-terminated array of attribute names for which values must be updated. The `SaImmAttrNameT` type is defined in Section 4.2.2 on page 26.

Description

The IMM Service invokes this callback function to request an Object Implementer to update the values of some attributes of a runtime object. These attributes are attributes whose values are not cached by the IMM Service. The target object is identified by the name to which `objectName` points. The process must use the `saImmOiRtObjectUpdate_2()` function to update the values of the attributes whose names are specified by the `attributeNames` parameter.

If a requested attribute has no value, the `SA_IMM_ATTR_VALUES_REPLACE` flag of the `SaImmAttrModificationTypeT` structure can be used in the `saImmOiRtObjectUpdate_2()` call to set the attribute value to the empty set.

On successful return of this callback, all requested attributes have been updated.

Return Values

`SA_AIS_OK` - The function completed successfully.

`SA_AIS_ERR_NO_MEMORY` - The implementer process is out of memory and cannot provide the service.

`SA_AIS_ERR_NO_RESOURCES` - The implementer process is out of required resources (other than memory) to provide the service.

`SA_AIS_ERR_FAILED_OPERATION` - The implementer process failed to update the requested attributes due to an error occurring in the `saImmOiRtObjectUpdate_2()` invocation.

See Also

`saImmOiInitialize_2()`

5.6 Configuration Objects Implementer

The IMM Service invokes callbacks provided by an implementer of configuration objects (called a **configuration object implementer**) when requests to change the objects they implement are added to a configuration change bundle (CCB), and also when the CCB is being applied. Each CCB-related callback is invoked with a CCB identifier as a parameter. In situations in which an Object Implementer needs to handle several CCBs in parallel (on disjoint sets of objects), the CCB identifier enables the Object Implementer to collect the particular changes associated with each CCB. The scope of a CCB identifier is limited to the process that implements the callbacks (that is, different CCBs may have the same identifier in different object implementers).

When any of the callbacks `saImmOiCcbObjectCreateCallback()`, `saImmOiCcbObjectDeleteCallback()`, or `saImmOiCcbObjectModifyCallback()` is invoked to indicate the addition of a change request to a CCB, the Object Implementer must check the CCB identifier given as parameter to determine whether the change belongs to a CCB already known by the Object Implementer or if this is the first change of a new set of changes. The Object Implementer is responsible for validating the change and memorizing it, so it can react appropriately when all change requests contained in the CCB are applied by invoking the `saImmOmCcbApply()` function.

After a CCB has been either applied (`saImmOiCcbApplyCallback()`) or aborted (`saImmOiCcbAbortCallback()`), the Object Implementer shall dispose of the corresponding CCB identifier (as well as of the associated memorized changes), as the IMM Service may re-use the same identifier to designate another set of changes later.

The same CCB initialized with `saImmOmCcbInitialize()` may hold changes handled by different Object Implementers. However, the IMM Service does not require that the CCB identifiers passed to the different Object Implementers be identical. Additionally, after the changes associated to a given CCB have been applied (or aborted), the same CCB can be re-used to apply another set of changes; however, it is not required that the CCB identifiers passed to Object Implementers' callbacks for the second set of changes be identical to the identifiers used the first time.

If a change is added to a CCB for a particular object, but its Object Implementer did not provide the appropriate callback for the change or the callbacks used by the IMM Service to eventually apply or abort the CCB, the change is rejected with an `SA_AIS_ERR_FAILED_OPERATION` error. Note that the change is rejected regardless of whether the `SA_IMM_CCB_REGISTERED_OI` flag is set or not in the CCB.

Each change request added to a CCB must be validated by the Object Implementer with the understanding that the new request will be applied after all requests already

present in the CCB are applied. Thus, the validation should not consider the current state of the IMM Information Model but the state it would have with all prior requests being applied. Before invoking the Object Implementer callbacks, the IMM Service validates that the Information Model tree hierarchy is consistent:

- It checks that a newly created object has a parent in the hierarchy,
- and it checks that an object being deleted has no child.

If changes are made on configuration objects for which there is no registered Object Implementer, the IMM Service still applies the changes when the CCB is applied, without invoking any Object Implementer callbacks for these changes.

If an Object Implementer either registers or unregisters itself, while some registered CCB changes are still pending for objects it implements (that is, the IMM Service has not yet passed the step of successfully invoking all `SaImmOiCcbCompletedCallbackT` functions of registered Object Implementers for the CCB), the IMM Service aborts the CCBs that hold these changes.

