iSCSI Management API

Version 2.0 rev 15

"Publication of this Working Draft for review and comment has been approved by the IP Storage TWG. This draft represents a "best effort" attempt by the IP Storage TWG to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a "work in progress." Suggestion for revision should be directed to snia-ips@snia.org."

Working DRAFT

June 30, 2008

Table of Contents

Foreword

This specification documents an API that allows a management application to discover and manage the iSCSI resources on a system. The API uses an architecture that allows multiple iSCSI HBAs, sometimes referred to as hardware initiators, and/or multiple iSCSI software initiators installed on a system to provide a common interface to clients of the library. This API can be used by host-based management applications. A client of the API should be able to move between platforms by simply recompiling.

This specification includes two informative annexes.

Introduction

Clause 1 defines the scope of this document.

Clause 2 lists the documents referenced within this standard.

Clause 3 describes the conventions used in presenting the API interfaces.

Clause 4 provides background technical information on terms used within this standard and concepts describing the relationship between the library and an iSCSI implementation.

Clause 5 defines the Constants and Structures of the standard.

Clause 6 defines the programmatic functions (APIs) of the standard.

Clause 7 provides information on implementation compliance.

Clause 8 provides guidance to implementers of the API to help achieve interoperability between implementation releases, and between implementations from different vendors.

Annex A is an informative annex with guidelines on device names.

Annex B is an informative annex with coding examples.

1 Scope

This API provides interfaces to discover and manage iSCSI resources on a system. The intended audience is vendors that deliver drivers that provide these resources to a system.

2 References

The following standards contain provisions which, through reference in this text, constitute provisions of this specification. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

IETF RFC 3720, Internet Small Computer Systems Interface (iSCSI)

IEEE 802.3-2000, IEEE Standard for Information technology

ANSI INCITS 408-2005, SCSI Primary Commands 3 (SPC-3)

IETF RFC 1994, PPP Challenge Handshake Authentication Protocol (CHAP)

IETF RFC 2945, The SRP Authentication and Key Exchange System

ANSI INCITS 386-2004, Fibre Channel HBA API (FC-HBA)

ISO/IEC 19501 Unified Modeling Language (UML)

ISO/IEC 9899:1999, Programming Languages -- C

3 Document Conventions

3.1 API Description Format

Each API's description is divided into seven sections. These sections are described below.

1. **Synopsis**
   This section gives a brief description of what action the API performs.

2. **Prototype**
   This section gives a C prototype of the function. The prototypes show the following:

   - The name of the API
   - The return type of the API

3. **Parameters**
   This section lists each parameter along with an explanation of what the parameter represents.

4. **Typical Return Values**
   This section lists the Typical Return Values of the API with an explanation of why a particular return value would be returned. It is important to note that this list is **not** a comprehensive list of all of the possible return values. There are certain errors, e.g. IMA_ERROR_INSUFFICIENT_MEMORY, that might be returned by any API. Return values such as these are not listed.

5. **Remarks**
   This section contains comments about the API that may be useful to the reader. In particular, this section will contain extra information about the information returned by the API.

6. **Support**
   This section says if an API is mandatory to be supported, optional to be supported, or mandatory to be supported under certain conditions.

   - If an API is mandatory to be supported a client can rely on the API functioning under all circumstances.
   - If the API is optional to be supported then a client cannot rely on the API functioning.
   - If the API is mandatory to be supported under certain conditions then a client can rely on the API functioning if the specified conditions are met. Otherwise a client should assume that the API is not supported.

7. **See Also**
   This section lists other related APIs or related code examples that the reader might find useful.

4 Background Technical Information

4.1 Terms

The terms that are used in this specification are defined in this section.

Alias[1]	An alias string can be associated with an iSCSI Node. The alias allows an organization to associate a user-friendly string with the iSCSI Name. However, the alias string is not a substitute for the iSCSI Name.
Discovery Address	An address used in a SendTargets discovery session. The discovery address is used to represent one or more targets to be discovered. One example of a discovery address is a gateway that exposes one or more targets to one or more iSCSI initiators.
Host	A compute node connected to the SAN.
iSCSI Node[1]	The iSCSI Node represents a single iSCSI initiator or iSCSI target. There are one or more iSCSI Nodes within a Network Entity. The iSCSI Node is accessible via one or more Network Portals. An iSCSI Node is identified by its iSCSI Name. The separation of the iSCSI Name from the addresses used by and for the iSCSI node allows multiple iSCSI nodes to use the same address, and the same iSCSI node to use multiple addresses.
LHBA	See **L**ogical **HBA**.
LNP	**L**ogical **N**etwork **P**ort
Logical HBA	A representation of a parallel SCSI HBA to the operating system.
Logical Network Port	A logical network port is a collection of one or more physical network ports that have been aggregated together. This can be done using IEEE 802.3ad, but may be done in other ways as well. Note: If more than one physical network port is used to create a logical network port those physical network ports do not have to be on the same PHBA.
LUN	***L***ogical ***U***nit ***N***umber
Mutual Authentication	Authentication of the target by the initiator in addition to authentication of the initiator by the target.
Network Entity[1]	The Network Entity represents a device or gateway that is accessible from the IP network. A Network Entity must have one or more Network Portals, each of which can be used to gain access to the IP network by some iSCSI Nodes contained in that Network Entity.

Network Portal[1]	The Network Portal is a component of a Network Entity that has a TCP/IP network address and that may be used by an iSCSI Node within that Network Entity for the connection(s) within one of its iSCSI sessions. A Network Portal in an initiator is identified by its IP address. A Network Portal in a target is identified by its IP address and its listening TCP port.
Object ID	A unique identifier assigned to any object within the IMA. Objects sometimes represent physical entities, e.g. physical HBAs. At other times, objects represent logical entities, e.g. network portals.
OID	**O**bject **ID**
One-way Authentication	Authentication of the initiator by the target.
Persistent	The quality of something being non-volatile. This usually means that it is recorded on some non-volatile medium such as flash RAM or magnetic disk. Implicitly, this shall also be readable from the non-volatile medium. Examples of persistent storage: ▪ Under Windows, the Registry would be a common place to find persistently stored values (assuming that the values are not stored as volatile). ▪ Under any OS a file on magnetic hard disk would be persistent.
PHBA	**P**hysical **HBA**
Physical HBA	A physical HBA (PHBA) is a controller card that has one or more physical network ports mounted on it and that plugs into a slot in a motherboard. If a motherboard has physical network ports mounted on it directly, in can be considered a PHBA *in regards to the requirements specified in this document*. Normally, a motherboard would not be considered a PHBA.
Physical Network Port	A physical connection on a physical HBA that connects the PHBA to the network, e.g. an RJ-45. A physical HBA has one or more physical network ports.
Plugin	A plugin is software, typically written by an HBA vendor, that provides support for one or more models of iSCSI HBAs. The plugin's job is to provide a bridge between the library's interface and the vendor's HBA device driver. A plugin is implemented as a loadable module: a DLL in Windows and a shared object in UNIX. A plugin is accessed by an application through the iSCSI Management API library. The SNIA FC HBA API's concept of a vendor library is the

	equivalent to a plugin.
PNP	**P**hysical **N**etwork **P**ort
Portal Groups[1]	iSCSI supports multiple connections within the same session; some implementations will have the ability to combine connections in a session across multiple Network Portals. A Portal Group defines a set of Network Portals within an iSCSI Network Entity that collectively supports the capability of coordinating a session with connections spanning these portals. Not all Network Portals within a Portal Group need participate in every session connected through that Portal Group. One or more Portal Groups may provide access to an iSCSI Node. Each Network Portal, as utilized by a given iSCSI Node, belongs to exactly one portal group within that node.
Portal Group Tag[1]	This 16-bit quantity identifies the Portal Group within an iSCSI Node. All Network Portals with the same portal group tag in the context of a given iSCSI Node are in the same Portal Group.
Primary Discovery Method	A discovery method that does not depend upon any other discovery method to discover targets. iSNS target discovery, SLP target discovery, and static target discovery are all primary discovery methods.
Secondary Discovery Method	A discovery method that depends upon other discovery methods to discover targets. `SendTargets` target discovery is a secondary discovery method because it cannot discover any targets without using some other discovery method.
Stale Data	Stale data is configuration data that refers to targets or logical units that are no longer present or that are no longer visible to the system. For example, setting iSCSI login parameters associated with a target using the IMA_SetFirstBurstLength API and then removing the target from the network will create stale data because the configuration data that's been set for the target is retained after the target is removed from the network. Similarly, calling the IMA_ExposeLu API for a logical unit and then removing the logical unit's target from the network will create stale data because configuration data indicating that the logical unit is exposed is retained after the target is removed from the network.
Unicode	Unicode is a system of uniquely identifying (numbering) characters such that nearly any character in any language is identified.
UTF-8[2]	Unicode Transformation Format, 8-bit encoding form. UTF-8 is the Unicode Transformation Format that serializes a Unicode scalar value as a sequence of one to four bytes.

[1] Definition taken from IETF RFC 3720.

[2] Definition taken from the glossary of the Unicode Consortium web site. See http://www.unicode.org/glossary/index.html.

4.2 Concepts

Library and Plugins

The iSCSI Management API shall be implemented in one of two ways:

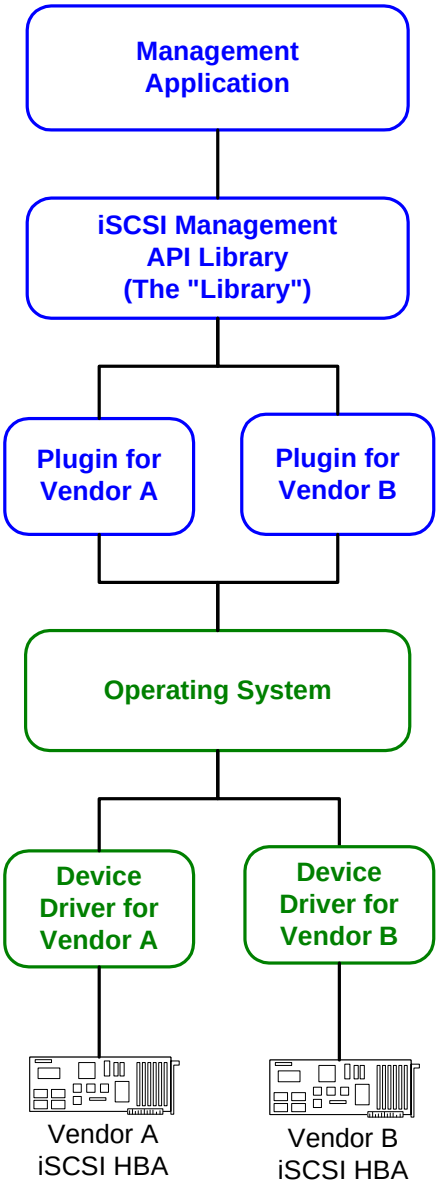
- A library.
- A library in combination with plugins.

If an implementation uses a library without plugins then either the plugin functionality is built into the library itself or the library is able to interface with vendor specific modules using a pre-existing interface. These vendor specific modules would provide functionality equivalent to plugins.

The library provides an interface that applications use to perform iSCSI management. Among other things, the library is responsible for loading plugins and dispatching requests from a management application to the appropriate plugin(s).

Plugins are provided by iSCSI HBA vendors to manage their hardware. Typically, a plugin will take a request in the generic format provided by the library and then translate that request into a vendor specific format and forward the request onto the vendor's device driver. In practice, a plugin may use a DLL or shared object library to communicate with the device driver. Also, it may communicate with multiple device drivers. Ultimately, the method a plugin uses to accomplish its work is entirely vendor specific. It is anticipated that most implementations will use plugins and such an implementation is generally assumed throughout this document. With the exception of two APIs (IMA_GetAssociatedPluginOid and IMA_GetPluginOidList), the method of implementation does not matter to the client.

The figure below shows a simple block diagram of how the iSCSI Management API library and plugins fit into a total iSCSI management application architecture.



Object ID

The core element of the iSCSI Management API (IMA) is the object ID (OID). An object ID is a structure that “uniquely” identifies an object. The reason uniquely is in quotes in the previous sentence is that it is possible, though *very* unlikely, that an object ID would be reused and refer to a different object.

An object ID consists of three fields:

1. An object type. This identifies the type of object, e.g. iSCSI node, PHBA, LHBA, etc., that the object ID refers to.

2. An object owner identifier. This is a number that is used to uniquely identify the owner of the object. Objects are owned by either the library or a plugin.

3. An object sequence number. This is a number used by the owner of an object, possibly in combination with the object type, to identify an object.

To a client that uses the library object IDs shall be considered opaque. A client shall use only documented APIs to access information found in the object ID.

There are several rules for object IDs that the library, plugins, and clients shall follow. They are:

An object ID can only refer to one object at a time.

An object can only have one object ID that refers to it at any one time. It is not permissible to have two or more object IDs that refer to the same object at the same time. In some cases this may be difficult, but the rule still shall be followed.

For example, suppose a PHBA is in a system. That PHBA will have an object ID. If the PHBA is removed and then reinserted (while the associated plugin is running) then one of two things can happen:

- The PHBA can retain the same object ID as it had before it was removed

  OR

- The PHBA can get a new object ID and the old object ID will no longer be usable.

This can only happen if the *same* PHBA is reinserted. If a PHBA is removed and another PHBA is inserted that has not been in the system while a particular instance of the library and plugins are running then that PHBA *shall* be given a new object ID.

The library and plugins can uniquely identify an object within their own object space by using either the object sequence number or by using the object sequence number in combination with the object type. Which method is used is up to the implementer of the library or plugin.

Object sequence numbers shall be reused in a conservative fashion to minimize the possibility that an object ID will ever refer to two (or more) different objects in any once instance of the library or plugin. This rule for reuse only applies to a particular instance of the library or plugin. Neither the library nor plugins are required or expected to persist object sequence numbers across instances.

Because neither the library nor plugins are required to persist object sequence numbers a client using the library shall not use persisted object IDs across instances of itself.

Similarly, different instances of the library and plugins *may* use different object IDs to represent the same physical entity.

Object ID List

An object ID list is a list of zero or more object IDs. There are several APIs, e.g.IMA_GetNonSharedNodeOidList, that return object ID lists. Once a client is finished using an object ID list the client shall free the memory used by the list by calling the IMA_FreeMemory API.

The Shared Node vs. Non-shared Nodes

The following is the definition of an iSCSI node found in IETF RFC 3720:

The iSCSI Node represents a single iSCSI initiator or iSCSI target. There are one or more iSCSI Nodes within a Network Entity. The iSCSI Node is accessible via one or more Network Portals. An iSCSI Node is identified by

its iSCSI Name. The separation of the iSCSI Name from the addresses used by and for the iSCSI node allows multiple iSCSI nodes to use the same addresses, and the same iSCSI node to use multiple addresses. iSCSI nodes also have addresses.

The following text from IETF RFC 3720 gives a somewhat clearer idea of what an iSCSI node is:

An iSCSI name really names a logical software entity, and is not tied to a port or other hardware that can be changed. For instance, an initiator name should name the iSCSI initiator node, not a particular NIC or HBA. When multiple NICs are used, they should generally all present the same iSCSI initiator name to the targets, because they are just paths to the same SCSI layer. In most operating systems, the named entity is the operating system image.

So, in simple terms, an iSCSI node is a uniquely named entity that runs on a particular operating system image. IETF RFC 3720 allows for more than one node to run on a particular image, but it strongly encourages that only one node be used. (For more information on why this is the case please refer to the various iSCSI related specifications.)

To this end the iSCSI Management API has the concept of a shared node. The shared node is intended to be the single node of an operating system image that is encouraged in the iSCSI specification. The reason that it's called the shared node is that the node is shared by all the vendors whose iSCSI HBAs are in a system.

All other nodes in an operating system image are considered to be non-shared nodes. That is, they are nodes that are created by an HBA vendor to be used exclusively by that vendor's iSCSI HBAs in combination with the operating system image. Non-shared nodes run counter to the spirit of what IETF RFC 3720 intended a node to be. However, the need for them in this specification reflect some limitations of existing HBA implementations.

Logical HBA

A logical HBA (LHBA) is a representation of a parallel SCSI HBA to the operating system. Typically, today's operating systems only have an understanding of the specifics of parallel SCSI. They don't understand other SCSI transports such as FCP or iSCSI. Therefore, device drivers for these other transports are required to map transport specific concepts to parallel SCSI concepts. For example, an iSCSI HBA device driver cannot identify a device to the OS using the iSCSI node name. Instead, it must conjure up a parallel SCSI ID (0-15) for the device. In addition, it may have to map eight byte SCSI LUNs to three *bit* SCSI LUNs.

In addition, some HBA vendors provide dynamic multi-pathing in their device drivers. This allows the host to communicate with targets using different initiator ports in the host, thus allowing the device driver to provide both load balancing and fail over capabilities transparently to the operating system. An LHBA allows a device driver to easily implement these features transparently to the operating system.

Target OIDs and Logical Unit OIDs

A target OID should not be confused with a parallel SCSI target ID. A target OID is an IMA_OID structure in which the object type field indicates that the OID structure specifies an iSCSI target accessible via a LHBA. An iSCSI target will have a unique OID on each LHBA that can access a LU of the target.

Similarly, a logical unit OID should not be confused with a SCSI LUN. A logical unit OID is an IMA_OID structure in which the object type field indicates that the OID structure specifies an iSCSI logical unit accessible via an iSCSI target. A logical unit will have a unique OID on each LHBA that can access the LU.

Software Initiators Versus Hardware Initiators

There are two basic types of implementations of iSCSI initiators, they are typically referred to as software initiators and hardware initiators.

- A software initiator usually is a device driver that runs on top of the TCP/IP stack that comes with the operating system. It connects to iSCSI targets using the NICs that are being used for traditional networking tasks.

- A hardware initiator usually includes a specialized adapter, usually referred to as an HBA, that includes special hardware and/or firmware for accelerating TCP/IP and sometimes iSCSI as well. This solution also includes a device driver to allow the operating system to use the HBA.

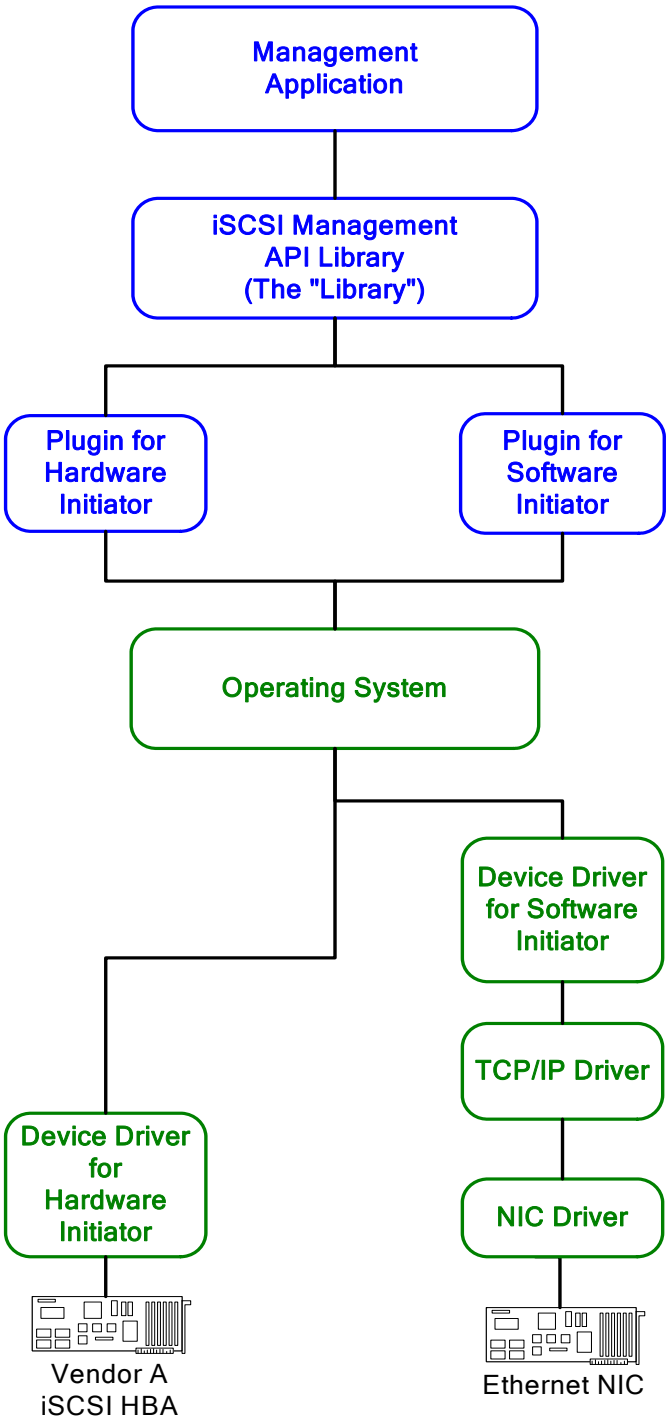
These terms are imprecise as both types of initiators use software and hardware. However, these terms have come into common use and so this specification uses them as well.

The reason to introduce these two types of initiators is that they provide different levels of discovery and management capabilities, which requires clients to manage them somewhat differently.

For example, a software initiator will typically know little, if anything, about the underlying hardware that is being used. Therefore, an IMA client cannot determine (using IMA) the topology of the storage network, the redundancy of connections between the initiator and a target, etc. Meanwhile, a hardware initiator will know the hardware that it is using and be able to provide an IMA client with this kind of information.

Another example, is that when querying and setting various discovery methods for iSCSI targets a software initiator will require this be done using the logical HBA, while a hardware initiator may allow either the logical or the physical HBA to be used. The software initiator requires that the logical HBA be used because it doesn't know anything about the underlying hardware and it cannot therefore expose any physical HBA objects.

The diagram below shows the software and hardware stack needed for both a hardware initiator and a software initiator.



iSCSI Session and Connection Parameters

iSCSI supports the negotiation of both session specific and connection specific parameters, e.g., max burst length. This API recognizes two levels at which a client can query and set these parameters: the target level, the LHBA level. There is a third level which is used, but is not queryable or settable: the driver level.

If a client sets one or more of these parameters at the target level then those values will be used as the proposed initial value when negotiating the actual value to use at runtime with a target. A client can only set a proposed initial value for a parameter, it cannot specify an actual value that is guaranteed to be used.

If a client sets a value for a parameter at the LHBA level than that value will be used as the proposed initial value when negotiating the actual value to use at runtime *unless* a value for that parameter has been set for that parameter on that specific target. Thus a target specific value overrides the setting of an LHBA specific value.