When the user of the Object Management API requests the IMM Service to apply all change requests contained in a CCB, the IMM Service gives a last chance to the Object Implementers to check that all changes will bring the set of configuration objects they implement in a consistent state. As a CCB may contain change requests for objects having different implementers, the IMM Service applies a CCB in two steps:

- In the first step, the IMM Service indicates to each Object Implementer that has at least one object changed by the CCB requests that the CCB is now complete and that the Object Implementer must validate the entire set of CCB changes. This indication is done by invoking the `SaImmOiCcbCompletedCallbackT` callback function. If one of the Object Implementers returns an error, the attempt to apply the CCB fails, and the `saImmOmCcbApply()` function returns an error.
- If all implementers agreed with the proposed changes, the IMM Service applies the changes. In a second step, the IMM Service informs the implementers that the changes have been applied by invoking the `SaImmOiCcbApplyCallbackT` callback function. If one implementer rejected the proposed changes, the IMM Service informs implementers affected by the CCB that the CCB is aborted by invoking the `SaImmOiCcbAbortCallbackT` callback function.

5.6.1 SaImmOiCcbObjectCreateCallbackT_2

Prototype

```
typedef SaAisErrorT (*SaImmOiCcbObjectCreateCallbackT_2)(
      SaImmOiHandleT immOiHandle,
      SaImmOiCcbIdT ccbId,
      const SaImmClassNameT className,
      const SaNameT *parentName,
      const SaImmAttrValuesT_2 **attr
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`ccbId` - [in] CCB identifier. The `SaImmOiCcbIdT` type is defined in Section 5.2.3 on page 101.

`className` - [in] Object class name. The `SaImmClassNameT` type is defined in Section 4.2.2 on page 26.

`parentName` - [in] Pointer to the name of the parent of the new object. The `SaNameT` type is defined in [1].

`attr` - [in] Pointer to a NULL-terminated array of pointers to attribute descriptors. The `SaImmAttrValuesT_2` type is defined in Section 4.2.8 on page 29.

Description

The IMM Service invokes this callback function to enable an Object Implementer to validate and register a change request being added to a CCB identified by `ccbId`. The change request is a creation request for a configuration object of a class that is implemented by the process implementing the callback.

All parameters of the creation request are provided as parameters of the callback function to enable the implementer process to validate and memorize the creation request. For details on these parameters, refer to the description of the `saImmOmCcbObjectCreate_2()` function. All the parameters of the creation

request may be memorized by the implementer process and associated with the `ccbId` identifier, because these parameters will not be provided later on when the CCB is finally applied.

The changes will only be applied by the IMM Service after a successful invocation of the `SaImmOiCcbCompletedCallbackT` callback.

Return Values

`SA_AIS_OK` - The function completed successfully.

`SA_AIS_ERR_NO_MEMORY` - The implementer process is out of memory and cannot allocate the memory required to register the request.

`SA_AIS_ERR_NO_RESOURCES` - The implementer process is out of required resources (other than memory) to register the request.

`SA_AIS_ERR_BAD_OPERATION` - The implementer process rejects the creation request.

See Also

`saImmOmCcbObjectCreate_2()`, `SaImmOiCcbCompletedCallbackT`

5.6.2 SaImmOiCcbObjectDeleteCallbackT

Prototype

```
typedef SaAisErrorT (*SaImmOiCcbObjectDeleteCallbackT)(
     SaImmOiHandleT immOiHandle,
     SaImmOiCcbIdT ccbId,
     const SaNameT *objectName
);
```

Parameters

`immOiHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`ccbId` - [`in`] CCB identifier. The `SaImmOiCcbIdT` type is defined in Section 5.2.3 on page 101.

`objectName` - [`in`] Pointer to the object name. The `SaNameT` type is defined in [1].

Description

The IMM Service invokes this callback function to enable an Object Implementer to validate and memorize a deletion request being added to a CCB identified by `ccbId`. The deletion request is a request to delete object(s) that are implemented by the process that provided the callback function. These objects are the object designated by the name to which the `objectName` parameter points and the entire subtree of objects rooted at that object.

The name to which the `objectName` parameter points may be memorized by the implementer process and associated with the `ccbId` identifier, because these parameters will not be provided later on when the CCB is finally applied.

The changes will only be applied by the IMM Service after a successful invocation of the `SaImmOiCcbCompletedCallbackT` callback.

Return Values

`SA_AIS_OK` - The function completed successfully.

`SA_AIS_ERR_NO_MEMORY` - The implementer process is out of memory and cannot allocate the memory required to validate and memorize the request.

SA_AIS_ERR_NO_RESOURCES - The implementer process is out of required resources (other than memory) to validate and memorize the request.

SA_AIS_ERR_BAD_OPERATION - The implementer process rejects the deletion request.

See Also

saImmOmCcbObjectDelete(), SaImmOiCcbCompletedCallbackT

5.6.3 SaImmOiCcbObjectModifyCallbackT_2

Prototype