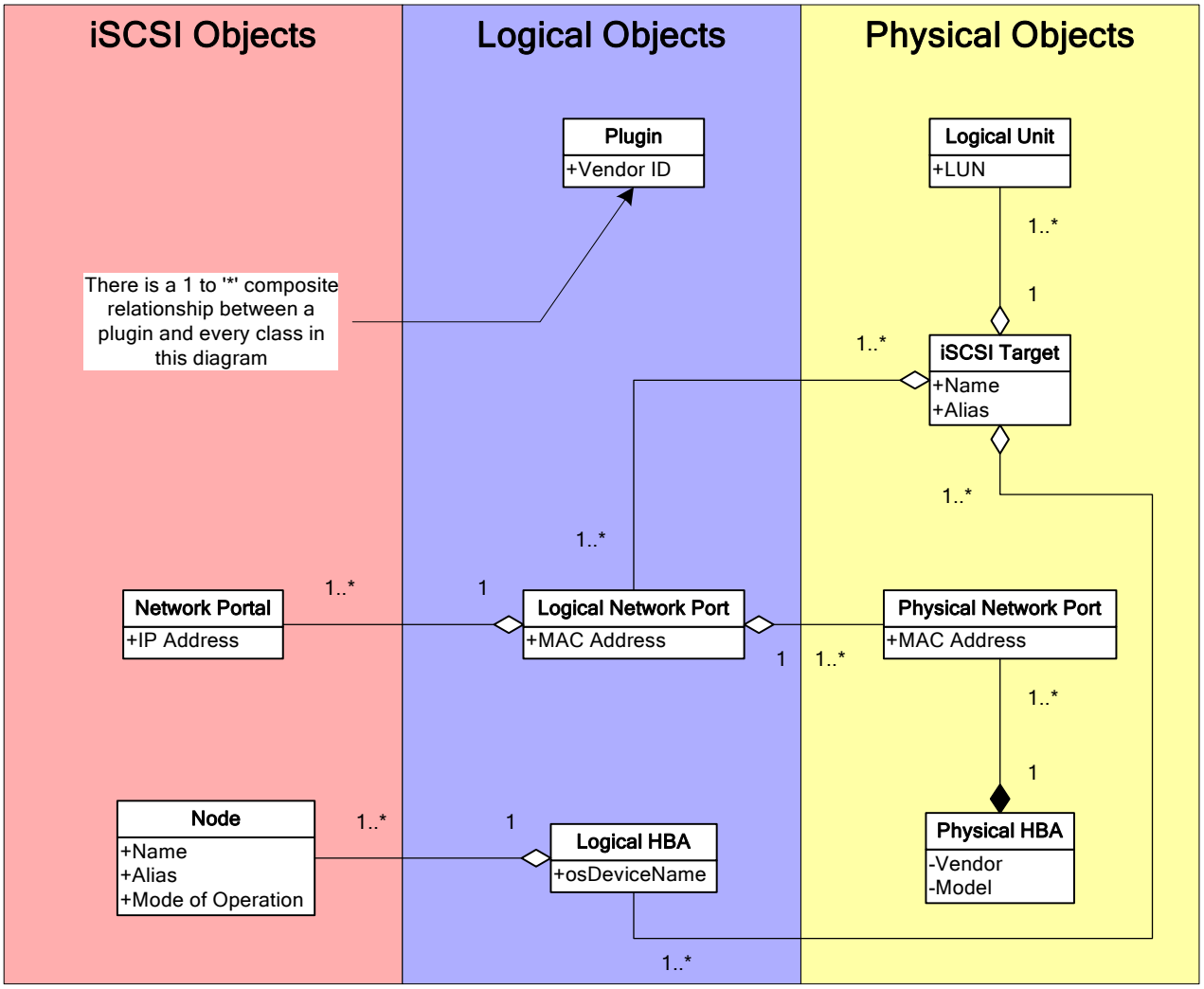
If a client does not set value for a parameter at either the target or the LHBA level then the driver level value, i.e., a default value specified by the driver implementation is used.

In summary:

- If a target level value for a parameter has been specified it is used.
- If no target level value has been specified for a parameter, but an LHBA level value has been specified then it is used.
- Finally, if neither a target level nor an LHBA level value for a parameter has been specified then the driver level (implementation default) value is used.

Class Relationship Diagram

Below is a Universal Modeling Language (ISO/IEC 19501 UML) diagram that shows the relationship between the various classes of objects in the IMA. Each class may contain a few example properties.



5 Constants and Types

5.1 IMA_WCHAR

Typedef'd as a wchar_t.

5.2 IMA_BYTE

The smallest unsigned integer that is at least 8 bits in length.

5.3 IMA_BOOL

Typedef'd to an IMA_UINT32. A variable of this type can have either of the following values:

- IMA_TRUE

  This symbol has the value 1.

- IMA_FALSE

  This symbol has the value 0.

5.4 IMA_XBOOL

Typedef'd to an IMA_UINT32. This is an extended boolean. A variable of this type can have any of the following values:

- IMA_TRUE

  This symbol has the value 1.

- IMA_FALSE

  This symbol has the value 0.

- IMA_UNKNOWN

  This symbol has the value 0xFFFFFFFF.

5.5 IMA_UINT

The smallest unsigned integer that is at least 32 bits in length.

5.6 IMA_UINT16

The smallest unsigned integer that is at least 16 bits in length.

5.7 IMA_UINT32

The smallest unsigned integer that is at least 32 bits in length.

5.8 IMA_UINT64

The smallest unsigned integer that is at least 64 bits in length.

5.9 IMA_DATETIME

Typedef'd to a `struct tm`. This is a structure declared in `time.h` that comes with the standard C runtime library.

5.10 IMA_OBJECT_VISIBILITY_FN

Format

```
typedef void (* IMA_OBJECT_VISIBILITY_FN)(
    /* in */    IMA_BOOL    becomingVisible,
    /* in */    IMA_OID     oid
);
```

Parameters

becomingVisible

A boolean indicating if the object specified by *oid* is becoming visible or is disappearing. If this parameter has the value IMA_TRUE then the object is becoming visible. If this parameter has the value IMA_FALSE then the object is disappearing.

oid

The object ID of the object whose visibility is changing.

Remarks

This type is used to declare client functions that can be used with the IMA_RegisterForObjectVisibilityChanges and IMA_DeregisterForObjectVisibilityChanges APIs.

5.11 IMA_OBJECT_PROPERTY_FN

Format

```
typedef void (* IMA_OBJECT_PROPERTY_FN)(
    /* in */        IMA_OID         oid
);
```

Parameters

oid

The object ID of the object whose property(ies) changed.

Remarks

This type is used to declare client functions that can be used with the IMA_RegisterForObjectPropertyChanges and IMA_DeregisterForObjectPropertyChanges APIs.

5.12 IMA_OBJECT_TYPE

Format

```
typedef enum IMA_object_type
{
    IMA_OBJECT_TYPE_UNKNOWN                 =  0,
    IMA_OBJECT_TYPE_PLUGIN                  =  1,
    IMA_OBJECT_TYPE_NODE                    =  2,
    IMA_OBJECT_TYPE_LHBA                    =  3,
    IMA_OBJECT_TYPE_PHBA                    =  4,
    IMA_OBJECT_TYPE_NETWORK_PORTAL          =  5,
    IMA_OBJECT_TYPE_PORTAL_GROUP            =  6,
    IMA_OBJECT_TYPE_LNP                     =  7,
    IMA_OBJECT_TYPE_PNP                     =  8,
    IMA_OBJECT_TYPE_TARGET                  =  9,
    IMA_OBJECT_TYPE_LU                      =  10,
    IMA_OBJECT_TYPE_DISCOVERY_ADDRESS       =  11,
    IMA_OBJECT_TYPE_STATIC_DISCOVERY_TARGET =  12,
    IMA_OBJECT_TYPE_CONNECTION              =  13,
    IMA_OBJECT_TYPE_SESSION                 =  14
} IMA_OBJECT_TYPE;
```

Fields

IMA_OBJECT_TYPE_UNKNOWN

The object has an unknown type. If an object has this type it's most likely an uninitialized object.

This symbol has the value 0.

IMA_OBJECT_TYPE_PLUGIN

The object represents a plugin to the IMA library.

This symbol has the value 1.

IMA_OBJECT_TYPE_NODE

The object represents an iSCSI node.

This symbol has the value 2.

IMA_OBJECT_TYPE_LHBA

The object represents a logical HBA.

This symbol has the value 3.

IMA_OBJECT_TYPE_PHBA

The object represents a physical HBA.

This symbol has the value 4.

IMA_OBJECT_TYPE_NETWORK_PORTAL

The object represents an iSCSI network portal.

This symbol has the value 5.

IMA_OBJECT_TYPE_PORTAL_GROUP

The object represetns an iSCSI portal group.

This symbol has the value 6.

IMA_OBJECT_TYPE_LNP

The object represents a logical network port.

This symbol has the value 7.

IMA_OBJECT_TYPE_PNP

The object represents a physical network port.

This symbol has the value 8.

IMA_OBJECT_TYPE_TARGET

The object represents an iSCSI target relative to an LHBA.

This symbol has the value 9.

IMA_OBJECT_TYPE_LU

The object represents a logical unit relative to a target.

This symbol has the value 10.

IMA_OBJECT_TYPE_DISCOVERY_ADDRESS

The object represents a discovery address relative to a PNP or an LHBA.

This symbol has the value 11.

IMA_OBJECT_TYPE_STATIC_DISCOVERY_TARGET

The object represents a static discovery target relative to a LNP or an LHBA.

This symbol has the value 12.

IMA_OBJECT_TYPE_CONNECTION

The object represents a connection within an iSCSI session

This symbol has the value 13.

IMA_OBJECT_TYPE_SESSION

The object represents an iSCSI session

This symbol has the value 14.

5.13 IMA_STATUS

IMA_STATUS is an enumerated type used to indicate the status of an API call. Most statuses are errors, however some are not. The non-error statuses indicate that an operation successfully completed. However, there is additional information that the client needs to be aware of and the status indicates what this information is.

Currently there are a limited number of status values that indicate that an invalid parameter was specified to an API call. Additional statuses which indicate more precisely why a parameter was invalid may be added to future versions of the iSCSI Management API specification. Statuses in the range of 0xC0000000 to 0xCFFFFFFF are reserved for indicating conditions that mean an invalid parameter was specified to an API. Clients shall be written to handle invalid parameter statuses that may be added in later versions of this specification.

Macros

IMA_SUCCESS

This macro returns IMA_TRUE if the specified status code indicates that a call succeeded. It returns IMA_FALSE if the specified status code indicates that the call failed.

IMA_ERROR

This macro returns IMA_TRUE if the specified status code indicates that a call failed. It reutrns IMA_FALSE if the specified status code indicates that the call succeeded.

Non-error Statuses

IMA_STATUS_SUCCESS

This status indicates that the API call succeeded.

This symbol has the value 0x00000000.

IMA_STATUS_REBOOT_NECESSARY

This status indicates that the operation succeeded, but a reboot is necessary to have the change take affect.

This symbol has the value 0x00000001.

IMA_STATUS_INCONSISTENT_NODE_PROPERTIES

This status indicates there is an inconsistency between the node properties, specifically either the node name or node alias, kept by the library and one or more plugins. The client should set both the node name (using IMA_SetNodeName) and node alias (using IMA_SetNodeAlias) to fix this problem.

This symbol has the value 0x00000002.

IMA_STATUS_SCSI_STATUS_CONDITION_MET

This status indicates that a SCSI command succeeded with a CONDITION MET status.

This symbol has the value 0x00000100.

Error Statuses

IMA_ERROR_NOT_SUPPORTED

This error indicates that the specified API is not supported by the owner of the object.

This symbol has the value 0x80000001.

IMA_ERROR_INSUFFICIENT_MEMORY

This error indicates that there was insufficient memory to complete the request. It is possible that any API can return this error.

This symbol has the value 0x80000002.

IMA_ERROR_LAST_PRIMARY_DISCOVERY_METHOD

This error indicates that the call would disable the last primary discovery method for the PHBA specified in the call. A client is not allowed to disable all primary discovery methods for a PHBA.

This symbol has the value 0x80000003.

IMA_ERROR_UNEXPECTED_OS_ERROR

This error indicates that either the library or plugin encountered an unexpected error from an OS API while attempting to perform the requested operation.

This symbol has the value 0x80000004.

IMA_ERROR_SYNC_TIMEOUT

This error indicates that an attempt to acquire ownership of some synchronization mechanism, e.g. a mutex or semaphore, has timed out.

This symbol has the value 0x80000005.

IMA_ERROR_LU_EXPOSED

This error indicates the requested operation cannot be completed because an LU is currently exposed to the operating system. For the called API to succeed the logical unit shall not be exposed to the operating system. This error is returned by the IMA_ExposeLu and IMA_RemoveStaticDiscoveryTarget APIs.

This symbol has the value 0x80000006.

IMA_ERROR_LU_NOT_EXPOSED

This error indicates an attempt to use a logical unit that is not currently exposed to the operating system. For the called API to succeed the logical unit shall be exposed to the operating system. This error is returned by the IMA_UnexposeLu and the IMA_GetDeviceStatistics APIs.

This symbol has the value 0x80000007.

IMA_ERROR_LU_IN_USE

This error indicates an attempt to unexpose a logical unit that is in use by the operating system. This error is returned by the IMA_UnexposeLu API.

This symbol has the value 0x80000008.

IMA_ERROR_TARGET_TIMEOUT

This error indicates that communication with a target was necessary to perform the requested API and that the target didn't respond to a command that was sent to it.

This symbol has the value 0x80000009.

IMA_ERROR_LOGIN_REJECTED

This error indicates that a login to a target was needed to perform the requested API and the target rejected the attempt.

This symbol has the value 0x8000000A.

IMA_ERROR_STATS_COLLECTION_NOT_ENABLED

This error indicates that an attempt was made to retrieve statistics from an object that did not have statistics collection enabled.

This symbol has the value 0x8000000B.

IMA_ERROR_SCSI_STATUS_CHECK_CONDITION

This error indicates that a SCSI command failed with a CHECK CONDITION status.

This symbol has the value 0x80000100.

IMA_ERROR_SCSI_STATUS_BUSY

This error indicates that a SCSI command failed with a BUSY status.

This symbol has the value 0x800000101.

IMA_ERROR_SCSI_STATUS_RESERVATION_CONFLICT

This error indicates that a SCSI command failed with a RESERVATION CONFLICT status.

This symbol has the value 0x800000102.

IMA_ERROR_SCSI_STATUS_TASK_SET_FULL

This error indicates that a SCSI command failed with a TASK SET FULL status.

This symbol has the value 0x80000103.

IMA_ERROR_SCSI_STATUS_ACA_ACTIVE

This error indicates that a SCSI command failed with a ACA ACTIVE status.

This symbol has the value 0x80000104.

IMA_ERROR_SCSI_STATUS_TASK_ABORTED

This error indicates that a SCSI command failed with a TASK ABORTED status.

This symbol has the value 0x80000105.

IMA_ERROR_PLUGINS_NOT_SUPPORTED

This error indicates that the library implementation does not support plugins.

This symbol has the value 0x80000106.

IMA_ERROR_INVALID_PARAMETER

This error indicates that a specified parameter was invalid.

This error can be returned in a number of situations, such as

- When a client calls an API and specifies a NULL pointer as a parameter to an API that does not accept NULL pointers.
- When a client calls an API and specifies an integer parameter that is out range of the acceptable values for the parameter.

- When a client specifies a pointer to a structure as a parameter to an API and the contents of the structure contain invalid pointers or out of range integer parameters as stated above.

This symbol has the value 0xC0000000.

IMA_ERROR_INVALID_OBJECT_TYPE

This error indicates that the object type of the specified IMA_OID structure is invalid. Most likely an uninitialized variable or a corrupted variable was used in an API call.

This symbol has the value 0xC0000001.

IMA_ERROR_INCORRECT_OBJECT_TYPE

This error indicates that an object with an incorrect type was specified in an API call. This can be caused by passing an object ID of the wrong type to an API call. It can also be caused by using an uninitialized variable or a corrupted variable.

This symbol has the value 0xC0000002.

IMA_ERROR_OBJECT_NOT_FOUND

This error indicates an object specified in the API call was not found. This can be caused by using an uninitialized variable or a corrupted variable. It can also be caused by using an object ID that referred to an object or plugin that is no longer known to the system.

This symbol has the value 0xC0000003.

IMA_ERROR_NAME_TOO_LONG

This error indicates that a name specified in an API call is too long.

This symbol has the value 0xC0000004.

IMA_ERROR_UNKNOWN_ERROR

This error indicates that some sort of unknown error has occurred.

This symbol has the value 0x8FFFFFFF.

5.14 IMA_OID

Format

```
typedef struct IMA_oid
{
    IMA_OBJECT_TYPE    objectType;
    IMA_UINT32         ownerId;
    IMA_UINT64         objectSequenceNumber;
} IMA_OID;
```

Fields

objectType

The type of the object. When an object ID is supplied as a parameter to an API the library uses this value to ensure that the supplied object's type is appropriate for the API that was called.

ownerId

A number determined by the library that it uses to uniquely identify the owner of an object. The owner of an object is either the library itself or a plugin. When an object ID is supplied as a parameter to an API the library uses this value to determine if it should handle the call itself or direct the call to one or more plugins.

objectSequenceNumber

A number determined by the owner of an object, that is used by the owner possibly in combination with the object type, to uniquely identify an object.

Remarks

This structure shall be treated as opaque by clients of the API. Appropriate APIs, e.g. IMA_GetObjectType and IMA_GetAssociatedPluginOid, shall be used to extract information from the structure.

5.15 IMA_OID_LIST

Format

```
typedef struct IMA_oid_list
{
    IMA_UINT            oidCount;
    IMA_OID             oids[1];
} IMA_OID_LIST;
```

Fields

oidCount

The number of object IDs in the *oids* array.

oids

A variable length array of zero or more object IDs. There are *oidCount* object IDs in this array.

Remarks

This structure is used by a number of APIs to return lists of objects. Any instance of this structure returned by an API shall be freed by a client using the IMA_FreeMemory API.

Although *oids* is declared to be an array of one IMA_OID structure it can in fact contain any number of IMA_OID structures.

5.16 IMA_NODE_NAME

Format

typedef IMA_WCHAR IMA_NODE_NAME[224];

Remarks

This type is used to represent an iSCSI node name. An iSCSI node name is a unique identifier for an iSCSI node.

The name shall be terminated with a Unicode nul character.

Both initiators and targets have node names.

5.17 IMA_NODE_ALIAS

Format

typedef IMA_WCHAR IMA_NODE_ALIAS[256];

Remarks

This type is used to represent an iSCSI node alias. An iSCSI node alias is intended to be used as a human useful description of an iSCSI node.

The alias shall be terminated with a Unicode nul character.

Both initiators and targets may have node aliases.

5.18 IMA_IP_ADDRESS

Format

```
typedef struct IMA_ip_address
{
    IMA_BOOL        ipv4Address;
    IMA_BYTE        ipAddress[16];

} IMA_IP_ADDRESS;
```

Fields

ipv4Address

A boolean indicating the type of IP address this structure represents. If this field has value IMA_TRUE then this is an IPv4 address; if this field has the value IMA_FALSE then this is an IPv6 address.

ipAddress

An array of bytes containing the IP address.

- If *ipv4Address* has the value IMA_TRUE then bytes 0 through 3 of this array contain the IP address. Byte 0 of the array contains the most significant byte of the IP address and byte 3 contains the least significant byte of the address.
- If *ipv4Address* has the value IMA_FALSE then bytes 0 through 15 of this array contain the IP address. Byte 0 of the array contains the most significant byte of the IP address and byte 15 contains the least significant byte of the address.

5.19 IMA_HOST_NAME

Format

typedef IMA_WCHAR IMA_HOST_NAME[256];

Remarks

This type is used to represent a DNS hostname.

This type is used as part of the IMA_HOST_ID union.

The hostname shall be terminated by a Unicode nul character.

5.20 IMA_HOST_ID

Format

```
typedef struct IMA_host_id
{
    IMA_BOOL            hostnameInUse;

    union
    {
            IMA_HOST_NAME                   hostname;
            IMA_IP_ADDRESS                  ipAddress;
    } id;

} IMA_HOST_ID;
```

Fields

hostnameInUse

A boolean indicating whether a DNS hostname or IP address is represented by the *IpAddress* field. If this field has the value IMA_TRUE then the *hostname* field contains the value. If this field has the value IMA_FALSE then the *ipAddress* field contains the value.

hostname

If *hostnameInUse* has the value IMA_TRUE then this field contains a DNS hostname, otherwise the value of this field is undefined.

ipAddress

If *hostnameInUse* has the value IMA_FALSE then this field contains an IP address, otherwise the value of this field is undefined.

Remarks

This type is used to represent a hostname or IP address used in a iSCSI target address.

5.21 IMA_TARGET_ADDRESS

Format

```
typedef struct IMA_target_address
{
    IMA_HOST_ID         hostnameIpAddress;
    IMA_UINT16          portNumber;
} IMA_TARGET_ADDRESS;
```

Fields

hostnameIpAddress

A DNS hostname or IP address at which a target can be contacted.

portNumber

The IP port number which can used in conjunction with the DNS hostname or IP address to contact an iSCSI target.

Remarks

This structure is used to represent a target address that can be used to contact an iSCSI target.

5.22 IMA_ADDRESS_KEY

Format

```
typedef struct IMA_address_key
{
    IMA_IP_ADDRESS      ipAddress;

    IMA_BOOL            portNumberValid;
    IMA_UINT16          portNumber;

    IMA_BOOL            portalGroupTagValid;
    IMA_UINT16          portalGroupTag;

} IMA_ADDRESS_KEY;
```

Fields

ipAddress

An IP address at which a target can be contacted.

portNumberValid

A boolean indicating if the field *portNumber* has a valid value..

portNumber

The IP port number that is used in conjunction with the IP address to contact an iSCSI target. If *portNumberValid* has the value IMA_TRUE then this value is valid. If *portNumberValid* has the value IMA_FALSE then this value is undef+ined.

If *portNumberValid* has the value IMA_TRUE then this field shall not be the value zero.

portalGroupTagValid

A boolean indicating if the field *portalGroupTag* has a valid value.

portalGroupTag

A 16-bit unsigned integer indicating the portal group tag to be used when connecting to the target. If *portalGroupTagValid* has the value IMA_TRUE then this value is valid. If *portalGroupTagValid* has the value IMA_FALSE then this value is undefined.

Remarks

This structure is used as a part of the IMA_ADDRESS_KEYS structure.

5.23 IMA_ADDRESS_KEYS

Format

```
typedef struct IMA_address_keys
{
    IMA_UINT          addressKeyCount;
    IMA_ADDRESS_KEY   addressKeys[1];

} IMA_ADDRESS_KEYS;
```

Fields

addressKeyCount

The count of address keys in the *addressKeys* array.

addressKeys

This is an array of one or more target address keys that can be used to contact a target.

Remarks

This structure is returned by the IMA_GetAddressKeys API.

5.24 IMA_STATIC_DISCOVERY_TARGET

Format

```
typedef struct IMA_static_discovery_target
{
    IMA_NODE_NAME       targetName;
    IMA_TARGET_ADDRESS targetAddress;

    IMA_BOOL            portalGroupTagValid;
    IMA_UINT16          portalGroupTag;

} IMA_STATIC_DISCOVERY_TARGET;
```

Fields

targetName

The iSCSI node name of the statically discovered target.

targetAddress

A target address at which the statically discovered target resides.

portalGroupTagValid

A boolean indicating if the field *portalGroupTag* has a valid value.

portalGroupTag

An 16-bit unsigned integer indicating the portal group tag to be used when connecting to the target. If *portalGroupTagValid* has the value IMA_TRUE then this value is valid. If *portalGroupTagValid* has the value IMA_FALSE then this value is undefined.

Remarks

This structure is used to represent a statically discovered target. This structure is used in the IMA_AddStaticDiscoveryTarget API.

If a valid *portalGroupTag* value is not provided then the initiator should use a SendTargets discovery session to determine a value to use.

5.25 IMA_DISCOVERY_ADDRESS_PROPERTIES

Format

```
typedef struct IMA_discovery_address_properties
{
    IMA_OID                 associatedNodeOid;
    IMA_OID                 associatedLhbaOid;
    IMA_TARGET_ADDRESS discoveryAddress;
} IMA_DISCOVERY_ADDRESS_PROPERTIES;
```

Fields

associatedNodeOid

The object ID of the node through which the target is being accessed.

associatedLhbaOid

The object ID of the LHBA that is used to communicate with the discovery address.

discoveryAddress

A target address representing the discovery address.

Remarks

This structure is returned by the IMA_GetDiscoveryAddressProperties API.

5.26 IMA_STATIC_DISCOVERY_TARGET_PROPERTIES

Format

```
typedef struct IMA_static_discovery_target_properties
{
    IMA_OID                     associatedNodeOid;
    IMA_OID                     associatedLhbaOid;
    IMA_STATIC_DISCOVERY_TARGET staticConfigTarget;

} IMA_STATIC_DISCOVERY_TARGET_PROPERTIES;
```

Fields

associatedNodeOid

The object ID of the node through which the target is being accessed.

associatedLhbaOid

The object ID of the LHBA that is used to communicate with the target.

staticConfigTarget

The statically configured target.