```
typedef SaAisErrorT (*SaImmOiCcbObjectModifyCallbackT_2)(
     SaImmOiHandleT immOiHandle,
     SaImmOiCcbIdT ccbId,
     const SaNameT *objectName,
     const SaImmAttrModificationT_2 **attrMods
);
```

Parameters

immOiHandle - [in] The handle which was obtained by a previous invocation of the saImmOiInitialize_2() function and which identifies this particular initialization of the Information Model Management Service. The SaImmOiHandleT type is defined in Section 5.2.1 on page 101.

ccbId - [in] CCB identifier. The SaImmOiCcbIdT type is defined in Section 5.2.3 on page 101.

objectName - [in] Pointer to the object name. The SaNameT type is defined in [1].

attrMods - [in] Pointer to a NULL-terminated array of pointers to descriptors of the modifications to perform. The SaImmAttrModificationT_2 type is defined in Section 4.2.10 on page 30.

Description

The IMM Service invokes this callback function to enable an Object Implementer to validate and memorize a change request being added to a CCB identified by ccbId. The change request is a request to modify configuration attributes of a configuration object implemented by the process implementing the callback.

All parameters of the modification request are provided as parameters of the callback function to enable the implementer process to validate and memorize the modification request. For details on these parameters, refer to the description of the `saImmOmCcbObjectModify_2()` function. All the parameters of the modification request may be memorized by the implementer process and associated with the `ccbId` identifier, because these parameters will not be provided later on when the CCB is finally applied.

The changes will only be applied by the IMM Service after a successful invocation of the `SaImmOiCcbCompletedCallbackT` callback.

Return Values

`SA_AIS_OK` - The function completed successfully.

`SA_AIS_ERR_NO_MEMORY` - The implementer process is out of memory and cannot allocate the memory required to validate and memorize the request.

`SA_AIS_ERR_NO_RESOURCES` - The implementer process is out of required resources (other than memory) to validate and memorize the request.

`SA_AIS_ERR_BAD_OPERATION` - The implementer process rejects the modification request.

See Also

`saImmOmCcbObjectModify_2()`, `SaImmOiCcbCompletedCallbackT`

5.6.4 SaImmOiCcbCompletedCallbackT

Prototype

```
typedef SaAisErrorT (*SaImmOiCcbCompletedCallbackT)(
     SaImmOiHandleT immOiHandle,
     SaImmOiCcbIdT ccbId
);
```

Parameters

`immOiHandle` - [`in`] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`ccbId` - [`in`] CCB identifier. The `SaImmOiCcbIdT` type is defined in Section 5.2.3 on page 101.

Description

The IMM Service invokes this callback function to inform an Object Implementer that the CCB identified by `ccbId` is now complete (no additional requests will be added). The implementer process must check that the sequence of change requests contained in the CCB is valid and that no errors will be generated when these changes are applied.

If all Object Implementers that implement objects changed by the CCB agree with the changes, the IMM Service will apply the changes and then invoke the `SaImmOiCcbApplyCallbackT` callback to notify all Object Implementers that the CCB has been applied.

Return Values

`SA_AIS_OK` - The function completed successfully.

`SA_AIS_ERR_NO_MEMORY` - The implementer process is out of memory and cannot allocate the memory required to later apply all requested changes.

`SA_AIS_ERR_NO_RESOURCES` - The implementer process is out of required resources (other than memory) to later apply all requested changes.

`SA_AIS_ERR_BAD_OPERATION` - The validation by the implementer process of all change requests contained in the CCB failed.

See Also

`saImmOmCcbApply()`, `SaImmOiCcbObjectCreateCallbackT_2`, `SaImmOiCcbObjectDeleteCallbackT`, `SaImmOiCcbObjectModifyCallbackT_2`

5.6.5 SaImmOiCcbApplyCallbackT

Prototype

```
typedef void (*SaImmOiCcbApplyCallbackT)(
      SaImmOiHandleT immOiHandle,
      SaImmOiCcbIdT ccbId
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`ccbId` - [in] CCB identifier. The `SaImmOiCcbIdT` type is defined in Section 5.2.3 on page 101.

Description

The IMM Service invokes this callback function to inform an Object Implementer that the CCB identified by `ccbId` has been applied by the IMM Service.

All configuration changes have already been validated by the Object Implementer in a previous call to `SaImmOiCcbCompletedCallbackT`.

Each Object Implementer is responsible for determining the effect of the configuration changes.

Return Values

None

See Also

`saImmOmCcbApply()`, `SaImmOiCcbCompletedCallbackT`

5.6.6 SaImmOiCcbAbortCallbackT

Prototype

```
typedef void (*SaImmOiCcbAbortCallbackT)(
     SaImmOiHandleT immOiHandle,
     SaImmOiCcbIdT ccbId
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`ccbId` - [in] CCB identifier. The `SaImmOiCcbIdT` type is defined in Section 5.2.3 on page 101.

Description

The IMM Service invokes this callback function to inform an Object Implementer that the CCB identified by `ccbId` is aborted, so that the Object Implementer can remove all change requests memorized for this CCB.

Return Values

None

See Also

`saImmOmCcbApply()`

5.7 Administrative Operations

5.7.1 SaImmOiAdminOperationCallbackT_2

Prototype

```
typedef void (*SaImmOiAdminOperationCallbackT_2) (
      SaImmOiHandleT immOiHandle,
      SaInvocationT invocation,
      const SaNameT *objectName,
      SaImmAdminOperationIdT operationId,
      const SaImmAdminOperationParamsT_2 **params
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`invocation` - [in] Used to match this invocation of `SaImmOiAdminOperationCallbackT_2` with the corresponding invocation of `saImmOiAdminOperationResult()`. The `SaInvocationT` type is defined in [1].

`objectName` - [in] Pointer to the object name. The `SaNameT` type is defined in [1].

`operationId` - [in] Identifier of the administrative operation. The `SaImmAdminOperationIdT` type is defined in Section 4.2.16 on page 33.

`params` - [in] Pointer to a NULL-terminated array of pointers to parameter descriptors. The `SaImmAdminOperationParamsT_2` type is defined in Section 4.2.17 on page 34.

Description

The IMM Service invokes this callback function to request an Object Implementer to execute an administrative operation on the object designated by the name to which `objectName` points. The administrative operation identified by the `operationId` parameter has been initiated by an invocation of the

`saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()` functions.

Each element referred to by a pointer of the array of pointers to which the `params` parameter points represents an input parameter of the administrative operation to execute.

The Object Implementer indicates the success or failure of the administrative operation by invoking the `saImmOiAdminOperationResult()` function. The `saImmOiAdminOperationResult()` function can be invoked from the callback itself or outside the callback by any thread of the process that initialized the `immOiHandle`.

Return Values

None

See Also

`saImmOiInitialize_2()`, `saImmOmAdminOperationInvoke_2()`, `saImmOmAdminOperationInvokeAsync_2()`, `saImmOiAdminOperationResult()`

5.7.2 saImmOiAdminOperationResult()

Prototype

```
SaAisErrorT saImmOiAdminOperationResult(
      SaImmOiHandleT immOiHandle,
      SaInvocationT invocation,
      SaAisErrorT result
);
```

Parameters

`immOiHandle` - [in] The handle which was obtained by a previous invocation of the `saImmOiInitialize_2()` function and which identifies this particular initialization of the Information Model Management Service. The `SaImmOiHandleT` type is defined in Section 5.2.1 on page 101.

`invocation` - [in] Used to match this invocation of `saImmOiAdminOperationResult()` with the previous corresponding invocation of the `SaImmOiAdminOperationCallbackT_2` callback. The `SaInvocationT` type is defined in [1].

`result` - [in] Result of the execution of the administrative operation. The `SaAisErrorT` type is defined in [1].

Description

An Object Implementer invokes this function to inform the IMM Service about the result of the execution of an administrative operation requested by the IMM Service by an invocation of the `SaImmOiAdminOperationCallbackT_2` callback.

This function can be called only by the process for which the `SaImmOiAdminOperationCallbackT_2` 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 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 `immOiHandle` is invalid, since it is corrupted, uninitialized, has already been finalized, or it is not associated with an implementer name.

`SA_AIS_ERR_INVALID_PARAM` - A parameter is not set correctly.

`SA_AIS_ERR_NO_MEMORY` - Either the Information Model Management 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 operation requested in this call is unavailable on this cluster node due to one of the two reasons:

- the cluster node has left the cluster membership;
- the cluster node has rejoined the cluster membership, but the handle `immOiHandle` was acquired before the cluster node left the cluster membership.

See Also

`saImmOiInitialize_2()`, `SaImmOiAdminOperationCallbackT_2`

6 IMM Service UML Information Model

The IMM Service Information Model is described in UML and has been organized in a UML class diagram.

The IMM Service UML model is implemented by the IMM Service. For further details on this implementation, refer to the SA Forum Overview document ([1]).