Remarks

This structure is returned by the IMA_GetStaticDiscoveryTargetProperties API.

5.27 IMA_IP_PROPERTIES

Format

```
typedef struct IMA_ip_properties
{
    IMA_BOOL            ipConfigurationMethodSettable;
    IMA_BOOL            dhcpConfigurationEnabled;

    IMA_BOOL            subnetMaskSettable;
    IMA_BOOL            subnetMaskValid;
    IMA_IP_ADDRESS      subnetMask;

    IMA_BOOL            defaultGatewaySettable;
    IMA_BOOL            defaultGatewayValid;
    IMA_IP_ADDRESS      defaultGateway;

    IMA_BOOL            primaryDnsServerAddressSettable;
    IMA_BOOL            primaryDnsServerAddressValid;
    IMA_IP_ADDRESS      primaryDnsServerAddress;

    IMA_BOOL            alternateDnsServerAddressSettable;
    IMA_BOOL            alternateDnsServerAddressValid;
    IMA_IP_ADDRESS      alternateDnsServerAddress;

    IMA_BYTE            reserved[64];

} IMA_IP_PROPERTIES;
```

Fields

ipConfigurationMethodSettable

A boolean indicating if the IP configuration method is settable using the IMA_SetIpConfigMethod API. If the value of this field is IMA_TRUE then the IMA_SetIpConfigMethod API is supported for the associated object. If the value of this field is IMA_FALSE then the API is not supported for that object.

dhcpConfigurationEnabled

A boolean indicating if the *subnetMask*, *defaultGateway*, *primaryDnsServerAddress*, and *alternateDnsServerAddress* fields have been set via DHCP or using static configuration. If the value of this field is IMA_TRUE then these fields have been set using DHCP. If the value of this field is IMA_FALSE then these fields have been set via static configuration or they are not otherwise settable.

subnetMaskSettable

A boolean indicating if the subnet mask can be set using the IMA_SetSubnetMask API. If the *dhcpConfigurationEnabled* field has the value IMA_TRUE then this field shall have the value IMA_FALSE. The subnet mask can only be set when static configuration is enabled.

subnetMaskValid

A boolean indicating if the field *subnetMask* has a valid value.

subnetMask

A structure containing the subnet mask. If *subnetMaskValid* has the value IMA_TRUE then this value is valid. If *subnetMaskValid* has the value IMA_FALSE then this value is undefined.

defaultGatewaySettable

A boolean indicating if the default gateway can be set using the IMA_SetDefaultGateway API. If the *dhcpConfigurationEnabled* field has the value IMA_TRUE then this field shall have the value IMA_FALSE. The default gateway can only be set when static configuration is enabled.

defaultGatewayValid

A boolean indicating if the field *defaultGateway* has a valid value.

defaultGateway

A structure containing the default gateway. If *defaultGatewayValid* has the value IMA_TRUE then this value is valid. If *defaultGatewayValid* has the value IMA_FALSE then this value is undefined.

primaryDnsServerAddressSettable

A boolean indicating if the primary DNS server address can be set using the IMA_SetDnsServerAddress API. If the *dhcpConfigurationEnabled* field has the value IMA_TRUE then this field shall have the value IMA_FALSE. The primary DNS server can only be set when static configuration is enabled.

primaryDnsServerAddressValid

A boolean indicating if the field *primaryDnsServerAddress* has a valid value.

primaryDnsServerAddress

An array containing the name or address of the primary DNS server. If *primaryDnsServerAddressValid* has the value IMA_TRUE then this value is valid. If *primaryDnsServerAddressValid* has the value IMA_FALSE then this value is undefined.

alternateDnsServerAddressSettable

A boolean indicating if the alternate DNS server can be set using the IMA_SetDnsServerAddress API. If the *dhcpConfigurationEnabled* field has the value IMA_TRUE then this field shall have the value IMA_FALSE. The alternate DNS server can only be set when static configuration is enabled. If *primaryDnsServerAddressSettable* has the value IMA_FALSE then this field shall have the value IMA_FALSE. The alternate DNS server address can only be set if the primary DNS server can be set.

alternateDnsServerAddressValid

A boolean indicating if the field *alternateDnsServerAddress* has a valid value.

alternateDnsServerAddress

An array containing the name or address of the alternate DNS server. If *alternateDnsServerAddressValid* has the value IMA_TRUE then this value is valid. If *alternateDnsServerAddressValid* has the value IMA_FALSE then this value is undefined.

reserved

This field is reserved.

Remarks

If both the primary and alternate DNS servers are valid then the implementation tries to use the primary DNS server for domain name resolution. If there are errors using the primary DNS server the implementation will use the alternate DNS server. The method that an implementation uses to determine which DNS server to use is implementation specific.

5.28 IMA_LIBRARY_PROPERTIES

Format

```
typedef struct IMA_library_properties
{
    IMA_UINT        supportedImaVersion;
    IMA_WCHAR       vendor[256];
    IMA_WCHAR       implementationVersion[256];
    IMA_WCHAR       fileName[256];
    IMA_DATETIME    buildTime;

    IMA_BYTE        reserved[64];

} IMA_LIBRARY_PROPERTIES;
```

Fields

supportedImaVersion

The version of the iSCSI Management API implemented by the library. The value returned by a library for the API as described in this document is two.

vendor

A nul terminated Unicode string containing the name of the vendor that created the binary version of the library.

implementationVersion

A nul terminated Unicode string containing the implementation version of the library from the vendor specified in *vendor*.

fileName

A nul terminated Unicode string ideally containing the path and file name of the library that is being used by the currently executing process can be found. If the path cannot be determined then it is acceptable to fill this field with only the name (and extension if applicable) of the file of the library. If this cannot be determined then this field shall be an empty string.

buildTime

The time and date that the library that is executing was built.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetLibraryProperties API.

5.29 IMA_PLUGIN_PROPERTIES

Format

```
typedef struct IMA_plugin_properties
{
    IMA_UINT        supportedImaVersion;
    IMA_WCHAR       vendor[256];
    IMA_WCHAR       implementationVersion[256];
    IMA_WCHAR       fileName[256];
    IMA_DATETIME    buildTime;

    IMA_BOOL        lhbasCanBeCreatedAndDestroyed;

    IMA_BOOL        autoPersistenceSupported;

    IMA_BYTE        reserved[60];

} IMA_PLUGIN_PROPERTIES;
```

Fields

supportedImaVersion

The version of the iSCSI Management API implemented by a plugin. The value returned by a library for the API as described in this document is two.

vendor

A nul terminated Unicode string containing the name of the vendor that created the binary version of the plugin.

implementationVersion

A nul terminated Unicode string containing the implementation version of the plugin from the vendor specified in *vendor*.

fileName

A nul terminated Unicode string ideally containing the path and file name of the plugin that is filing in this structure.

If the path cannot be determined then this field will contain only the name (and extension if applicable) of the file of the plugin. If this cannot be determined then this field will be be an empty string.

buildTime

The time and date that the plugin that is specified by this structure was built.

lhbasCanBeCreatedAndDestroyed

A boolean indicating if LHBAs can be created or destroyed. For this version of the IMA this value is always IMA_FALSE.

autoPersistenceSupported

A boolean indicating if the plugin supports automatic persistence of settings. If this setting is true, all set operations will be persistent upon successful completion of the API call. If this setting is false, the function IMA_PersistHbaParameters must be called to make the setting persistent.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetPluginProperties API.

5.30 IMA_NODE_PROPERTIES

Format

```
typedef struct IMA_node_properties
{
    IMA_BOOL            runningInInitiatorMode;
    IMA_BOOL            runningInTargetMode;

    IMA_BOOL            nameValid;
    IMA_NODE_NAME       name;

    IMA_BOOL            aliasValid;
    IMA_NODE_ALIAS      alias;

    IMA_BOOL            nameAndAliasSettable;

    IMA_BYTE            reserved[64];

} IMA_NODE_PROPERTIES;
```

Fields

runningInInitiatorMode

A boolean indicating if the node is running as initiator or not.

runningInTargetMode

A boolean indicating if the node is running as a target or not.

nameValid

A boolean indicating if the node's name is valid or not.

name

A nul terminated Unicode string that contains the name of the node.

The value in this field is only valid if *nameValid* is set to IMA_TRUE. If *nameValid* is set to IMA_FALSE then this field will contain an empty string.

aliasValid

A boolean indicating if the node's alias is valid or not.

alias

A nul terminated Unicode string that contains the alias of the node.

This field is only valid if *aliasValid* is set to IMA_TRUE. If *aliasValid* is set to IMA_FALSE then this field will contain an empty string.

nameAndAliasSettable

A boolean indicating if both the name and alias are settable using IMA_SetNodeName and IMA_SetNodeAlias.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetNodeProperties API.

It is possible for both *runningInInitiatorMode* and *runningInTargetMode* to be set to IMA_TRUE. This means that the node is operating both as an initiator and as a target.

5.31 IMA_LHBA_PROPERTIES

Format

```
typedef struct IMA_lhba_properties
{
    IMA_WCHAR       osDeviceName[256];
    IMA_BOOL        luExposingSupported;
    IMA_BOOL        isDestroyable;

    IMA_BOOL        staleDataRemovable;
    IMA_UINT        staleDataSize;

    IMA_BOOL        initiatorAuthMethodsSettable;
    IMA_BOOL        targetAuthMethodsSettable;
    IMA_BOOL        initiatorLocalAuthParmsSupported;  IMA_BOOL
    mutualLocalAuthParmsSupported;                     IMA_BOOL
    mutualLhbaAuthParmsListSupported;
    IMA_BOOL        mutualAuthSupported;
    IMA_BYTE        reserved[116];

} IMA_LHBA_PROPERTIES;
```

Fields

osDeviceName

A nul terminated Unicode string that contains the operating system's name for a logical HBA. If this is an empty string then the device name for the LHBA is unknown.

See Annex A (informative) Device Names for details on the value(s) of this field for each operating system.

luExposingSupported

A boolean indicating if the LHBA supports exposing and unexposing of individual LUs using the IMA_ExposeLu and IMA_UnexposeLu APIs. If the value of this field is IMA_TRUE then a client can control the exposing and unexposing of all LUs associated with the LHBA using the IMA_ExposeLu and IMA_UnexposeLu APIs. If th value of this field is IMA_FALSE then a client cannot control the exposing and unexposing of LUs using the IMA.

isDestroyable

A boolean indicating if the LHBA can be destroyed.

For this version of the IMA this value shall always be IMA_FALSE.

staleDataRemovable

A boolean indicating if stale data associated with the LHBA is removable. If this value is IMA_TRUE then the data can be removed using IMA_RemoveStaleData.

staleDataSize

The **approximate** size, in bytes, of the amount of stale data associated with the LHBA. If this value is zero then there is no stale data associated with the LHBA.

initiatorAuthMethodsSettable

A boolean indicating if the initiator authentications methods used by the LHBA can be set by a client using the IMA_SetInitiatorAuthMethods API.

targetAuthMethodsSettable

A boolean that, for this version of the IMA specification, shall be set to IMA_FALSE.

initiatorLocalAuthParmsSupported

A boolean indicating if the LHBA supports the setting and enabling of local initiator authentication parameters. When enabled, these local initiator authentication parameters are used in place of any authentication parameters that have been applied to the LHBA.

mutualLocalAuthParmsSupported

A boolean indicating if the LHBA supports the setting and enabling of target parameters. When enabled using IMA_SetMutualLocalAuthParms, these target authentication parameters are used when mutual authentication is performed. When this is enabled, mutualLhbaAuthParmsListSupported must be disabled.mutualLhbaAuthParmsListSupported

A boolean indicating if the LHBA supports the setting and enabling of target parameters as a list. When set using IMA_AddLhbaMutualAuthParms, these target authentication parameters are used when mutual authentication is performed. When this is enabled, mutualLocalAuthParmsSupported must be disabled.

mutualAuthSupported

A boolean indicating if the LHBA supports the enabling of mutual authentication. If supported, mutual authentication can be set using IMA_SetMutualLocalAuth.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetLhbaProperties API.

5.32 Upper Level Protocol (ULP) Flags

These are flag values used in the *supportedUlps* field of the IMA_PHBA_PROPERTIES structure.

IMA_ULP_TCP

This flag indicates that TCP is supported.

This symbol has the value 0x01.

IMA_ULP_SCTP

This flag indicates that SCTP is supported.

This symbol has the value 0x02.

IMA_ULP_UDP

This flag indicates that UDP is supported.

This symbol has the value 0x04.

These flags may be OR'd together. For example, it's valid for a PHBA to support both TCP and SCTP. In this case, a value of 0x03 would be returned in the *supportedUlps* field of the IMA_PHBA_PROPERTIES structure filled in by a call to IMA_GetPhbaProperties.

5.33 IMA_PHBA_PROPERTIES

Format

```
typedef struct IMA_phba_properties
{
    IMA_WCHAR       vendor[64];
    IMA_WCHAR       model[256];
    IMA_WCHAR       description[256];
    IMA_WCHAR       serialNumber[64];
    IMA_WCHAR       hardwareVersion[256];
    IMA_WCHAR       asicVersion[256];
    IMA_WCHAR       firmwareVersion[256];
    IMA_WCHAR       optionRomVersion[256];
    IMA_WCHAR       driverName[256];
    IMA_WCHAR       driverVersion[256];

    IMA_UINT        supportedUlps;

    IMA_XBOOL       bidirectionalTransfersSupported.
    IMA_UINT        maximumCdbLength;

    IMA_XBOOL       canBeNic;
    IMA_XBOOL       isNic;

    IMA_XBOOL       isInitiator;
    IMA_XBOOL       isTarget;

    IMA_XBOOL       usingTcpOffloadEngine;
    IMA_XBOOL       usingIscsiOffloadEngine;
    IMA_BYTE        reserved[128];

} IMA_PHBA_PROPERTIES;
```

Fields

vendor

A nul terminated Unicode string that contains the name of the vendor of a PHBA. If the first character in this field is nul then the vendor is unknown.

model

A nul terminated Unicode string that contains the name of the model of a PHBA. If the first character in this field is nul then the model is unknown.

description

A nul terminated Unicode string that contains a description of a PHBA. This is a user friendly description of the PHBA. If the first character in this field is nul then there is no description.

serialNumber

A nul terminated Unicode string that contains the serial number of a PHBA. If the first character in this field is nul then the serial number is unknown.

hardwareVersion

A nul terminated Unicode string that contains the hardware version of a PHBA. If the first character in this field is nul then the hardware version is unknown.

asicVersion

A nul terminated Unicode string that contains the ASIC version of a PHBA. If the first character in this field is nul then the ASIC version is unknown or is not applicable.

firmwareVersion

A nul terminated Unicode string that contains the firmware version of a PHBA. If the first character in this field is nul then the firmware version is unknown or is not applicable.

optionRomVersion

A nul terminated Unicode string that contains the option ROM version of a PHBA. If the first character in this field is nul then the option ROM version is unknown or is not applicable.

driverName

A nul terminated Unicode string that contains the full name of the driver controlling a PHBA. If the first character in this field is nul then the name of the driver is unknown.

On operating systems, such as Windows, where files have extensions the full name includes the extension. The full name does not include the path to the file.

driverVersion

A nul terminated Unicode string that contains the version of the driver specified in *driverName*. If the first character in this field is nul then the version of the driver is unknown.

This field can have a known value only if *driverName* has a known value as well.

supportedUlps

A field containing flags that indicate what upper level protocols are supported by a PHBA. Examples of upper level protocols include:

- TCP, represented by IMA_ULP_TCP
- SCTP, represented by IMA_ULP_SCTP
- UDP, represented by IMA_ULP_UDP

bidirectionalTransfersSupported

A extended boolean that indicates if a PHBA supports executing SCSI commands that cause bidirectional transfers.

Note: The value of this field applies to the entire “stack”: the hardware, ASIC, firmware, driver, etc. All shall support SCSI commands that cause bidirectional transfers for this field to be set to IMA_TRUE.

maximumCdbLength

The maximum length, in bytes, of a CDB that can be transferred by a PHBA. If this field has a value of zero that indicates that this value is unknown.

Note: The value of this field applies to the entire “stack”: the hardware, ASIC, firmware, driver, etc. All shall support the maximum CDB length returned in this field.

canBeNic

An extended boolean that indicates if a PHBA can also function as a “standard” NIC concurrently with functioning as an iSCSI PHBA.

Note: The value of this field applies to the entire “stack”: the hardware, ASIC, firmware, driver, etc. All shall support the PHBA functioning concurrently as a NIC for the value returned in this field to be IMA_TRUE.

isNic

A extended boolean that indicates if a PHBA is functioning as a “standard” NIC concurrently with functioning as an iSCSI PHBA.

isInitiator

An extended boolean indicating if the PHBA is functioning as an initiator.

isTarget

An extended boolean indicating if the PHBA is functioning as a target.

usingTcpOffloadEngine

An extended boolean indicating if the PHBA is using a TCP offload engine.

Note: This value shall only be set to IMA_TRUE if a TCP offload engine is present and is being used. If it can be determined that a TCP offload engine is present, but it cannot be determined if that offload engine is being used then this value shall be set to IMA_UNKNOWN.

usingIscsiOffloadEngine

An extended boolean indicating if the PHBA is using a iSCSI offload engine.

Note: This value shall only be set to IMA_TRUE if a iSCSI offload engine is present and is being used. If it can be determined that an iSCSI offload engine is present, but it cannot be determined if that offload engine is being used then this value shall be set to IMA_UNKNOWN.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetPhbaProperties API.

Both *isInitiator* and *isTarget* cannot be set to IMA_FALSE as this would mean that the PHBA was not functioning as either an initiator or target: which means that it’s not functioning.

5.34 IMA_DISCOVERY_PROPERTIES

Format

```
typedef struct IMA_discovery_properties
{
    IMA_BOOL                    iSnsDiscoverySettable;
    IMA_XBOOL                   iSnsDiscoveryEnabled;
    IMA_ISNS_DISCOVERY_METHOD   iSnsDiscoveryMethod;
    IMA_HOST_ID                 iSnsHost;

    IMA_BOOL                    slpDiscoverySettable;
    IMA_XBOOL                   slpDiscoveryEnabled;

    IMA_BOOL                    staticDiscoverySettable;
    IMA_XBOOL                   staticDiscoveryEnabled;

    IMA_BOOL                    sendTargetsDiscoverySettable;
    IMA_XBOOL                   sendTargetsDiscoveryEnabled;

    IMA_BYTE                    reserved[128];

} IMA_DISCOVERY_PROPERTIES;
```

Fields

iSnsDiscoverySettable

A boolean that indicates if iSNS target discovery can be enabled and disabled using the IMA_SetIsnsDiscovery API.

iSnsDiscoveryEnabled

An extended boolean that indicates if iSNS target discovery is currently enabled or disabled.

Note: It is possible for this field to have a value of IMA_TRUE and the *iSnsDiscoverySettable* field to have a value of IMA_FALSE. This means that the associated PHBA/LHBA performs iSNS target discovery and that it cannot be disabled.

iSnsDiscoveryMethod

A value which indicates how the iSNS server is being discovered. This field is valid only if *iSnsDiscoveryEnabled* has the value IMA_TRUE, otherwise the value of this field is undefined.

iSnsHost

A nul terminated Unicode string containing the domain name of the iSNS server the specified PHBA is using. If there is no iSNS server in use, either because iSNS target discovery is disabled or because no iSNS server is found, then this shall be an empty string.

slpDiscoverySettable

A boolean that indicates if SLP target discovery can be enabled and disabled using the IMA_SetSlpDiscovery API.

slpDiscoveryEnabled

An extended boolean that indicates if SLP target discovery is currently enabled or disabled.

Note: It is possible for this field to have a value of IMA_TRUE and the *slpDiscoverySettable* field to have a value of IMA_FALSE. This means that the associated PHBA/LHBA performs SLP target discovery and that it cannot be disabled.

staticDiscoverySettable

A boolean that indicates if static target discovery can be enabled and disabled using the IMA_SetStaticDiscovery API.

staticDiscoveryEnabled

An extended boolean that indicates if static target discovery is currently enabled or disabled.

Note: It is possible for this field to have a value of IMA_TRUE and the *staticDiscoverySettable* field to have a value of IMA_FALSE. This means that the specified PHBA performs static target discovery and that it cannot be disabled.

sendTargetsDiscoverySettable

A boolean that indicates if `SendTargets` target discovery can be enabled and disabled using the IMA_SetSendTargetsDiscovery API.

sendTargetsDiscoveryEnabled

An extended boolean that indicates if `SendTargets` target discovery is currently enabled or disabled.

Note: It is possible for this field to have a value of IMA_TRUE and the *sendTargetsDiscoverySettable* field to have a value of IMA_FALSE. This means that the associated PHBA/LHBA performs `SendTargets` target discovery and that it cannot be disabled.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetDiscoveryProperties API.

It is possible for all of the *xxxDiscoveryEnabled* fields to have the value IMA_FALSE and for all of the *xxxDiscoverySettable* fields to also have the value IMA_FALSE. This indicates that no target discovery is being done by the PHBA/LHBA and no target discovery can be enabled for the PHBA/LHBA. Such a set of discovery properties would indicate that a PHBA/LHBA is acting as a target, but not as an initiator.

5.35 IMA_PHBA_DOWNLOAD_IMAGE_TYPE

Format

```
typedef enum IMA_phba_download_image_type
{
    IMA_DOWNLOAD_IMAGE_TYPE_FIRMWARE        =  0,
    IMA_DOWNLOAD_IMAGE_TYPE_OPTION_ROM      =  1,
    IMA_DOWNLOAD_IMAGE_TYPE_ALL             =  2,
    IMA_DOWNLOAD_IMAGE_TYPE_BOOTCODE        =  3
} IMA_PHBA_DOWNLOAD_IMAGE_TYPE;
```

Fields

IMA_IMAGE_TYPE_FIRMWARE

This value indicates that a file contains a firmware download image.

This symbol has the value 0.

IMA_IMAGE_TYPE_OPTION_ROM

This value indicates that a file contains an option ROM download image.

This symbol has the value 1.

IMA_IMAGE_TYPE_ALL

This value indicates that a file contains all of the download images, i.e. it contains both a firmware download image and an option ROM download image.

This symbol has the value 2.

IMA_DOWNLOAD_IMAGE_TYPE_BOOTCODE

This value indicates that the file contains a bootcode image, such as FCode, BIOS or EFI.

This symbol has the value 3.

Remarks

This type is used in the IMA_PHBA_DOWNLOAD_IMAGE_PROPERTIES structure and by the IMA_PhbaDownload API.

5.36 IMA_PHBA_DOWNLOAD_IMAGE_PROPERTIES

Format

```
typedef struct IMA_phba_download_image_properties
{
    IMA_PHBA_DOWNLOAD_IMAGE_TYPE    imageType;
    IMA_WCHAR                       version[32];
    IMA_WCHAR                       description[512];
    IMA_XBOOL                       upgrade;
} IMA_PHBA_DOWNLOAD_IMAGE_PROPERTIES;
```

Fields

imageType

This field indicates the type of the associated download image.

version

This field contains a nul terminated Unicode string which indicates the version of the download image. If the version cannot be determined or there is no version then this string is empty.

description

This field contains a nul terminated Unicode string which describes the download image and possibly any bug fixes that the download image contains. If there is no description then this string is empty.

upgrade

An extended boolean indicating if the specified download image would be an upgrade, i.e. a later version, than the image that currently resides in the associated PHBA. If this field has the value IMA_TRUE then it is an upgrade. If this field has the value IMA_FALSE then it is not an upgrade, i.e. it may be the same version or it may be a downgrade. If this field has the value IMA_UNKNOWN then it cannot be determined if this is an upgrade or not.

Remarks

This type is used by both the IMA_IsPhbaDownloadFile and the IMA_PhbaDownload APIs.

5.37 IMA_ISNS_DISCOVERY_METHOD

Format

```
typedef enum IMA_isns_discovery_method
{
    IMA_ISNS_DISCOVERY_METHOD_STATIC        =  0,
    IMA_ISNS_DISCOVERY_METHOD_DHCP          =  1,
    IMA_ISNS_DISCOVERY_METHOD_SLP           =  2
} IMA_ISNS_DISCOVERY_METHOD;
```

Fields

IMA_ISNS_DISCOVERY_METHOD_STATIC

This value indicates that the discovery method is static.

This symbol has the value 0.

IMA_ISNS_DISCOVERY_METHOD_DHCP

This value indicates that the discovery method is DHCP.

This symbol has the value 1.

IMA_ISNS_DISCOVERY_METHOD_SLP

This value indicates that the discovery method is SLP.

This symbol has the value 2.

Remarks

This type is used to indicate how a the iSNS server is discovered. It is used as a field in the IMA_DISCOVERY_PROPERTIES structure and as a parameter to the IMA_SetIsnsDiscovery API.

5.38 IMA_PHBA_DOWNLOAD_PROPERTIES

Format

```
typedef struct IMA_phba_download_properties
{
    IMA_BOOL        isPhbaDownloadFileSupported;
    IMA_BOOL        optionRomDownloadSupported;
    IMA_BOOL        firmwareDownloadSupported;

    IMA_BYTE        reserved[32];

} IMA_PHBA_DOWNLOAD_PROPERTIES;
```

Fields

isPhbaDownloadFileSupported

A boolean indicating if the PHBA supports the IMA_IsPhbaDownloadFile API.

optionRomDownloadSupported

A boolean indicating if the PHBA supports downloading option ROM code.

firmwareDownloadSupported

A boolean indicating if the PHBA supports downloading firmware code.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetPhbaDownloadProperties API.

5.39 IMA_IPSEC_PROPERTIES

Format

```
typedef struct IMA_ipsec_properties
{
    IMA_BOOL        ipsecSupported;
    IMA_BOOL        implementedInHardware;
    IMA_BOOL        implementedInSoftware;

    IMA_BYTE        reserved[32];

} IMA_IPSEC_PROPERTIES;
```

Fields

ipsecSupported

A boolean indicating if IPsec is supported in accordance with the requirements of the iSCSI standard.

implementedInHardware

An boolean indicating if IPsec is provided in hardware.

implementedInSoftware

An boolean indicating if IPsec is provided in software.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetIpsecProperties API.

It is possible for both *implementedInHardware* and *implementedInSoftware* to both have the value of IMA_TRUE. This means that some IPsec algorithms are implemented in hardware, while other algorithms are implemented in software. It is not possible to determine which algorithms are implemented in which location.

It is not valid to return this structure where *ipsecSupported* is set to IMA_TRUE and both implementedInHardware and implementedInSoftware are set to IMA_FALSE.

5.40 IMA_MIN_MAX_VALUE

Format

```
typedef struct IMA_min_max_value
{
    IMA_BOOL        currentValueValid;
    IMA_BOOL        settable;

    IMA_UINT32      currentValue;
    IMA_UINT32      defaultValue;
    IMA_UINT32      minimumValue;
    IMA_UINT32      maximumValue;
    IMA_UINT32      incrementValue;

} IMA_MIN_MAX_VALUE;
```

Fields

currentValueValid

A boolean indicating if the *currentValue* field contains a valid value.

settable

Indicates if the corresponding property is settable. If this field has the value IMA_TRUE then the *defaultValue*, *minimumValue*, *maximumValue*, and *incrementValue* fields shall contain valid values. If this field has the value IMA_FALSE then these fields have undefined values.

currentValue

If *currentValueValid* has the value IMA_TRUE then this field contains the current value of the associated property. If *currentValueValid* has the value IMA_FALSE then the value of this field is undefined.

defaultValue

If *settable* has the value IMA_TRUE then this field contains the implementation's default value of the associated property. If *settable* has the value IMA_FALSE then the value of this field is undefined.

minimumValue

If *settable* has the value IMA_TRUE then this field contains the implementation's minimum value of the associated property. If *settable* has the value IMA_FALSE then the value of this field is undefined.

maximumValue

If *settable* has the value IMA_TRUE then this field contains the implementation's maximum value of the associated property. If *settable* has the value IMA_FALSE then the value of this field is undefined.

incrementValue

If *settable* has the value IMA_TRUE then this field contains a value that can be added to or subtracted from *currentValue* to obtain other possible values of the associated property. If *settable* has the value IMA_FALSE then the value of this field is undefined.

Remarks

If the *currentValueValid* field is IMA_FALSE then the value of *settable* shall also be set to IMA_FALSE.

The fields in this structure contain values that are defined by the implementation and not by IETF RFC 3720. It is possible that an implementation may be more or less restrictive in the values that it can accept than IETF RFC 3720 allows.

An example of how to use *incrementValue*: Suppose that a structure is obtained where *currentValueValid* is IMA_TRUE, *settable* is IMA_TRUE, *currentValue* is 50, *defaultValue* is 50, *minimumValue* is 30, *maximumValue* is 70 and *incrementValue* is 10. In this case, the possible values that the property can be set to are 30, 40, 50, 60, and 70. The new value shall be the current value plus or minus some multiple of *incrementValue*.

5.41 IMA_BOOL_VALUE

Format

```
typedef struct IMA_bool_value
{
    IMA_BOOL        currentValueValid;
    IMA_BOOL        settable;

    IMA_BOOL        currentValue;
    IMA_BOOL        defaultValue;

} IMA_BOOL_VALUE;
```

Fields

currentValueValid

A boolean indicating if the *currentValue* field contains a valid value.

settable

Indicates if the corresponding property is settable. If this field has the value IMA_TRUE then the *defaultValue* shall contain a valid value. If this field has the value IMA_FALSE then the *defaultValue* field contains an undefined value.

currentValue

If *currentValueValid* has the value IMA_TRUE then this field contains the current value of the associated property. If *currentValueValid* has the value IMA_FALSE then the value of this field is undefined.

defaultValue

If *settable* has the value IMA_TRUE then this field contains the implementation's default value of the associated property. If *settable* has the value IMA_FALSE then the value of this field is undefined.

5.42 IMA_MAC_ADDRESS

Format

typedef IMA_BYTE IMA_MAC_ADDRESS[6];

Remarks

A MAC address is a series of six bytes which uniquely identifies a physical or logical network port. Byte 0 of the array is the most significant byte of the MAC address, byte 1 is the next most significant byte, etc., on to byte 5 of the array which is the least significant byte of the MAC address.

5.43 IMA_LNP_PROPERTIES

Format

```
typedef struct IMA_lnp_properties
{
    IMA_MAC_ADDRESS   macAddress;
    IMA_BOOL          macAddressSettable;

    IMA_BYTE          reserved[32];

} IMA_LNP_PROPERTIES;
```

Fields

macAddress

An array of bytes containing the MAC address.

It is possible that an LNP will have the same MAC address as a PNP. This means that there is a one to one relationship between the LNP and the PNP.

macAddressSettable

A boolean indicating if the MAC address of the logical network port can be set.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetLnpProperties API.

5.44 IMA_PNP_PROPERTIES

Format

```
typedef struct IMA_pnp_properties
{
    IMA_OID             associatedPhbaOid;

    IMA_MAC_ADDRESS     macAddress;
    IMA_BOOL            macAddressSettable;

    IMA_UINT            maximumTransferRate;
    IMA_UINT            currentTransferRate;

    IMA_UINT            maximumFrameSize;

    IMA_BYTE            reserved[64];

} IMA_PNP_PROPERTIES;
```

Fields

associatedPhbaOid

The object ID of the PHBA to which the PNP the structure describes is attached.

macAddress

An array of bytes containing the MAC address.

It is possible that a PNP will have the same MAC address as an LNP. This means that there is a one to one relationship between the PNP and the LNP.

macAddressSettable

A boolean indicating if the MAC address of the physical network port can be set.

maximumTransferRate

The maximum number of megabits that can be transferred in one second through the port at any time.

So, if the maximum transfer rate of the port is 10 Mb then this field will contain 10. If the maximum transfer rate of the port is 100 Mb then this field will contain 100, etc.

currentTransferRate

The maximum number of megabits that can be transferred in one second through the port at the current time.

So, if the current transfer rate of the port is 10 Mb then this field will contain 10. If the current transfer rate of the port is 100 Mb then this field will contain 100, etc.

This value is the current transfer rate of the port, it has nothing to do with how much data is actually being transferred through the port.

maximumFrameSize

The maximum size of a frame, in bytes.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetPnpProperties API.

5.45 IMA_PNP_STATISTICS

Format

```
typedef struct IMA_pnp_statistics
{
    IMA_UINT64          bytesSent;
    IMA_UINT32          pdusSent;
    IMA_UINT64          bytesReceived;
    IMA_UINT32          pdusReceived;

} IMA_PNP_STATISTICS;
```

Fields

bytesSent

The number of bytes sent in iSCSI PDUs on a physical network port.

bytesReceived

The number bytes received in iSCSI PDUs on a physical network port.

pdusSent

The number of iSCSI PDUs sent on a physical network port.

pdusReceived

The number of iSCSI PDUs received on a physical network port.

Remarks

This structure is returned by the IMA_GetPnpStatistics API.

5.46 IMA_NETWORK_PORTAL_PROPERTIES

Format

```
typedef struct IMA_network_portal_properties
{
    IMA_IP_ADDRESS      ipAddress;
    IMA_OID             associatedLnp;

    IMA_BYTE            reserved[32];

} IMA_NETWORK_PORTAL_PROPERTIES;
```

Fields

ipAddress

The IP address of the network portal.

associatedLnp

The OID of the LNP with which the network portal is associated.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetNetworkPortalProperties API.

5.47 IMA_PHBA_STATUS

Format

```
typedef enum IMA_phba_status
{
    IMA_PHBA_STATUS_WORKING                 =   0,
    IMA_PHBA_STATUS_FAILED                  =   1.
} IMA_PHBA_STATUS;
```

Values

IMA_PHBA_STATUS_WORKING

This status indicates that the PHBA is working properly.

This symbol has the value 0.

IMA_PHBA_STATUS_FAILED

This status indicates that the PHBA has failed and is no longer functioning.

This symbol has the value 1.

Remarks

This status is returned by the IMA_GetPhbaStatus API.

5.48 IMA_ NETWORK_PORT_STATUS

Format

```
typedef enum IMA_network_port_status
{
    IMA_NETWORK_PORT_STATUS_WORKING          =  0,
    IMA_NETWORK_PORT_STATUS_DEGRADED         =  1,
    IMA_NETWORK_PORT_STATUS_CRITICAL         =  2,
    IMA_NETWORK_PORT_STATUS_FAILED           =  3,
    IMA_NETWORK_PORT_STATUS_DISCONNECTED     =  4
} IMA_NETWORK_PORT_STATUS;
```

Fields

IMA_NETWORK_PORT_STATUS_WORKING

This status indicates that the specified link is in a normal operational state.

- For a PNP this status indicates that the port is working correctly.
- For an LNP this status indicates means that all of the PNPs associated with the LNP are working.

IMA_NETWORK_PORT_STATUS_DEGRADED

This status indicates that the specified link is in a degraded operational state.

- This status cannot be specified for a PNP.
- For an LNP this status indicates that two or more of the PNPs associated with the LNP are in a working state, but one or more of the PNP's associated with the LNP has failed or is disconnected.

  This means that some failures have occurred, but there is still redundancy.

IMA_NETWORK_PORT_STATUS_CRITICAL

This status indicates that the specified link is in a critical operational state.

- This status cannot be specified for a PNP.
- For an LNP this status indicates that only one of the PNPs associated with the LNP is in a working state; all other associated PNPs have either failed or are disconnected.

  This means that some failures have occurred, and there is no redundancy.

IMA_NETWORK_PORT_STATUS_FAILED

This status indicates the specified link has failed.

- For a PNP this status indicates that the port has failed; that it cannot be used to transmit data.
- For an LNP this status indicates that all of the PNPs associated with the LNP have failed.

IMA_NETWORK_PORT_STATUS_DISCONNECTED

This status indicates that the specified link is disconnected.

- For a PNP this status indicates that the port is disconnected from the network.

- For an LNP this status indicates that one of the PNPs associated with the LNP is in a disconnected state and that all PNPs associated with the LNP are not in a working state.

  Another way of thinking of this is that none of the PNPs associated with the LNP is in a working state and at least one of the PNPs is in a disconnected state.

Remarks

This status is returned by the IMA_GetLinkStatus API.

A link's status should in no way be confused with iSCSI connection or iSCSI session failures that may have used that link. An LNP that fails or becomes disconnected may result iSCSI connection failures and possibly iSCSI session failures as well. However, iSCSI connection and iSCSI session failures can occur for reasons other than an LNP failure or disconnect.

5.49 IMA_TARGET_DISCOVERY_METHOD

Format

```
typedef enum IMA_TARGET_DISCOVERY_METHOD
{
    IMA_TARGET_DISCOVERY_METHOD_STATIC      =  1,
    IMA_TARGET_DISCOVERY_METHOD_SLP         =  2,
    IMA_TARGET_DISCOVERY_METHOD_ISNS        =  4,
    IMA_TARGET_DISCOVERY_METHOD_SENDTARGETS =  8
} IMA_TARGET_DISCOVERY_METHOD;
```

Fields

IMA_TARGET_DISCOVERY_METHOD_STATIC

This value indicates that a target was discovered using static discovery.

This symbol has the value 1.

IMA_TARGET_DISCOVERY_METHOD_SLP

This value indicates that a target was discovered using SLP.

This symbol has the value 2.

IMA_TARGET_DISCOVERY_METHOD_ISNS

This value indicates that a target was discovered using iSNS.

This symbol has the value 4.

IMA_TARGET_DISCOVERY_METHOD_SENDTARGETS

This value indicates that a target was discovered using `SendTargets`.

This symbol has the value 8.

5.50 IMA_TARGET_PROPERTIES

Format

```
typedef struct IMA_target_properties
{
    IMA_OID             associatedNodeOid;
    IMA_OID             associatedLhbaOid;

    IMA_NODE_NAME       name;
    IMA_NODE_ALIAS      alias;
    IMA_UINT32          discoveryMethodFlags;

    IMA_BOOL            sendTargetsDiscoverySettable;
    IMA_BOOL            sendTargetsDiscoveryEnabled;  IMA_BYTE
    reserved[128];

} IMA_TARGET_PROPERTIES;
```

Fields

associatedNodeOid

The object ID of the node through which the target is being accessed.

associatedLhbaOid

The object ID of the LHBA that is used to communicate with the target.

name

A nul terminated Unicode string that contains the name of the target.

alias

A nul terminated Unicode string that contains the alias of the target. If the target does not have an alias or the alias is not available then this value shall be an empty string.

discoveryMethodFlags

An integer which contains flags indicating how the target was discovered. This value is a bitwise OR'ing of the flags defined in IMA_TARGET_DISCOVERY_METHOD.

sendTargetsDiscoverySettable

A boolean indicating if a client can control if `SendTargets` commands are sent to the target in an attempt to discover additional targets. If this field has the value IMA_TRUE then a client can call IMA_SetSendTargetsDiscovery to enable/disable the sending `SendTargets` to the target associated with the properties structure. If this field has the value IMA_FALSE then if a client called the IMA_SetSendTargetsDiscovery API specifying the target associated with the properties structure that call would fail with an IMA_ERROR_NOT_SUPPORTED error.

sendTargetsDiscoveryEnabled

A boolean indicating if `SendTargets` commands are being sent to the target in an attempt to discover additional targets.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetTargetProperties API.

It is possible for the field *sendTargetsDiscoverySettable* to have the value IMA_FALSE and the field *sendTargetsDiscoveryEnabled* to have the value IMA_TRUE at the same time. If this is the case then `SendTargets` discovery is being performed on the target and the client cannot disable this.

5.51 IMA_TARGET_ERROR_STATISTICS

Format

```
typedef struct IMA_target_error_statistics
{
    IMA_BOOL        loginFailedCountValid;
    IMA_UINT32      loginFailedCount;

    IMA_BOOL        sessionFailedCountValid;
    IMA_UINT32      sessionFailedCount;

    IMA_BOOL        headerOrDigestSessionFailedCountValid;
    IMA_UINT32      headerOrDigestSessionFailedCount

    IMA_BOOL        timeLimitExceededSessionFailedCountValid;
    IMA_UINT32      timeLimitExceededSessionFailedCount;

    IMA_BOOL        formatErrorSessionFailedCountValid;
    IMA_UINT32      formatErrorSessionFailedCount;

    IMA_BOOL        closedConnectionDueToTimeoutCountValid;
    IMA_UINT32      closedConnectionDueToTimeoutCount;

    IMA_BOOL        lastLoginFailureTimeValid;
    IMA_DATETIME    lastLoginFailureTime;

    IMA_BYTE        reserved[64];

} IMA_TARGET_ERROR_STATISTICS;
```

Fields

loginFailedCountValid

A boolean indicating if the *loginFailedCount* field contains a valid value.

loginFailedCount

If the *loginFailedCountValid* field has the value IMA_TRUE then this field contains the number of times that iSCSI login attempts failed across all attempted sessions with the target. If the *loginFailedCountValid* field has the value IMA_FALSE then the value of this field is undefined.

sessionFailedCountValid

A boolean indicating if the *sessionFailedCount* field contains a valid value.

sessionFailedCount

If the *sessionFailedCountValid* field has the value IMA_TRUE then this field contains the number of times iSCSI sessions failed with the associated target. If the *sessionFailedCountValid* field has the value IMA_FALSE then the value of this field is undefined.

headerOrDigestSessionFailedCountValid

A boolean indicating if the *headerOrDigestSessionFailedCount* field contains a valid value.

headerOrDigestSessionFailedCount

If the *headerOrDigestSessionFailedCountValid* field has the value IMA_TRUE then this field contains the total number of iSCSI PDU header or data digest errors across all iSCSI sessions of the associated target. If the *headerOrDigestSessionFailedCountValid* field has the value IMA_FALSE then the value of this field is undefined.

timeLimitExceededSessionFailedCountValid

A boolean indicating if the *timeLimitExceededSessionFailedCount* field contains a valid value.

timeLimitExceededSessionFailedCount

If the *timeLimitExceededSessionFailedCountValid* field has the value IMA_TRUE then this field contains the number of sessions which were failed due to a sequence exceeding a time limit. If the *timeLimitExceededSessionFailedCountValid* field has the value IMA_FALSE then the value of this field is undefined.

formatErrorSessionFailedCountValid

A boolean indicating if the *formatErrorSessionFailedCount* field contains a valid value.

formatErrorSessionFailedCount

If the *formatErrorSessionFailedCountValid* field has the value IMA_TRUE then this field contains the total number of sessions which were failed due to receipt of a PDU which contained a format error. If the *formatErrorSessionFailedCountValid* field has the value IMA_FALSE then the value of this field is undefined.

closedConnectionDueToTimeoutCountValid

A boolean indicating if the *closedConnectionDueToTimeoutCount* field contains a valid value.

closedConnectionDueToTimeoutCount

If the *closedConnectionDueToTimeoutCountValid* has the value IMA_TRUE then this field contains the number of times iSCSI connections with the associated target were terminated due to timeout. If the *closedConnectionDueToTimeoutCountValid* has the value IMA_FALSE then the value of this field is undefined.

lastLoginFailureTimeValid

A boolean indicating if the *lastLoginFailureTime* field contains a valid value.

lastLoginFailureTime

If the *lastLoginFailureTimeValid* field has the value IMA_TRUE then this field contains the time stamp of the last failed iSCSI login attempt with the associated target. If the *lastLoginFailureTimeValid* field has the value IMA_FALSE then the value of this field is undefined.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetTargetErrorStatistics API.

5.52 IMA_LU_PROPERTIES

Format

```
typedef struct IMA_lu_properties
{
    IMA_OID             associatedTargetOid;
    IMA_BYTE            targetLun[8];

    IMA_BOOL            exposedToOs;
    IMA_DATETIME        timeExposedToOs;

    IMA_BOOL            osDeviceNameValid;
    IMA_WCHAR           osDeviceName[64];

    IMA_BOOL            osParallelIdsValid;
    IMA_UINT32          osBusNumber;
    IMA_UINT32          osTargetId;
    IMA_UINT32          osLun;

    IMA_BYTE            reserved[128];

} IMA_LU_PROPERTIES;
```

Fields

associatedTargetOid

The object ID of the target to which the specified logical unit is attached.

targetLun

The logical unit number of the logical unit on the target.

exposedToOs

A boolean indicating if the logical unit is currently exposed to the operating system, i.e., the operating system can communicate with the logical unit using standard operating system device interfaces.

timeExposedToOs

If *exposedToOs* contains the value IMA_TRUE this field contains the date and time the LU was exposed to the OS.

osDeviceNameValid

A boolean indicating if the osDeviceName field contains a valid value. This field shall be IMA_TRUE if the name is known. This field shall be IMA_FALSE if the name is not known or if the logical unit has no name that indentifies it to the operating system.

osDeviceName

If *osDeviceNameValid* has the value IMA_TRUE then this field contains a nul terminated Unicode string that indicates the name that the logical unit was declared to the operating system with.

If *osDeviceNameValid* has the value IMA_FALSE then the value of this field is undefined.

See Annex A for details on how to set this field for each operating system.

osParallelIdsValid

A boolean indicating if the subsequent parallel SCSI identifiers are valid or not. If the parallel SCSI identifiers are known then this field shall have the value IMA_TRUE.

If these values are not known or if the logical unit is declared to the operating system as a parallel SCSI logical unit then this field shall have the value IMA_FALSE.

osBusNumber

If *osParallelIdsValid* has the value IMA_TRUE then this field contains the parallel SCSI bus number that the LU was declared to the operating system with. If the operating system does not have the concept of buses on a parallel SCSI controller then this field shall have the value zero.

If *osParallelIdsValid* has the value IMA_FALSE then the value of this field is undefined.

osTargetId

If *osParallelIdsValid* has the value IMA_TRUE then this field contains the parallel SCSI target ID that the logical unit was declared to the operating system with.

If *osParallelIdsValid* has the value IMA_FALSE then the value of this field is undefined.

osLun

If *osParallelIdsValid* has the value IMA_TRUE then this field contains the parallel SCSI logical unitnumber that the logical unit was declared to the operating system with.

If *osParallelIdsValid* has the value IMA_FALSE then the value of this field is undefined.

reserved

This field is reserved.

Remarks

This structure is returned by the IMA_GetLuProperties API.

5.53 IMA_DEVICE_STATISTICS

Format

```
typedef struct IMA_device_statistics
{
    IMA_UINT64          scsiPayloadBytesSent;
    IMA_UINT64          scsiPayloadBytesReceived;

    IMA_UINT64          iScsiPduBytesSent;
    IMA_UINT64          iScsiPduBytesReceived;

    IMA_UINT64          iScsiPdusSent;
    IMA_UINT64          iScsiPdusReceived;

    IMA_UINT64          millisecondsSpentSending;
    IMA_UINT64          millisecondsSpentReceiving;

} IMA_DEVICE_STATISTICS;
```

Fields

scsiPayloadBytesSent

This value contains the number of bytes sent in SCSI payloads.

scsiPayloadBytesReceived

This value contains the number of bytes received in SCSI payloads.

iScsiPduBytesSent

This value contains the number of bytes sent in iSCSI PDUs.

iScsiPduBytesReceived

This value contains the number of bytes received in iSCSI PDUs.

iScsiPdusSent

This value contains the number of iSCSI PDUs sent.

iScsiPdusReceived

This value contains the number of iSCSI PDUs received.

millisecondsSpentSending

This value is the **approximate** number of milliseconds that have been spent sending data. If this value is zero then the value is unknown.

millisecondsSpentReceiving

This value is the **approximate** number of milliseconds that have been spent receiving data. If this value is zero then the value is unknown.

Remarks

This structure is returned by the IMA_GetDeviceStatistics API.