The IMM Service UML class diagram has one object class, which shows the contained attributes and the administrative operations applicable on this class.

6.1 DN Format for the IMM Service UML Class

Table 3 DN Formats for Objects of the IMM Service Class

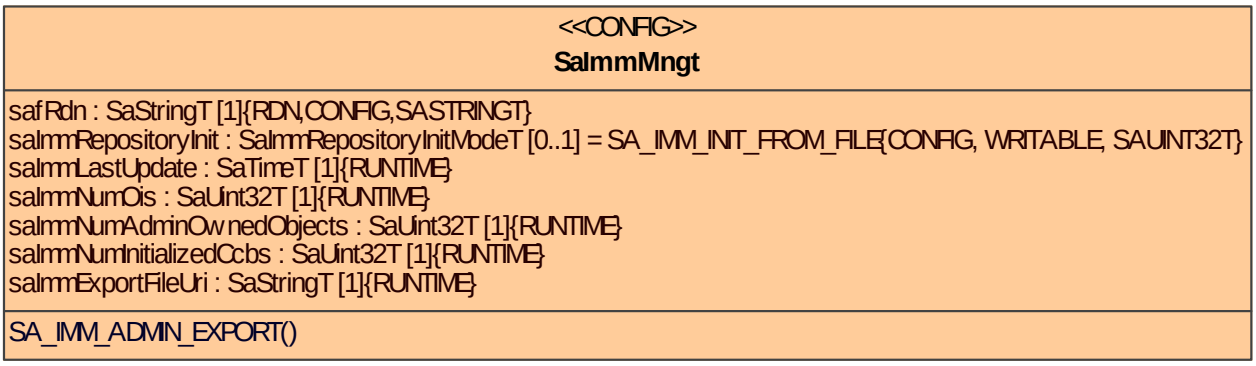
Object Class	DN Formats for Objects of the Class
`SaImmMngt`	"`safRdn=immManagement, safApp=safImmService`"

6.2 IMM Service UML Class

The `SaImmMngt` runtime object class exports all IMM global attributes and administrative operations.

FIGURE 2 shows the `SaImmMngt` class. A description of each attribute of these classes is found in the XMI file (see [2]). For additional details, refer to the SA Forum Overview document ([1]).

FIGURE 2 IMM Service UML Class



7 IMM Service Administration API

This section describes the administrative API functions that the IMM Service exposes on behalf of itself to a system administrator. These API functions are described using a ‘C’ API syntax. The main clients of this administrative API are system management applications.

7.1 Administrative Operations on the IMM Service

Administrative operations on the IMM Service can be carried out using the IMM Service API functions `saImmOmAdminOperationInvoke_2()` or `saImmOmAdminOperationInvokeAsync_2()` (refer to Section 4.9 on page 85) on an object that represents the IMM Service and for which the IMM Service is the Object Implementer.

Return values are passed in the `operationReturnValue` parameter (see Section 4.9.1 on page 86).

7.2 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 IMM Service Administration API:

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

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

```
libSaImm.so
```

7.3 IMM Service Administration API

7.3.1 SA_IMM_ADMIN_EXPORT

Parameters

`operationId` - [`in`] = `SA_IMM_ADMIN_EXPORT`

`objectName` - [`in`] = The LDAP name of the object of class `SaImmService` that represents the IMM Service. The DN of this object is "`safRdn=immManagement,safApp=safImmService`".
See [1] for SA Forum naming conventions and rules.

`filePathname` - [`in`] The standard relative POSIX pathname of the file to which the IMM contents must be exported. This pathname is relative to an implementation defined root directory. The type of this parameter is `SaStringT`, defined in [1].

Description

This administrative operation requests the IMM Service to export all its persistent contents (class definitions as well as persistent objects and attributes) into a file whose relative pathname is specified by the `filePathname` parameter.

The persistent contents will be stored into the file according to the IMM XML Schema Definition (see [3]).

The `saImmExportFileUri` attribute of the `SaImmMngt` IMM configuration class (see Section 6.2 on page 143) shall be used to retrieve the file after the export operation completed.

operationReturnValue

`SA_AIS_OK` - The operation completely successfully.

`SA_AIS_ERR_TRY_AGAIN` - The operation cannot be provided at this time. The caller may retry later. This error should be generally returned in cases where the requested administrative operation is valid but not currently possible.

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

See Also

-

8 IMM Service Alarms and Notifications

The IMM Service does not issue any alarms and notifications at the time of publication of this specification.

9 IMM Service Management Interface

The IMM Service has no management interface at the time of publication of this specification.

Index of Definitions