5.54 IMA_STATISTICS_PROPERTIES

Format

```
typedef struct IMA_statistics_properties
{
    IMA_BOOL        statisticsCollectionSettable;
    IMA_BOOL        statisticsCollectionEnabled;

} IMA_STATISTICS_PROPERTIES;
```

Fields

statisticsCollectionSettable

A boolean indicating if statistics collection is settable, i.e., it can be enabled and disabled.

statisticsCollectionEnabled

A boolean indicating if statistics collection is enabled.

Remarks

This structure is returned by the IMA_GetStatisticsProperties API.

It is possible for *statisticsCollectionEnabled* to have the value IMA_TRUE and *statisticsCollectionSettable* to have the value IMA_FALSE. In this case, statistics collection is always enabled and cannot be disabled.

5.55 IMA_AUTHMETHOD

Format

```
typedef enum IMA_authmethod
{
    IMA_AUTHMETHOD_NONE     =  0,
    IMA_AUTHMETHOD_CHAP     =  1,
    IMA_AUTHMETHOD_SRP      =  2,
    IMA_AUTHMETHOD_KRB5     =  3,
    IMA_AUTHMETHOD_SPKM1    =  4,
    IMA_AUTHMETHOD_SPKM2    =  5
} IMA_AUTHMETHOD;
```

Fields

IMA_AUTHMETHOD_NONE

This indicates that no authentication is performed.

This symbol has the value 0.

IMA_AUTHMETHOD_CHAP

This indicates that Challenge Handshake Authentication Protocol (CHAP) authentication compatible with IETF RFC 3720 is performed.

This symbol has the value 1.

IMA_AUTHMETHOD_SRP

This indicates that Secure Remote Password (SRP) authentication compatible with IETF RFC 3720 is performed.

This symbol has the value 2.

IMA_AUTHMETHOD_KRB5

This indicates that Kerberos V5 (KRB5) authentication compatible with IETF RFC 3720 is performed.

This symbol has the value 3.

IMA_AUTHMETHOD_SPKM1

This indicates that Simple Public Key Mechanism one (SPKM1) authentication compatible with IETF RFC 3720 is performed.

This symbol has the value 4.

IMA_AUTHMETHOD_SPKM2

This indicates that Simple Public Key Mechanism two (SPKM2) authentication compatible with IETF RFC 3720 is performed.

This symbol has the value 5.

5.56 IMA_CHAP_INITIATOR_AUTHPARMS

Format

```
typedef struct IMA_chap_initiator_authparms
{

    IMA_UINT        retries;

    IMA_BYTE        name[512];
    IMA_UINT        nameLength;

    IMA_UINT        minValueLength;
    IMA_UINT        maxValueLength;

    IMA_BYTE        challengeSecret[256];
    IMA_UINT        challengeSecretLength;

    IMA_BYTE        reserved[512];

} IMA_CHAP_INITIATOR_AUTHPARMS;
```

Fields

retries

When this structure is retrieved this field contains the number of retries that the implementation will perform when a challenge fails.

When this structure is being set this field contains the recommended number of retries that the implementation should use. The implementation is free to ignore this value if it chooses.

name

An array of one or more bytes and shall be formatted per section 4.1 of RFC 1994.

nameLength

The number of bytes that have been set in *name*.

minValueLength

When this structure is retrieved this field contains the minimum number of bytes that can be used by the implementation to construct the CHAP Value.

When this structure is being set this field contains the recommended minimum number of bytes to be used by the implementation to constructor the CHAP Value.

maxValueLength

When this structure is retrieved this field contains the maximum number of bytes that can be used by the implementation to construct the CHAP Value.

When this structure is being set this field contains the recommended maximum number of bytes to be used by the implementation to constructor the CHAP Value.

challengeSecret

When this structure is retrieved this field is unused and this array shall contain all zeros.

When this structure is set this field contains the CHAP Secret that is used when computing the hash value that is used to challenge a target. It is also used to compute the hash value when challenged by a target.

challengeSecretLength

When this structure is retrieved this field is unused and shall contain the value zero.

When this structure is being set this field contains the the length, in bytes, of the secret that has been stored in *challengeSecret*.

reserved

This field is reserved and shall be set to all zeros.

Remarks

This structure is used to set the CHAP initiator authentication parameters for an LHBA.

This structure is included in the IMA_INITIATOR_AUTHPARMS union.

Currently, no method is provided to determine the hash algorithms that may be used, the hash algorithms to use, or their preferred order of use.

5.57 IMA_SRP_INITIATOR_AUTHPARMS

Format

```
typedef struct IMA_srp_initiator_authparms
{

    IMA_BYTE        userName[512];
    IMA_UINT        userNameLength;

    IMA_BYTE        reserved[512];

} IMA_SRP_INITIATOR_AUTHPARMS;
```

Fields

userName

The name of the user to be transmitted to the host (in this case the target) as specified in RFC 2945.

userNameLength

The length, in bytes, of the name specified in *userName*.

reserved

This field is reserved and shall be set to zero.

Remarks

This structure is used to set the SRP initiator authentication parameters for an LHBA.

This structure is included in the IMA_INITIATOR_AUTHPARMS union .

5.58 IMA_KRB5_INITIATOR_AUTHPARMS

Format

```
typedef struct IMA_krb5_initiator_authparms
{

    IMA_BYTE            clientKey[1024];
    IMA_UINT            clientKeyLength;

    IMA_BYTE            reserved[2048];

} IMA_KRB5_INITIATOR_AUTHPARMS;
```

Fields

clientKey

When this structure is retrieved this field shall be set to zero.

When this structure is set this field shall contain the client key to be used.

clientKeyLength

When this structure is retrieved this field shall be set to zero.

When this structure is set this field shall contain the length, in bytes, of the client key stored in *clientKey*.

reserved

This field is reserved and shall be set to 0.

Remarks

This structure is used to set the Kerberos initiator authentication parameters for an LHBA .

This structure is included in the IMA_INITIATOR_AUTHPARMS union .

5.59 IMA_SPKM_INITIATOR_AUTHPARMS

Format

```
typedef struct IMA_spkm_initiator_authparms
{

    IMA_BYTE        privateKey[4096];
    IMA_UINT        privateKeyLength;

    void            publicKey[4096];
    IMA_UINT        publicKeyLength;

    IMA_BYTE        reserved[4096];

} IMA_SPKM_INITIATOR_AUTHPARMS;
```

Fields

privateKey

When this structure is retrieved this field shall be set to all zeros.

When this structure is set this field contains the private key of the associated object.

privateKeyLength

When this structure is retrieved this field shall be zero.

When this structure is set this field contains the length, in bytes, of the private key stored in *privateKey*.

publicKey

When this structure is retrieved this field shall be the currently set public key of the associated object.

When this structure is set this field contains the new public key of the associated object.

publicKeyLength

When this structure is retrieved this field shall be the length, in bytes, of the public key stored in *publicKey*.

When this structure is set this field contains the length, in bytes, of the public key stored in *publicKey*.

Remarks

This structure is used to set the SPKM1 and SPKM2 initiator authentication parameters for an LHBA .

This structure is included in the IMA_INITIATOR_AUTHPARMS union .

No method is provided to retrieve or control the algorithm identifiers used by SPKM1 or SPKM2.

5.60 IMA_INITIATOR_AUTHPARMS

Format

```
typedef union IMA_initiator_authparms
{

    IMA_CHAP_INITIATOR_AUTHPARMS        chapParms;
    IMA_SRP_INITIATOR_AUTHPARMS         srpParms;
    IMA_KRB5_INITIATOR_AUTHPARMS        kerberosParms;
    IMA_SPKM_INITIATOR_AUTHPARMS        spkmParms;

} IMA_INITIATOR_AUTHPARMS;
```

Fields

chapParms

The initiator authentication parameters for CHAP if the authentication method is IMA_AUTHMETHOD_CHAP.

srpParms

The initiator authentication parameters for SRP if the authentication method is IMA_AUTHMETHOD_SRP.

kerberosParms

The initiator authentication parameters for Kerberos if the authentication method is IMA_AUTHMETHOD_KRB5.

spkmParms

The initiator authentication parameters for SPKM if the authentication method is IMA_AUTHMETHOD_SPKM1 or IMA_AUTHMETHOD_SPKM2.

Remarks

5.61 This structure is used as a parameter to the IMA_GetInitiatorAuthParms and the IMA_SetInitiatorAuthParms APIs. IMA_CHAP_TARGET_AUTHPARMS

Format

typedef IMA_CHAP_INITIATOR_AUTHPARMS
IMA_CHAP_TARGET_AUTHPARMS;

Remarks

This is based on IMA_CHAP_INITIATOR_AUTHPARMS and is used to represent target CHAP parameters for mutual authentication.

5.62 IMA_SRP_TARGET_AUTHPARMS

Format

typedef IMA_SRP_INITIATOR_AUTHPARMS IMA_SRP_TARGET_AUTHPARMS;

Remarks

This is based on IMA_SRP_INITIATOR_AUTHPARMS and is used to represent target SRP parameters for mutual authentication.

5.63 IMA_KRB5_TARGET_AUTHPARMS

Format

typedef IMA_KRB5_INITIATOR_AUTHPARMS IMA_KRB5_TARGET_AUTHPARMS;

Remarks

This is based on IMA_KRB5_INITIATOR_AUTHPARMS and is used to represent target Kerberos parameters for mutual authentication.

5.64 IMA_SPKM_TARGET_AUTHPARMS

Format

typedef IMA_SPKM_INITIATOR_AUTHPARMS
IMA_SPKM_TARGET_AUTHPARMS;

Remarks

This is based on IMA_SPKM_INITIATOR_AUTHPARMS and is used to represent target SPKM parameters for mutual authentication.

5.65 IMA_TARGET_AUTHPARMS

Format

```
typedef union IMA_target_authparms
{

    IMA_CHAP_TARGET_AUTHPARMS       chapParms;
    IMA_SRP_TARGET_AUTHPARMS        srpParms;
    IMA_KRB5_TARGET_AUTHPARMS       kerberosParms;
    IMA_SPKM_TARGET_AUTHPARMS       spkmParms;

} IMA_TARGET_AUTHPARMS;
```

Fields

chapParms

The target authentication parameters for CHAP if the authentication method is IMA_AUTHMETHOD_CHAP.

srpParms

The target authentication parameters for SRP if the authentication method is IMA_AUTHMETHOD_SRP.

kerberosParms

The target authentication parameters for Kerberos if the authentication method is IMA_AUTHMETHOD_KRB5.

spkmParms

The target authentication parameters for SPKM if the authentication method is IMA_AUTHMETHOD_SPKM1 or IMA_AUTHMETHOD_SPKM2.

Remarks

This structure is used as a parameter to the IMA_SetMutualLocalAuthParms and IMA_GetMutualLocalAuthParms APIs.

5.66 IMA_TARGET_AUTHPARMS_LIST

Format

```
typedef union IMA_target_authparms_list
{
    IMA_UINT                    authparmsCount;
    IMA_TARGET_AUTHPARMS        authParmsList[1];
} IMA_TARGET_AUTHPARMS_LIST;
```

Fields

authparmsCount

The number of IMA_TARGET_AUTHPARMS structures in the authParmsList array.

authParmsList

The target authentication parameters array.

Remarks

This structure is used as a parameter for the IMA_GetLhbaMutualAuthParmsList API.

5.67 IMA_DIGEST_PROPERTIES

Format

```
typedef struct _IMA_DIGEST_PROPERTIES
{
    IMA_BOOL        headerDigestsSettable;
    IMA_BOOL        dataDigestsSettable;

} IMA_DIGEST_PROPERTIES;
```

Fields

headerDigestsSettable

A boolean indicating if header digests are settable.

dataDigestsSettable

A boolean indicating if data digests are settable.

Remarks

This structure is returned by the IMA_GetDigestProperties API.

5.68 IMA_DIGEST_TYPE

Format

```
enum IMA_DIGEST_TYPE
{
    IMA_DIGEST_NONE                    = 0,
    IMA_DIGEST_CRC32C                  = 1
};
```

Fields

IMA_DIGEST_NONE

This indicates that no digest is enabled.

This symbol has the value 0.

IMA_DIGEST_CRC32C

This indicates that 32-bit CRC is enabled.

This symbol has the value 1.

5.69 IMA_SESSION_PROPERTIES

Format

```
typedef struct _IMA_session_properties
{
    IMA_OID             associatedLhbaOid;
    IMA_OID             associatedTargetOid;
    IMA_AUTHMETHOD      authMethod;
    IMA_BOOL            dataPduInOrder;
    IMA_BOOL            dataSequenceInOrder;
    IMA_UINT32          defaultTime2Retain;
    IMA_UINT32          defaultTime2Wait;
    IMA_UINT32          errorRecoveryLevel;
    IMA_UINT32          firstBurstLength;
    IMA_BOOL            immediateData;
    IMA_BOOL            initialR2T;
    IMA_BYTE            isid[6];
    IMA_UINT32          maxBurstLength;
    IMA_UINT32          maxConnections;
    IMA_UINT32          maxOutstandingR2T;
    IMA_UINT16          targetPortalGroupTag;
    IMA_UINT16          tsih;
} IMA_SESSION_PROPERTIES;
```

Fields

associatedLhbaOid
The object ID of the LHBA responsible for this session.

associatedTargetOid
The object ID of the target associated with this session.

authMethod
The authentication method being used for this session.

dataPduInOrder
The negotiated boolean value of the DataPDUInOrder login parameter.

dataSequenceInOrder
The negotiated boolean value of the DataSequenceInOrder login parameter.

defaultTime2Retain
The negotiated numerical value of the DefaultTime2Retain login parameter.

defaultTime2Wait
The negotiated numerical value of the DefaultTime2Wait login parameter.

errorRecoveryLevel
The negotiated numerical value of the ErrorRecoveryLevel login parameter.

firstBurstLength
The negotiated numerical value of the FirstBurstLength login parameter.

immediateData
The negotiated boolean value of the ImmediateData login parameter.

initialR2T
The negotiated boolean value of the InitialR2T login parameter.

isid
The initiator session identifier associated with this session object.

maxBurstLength
The negotiated numerical value of the MaxBurstLength login parameter.

maxConnections
The negotiated numerical value of the MaxConnections login parameter.

maxOutstandingR2T
The negotiated numerical value of the MaxOutstandingR2T login parameter.

targetPortalGroupTag
The target portal group tag value returned from the target.

tsih
The target session identifying handle for this session object.

Remarks

This structure is returned by the IMA_GetSessionProperties API.

5.70 IMA_CONNECTION_PROPERTIES

Format

```
typedef struct _IMA_connection_properties
{
    IMA_OID             associatedSessionOid;
    IMA_UINT16          connectionId;
    IMA_DIGEST_TYPE     dataDigest;
    IMA_DIGEST_TYPE     headerDigest;
    IMA_BOOL            ifMarker;
    IMA_UINT32          ifMarkInt;
    IMA_UINT32          maxRecvDataSegmentLength;
    IMA_UINT32          maxTransmitDataSegmentLength;
    IMA_BOOL            ofMarker;
    IMA_UINT32          ofMarkInt;
} IMA_CONNECTION_PROPERTIES;
```

Fields

associatedSessionOid

The object ID of the session which encompasses this connection.

connectionId

The connection identifier for this connection object.

dataDigest

The negotiated value for the data digest parameter. This field will be set to one of the values in IMA_DIGEST_TYPE.

headerDigest

The negotiated value for the header digest parameter. This field will be set to one of the values in IMA_DIGEST_TYPE.

ifMarker

The negotiated value for the ifMarker parameter.

ifMarkInt

The negotiated value for the ifMarkInt parameter. This value is undefined when ifMarker has the value IMA_FALSE.

maxRecvDataSegmentLength

The declared numerical value of the MaxRecvDataSegmentLength parameter. This is the value declared by the initiator.

maxTransmitDataSegmentLength

The declared numerical value of the MaxRecvDataSegmentLength parameter. This is the value declared by the target.

ofMarker

The negotiated value for the ofMarker parameter.

ofMarkInt

The negotiated value for the ofMarkInt parameter. This value is undefined when ofMarker has the value IMA_FALSE.

Remarks

This structure is returned by the IMA_GetConnectionProperties API.

5.71 IMA_RADIUS_PROPERTIES

Format

```
typedef struct _IMA_radius_properties
{
    IMA_BOOL        radiusEnabled;
    IMA_BOOL        radiusHostAddressValid;
    IMA_HOST_ID     radiusHost;
    IMA_UINT16      radiusPort;
    IMA_BOOL        sharedSecretValid;
    IMA_UINT        sharedSecretLength;
    IMA_BYTE        sharedSecret[128];
} IMA_RADIUS_PROPERTIES;
```

Fields

radiusEnabled

A boolean indicating whether RADIUS access is enabled.

radiusHostAddressValid

A boolean indicating whether the radiusHost and radiusPort fields contain valid values.

radiusHost

The IP address or hostname of the RADIUS server.

radiusPort

The TCP/IP port number of the RADIUS server.

sharedSecretValid

A boolean indicating whether the sharedSecretLength and sharedSecret fields contain valid values. This field will always be IMA_FALSE when this structure is returned in a call to IMA_GetRadiusConfig.

sharedSecretLength

The length of the value in the field sharedSecret.

sharedSecret

The shared secret to be used when contacting the RADIUS server.

Remarks

This structure is used in calls to IMA_SetRadiusConfig and IMA_GetRadiusConfig. If either radiusHostAddressValid or sharedSecretValid are set to IMA_FALSE in a call to IMA_SetRadiusConfig, any previous values that were set for radiusHost, radiusPort, sharedSecretLength or sharedSecret will be left unchanged.

5.72 IMA_MARKER_INT

Format

```
typedef struct_IMA_MARKER_INT
{
    IMA_BOOL        rangeInUse;
    IMA_UINT32      markerIntValue;
    IMA_UINT32      markerIntValueHigh;
} IMA_MARKER_INT;
```

Fields

rangeInUse

A boolean indicating whether a range of marker interval values is used in this structure. If rangeInUse has the value IMA_TRUE markerIntValue contains the low value of the marker interval range while markerIntValueHigh contains the high value of the marker interval range. If rangeInUse has the value IMA_FALSE markerIntValue contains the only value of the marker interval.

markerIntValue

The low value of the marker interval range if rangeInUse has the value IMA_TRUE or the only value if rangeInUse has the value IMA_FALSE.

markerIntValueHigh

The high value of the marker interval range if rangeInUse has the value IMA_TRUE. If rangeInUse has the value IMA_FALSE, the contents of this field is undefined.

Remarks

This structure is used in the IMA_GetIfMarkIntProperties, IMA_GetOfMarkIntProperties, IMA_SetIfMarkIntProperties and IMA_SetOfMarkIntProperties APIs.

5.73 IMA_MARKER_INT_PROPERTIES

Format

```
typedef struct_IMA_MARKER_INT_PROPERTIES
{
    IMA_BOOL            currentValueValid;
    IMA_BOOL            settable;
    IMA_MARKER_INT      currentValue;
    IMA_MARKER_INT      defaultValue;
    IMA_UINT32          minimumValue;
    IMA_UINT32          maximumValue;
} IMA_MARKER_INT_PROPERTIES;
```

Fields

currentValueValid

A boolean indicating if the *currentValue* field contains a valid value.

settable

Indicates if the marker interval property is settable. If this field has the value IMA_TRUE then the *defaultValue*, *minimumValue*, and *maximumValue* fields shall contain valid values. If this field has the value IMA_FALSE then these fields have undefined values.

currentValue

If *currentValueValid* has the value IMA_TRUE then this field contains the current value of the marker interval. If *currentValueValid* has the value IMA_FALSE then the value of this field is undefined.

defaultValue

If *settable* has the value IMA_TRUE then this field contains the implementation's default value of the marker interval. If *settable* has the value IMA_FALSE then the value of this field is undefined.

minimumValue

If *settable* has the value IMA_TRUE then this field contains the implementation's minimum value of the marker interval. This value applies to the low end of the range when a range is used or the only value when a range is not used. If *settable* has the value IMA_FALSE then the value of this field is undefined.

maximumValue

If *settable* has the value IMA_TRUE then this field contains the implementation's maximum value of the marker interval. This value applies to the high end of the range when a range is used or the only value when a range is not used. If *settable* has the value IMA_FALSE then the value of this field is undefined.

Remarks

If the *currentValueValid* field is IMA_FALSE then the value of *settable* shall also be set to IMA_FALSE.

The fields in this structure contain values that are defined by the implementation and not by the iSCSI specification. It is possible that an implementation may be more or less restrictive in the values that it can accept than the iSCSI specification allows.

6 APIs

There are ten groups of APIs in the iSCSI Management API. These groups are:

1. Library and Plugin APIs
2. Node APIs
3. Logical HBA APIs
4. Physical HBA APIs
5. Network Portal APIs
6. Logical Network Port (LNP) APIs
7. Physical Network Port (PNP) APIs
8. Target APIs
9. Logical Unit (LU) APIs
10. Miscellaneous APIs

6.1 APIs by Category

6.1.1 Library and Plugin APIs

There are five APIs that deal with the library and plugins. They are:

1. IMA_GetLibraryProperties
2. IMA_GetPluginOidList
3. IMA_GetPluginProperties
4. IMA_PluginIOCtl
5. IMA_GetAssociatedPluginOid

6.1.2 Node APIs

There are seven node related APIs. They are:

1. IMA_GetSharedNodeOid
2. IMA_GetNonSharedNodeOidList
3. IMA_GetNodeProperties
4. IMA_SetNodeName
5. IMA_GenerateNodeName
6. IMA_SetNodeAlias
7. IMA_GetAssociatedPluginOid

6.1.3 Logical HBA APIs

There are 43 logical HBA related APIs. They are:

1. IMA_GetLhbaOidList
2. IMA_GetLhbaProperties
3. IMA_GetNetworkPortalOidList
4. IMA_GetTargetOidList
5. IMA_GetLuOidList
6. IMA_GetDiscoveryProperties
7. IMA_SetIsnsDiscovery
8. IMA_SetSlpDiscovery
9. IMA_SetStaticDiscovery
10. IMA_AddStaticDiscoveryTarget
11. IMA_RemoveStaticDiscoveryTarget

12. IMA_SetSendTargetsDiscovery
13. IMA_GetIpsecProperties
14. IMA_RemoveStaleData
15. IMA_GetFirstBurstLengthProperties
16. IMA_SetFirstBurstLength
17. IMA_GetMaxBurstLengthProperties
18. IMA_SetMaxBurstLength
19. IMA_GetMaxRecvDataSegmentLengthProperties
20. IMA_SetMaxRecvDataSegmentLength
21. IMA_GetMaxConnectionsProperties
22. IMA_SetMaxConnections
23. IMA_GetDefaultTime2RetainProperties
24. IMA_SetDefaultTime2Retain
25. IMA_GetDefaultTime2WaitProperties
26. IMA_SetDefaultTime2Wait
27. IMA_GetInitialR2TProperties
28. IMA_SetInitialRT2
29. IMA_GetMaxOutstandingRT2Properties
30. IMA_SetMaxOutstandingR2T
31. IMA_GetErrorRecoveryLevelProperties
32. IMA_SetErrorRecoveryLevel
33. IMA_GetImmediateDataProperties
34. IMA_SetImmediateData
35. IMA_GetDataPduInOrderProperties
36. IMA_SetDataPduInOrder
37. IMA_GetDataSequenceInOrderProperties
38. IMA_SetDataSequenceInOrder
39. IMA_GetSupportedAuthMethods
40. IMA_GetInUseInitiatorAuthMethods
41. IMA_SetInitiatorAuthMethods
42. IMA_GetInitiatorAuthParms
43. IMA_SetInitiatorAuthParms

44. IMA_GetSessionOidList
45. IMA_GetDigestProperties
46. IMA_GetDigestValues
47. IMA_SetDataDigestValues
48. IMA_SetHeaderDigestValues
49. IMA_GetIFMarkerProperties
50. IMA_SetIFMarkerProperties
51. IMA_GetOFMarkerProperties
52. IMA_SetOFMarkerProperties
53. IMA_GetIFMarkIntProperties
54. IMA_SetIFMarkIntProperties
55. IMA_GetOFMarkIntProperties
56. IMA_SetOFMarkIntProperties

6.1.4 Physical HBA APIs

There are 15 physical HBA related APIs. They are:

1. IMA_GetPhbaOidList
2. IMA_GetPhbaProperties
3. IMA_GetPhbaStatus
4. IMA_GetDiscoveryProperties
5. IMA_SetIsnsDiscovery
6. IMA_SetSlpDiscovery
7. IMA_SetStaticDiscovery
8. IMA_AddStaticDiscoveryTarget
9. IMA_RemoveStaticDiscoveryTarget
10. IMA_SetSendTargetsDiscovery
11. IMA_GetPnpOidList
12. IMA_GetPhbaDownloadProperties
13. IMA_IsPhbaDownloadFile
14. IMA_PhbaDownload
15. IMA_SetStatisticsCollection

6.1.5 Network Portal APIs

There are currently three network portal related APIs. They are:

1. IMA_GetNetworkPortalOidList
2. IMA_GetNetworkPortalProperties
3. IMA_SetNetworkPortalIpAddress

6.1.6 Logical Network Port (LNP) APIs

There are currently four logical network prot related APIs. They are:

1. IMA_GetLnpOidList
2. IMA_GetPnpOidList
3. IMA_GetLnpProperties
4. IMA_GetNetworkPortStatus

6.1.7 Physical Network Port (PNP) APIs

There are currently four physical network port related APIs. They are:

1. IMA_GetPnpOidList
2. IMA_GetNetworkPortStatus
3. IMA_GetPnpProperties
4. IMA_GetStatisticsProperties
5. IMA_GetPnpStatistics
6. IMA_GetIpProperties
7. IMA_SetIpConfigMethod
8. IMA_SetDefaultGateway
9. IMA_SetDnsServerAddress
10. IMA_SetSubnetMask

6.1.8 Target APIs

There are currently 32 target related APIs. They are:

1. IMA_GetAddressKeys
2. IMA_GetTargetOidList
3. IMA_GetTargetProperties
4. IMA_GetStatisticsProperties

5. IMA_GetTargetErrorStatistics
6. IMA_SetSendTargetsDiscovery
7. IMA_GetLuOidList
8. IMA_GetFirstBurstLengthProperties
9. IMA_SetFirstBurstLength
10. IMA_GetMaxBurstLengthProperties
11. IMA_SetMaxBurstLength
12. IMA_GetMaxRecvDataSegmentLengthPropertie
13. IMA_SetMaxRecvDataSegmentLength
14. IMA_GetMaxConnectionsProperties
15. IMA_SetMaxConnections
16. IMA_GetDefaultTime2RetainProperties
17. IMA_SetDefaultTime2Retain
18. IMA_GetDefaultTime2WaitProperties
19. IMA_SetDefaultTime2Wait
20. IMA_GetInitialR2TProperties
21. IMA_SetInitialRT2
22. IMA_GetMaxOutstandingRT2Properties
23. IMA_SetMaxOutstandingR2T
24. IMA_GetErrorRecoveryLevelProperties
25. IMA_SetErrorRecoveryLevel
26. IMA_GetImmediateDataProperties
27. IMA_SetImmediateData
28. IMA_GetDataPduInOrderProperties
29. IMA_SetDataPduInOrder
30. IMA_GetDataSequenceInOrderProperties
31. IMA_SetDataSequenceInOrder
32. IMA_GetDeviceStatistics
33. IMA_SetStatisticsCollection
34. IMA_GetSessionOidList
35. IMA_GetDigestProperties
36. IMA_GetDigestValues

37. IMA_SetDataDigestValues
38. IMA_SetHeaderDigestValues
39. IMA_GetIFMarkerProperties
40. IMA_SetIFMarkerProperties
41. IMA_GetOFMarkerProperties
42. IMA_SetOFMarkerProperties
43. IMA_GetIFMarkIntProperties
44. IMA_SetIFMarkIntProperties
45. IMA_GetOFMarkIntProperties
46. IMA_SetOFMarkIntProperties
47. IMA_GetInitiatorLocalAuthParms
48. IMA_SetInitiatorLocalAuthParms
49. IMA_GetMutualLocalAuthParms
50. IMA_SetMutualLocalAuthParms

6.1.9 Logical Unit (LU) APIs

There are currently ten logical unit related APIs. They are:

1. IMA_GetLuOid
2. IMA_GetLuOidList
3. IMA_GetLuProperties
4. IMA_LuInquiry
5. IMA_LuReadCapacity
6. IMA_LuReportLuns
7. IMA_ExposeLu
8. IMA_UnexposeLu
9. IMA_GetStatisticsProperties
10. IMA_GetDeviceStatistics
11. IMA_SetStatisticsCollection

6.1.10 Miscellaneous APIs

There are six miscellaneous APIs. They are:

1. IMA_GetObjectType
2. IMA_FreeMemory

3. IMA_RegisterForObjectVisibilityChanges
4. IMA_DeregisterForObjectVisibilityChanges
5. IMA_RegisterForObjectPropertyChanges
6. IMA_DeregisterForObjectPropertyChanges

6.2 APIs by Name

There are 98 APIs in the iSCSI Management API. They are:

1. IMA_AddDiscoveryAddress
2. IMA_AddLhbaMutualAuthParms
3. IMA_AddStaticDiscoveryTarget
4. IMA_DeregisterForObjectPropertyChanges
5. IMA_DeregisterForObjectVisibilityChanges
6. IMA_ExposeLu
7. IMA_FreeMemory
8. IMA_GenerateNodeName
9. IMA_GetAddressKeyProperties
10. IMA_GetAssociatedPluginOid
11. IMA_GetConnectionOidList
12. IMA_GetConnectionProperties
13. IMA_GetDigestProperties
14. IMA_GetDigestValues
15. IMA_GetDataPduInOrderProperties
16. IMA_GetDataSequenceInOrderProperties
17. IMA_GetDefaultTime2RetainProperties
18. IMA_GetDefaultTime2WaitProperties
19. IMA_GetDeviceStatistics
20. IMA_GetDiscoveryAddressOidList
21. IMA_GetDiscoveryAddressProperties
22. IMA_GetDiscoveryProperties
23. IMA_GetErrorRecoveryLevelProperties
24. IMA_GetFirstBurstLengthProperties
25. IMA_GetIFMarkerProperties
26. IMA_GetIFMarkIntProperties
27. IMA_GetImmediateDataProperties
28. IMA_GetInitialR2TProperties
29. IMA_GetInitiatorAuthParms
30. IMA_GetInitiatorLocalAuthParms
31. IMA_GetInUseInitiatorAuthMethods
32. IMA_GetIpProperties
33. IMA_GetIpsecProperties
34. IMA_GetLhbaMutualAuthParmsList
35. IMA_GetLhbaOidList
36. IMA_GetLhbaProperties
37. IMA_GetLibraryProperties
38. IMA_GetLnpOidList
39. IMA_GetLnpProperties
40. IMA_GetLuOid
41. IMA_GetLuOidList
42. IMA_GetLuProperties
43. IMA_GetMaxBurstLengthProperties
44. IMA_GetMaxConnectionsProperties
45. IMA_GetMaxOutstandingRT2Properties
46. IMA_GetMaxRecvDataSegmentLengthProperties
47. IMA_GetMutualLocalAuth
48. IMA_GetMutualLocalAuthParms
49. IMA_GetNetworkPortalOidList
50. IMA_GetNetworkPortalProperties
51. IMA_GetNetworkPortStatus
52. IMA_GetNodeProperties
53. IMA_GetNonSharedNodeOidList
54. IMA_GetOFMarkerProperties
55. IMA_GetOFMarkIntProperties
56. IMA_GetObjectType
57. IMA_GetPhbaDownloadProperties
58. IMA_GetPhbaOidList

59. IMA_GetPhbaProperties
60. IMA_GetPhbaStatus
61. IMA_GetPluginOidList
62. IMA_GetPluginProperties
63. IMA_GetPnpOidList
64. IMA_GetPnpProperties
65. IMA_GetPnpStatistics
66. IMA_GetRadiusAccess
67. IMA_GetSessionOidList
68. IMA_GetSessionProperties
69. IMA_GetSharedNodeOid
70. IMA_GetStaticDiscoveryTargetOidList
71. IMA_GetStaticDiscoveryTargetProperties
72. IMA_GetStatisticsProperties
73. IMA_GetSupportedAuthMethods
74. IMA_GetTargetErrorStatistics
75. IMA_GetTargetOidList
76. IMA_GetTargetProperties
77. IMA_IsPhbaDownloadFile
78. IMA_LuInquiry
79. IMA_LuReadCapacity
80. IMA_LuReportLuns
81. IMA_PersistHbaParameters
82. IMA_PhbaDownload
83. IMA_PluginIOCtl
84. IMA_RegisterForObjectPropertyChanges
85. IMA_RegisterForObjectVisibilityChanges
86. IMA_RemoveDiscoveryAddress
87. IMA_RemoveLhbaMutualAuthParms
88. IMA_RemoveStaleData
89. IMA_RemoveStaticDiscoveryTarget
90. IMA_SetDataDigestValues
91. IMA_SetDataPduInOrder
92. IMA_SetDataSequenceInOrder
93. IMA_SetDefaultGateway
94. IMA_SetDefaultTime2Retain
95. IMA_SetDefaultTime2Wait
96. IMA_SetDnsServerAddress
97. IMA_SetErrorRecoveryLevel
98. IMA_SetFirstBurstLength
99. IMA_SetHeaderDigestValues
100. IMA_SetIFMarkerProperties
101. IMA_SetIFMarkIntProperties
102. IMA_SetImmediateData
103. IMA_SetInitialRT2
104. IMA_SetInitiatorAuthMethods
105. IMA_SetInitiatorAuthParms
106. IMA_SetInitiatorLocalAuthParms
107. IMA_SetIpConfigMethod
108. IMA_SetIsnsDiscovery
109. IMA_SetMaxBurstLength
110. IMA_SetMaxConnections
111. IMA_SetMaxOutstandingR2T
112. IMA_SetMaxRecvDataSegmentLength
113. IMA_SetMutualLocalAuth
114. IMA_SetMutualLocalAuthParms
115. IMA_SetNetworkPortalIpAddress
116. IMA_SetNodeAlias
117. IMA_SetNodeName
118. IMA_SetOFMarkerProperties
119. IMA_SetOFMarkIntProperties
120. IMA_SetRadiusAccess
121. IMA_SetSendTargetsDiscovery
122. IMA_SetSlpDiscovery

123. IMA_SetStaticDiscovery
124. IMA_SetStatisticsCollection
125. IMA_SetSubnetMask
126. IMA_UnexposeLu

6.2.1 IMA_AddDiscoveryAddress

Synopsis

Adds a discovery address to be used for send targets discovery by the specified physical network port or logical HBA.

Prototype

```
IMA_STATUS IMA_AddDiscoveryAddress(
    /* in */      IMA_OID oid,
    /* in */      const IMA_TARGET_ADDRESS discoveryAddress,
    /* out */     IMA_OID *pDiscoveryAddressOid
);
```

Parameters

oid

The object ID of the PNP or LHBA to which the discovery address is being added.

discoveryAddress

The target address of the target to add to the specified PNP's or LHBA's list of discovery addresses that are to be used in a send targets discovery session.

pDiscoveryAddressOid

A pointer to a IMA_OID structure allocated by the caller or NULL. If not NULL, on successful return it will contain the OID of the discovery address added by this API.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the discovery address is used by the PNP or the LHBA in a send targets discovery session.

IMA_ERROR_NOT_SUPPORTED

Returned if send targets discovery is not supported by the specified PNP or LHBA.

IMA_ERROR_INVALID_PARAMETER

Returned if *discoveryAddress* is NULL or specifies a memory area from which data cannot be read. Also, returned if *discoveryAddress* specifies an empty structure.

Returned if *pDiscoveryAddressOid* is not NULL and specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PNP or LHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PNP or LHBA that is currently known to the system.

Remarks

The discovery address is persistent, i.e., it will continue to be used until removed by the IMA_RemoveDiscoveryAddress API.

It is not an error to specify a discovery address that is already being used.

This call does not verify that the specified discovery address exists or that the PNP or LHBA has sufficient rights to login into the specified discovery address.

If the return value from this API is IMA_SUCCESS then an attempt to use the discovery address in a send targets discovery session will be made by the iSCSI stack as soon as possible, possibly before the API returns control to the caller. If a client wishes to know when the target(s) represented by the discovery address are discovered it should call the IMA_RegisterForObjectVisibilityChanges API before calling this API to register a notification function.

If the return value from this API is IMA_STATUS_REBOOT_NECESSARY then an attempt to use the discovery address will be made by the iSCSI stack on reboot. This discovery is guaranteed to have been attempted by the time the top of the iSCSI stack finishes initializing. This may or may not occur prior to the time an IMA client can execute.

Support

Mandatory if the *sendTargetsDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by IMA_GetDiscoveryProperties is true for the same *oid*.

See Also

IMA_RemoveDiscoveryAddress

IMA_GetDiscoveryAddressProperties

IMA_RegisterForObjectVisibilityChanges

IMA_GetDiscoveryProperties

6.2.2 IMA_AddLhbaMutualAuthParms

Synopsis

Adds a mutual auth parm to the list for a specified lhba.

Prototype

```
IMA_STATUS IMA_AddLhbaMutualAuthParms(
    /* in */    IMA_OID oid,
    /* in */    IMA_AUTHMETHOD method,
    /* in */    IMA_TARGET_AUTHPARMS pParms
);
```

Parameters

oid

The object ID of an LHBA to which the authentication parameters will be added.

method

The authentication method of the object ID whose authentication parameters are to be added.

pParms

An IMA_TARGET_AUTHPARMS structure which contains the new auth parms to add to the list for the specified LHBA.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA.

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting of the mutual authentication parameters list.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *o*id does not specify an LHBA that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory if the *mutualLhbaAuthParmsListSupported* field of the IMA_LHBA_PROPERTIES returned by the IMA_GetLhbaProperties API true for the LHBA associated with this target.

See Also

IMA_GetLhbaMutualAuthParmsList

IMA_RemoveLhbaMutualAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.3 IMA_AddStaticDiscoveryTarget

Synopsis

Adds a target to be statically discovered by the specified physical network port or logical HBA.

Prototype

```
IMA_STATUS IMA_AddStaticDiscoveryTarget(
    /* in */      IMA_OID oid,
    /* in */      const IMA_STATIC_DISCOVERY_TARGET staticDiscoveryTarget,
    /* out */     IMA_OID *pStaticDiscoveryTargetOid
);
```

Parameters

oid

The object ID of the PNP or LHBA to which the target to be discovered is being added.

staticDiscoveryTarget

The name and target address of the target to add to the specified PNP's or LHBA's list of targets that are to be statically discovered.

pStaticDiscoveryTargetOid

A pointer to a IMA_OID structure allocated by the caller or NULL. If not NULL, on successful return it will contain the OID of the static discovery target added by this API.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before discovery of the specified target takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if static target discovery is not supported by the specified PNP or LHBA.

IMA_ERROR_INVALID_PARAMETER

Returned if *staticDiscoveryTarget* is NULL or specifies a memory area from which data cannot be read. Also, returned if *staticDiscoveryTarget* specifies an empty structure.

Returned if *pStaticDiscoveryTargetOid* is not NULL and specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PNP or LHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PNP or LHBA that is currently known to the system.

Remarks

The discovery of the target is persistent, i.e., it will continue to be “discovered” until it is removed using IMA_RemoveStaticDiscoveryTarget.

It is not an error to specify a target that is already being discovered, either statically or by other methods.

This call does not verify that the specified target exists or that the PNP or LHBA has sufficient rights to login into the specified target.

If the return value from this API is IMA_SUCCESS then an attempt to discover the specified target(s) will be made by the iSCSI stack as soon as possible, possibly before the API returns control to the caller. If a client wishes to know when the target(s) is/are actually discovered it should call the IMA_RegisterForObjectVisibilityChanges API before calling this API to register a notification function.

If the return value from this API is IMA_STATUS_REBOOT_NECESSARY then an attempt to discover the specified target(s) will be made by the iSCSI stack on reboot. This discovery is guaranteed to have been attempted by the time the top of the iSCSI stack finishes initializing. This may or may not occur prior to the time an IMA client can execute.

When a client is finished using the returned list it shall free the memory used by the returned list by calling IMA_FreeMemory.

Support

Mandatory if the *staticDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by IMA_GetDiscoveryProperties is true for the same *oid*.

See Also

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetProperties

IMA_RegisterForObjectVisibilityChanges

IMA_GetDiscoveryProperties

IMA_SetStaticDiscovery

6.2.4 IMA_DeregisterForObjectPropertyChanges

Synopsis

Deregisters a client function to be called whenever an object's property changes.

Prototype

```
IMA_STATUS IMA_DeregisterForObjectPropertyChanges (
    /* in */        IMA_OBJECT_PROPERTY_FN pClientFn
);
```

Parameters

pClientFn

A pointer to an IMA_OBJECT_PROPERTY_FN function defined by the client that was previously registered using the IMA_RegisterForObjectPropertyChanges API. On successful return this function will no longer be called to inform the client of object property changes.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pClientFn* is NULL or specifies a memory area that is not executable.

Support

Mandatory

Remarks

The function specified by *pClientFn* will no longer be called whenever an object's property changes.

It is not an error to unregister a client function that is not registered.

See Also

IMA_RegisterForObjectPropertyChanges

6.2.5 IMA_DeregisterForObjectVisibilityChanges

Synopsis

Deregisters a client function to be called whenever a high level object appears or disappears.

Prototype

```
IMA_STATUS IMA_DeregisterForObjectVisibilityChanges (
    /* in */        IMA_OBJECT_VISIBILITY_FN pClientFn
);
```

Parameters

pClientFn

A pointer to an IMA_OBJECT_VISIBILITY_FN function defined by the client that was previously registered using the IMA_RegisterForObjectVisibilityChanges API. On successful return this function will no longer be called to inform the client of object visibility changes.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pClientFn* is NULL or specifies a memory area that is not executable.

Support

Mandatory

Remarks

The function specified by *pClientFn* will be no longer be called whenever high level objects appear or disappear.

It is not an error to unregister a client function that is not registered.

See Also

IMA_RegisterForObjectVisibilityChanges

6.2.6 IMA_ExposeLu

Synopsis

Exposes the specified logical unit to the operating system.

Prototype

```
IMA_STATUS IMA_ExposeLu(
    /* in */        IMA_OID luOid
);
```

Parameters

luOid

The object ID of the LU to expose to the operating system.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the LU is exposed to the OS.

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA associated with the LU does not support selective exposing/exposing of logical units.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *luOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *luOid* does not specify a LU object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *luOid* does not specify a LU that is currently known to the system.

IMA_ERROR_LU_EXPOSED

Returned if *luOid* specifies a LU that is currently exposed to the operating system.

IMA_ERROR_TARGET_TIMEOUT

Returned if the target associated with the specified LU failed to respond to an iSCSI login request from an initiator.

IMA_ERROR_LOGIN_REJECTED

Returned if the target associated with the specified LU rejected an iSCSI login request from an initiator.

Remarks

The exposing of the LU is persistent, i.e. the LU will continue to be exposed to the OS whenever the associated device driver loads, until a successful call to IMA_UnexposeLu for the same LU.

If a session has not already been created with the target associated with the logical unit then an iSCSI login will be performed with the target to ensure that the initiator has permission to access the target. If a session already exists between the LHBA

and the target then another session may or may not be created, at the discretion of the LHBA software and firmware.

Support

Mandatory if the *luExposingSupported* field of the IMA_LHBA_PROPERTIES structure returned by the IMA_GetLhbaProperties API for the LHBA that is associated with the LU has the value IMA_TRUE.

See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_UnexposeLu

IMA_GetLuId

IMA_GetLuOidList

6.2.7 IMA_FreeMemory

Synopsis

Frees memory returned by an IMA API.

Prototype

```
IMA_STATUS IMA_FreeMemory(
    /* in */        void *pMemory
);
```

Parameters

pMemory

A pointer to memory returned by an IMA API, such as IMA_OID_LIST structure returned by the library. If the specified pointer is NULL then the function succeeds, but takes no action. On successful return if the specified pointer is non-NULL the allocated memory is freed.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pMemory* is non-NULL and specifies a memory area to which data cannot be written.

Support

Mandatory

Remarks

Clients shall free all IMA_OID_LIST structures returned by any API by using this function.

See Also

IMA_GetLhbaOidList

IMA_GetLnpOidList

IMA_GetLuOidList

IMA_GetNonSharedNodeOidList

IMA_GetPhbaOidList

IMA_GetPnpOidList

IMA_GetTargetOidList

6.2.8 IMA_GenerateNodeName

Synopsis

Generates a ‘unique’ node name for the currently running system.

Prototype

```
IMA_STATUS IMA_GenerateNodeName(
    /* out */       IMA_NODE_NAME generatedName
);
```

Parameters

generatedName

On successful return contains the generated node name.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *generatedName* is NULL or specifies a memory area to that data cannot be written.

Remarks

This API only generates a single name for a system, therefore it shall only be used to generate the shared node name; it shall not be used to generate nonshared node names.

This API call only generates a node name, it does not set the node name. To do that call the IMA_SetNodeName API.

The name generated by this API is based on the name of the computer that the client is running on. This API does not search the network to ensure that the generated name is not already in use. If the client chooses to use the node name generated by this API the client should ensure that the node name is not already being used by another node.

Support

Mandatory

See Also

IMA_GetSharedNodeOid

IMA_GetNonSharedNodeOidList

IMA_SetNodeName

Example of Setting a Node Name

6.2.9 IMA_GetAddressKeys

Synopsis

Gets the address keys of a target.

Prototype

```
IMA_STATUS IMA_GetAddressKeys(
    /* in */     IMA_OID targetOid,
    /* out */    IMA_ADDRESS_KEYS **ppKeys
);
```

Parameters

targetOid

The object ID of the target whose address keys are being retrieved.

ppKeys

A pointer to a pointer to an IMA_ADDRESS_KEYSstructure allocated by the caller. On successful return this will contain a pointer to an IMA_ADDRESS_KEYS structure.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppKeys* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *targetOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *targetOid* does not specify a target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *targetOid* does not specify a target that is currently known to the system.

IMA_ERROR_TARGET_TIMEOUT

Returned if the target failed to respond to an iSCSI discovery session creation from an initiator.

Remarks

The returned list of address keys is guaranteed to contain no duplicate values.

When a client is finished using the returned keys structure it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

6.2.10 IMA_GetAssociatedPluginOid

Synopsis

Gets the object ID for the plugin associated with the specified object ID.

Prototype

```
IMA_STATUS IMA_GetAssociatedPluginOid(
    /* in */        IMA_OID oid
    /* out */       IMA_OID *pPluginOid
);
```

Parameters

oid

The object ID of an object that has been received from a previous API call.

pPluginOid

A pointer to an IMA_OID structure allocated by the caller. On successful return this will contain the object ID of the plugin associated with the object specified by *oid*. This can then be used to work with the plugin, e.g., to get the properties of the plugin or the send the plugin an IOCtl.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *oid* specifies an object not owned by a plugin, but instead one that is owned by the library.

Returned if *pPluginOid* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* specifies an object with an invalid type.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a plugin that is currently known to the system.

IMA_ERROR_PLUGINS_NOT_SUPPORTED

Returned if the library implementation does not support plugins.

Remarks

None

Support

Mandatory

See Also

Example of Getting an Associated Plugin ID

IMA_PluginIOCtl

6.2.11 IMA_GetConnectionOidList

Synopsis

Gets a list of the object IDs of connections associated with the specified session.

Prototype

```
IMA_STATUS IMA_GetConnectionOidList(
    /* in */      IMA_OID sessionOid,
    /* out */     IMA_OID_LIST **ppList
);
```

Parameters

sessionOid

The object ID of the session object for which to retrieve the managed connections.

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the connections associated with the specified session object.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a session object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an object that is currently known to the system.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

IMA_GetConnectionProperties

6.2.12 IMA_GetConnectionProperties

Synopsis

Gets the properties of the specified connection.

Prototype

```
IMA_STATUS IMA_GetConnectionProperties(
    /* in */     IMA_OID connectionOid,
    /* out */    IMA_CONNECTION_PROPERTIES *pProps
);
```

Parameters

connectionOid

The object ID of the connection whose properties are being retrieved.

pProps

A pointer to an IMA_CONNECTION_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the connection specified by *connectionOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *connectionOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *connectionOid* does not specify a connection.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *connectionOid* does not specify a connection that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

6.2.13 IMA_GetDataDigestValues

Synopsis

Gets the list of checksums that can be negotiated for data digests.

Prototype

```
IMA_STATUS IMA_GetDataDigestValues(
    /* in */        IMA_OID oid,
    /* in,out */    IMA_UINT *pDigestValueCount,
    /* out */       IMA_DIGEST_TYPE *pDigestValueList;
);
```

Parameters

oid

The object ID of an LHBA or target whose digest values are to be retrieved.

pDigestValueCount

A pointer to an IMA_UINT allocated by the caller. On entry the pointed to value shall contain the maximum number of entries that can be placed into *pDigestValueList*. On return it will contain the number of entries that could be placed into *pDigestValueList*.

pDigestValueList

A pointer to an array of IMA_DIGEST_TYPEs allocated by the caller. This value may be NULL. If this value is not NULL on successful return the array will be filled in with the digest values currently being used by the LHBA or target. These entries will be sorted in decreasing order of preference for use by the LHBA or target.

If this value is NULL then the value pointed to by *pDigestValueCount* on entry shall be zero.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pDigestValueList* specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA or target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an LHBA or target that is currently known to the system.

Remarks

This API is the counterpart of the IMA_SetDataDigestValues API. The list of data digest values returned by this API is the same list, in the same order, as those specified to the last successful call to IMA_SetDataDigestValues. If no

successful call to IMA_SetDataDigestValues has been made then the data digest values in the returned list and their order in the list is vendor specific.

A successful call to this API will always return at least one data digest value.

The returned list is guaranteed to contain no duplicate values.

The number of data digest values returned in *pDigestValueList* will be the minimum of the value in **pDigestValueCount* on entry to the API and the value of **pDigestValueCount* on succesful return from the API.

If the call fails and the value of *pDigestValueList* is not NULL then the data pointed to by *pDigestValueList* will be unchanged.

Support

Mandatory

6.2.14 IMA_GetDigestProperties

Synopsis

Gets the digest properties of an LHBA or target.

Prototype

```
IMA_STATUS IMA_GetDataDigestProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_DIGEST_PROPERTIES *pProps;
);
```

Parameters

oid

The object ID of an LHBA or target whose digest properties are to be retrieved.

pProps

A pointer to an IMA_DIGEST_PROPERTIES structure allocated by the caller. On successful return this structure will contain the digest properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

6.2.15 IMA_GetDataPduInOrderProperties

Synopsis

Gets the `DataPDUInOrder` iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetDataPduInOrderProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_BOOL_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `DataPDUInOrder` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `DataPDUInOrder` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetDataPduInOrder

iSCSI Session and Connection Parameters

6.2.16 IMA_GetDataSequenceInOrderProperties

Synopsis

Gets the `DataSequenceInOrder` iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetDataSequenceInOrderProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_BOOL_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `DataSequenceInOrder` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `DataSequenceInOrder` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetDataSequenceInOrder

iSCSI Session and Connection Parameters

6.2.17 IMA_GetDefaultTime2RetainProperties

Synopsis

Gets the `DefaultTime2Retain` iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetDefaultTime2RetainProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `DefaultTime2Retain` properties are to be retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `DefaultTime2Retain` properties of this LHBA.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetDefaultTime2Retain

iSCSI Session and Connection Parameters

6.2.18 IMA_GetDefaultTime2WaitProperties

Synopsis

Gets the `DefaultTime2Wait` iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetDefaultTime2WaitProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `DefaultTime2Wait` properties are to be retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `DefaultTime2Wait` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetDefaultTime2Wait

iSCSI Session and Connection Parameters

6.2.19 IMA_GetDeviceStatistics

Synopsis

Gets the statistics of the specified target or logical unit which is exposed to the operating system.

Prototype

```
IMA_STATUS IMA_GetDeviceStatistics(
    /* in */        IMA_OID oid,
    /* out */       IMA_DEVICE_STATISTICS *pStats
);
```

Parameters

oid

The object ID of the target or LU whose statistics are being retrieved.

pStats

A pointer to an IMA_DEVICE_STATISTICS structure allocated by the caller. On successful return it will contain the statistics of the specified target or LU.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pStats* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LU object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LU that is currently known to the system.

IMA_ERROR_LU_NOT_EXPOSED

Returned if oid specifies a target that has not no LUs exposed to the operating system.

Returned if *oid* specifies a LU that is not exposed to the operating system.

IMA_ERROR_STATS_COLLECTION_NOT_ENABLED

Returned if statistics collection is not enabled for *oid*.

Remarks

If *oid* specifies a target then the statistics for the target are returned. At least one LU shall be exposed to the operating system for the target to have statistics.

If *oid* specifies a LU then the statistics for the LU are returned. The LU shall be exposed to the operating system for the LU to have statistics.

Support

Mandatory if the *statisticsCollectionEnabled* field of the IMA_STATISTICS_PROPERTIES structure returned by the IMA_GetStatisticsProperties API has the value IMA_TRUE.

See Also

IMA_STATISTICS_PROPERTIES

IMA_GetLuOidList

IMA_SetStatisticsCollection

6.2.20 IMA_GetDiscoveryAddressOidList

Synopsis

Gets a list of the object IDs of all the discovery addresses associated with the specified logical HBA or physical network port.

Prototype

```
IMA_STATUS IMA_GetDiscoveryAddressOidList(
    /* in */      IMA_OID oid,
    /* out */     IMA_OID_LIST **ppList
);
```

Parameters

oid

The object ID of the logical HBA object or a physical network port object for which to retrieve the known discovery addresses.

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the discovery addresses associated with the specified object.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_NOT_SUPPORTED

Returned if send targets discovery is not supported by the specified PNP or LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or a PNP object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an object that is currently known to the system.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory if the *sendTargetsDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by IMA_GetDiscoveryProperties is true for the same *oid*.

See Also

IMA_FreeMemory

IMA_AddDiscoveryAddress

IMA_RemoveDiscoveryAddress

IMA_GetDiscoveryAddressProperties

6.2.21 IMA_GetDiscoveryAddressProperties

Synopsis

Gets the properties of the specified discovery address.

Prototype

```
IMA_STATUS IMA_GetDiscoveryAddressProperties(
    /* in */     IMA_OID discoveryAddressOid,
    /* out */    IMA_DISCOVERY_ADDRESS_PROPERTIES *pProps
);
```

Parameters

discoveryAddressTargetOid

The object ID of the discovery address whose properties are being retrieved.

pProps

A pointer to an IMA_DISCOVERY_ADDRESS_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the discovery address specified by *discoveryAddressOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *discoveryAddressOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *discoveryAddressOid* does not specify a discovery address.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *discoveryAddressOid* does not specify a discovery address that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

IMA_AddStaticDiscoveryTarget

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetOidList

6.2.22 IMA_GetDiscoveryProperties

Synopsis

Gets the discovery properties of the specified physical HBA, logical HBA, or network portal.

Prototype

```
IMA_STATUS IMA_GetDiscoveryProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_DISCOVERY_PROPERTIES *pProps
);
```

Parameters

oid

The object ID of the PHBA, LHBA, or network portal whose discovery properties are being retrieved.

pProps

A pointer to an IMA_DISCOVERY_PROPERTIES structure allocated by the caller. On successful return it will contain the discovery properties of the specified PHBA,LHBA, or network portal.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA, LHBA, or network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA, LHBA, or network portal that is currently known to the system.

Remarks

None

Support

Mandatory for PHBAs.

Optional for LHBAs.

See Also

IMA_GetPhbaOidList

IMA_GetLhbaOidList

IMA_SetIsnsDiscovery

IMA_SetSlpDiscovery

6.2.23 IMA_GetErrorRecoveryLevelProperties

Synopsis

Gets the `ErrorRecoveryLevel` iSCSI login properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetErrorRecoveryLevelProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `ErrorRecoveryLevel` properties are to be retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `ErrorRecoveryLevel` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetErrorRecoveryLevel

iSCSI Session and Connection Parameters

6.2.24 IMA_GetFirstBurstLengthProperties

Synopsis

Gets the `FirstBurstLength` iSCSI login parameter properties of the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetFirstBurstLengthProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `FirstBurstLength` properties are to be retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `FirstBurstLength` properties of the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_MIN_MAX_VALUE

IMA_SetFirstBurstLength

6.2.25 IMA_GetHeaderDigestValues

Synopsis

Gets the list of checksums that can be negotiated for digests.

Prototype

```
IMA_STATUS IMA_GetHeaderDigestValues(
    /* in */        IMA_OID oid,
    /* in,out */    IMA_UINT *pDigestValueCount,
    /* out */       IMA_DIGEST_TYPE *pDigestValueList;
);
```

Parameters

oid

The object ID of an LHBA or target whose digest values are to be retrieved.

pDigestValueCount

A pointer to an IMA_UINT allocated by the caller. On entry the pointed to value shall contain the maximum number of entries that can be placed into *pDigestValueList*. On return it will contain the number of entries that could be placed into *pDigestValueList*.

pDigestValueList

A pointer to an array of IMA_DIGEST_TYPEs allocated by the caller. This value may be NULL. If this value is not NULL on successful return the array will be filled in with the digest values currently being used by the LHBA or target. These entries will be sorted in decreasing order of preference for use by the LHBA or target.

If this value is NULL then the value pointed to by *pDigestValueCount* on entry shall be zero.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pDigestValueList* specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA or target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an LHBA or target that is currently known to the system.

Remarks

This API is the counterpart of the IMA_SetHeaderDigestValues API. The list of header digest values returned by this API is the same list, in the same order, as those specified to the last successful call to IMA_SetHeaderDigestValues. If no

successful call to IMA_SetHeaderDigestValues has been made then the header digest values in the returned list and their order in the list is vendor specific.

A successful call to this API will always return at least one header digest value.

The returned list is guaranteed to contain no duplicate values.

The number of header digest values returned in *pDigestValueList* will be the minimum of the value in **pDigestValueCount* on entry to the API and the value of **pDigestValueCount* on succesful return from the API.

If the call fails and the value of *pDigestValueList* is not NULL then the data pointed to by *pDigestValueList* will be unchanged.

Support

Mandatory

6.2.26 IMA_GetIFMarkerProperties

Synopsis

Gets the `IFMarker` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetIFMarkerProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_BOOL_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `IFMarker` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `IFMarker` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

6.2.27 IMA_GetIFMarkIntProperties

Synopsis

Gets the `IFMarkInt` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetIFMarkInt Properties(
    /* in */      IMA_OID oid,
    /* out */     IMA_MARKER_INT_PROPERTIES *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `IFMarkInt` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `IFMarkInt` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

6.2.28 IMA_GetImmediateDataProperties

Synopsis

Gets the `ImmediateData` iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetImmediateDataProperties(
    /* in */        IMA_OID oid,
    /* out */       IMA_BOOL_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `ImmediateData` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `ImmediateData` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetImmediateData

iSCSI Session and Connection Parameters

6.2.29 IMA_GetInitialR2TProperties

Synopsis

Gets the `InitialR2T` iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetInitialR2TProperties(
    /* in */        IMA_OID oid,
    /* out */       IMA_BOOL_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `InitialR2T` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `InitialR2T` properties of the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetInitialRT2

iSCSI Session and Connection Parameters

6.2.30 IMA_GetInitiatorAuthParms

Synopsis

Gets the parameters for the specified authentication method on the specified LHBA.

Prototype

```
IMA_STATUS IMA_GetInitiatorAuthParms(
    /* in */        IMA_OID lhbaOid,
    /* in */        IMA_AUTHMETHOD method,
    /* out */       IMA_INITIATOR_AUTHPARMS *pParms
);
```

Parameters

lhbaOid

The object ID of the LHBA whose authentication parameters are to be retrieved.

method

The authentication method of the LHBA whose authentication parameters are to be retrieved.

pParms

A pointer to an IMA_INITIATOR_AUTHPARMS structure. On successful return this will contain the initiator authentication parameters for the specified authentication method on the specified LHBA.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the getting of authentication parameters is not supported by the specified LHBA. In this case, it is likely that LHBA does not support any authentication methods.

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lhbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lhbaOid* does not specify an LHBA.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lhbaOid* does not specify an LHBA that is currently known to the system.

Remarks

None.

Support

Optional

See Also

IMA_SetInitiatorAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.31 IMA_GetInitiatorLocalAuthParms

Synopsis

Gets the parameters for the specified authentication method for the specified object ID to be used for one-way authentication.

Prototype

```
IMA_STATUS IMA_GetInitiatorLocalAuthParms(
    /* in */    IMA_OID oid,
    /* in */    IMA_AUTHMETHOD method,
    /* out */   IMA_INITIATOR_AUTHPARMS *pParms,
    /* out */   IMA_BOOL *enabled
);
```

Parameters

oid

The object ID of the target, static target or discovery address whose authentication parameters are to be retrieved.

method

The authentication method of the object ID whose authentication parameters are to be retrieved.

pParms

A pointer to an IMA_INITIATOR_AUTHPARMS structure. On successful return this will contain the initiator authentication parameters for the specified authentication method on the specified target.

enabled

A pointer to a boolean indicating whether use of the local initiator authentication parameters is enabled or disabled. If this parameter is set to IMA_TRUE the authentication parameters set for this object ID will be used for initiator authentication.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a target that is currently known to the system.

Remarks

None

Support

Mandatory if the *initiatorLocalAuthParmsSupported* field in the IMA_LHBA_PROPERTIES structure returned by IMA_GetLhbaProperties is true for the LHBA associated with this target oid.

See Also

IMA_SetInitiatorLocalAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.32 IMA_GetInUseInitiatorAuthMethods

Synopsis

Gets the authentication methods currently in use by the specified logical HBA.

Prototype

```
IMA_STATUS IMA_GetInUseInitiatorAuthMethods(
    /* in */        IMA_OID    lhbaOid,
    /* in, out */   IMA_UINT   *pMethodCount,
    /* out */       IMA_AUTHMETHOD *pMethodList;
);
```

Parameters

lhbaOid

The object ID of an LHBA whose authentication methods are to be retrieved.

pMethodCount

A pointer to an IMA_UINT allocated by the caller. On entry the pointed to value shall contain the maximum number of entries that can be placed into *pMethodList*. On return it will contain the number of entries that could be placed into *pMethodList*.

pMethodList

A pointer to an array of IMA_AUTHMETHODs allocated by the caller. This value may be NULL. If this value is not NULL on successful return the array will be filled in with the authentication methods currently being used by the LHBA. These entries will be sorted in decreasing order of preference for use by the LHBA.

If this value is NULL then the value pointed to by *pMethodCount* on entry shall be zero.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pMethodList* specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lhbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lhbaOid* does not specify an LHBA.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lhbaOid* does not specify an LHBA that is currently known to the system.

Remarks

This API is the counterpart of the IMA_SetInitiatorAuthMethods API. The list of authentication methods returned by this API is the same list, in the same order, as those specified to the last successful call to IMA_SetInitiatorAuthMethods. If no

successful call to IMA_SetInitiatorAuthMethods has been made then the authentication methods in the returned list and their order in the list is vendor specific.

A successful call to this API will always return at least one authentication method.

The returned list is guaranteed to contain no duplicate values.

The number of authentication methods returned in *pMethodList* will be the minimum of the value in **pMethodCount* on entry to the API and the value of **pMethodCount* on succesful return from the API.

If the call fails and the value of *pMethodList* is not NULL then the data pointed to by *pMethodList* will be unchanged.

Support

Mandatory

See Also

IMA_GetSupportedAuthMethods

IMA_SetInitiatorAuthMethods

6.2.33 IMA_GetIpProperties

Synopsis

Gets the IP properties of the specified physical network port.

Prototype

```
IMA_STATUS IMA_GetIpProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_IP_PROPERTIES *pProps
);
```

Parameters

oid

The object ID of the PNP whose IP properties are to be retrieved.

pProps

A pointer to an IMA_IP_PROPERTIES structure. On successful return this will contain the IP properties of the PNP specified by *oid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PNP object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PNP that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

IMA_SetDefaultGateway

IMA_SetDnsServerAddress

IMA_SetIpConfigMethod

IMA_SetSubnetMask

6.2.34 IMA_GetIpsecProperties

Synopsis

Gets the IPsec properties of the specified physical network port or logical HBA.

Prototype

```
IMA_STATUS IMA_GetIpsecProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_IPSEC_PROPERTIES *pProps
);
```

Parameters

phbaOid

The object ID of the PNP or LHBA whose IPsec properties are being retrieved.

pProps

A pointer to an IMA_IPSEC_PROPERTIES structure allocated by the caller. On successful return this will contain the IPsec properties of the specified PNP or LHBA.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *pid* does not specify a PNP or LHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PNP or LHBA that is currently known to the system.

Remarks

If the *ipsecSupported* field in the returned IMA_IPSEC_PROPERTIES structure has the value of IMA_FALSE then the values of all other fields in the structure are not defined.

Support

Mandatory

See Also

IMA_GetLhbaOidList

IMA_GetPnpOidList

IPsec Security in Client Usage Notes

6.2.35 IMA_GetLhbaMutualAuthParmsList

Synopsis

Gets the mutual authentication parameters list for the specified authentication method for the specified LHBA object ID.

Prototype

```
IMA_STATUS IMA_GetLhbaMutualAuthParmsList(
    /* in */      IMA_OID oid,
    /* in */      IMA_AUTHMETHOD method,
    /* out */     IMA_TARGET_AUTHPARMS_LIST **ppParmsList,
);
```

Parameters

oid

The object ID of an LHBA whose authentication parameters are to be retrieved.

method

The authentication method of the object ID whose authentication parameters are to be retrieved.

ppParmsList

A pointer to a pointer to an IMA_TARGET_AUTHPARMS_LIST structure. On successful return this will contain a list of the mutual authentication parameters for the specified authentication method on the specified object ID.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA.

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support the mutual authentication parameters list

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *o*id does not specify an LHBA that is currently known to the system.

Remarks

None

Support

Mandatory if the *mutualLhbaAuthParmsListSupported* field of the IMA_LHBA_PROPERTIES returned by the IMA_GetLhbaProperties API true for the LHBA associated with this target.

See Also

IMA_AddLhbaMutualAuthParms

IMA_RemoveLhbaMutualAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.36 IMA_GetLhbaOidList

Synopsis

Gets a list of the object IDs of all the logical HBAs in the system.

Prototype

```
IMA_STATUS IMA_GetLhbaOidList(
    /* out */       IMA_OID_LIST **ppList
);
```

Parameters

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the LHBAs currently in the system.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

Remarks

If there are no LHBAs then the call completes successfully and the returned list contains zero entries.

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

Example of Getting LHBA Properties

6.2.37 IMA_GetLhbaProperties

Synopsis

Gets the properties of the specified logical HBA.

Prototype

```
IMA_STATUS IMA_GetLhbaProperties(
    /* in */     IMA_OID lhbaOid,
    /* out */    IMA_LHBA_PROPERTIES *pProps
);
```

Parameters

lhbaOid

The object ID of the LHBA whose properties are being retrieved.

pProps

A pointer to an IMA_LHBA_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the LHBA specified by *lhbaOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lhbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lhbaOid* does not specify anLHBA.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lhbaOid* does not specify an LHBA that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

Example of Getting LHBA Properties

6.2.38 IMA_GetLibraryProperties

Synopsis

Gets the properties of the IMA library that is being used.

Prototype

```
IMA_STATUS IMA_GetLibraryProperties(
    /* out */       IMA_LIBRARY_PROPERTIES *pProps
);
```

Parameters

pProps

A pointer to an IMA_LIBRARY_PROPERTIES structure allocated by the caller. On successful return this structure will contain the properties of the IMA library that is being used.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

Remarks

None

Support

Mandatory

See Also

Example of Getting Library Properties

6.2.39 IMA_GetLnpOidList

Synopsis

Gets a list of the object IDs of all the iSCSI usable logical network ports in the system.

Prototype

```
IMA_STATUS IMA_GetLnpOidList(
    /* out */       IMA_OID_LIST **ppList
);
```

Parameters

ppList

A pointer to a pointer to an IMA_OID_LIST. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the iSCSI usable LNPs known to the system.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

When the caller is finished using the list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

6.2.40 IMA_GetLnpProperties

Synopsis

Gets the properties of the specified logical network port.

Prototype

```
IMA_STATUS IMA_GetLnpProperties(
    /* in */      IMA_OID lnpOid,
    /* out */     IMA_LNP_PROPERTIES *pProps
);
```

Parameters

lnpOid

The object ID of the LNP whose properties are being retrieved.

pProps

A pointer to an IMA_LNP_PROPERTIES structure. On successful return this will contain the properties of the LNP specified by *lnpOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lnpOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lnpOid* does not specify an LNP.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lnpOid* does not specify an LNP that is currently known to the system.

Remarks

None

Support

Mandatory

6.2.41 IMA_GetLuOid

Synopsis

Gets a logical unit object ID for the specified LUN connected to the specified target.

Prototype

```
IMA_STATUS IMA_GetLuOid(
    /* in */      IMA_OID targetOid,
    /* in */      IMA_BYTE lun[8],
    /* out */     IMA_OID *pluOid
);
```

Parameters

targetOid

The object ID of the target that controls the logical unit whose object ID is being retrieved.

lun

The LUN specifying the logical unit of the target whose object ID is being retrieved.

pluOid

A pointer to an IMA_OID structure allocated by the caller. On successful return it will contain the object ID of the logical unit for the specified LUN connected to the specified target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pluOid* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *targetOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *targetOid* does not specify a target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *targetOid* does not specify a target that is currently known to the system.

Remarks

This API does not verify that the specified LUN actually exists. The client should use the IMA_LuReportLuns API to retrieve the LUNs that are available on the specified target.

Support

Mandatory

See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_GetLuOidList

IMA_GetLuProperties

IMA_LuReportLuns

6.2.42 IMA_GetLuOidList

Synopsis

Gets a list of the object IDs of all the logical units associated with the specified LHBA or target object ID that are exposed to the operating system.

Prototype

```
IMA_STATUS IMA_GetLuOidList(
    /* in */        IMA_OID oid,
    /* out */       IMA_OID_LIST **ppList
);
```

Parameters

oid

The object ID of the LHBA or target whose logical unit object IDs are being retrieved.

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the logical units associated with the specified object that are exposed to the operating system.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The entries in the returned list represent LUs that are currently exposed to the operating system. These LUs may have been exposed to the OS using the IMA_ExposeLu API or may have been exposed to the OS via some other mechanism outside of IMA.

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_FreeMemory

IMA_GetLuOid

IMA_GetLuProperties

6.2.43 IMA_GetLuProperties

Synopsis

Gets the properties of the specified logical unit.

Prototype

```
IMA_STATUS IMA_GetLuProperties(
    /* in */     IMA_OID luOid,
    /* out */    IMA_LU_PROPERTIES *pProps
);
```

Parameters

luOid

The object ID of the LU whose properties are being retrieved.

pProps

A pointer to an IMA_LU_PROPERTIES structure allocated by the caller. On successful return it will contain the properties of the specified LU.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *luOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *luOid* does not specify a LU object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *luOid* does not specify a LU that is currently known to the system.

Remarks

A successful call to this API does not guarantee that the LU associated the *luOid* actually exists. The best way for a client to ensure that a LU exists is to call the IMA_LuInquiry API and examine the returned data to determine if it indicates that the LU exists.

Support

Mandatory

See Also

IMA_GetLuOid

IMA_GetLuOidList

6.2.44 IMA_GetMaxBurstLengthProperties

Synopsis

Gets the `MaxBurstLength` iSCSI login parameter properties of the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetMaxBurstLengthProperties(
    /* in */     IMA_OID oid,
    /* out */    IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxBurstLength` properties are to be retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `MaxBurstLength` properties of the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetMaxBurstLength

iSCSI Session and Connection Parameters

Example of Getting/Setting LHBA Max Burst Length

6.2.45 IMA_GetMaxConnectionsProperties

Synopsis

Gets the `MaxConnections` iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetMaxConnectionsProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxConnections` properties are to be retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `MaxConnections` properties of sessions associated with this LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetMaxConnections

iSCSI Session and Connection Parameters

6.2.46 IMA_GetMaxOutstandingR2TProperties

Synopsis

Gets the `MaxOutstandingR2T` per task iSCSI login parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetMaxOutstandingR2TProperties(
    /* in */     IMA_OID oid,
    /* out */    IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxOutstandingR2T` properties are to be retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `MaxOutstandingR2T` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetMaxOutstandingR2T

iSCSI Session and Connection Parameters

6.2.47 IMA_GetMaxRecvDataSegmentLengthProperties

Synopsis

Gets the `MaxRecvDataSegmentLength` iSCSI login parameter properties of the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetMaxRecvDataSegmentLengthProperties(
    /* in */     IMA_OID oid,
    /* out */    IMA_MIN_MAX_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxRecvDataSegmentLength` properties are to retrieved.

pProps

A pointer to an IMA_MIN_MAX_VALUE structure allocated by the caller. On successful return this structure will contain the `MaxRecvDataSegmentLength` properties of the LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The returned structure contains the minimum and maximum values for the implementation, which may or may not be the minimum and maximum values as found in IETF RFC 3720.

Support

Mandatory for both LHBAs and targets

See Also

IMA_SetMaxRecvDataSegmentLength

iSCSI Session and Connection Parameters

6.2.48 IMA_GetMutualLocalAuth

Synopsis

Gets the mutual authentication setting for the specified authentication method for the specified object ID.

Prototype

```
IMA_STATUS IMA_GetMutualAuth (
    /* in */      IMA_OID oid,
    /* in */      IMA_AUTHMETHOD method,
    /* out */     IMA_BOOL *mutualAuthEnabled
);
```

Parameters

oid

The object ID of the target, static target or discovery address whose mutual authentication behavior is being set.

method

The authentication method of the object ID for which mutual authentication behavior is being retrieved.

mutualAuthEnabled

A pointer to a boolean indicating whether mutual authentication will be performed for the specified object ID for the specified authentication method. If this parameter is set to IMA_TRUE the initiator will perform mutual authentication for the specified object ID for the specified authentication method.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the authentication parameters takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a target, static target or discovery address.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *o*id does not specify an object ID that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory if the *mutualAuthSupported* field of the IMA_LHBA_PROPERTIES returned by the IMA_GetLhbaProperties API is true for the LHBA associated with this target.

See Also

IMA_GetMutualLocalAuthParms

IMA_SetMutualLocalAuthParms

IMA_SetMutualLocalAuth

IMA_AddLhbaMutualAuthParms

IMA_RemoveLhbaMutualAuthParms

IMA_GetLhbaMutualAuthParmsList

Client Implementation Notes: Transmission of Authorization Parameters

6.2.49 IMA_GetMutualLocalAuthParms

Synopsis

Gets the parameters for the specified authentication method for the specified object ID for purposes of mutual authentication.

Prototype

```
IMA_STATUS IMA_GetMutualLocalAuthParms(
    /* in */      IMA_OID oid,
    /* in */      IMA_AUTHMETHOD method,
    /* out */     IMA_TARGET_AUTHPARMS *pParms,
);
```

Parameters

oid

The object ID of the target, static target or discovery address whose authentication parameters are to be retrieved.

method

The authentication method of the object ID whose authentication parameters are to be retrieved.

pParms

A pointer to an IMA_TARGET_AUTHPARMS structure. On successful return this will contain the mutual authentication parameters for the specified authentication method on the specified object ID.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a target, static target or discovery address.

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA associated with the target does not support setting authentication parameters for a target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *o*id does not specify a target that is currently known to the system.

Remarks

None

Support

Mandatory if the *targetAuthParmsSupported* field of the IMA_LHBA_PROPERTIES returned by the IMA_GetLhbaProperties API true for the LHBA associated with this target.

See Also

IMA_SetMutualLocalAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.50 IMA_GetNetworkPortalOidList

Synopsis

Gets a list of the object IDs of the network portals that can be used to enumerate the network portals of a logical HBA.

Prototype

```
IMA_STATUS IMA_GetNetworkPortalOidList(
    /* in */      IMA_OID oid,
    /* out */     IMA_OID_LIST **ppList
);
```

Parameters

oid

The object ID of the LNP whose network portals are being enumerated.

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the network portals associated with the specified OID.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LNP object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an LNP that is currently known to the system.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

6.2.51 IMA_GetNetworkPortalProperties

Synopsis

Gets the properties of the specified network portal.

Prototype

```
IMA_STATUS IMA_GetNetworkPortalProperties(
    /* in */     IMA_OID networkPortalOid,
    /* out */    IMA_NETWORK_PORTAL_PROPERTIES *pProps
);
```

Parameters

networkPortalOid

The object ID of the network portal whose properties are being retrieved.

pProps

A pointer to an IMA_NETWORK_PORTAL_PROPERITES structure allocated by the caller. On successful return this structure will contain the properties of the network portal specified by *networkPortalOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *networkPortalOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *networkPortalOid* does not specify a network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *networkPortalOid* does not specify an LNP that is currently known to the system.

Remarks

None

Support

Mandatory

6.2.52 IMA_GetNetworkPortStatus

Synopsis

Gets the status of a specified logical or physical network port.

Prototype

```
IMA_STATUS IMA_GetNetworkPortStatus(
    /* in */      IMA_OID portOid,
    /* out */     IMA_NETWORK_PORT_STATUS *pStatus
);
```

Parameters

portOid

The object ID of the logical or physical network port whose status is being retrieved.

pStatus

A pointer to an IMA_NETWORK_PORT_STATUS variable allocated by the caller. On successful return it will contain the current status of the specified logical or physical network port.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pStatus* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *portOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *portOid* does not specify a logical or physical network port object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *portOid* does not specify a logical or physical network port that is currently known to the system.

Remarks

None

Support

Mandatory for both LNPs and PNPs.

6.2.53 IMA_GetNodeProperties

Synopsis

Gets the properties of the specified iSCSI node.

Prototype

```
IMA_STATUS IMA_GetNodeProperties(
    /* in */     IMA_OID nodeOid,
    /* out */    IMA_NODE_PROPERTIES *pProps
);
```

Parameters

nodeOid

The ID of the node to get the properties of.

pProps

A pointer to an IMA_NODE_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the specified by *nodeOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *nodeOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *nodeOid* does not specify a node object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *nodeOid* does not specify a node that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

IMA_GetSharedNodeOid

IMA_GetNonSharedNodeOidList

Example of Getting Node Properties

6.2.54 IMA_GetNonSharedNodeOidList

Synopsis

Gets a list of the object IDs for the non-shared nodes of the currently executing operating system image.

Prototype

```
IMA_STATUS IMA_GetNonSharedNodeOidList(
    /* out */       IMA_OID_LIST **ppList
);
```

Parameters

ppList

A pointer to a pointer to an IMA_OID_LIST. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the non-shared nodes currently in the system.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

Remarks

If there are no non-shared nodes then the call completes successfully and the returned list contains zero entries.

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

Concepts: The Shared Node vs. Non-shared Nodes

IMA_FreeMemory

IMA_GetSharedNodeOid

IMA_GetNodeProperties

Example of Getting an Associated Plugin ID

6.2.55 IMA_GetOFMarkerProperties

Synopsis

Gets the `OFMarker` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetOFMarkerProperties(
    /* in */        IMA_OID oid,
    /* out */       IMA_BOOL_VALUE *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `OFMarker` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `OFMarker` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

6.2.56 IMA_GetOFMarkIntProperties

Synopsis

Gets the `OFMarkInt` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetOFMarkInt Properties(
    /* in */      IMA_OID oid,
    /* out */     IMA_MARKER_INT_PROPERTIES *pProps
);
```

Parameters

oid

The object ID of the LHBA or target whose `OFMarkInt` properties are to be retrieved.

pProps

A pointer to an IMA_BOOL_VALUE structure allocated by the caller. On successful return this structure will contain the `OFMarkInt` properties for the specified LHBA or target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if*oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

None

Support

Mandatory for both LHBAs and targets

See Also

6.2.57 IMA_GetObjectType

Synopsis

Gets the object type of an initialized object ID.

Prototype

```
IMA_STATUS IMA_GetObjectType(
    /* in */        IMA_OID oid,
    /* out */       IMA_OBJECT_TYPE *pObjectType
);
```

Parameters

oid

The initialized object ID to get the type of.

pObjectType

A pointer to an IMA_OBJECT_TYPE variable allocated by the caller. On successful return it will contain the object type of *oid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pObjectType* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type. This is most likely to happen if an uninitialized object ID is passed to the API.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* has an owner that is not currently known to the system.

Remarks

This API is provided so that clients can determine the type of object an object ID represents. This can be very useful for a client function that receives notifications of object visibility changes.

Support

Mandatory

See Also

IMA_RegisterForObjectVisibilityChanges

6.2.58 IMA_GetPhbaDownloadProperties

Synopsis

Gets the download properties for the specified PHBA.

Prototype

```
IMA_STATUS IMA_GetPhbaDownloadProperties(
    /* in */     IMA_OID phbaOid,
    /* out */    IMA_PHBA_DOWNLOAD_PROPERTIES *pProps
);
```

Parameters

phbaOid

The object ID of the PHBA whose download properties are being queried.

pProps

A pointer to an IMA_PHBA_DOWNLOAD_PROPERTIES structure allocated by the caller. On successful return it will contain the download properties of the specified PHBA.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *phbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *phbaOid* does not specify a PHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *phbaOid* does not specify a PHBA that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

IMA_GetPhbaOidList

IMA_IsPhbaDownloadFile

IMA_PhbaDownload

6.2.59 IMA_GetPhbaOidList

Synopsis

Gets a list of the object IDs of all the physical HBAs in the system.

Prototype

```
IMA_STATUS IMA_GetPhbaOidList(
    /* out */       IMA_OID_LIST **ppList
);
```

Parameters

ppList

A pointer to a pointer to an IMA_OID_LIST. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the PHBAs currently in the system.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

If there are no PHBAs then the call completes successfully and the returned list contains zero entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

6.2.60 IMA_GetPhbaProperties

Synopsis

Gets the general properties of a physical HBA.

Prototype

```
IMA_STATUS IMA_GetPhbaProperties(
    /* in */     IMA_OID phbaOid,
    /* out */    IMA_PHBA_PROPERTIES *pProps
);
```

Parameters

phbaOid

The object ID of the PHBA whose properties are being queried.

pProps

A pointer to an IMA_PHBA_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the PHBA specified by *phbaOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *phbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *phbaOid* does not specify a PHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *phbaOid* does not specify a PHBA that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

IMA_GetPhbaOidList

Example of Getting PHBA Properties

6.2.61 IMA_GetPhbaStatus

Synopsis

Gets the status of a specified physical HBA.

Prototype

```
IMA_STATUS IMA_GetPhbaStatus(
    /* in */      IMA_OID hbaOid,
    /* out */     IMA_PHBA_STATUS *pStatus
);
```

Parameters

hbaOid

The object ID of the physical HBA whose status is being retrieved.

pStatus

A pointer to an IMA_PHBA_STATUS variable allocated by the caller. On successful return it will contain the current status of the specified physical HBA.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pStatus* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *hbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *hbaOid* does not specify a physical HBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *hbaOid* does not specify a physical HBA that is currently known to the system.

Remarks

None

Support

Mandatory for PHBAs.

6.2.62 IMA_GetPluginOidList

Synopsis

Gets a list of the object IDs of all currently loaded plugins.

Prototype

```
IMA_STATUS IMA_GetPluginOidList(
    /* out */       IMA_OID_LIST **ppList
);
```

Parameters

ppList

A pointer to a pointer to an IMA_OID_LIST. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the plugins currently loaded by the library.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_PLUGINS_NOT_SUPPORTED

Returned if the library implementation does not support plugins.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

When the caller is finished using the list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

IMA_GetPluginProperties

Example of Getting Plugin Properties

6.2.63 IMA_GetPluginProperties

Synopsis

Gets the properties of the specified vendor plugin.

Prototype

```
IMA_STATUS IMA_GetPluginProperties(
    /* in */     IMA_OID pluginOid
    /* out */    IMA_PLUGIN_PROPERTIES *pProps
);
```

Parameters

pluginOid

The ID of the plugin whose properties are being retrieved.

pProps

A pointer to an IMA_PLUGIN_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the plugin specified by *pluginId*.

Typical Return Values

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *pluginOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *pluginOid* does not specify a plugin object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *pluginOid* does not specify a plugin that is currently known to the system.

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

Remarks

None

Support

Mandatory

See Also

IMA_GetAssociatedPluginOid

IMA_GetPluginOidList

Example of Getting Plugin Properties

6.2.64 IMA_GetPnpOidList

Synopsis

Gets a list of the object IDs of all the physical network ports associated with an object.

Prototype

```
IMA_STATUS IMA_GetPnpOidList(
    /* in */      IMA_OID oid,
    /* out */     IMA_OID_LIST **ppList
);
```

Parameters

oId

The object ID of the PHBA or LNP whose PNPs are being retrieved.

ppList

A pointer to a pointer to an IMA_OID_LIST. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the PNPs that associated with the specified PHBA or LNP.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA or LNP.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA or LNP that is currently known to the system.

Remarks

The returned list is guaranteed to contain at least one entry.

The returned list is guaranteed to not contain any duplicate entries.

When the caller is finished using the list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

IMA_GetPnpProperties

6.2.65 IMA_GetPnpProperties

Synopsis

Gets the properties of the specified physical network port.

Prototype

```
IMA_STATUS IMA_GetPnpProperties(
    /* in */      IMA_OID pnpOid,
    /* out */     IMA_PNP_PROPERTIES *pProps
);
```

Parameters

pnpOid

The object ID of the physical network port whose properties are being retrieved.

pProps

A pointer to an IMA_PNP_PROPERTIES structure. On successful return this will contain the properties of the physical network port specified by *pnpOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *pnpOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *pnpOid* does not specify a physical network port object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *pnpOid* does not specify a physical network port that is currently known to the system.

Remarks

None

Support

Mandatory

6.2.66 IMA_GetPnpStatistics

Synopsis

Gets the statistics related to a physical network port.

Prototype

```
IMA_STATUS IMA_GetPnpStatistics(
    /* in */     IMA_OID pnpOid,
    /* out */    IMA_PNP_STATISTICS *pStats
);
```

Parameters

pnpOid

The object ID of the physical network port whose statistics are being retrieved.

pStats

A pointer to an IMA_PNP_STATISTICS structure allocated by the caller. On successful return it will contain the current statistics of the specified physical network port.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pStats* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *pnpOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *pnpOid* does not specify a physical network port object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *pnpOid* does not specify a physical network port that is currently known to the system.

IMA_ERROR_STATS_COLLECTION_NOT_ENABLED

Returned if statistics collection is not enabled for the PNP specified by *pnpOid*.

Remarks

Statistics can only be retrieved if statistics collection is enabled. The IMA_GetStatisticsProperties API is used to determine if statistics collection is enabled and can be enabled. If statistics collection is not enabled, but can be enabled, the IMA_SetStatisticsCollection API is used to enable statistics collection.

Support

Mandatory if either the *statisticsCollectionEnabled* or *statisticsCollectionSettable* fields of the IMA_STATISTICS_PROPERTIES structure returned by the IMA_GetStatisticsProperties API for the specified PNP have the value IMA_TRUE.

See Also

IMA_STATISTICS_PROPERTIES

IMA_GetStatisticsProperties

IMA_SetStatisticsCollection

6.2.67 IMA_GetRadiusAccess

Synopsis

Retrieves RADIUS properties for the specified LHBA, PHBA or network portal.

Prototype

```
IMA_STATUS IMA_GetRadiusAccess(
    /* in */      IMA_OID oid,
    /* out */     IMA_RADIUS_PROPERTIES *radiusProps
);
```

Parameters

oid

The object ID of the LHBA, PHBA or network portal whose RADIUS access properties are being retrieved.

radiusProps

A pointer to an IMA_RADIUS_PROPERTIES structure allocated by the caller. On successful return this will contain the RADIUS properties of the specified object.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before this call will take effect.

IMA_ERROR_NOT_SUPPORTED

Returned if the retrieval of RADIUS server properties is unsupported by the LHBA, PHBA or network portal.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA, LHBA, or network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA, LHBA, or network portal that is currently known to the system.

Support

Optional

See Also

IMA_SetRadiusAccess

6.2.68 IMA_GetSessionOidList

Synopsis

Gets a list of the object IDs of sessions associated with the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_GetSessionOidList(
    /* in */      IMA_OID oid,
    /* out */     IMA_OID_LIST **ppList
);
```

Parameters

oid

The object ID of the LHBA or target whose session object IDs are being retrieved.

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the sessions associated with the specified logical HBA or target object.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an object that is currently known to the system.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

IMA_FreeMemory

IMA_GetSessionProperties

6.2.69 IMA_GetSessionProperties

Synopsis

Gets the properties of the specified session.

Prototype

```
IMA_STATUS IMA_GetSessionProperties(
    /* in */     IMA_OID sessionOid,
    /* out */    IMA_SESSION_PROPERTIES *pProps
);
```

Parameters

sessionOid

The object ID of the session whose properties are being retrieved.

pProps

A pointer to an IMA_SESSION_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the session specified by *sessionOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *sessionOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *sessionOid* does not specify a session.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *sessionOid* does not specify a session that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

6.2.70 IMA_GetSharedNodeOid

Synopsis

Gets the object ID of the shared node of the currently executing operating system image.

Prototype

```
IMA_STATUS IMA_GetSharedNodeOid(
    /* out */       IMA_OID *pSharedNodeOid
);
```

Parameters

pSharedNodeOid

A pointer to an IMA_OID structure allocated by the caller. On successful return it will contain the object ID of the shared node of the currently executing system is placed.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pSharedNodeOid* is NULL or specifies a memory area to which data cannot be written.

Remarks

None

Support

Mandatory

See Also

Concepts: The Shared Node vs. Non-shared Nodes

IMA_GetNonSharedNodeOidList

IMA_GetNodeProperties

Example of Getting Node Properties

Example of Setting a Node Name

6.2.71 IMA_GetStaticDiscoveryTargetOidList

Synopsis

Gets a list of the object IDs of all the static discovery targets associated with the specified logical HBA or physical network port.

Prototype

```
IMA_STATUS IMA_GetStaticDiscoveryTargetOidList(
    /* in */      IMA_OID oid,
    /* out */     IMA_OID_LIST **ppList
);
```

Parameters

oid

The object ID of the logical HBA object or a physical network port object for which to retrieve the known static discovery targets.

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the static discovery targets associated with the specified object.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or an PNP object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an object that is currently known to the system.

Remarks

The returned list is guaranteed to not contain any duplicate entries.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory if the *staticDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by IMA_GetDiscoveryProperties is true for the same *oid*.

See Also

IMA_FreeMemory

IMA_AddStaticDiscoveryTarget

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetProperties

6.2.72 IMA_GetStaticDiscoveryTargetProperties

Synopsis

Gets the properties of the specified static discovery target.

Prototype

```
IMA_STATUS IMA_GetStaticDiscoveryTargetProperties(
    /* in */      IMA_OID staticDiscoveryTargetOid,
    /* out */     IMA_STATIC_DISCOVERY_TARGET_PROPERTIES *pProps
);
```

Parameters

staticDiscoveryTargetOid

The object ID of the static discovery target whose properties are being retrieved.

pProps

A pointer to an IMA_STATIC_DISCOVERY_TARGET_PROPERTIES structure allocated by the caller. On successful return this will contain the properties of the static discovery target specified by *staticDiscoveryTargetOid*.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *staticDiscoveryTargetOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *staticDiscoveryTargetOid* does not specify a static discovery target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *staticDiscoveryTargetOid* does not specify a static discovery target that is currently known to the system.

Remarks

None

Support

Mandatory

See Also

IMA_AddStaticDiscoveryTarget

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetOidList

6.2.73 IMA_GetStatisticsProperties

Synopsis

Gets the statistics properties of the specified target, logical unit, or physical network port.

Prototype

```
IMA_STATUS IMA_GetStatisticsProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_STATISTICS_PROPERTIES *pProps
);
```

Parameters

oid

The object ID of the target, LU, or PNP whose statistics are to be retrieved.

pProps

A pointer to an IMA_STATISTICS_PROPERTIES structure. On successful return this will contain the statistics properties of the specified object.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if setting statistics collection is not supported for the specified object.

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a target, LU, or PNP object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a target, LU, or PNP that is currently known to the system.

Remarks

None.

Support

Mandatory

See Also

IMA_GetDeviceStatistics

IMA_GetTargetErrorStatistics

IMA_SetStatisticsCollection

6.2.74 IMA_GetSupportedAuthMethods

Synopsis

Gets a list of the authentication methods supported by the specified logical HBA.

Prototype

```
IMA_STATUS IMA_GetSupportedAuthMethods(
    /* in */        IMA_OID lhbaOid,
    /* in */        IMA_BOOL getSettableMethods,
    /* in, out */   IMA_UINT *pMethodCount,
    /* out */       IMA_AUTHMETHOD *pMethodList;
);
```

Parameters

lhbaOid

The object ID of an LHBA whose authentication methods are to be retrieved.

getSettableMethods

A boolean that indicates which set of authentication methods should be returned. If the value of this parameter is IMA_TRUE then a list of authentication methods that are currently settable is returned. Settable authentication methods are those methods whose authentication parameters have been set. If the value of this parameter is IMA_FALSE then a list of all of the supported authentication methods is returned. This list may include authentication methods which cannot be set, i.e. enabled, until the method's authentication parameters are set.

pMethodCount

A pointer to an IMA_UINT allocated by the caller. On entry the pointed to value shall contain the maximum number of entries that can be placed into *pMethodList*. On return it will contain the number of entries that could be placed into *pMethodList*.

pMethodList

A pointer to an array of IMA_AUTHMETHODs allocated by the caller. This value may be NULL. If this value is not NULL on successful return the array will be filled in with the authentication methods supported by the LHBA. These entries will be sorted in decreasing order of preference of the vendor of the LHBA. If this value is NULL then the value pointed to by *pMethodCount* on entry shall be zero.

Typical Return Values

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lhbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lhbaOid* does not specify an LHBA.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lhbaOid* does not specify an LHBA that is currently known to the system.

IMA_ERROR_INVALID_PARAMETER

Returned if *pMethodList* specifies a memory area to which data cannot be written.

Remarks

A successful call to this API will always return at least one authentication method: IMA_AUTHMETHOD_NONE.

The returned list is guaranteed to contain no duplicate values.

The number of authentication methods returned in *pMethodList* will be the minimum of the value in **pMethodCount* on entry to the API and the value of **pMethodCount* on succesful return from the API.

If the call fails and the value of *pMethodList* is not NULL then the data pointed to by *pMethodList* will be unchanged.

Support

Mandatory

See Also

IMA_GetInUseInitiatorAuthMethods

IMA_SetInitiatorAuthMethods

6.2.75 IMA_GetTargetErrorStatistics

Synopsis

Gets the error statistics of the specified target.

Prototype

```
IMA_STATUS IMA_GetTargetErrorStatistics(
    /* in */     IMA_OID targetOid,
    /* out */    IMA_TARGET_ERROR_STATISTICS *pStats
);
```

Parameters

targetOid

The object ID of the target whose error statistics are being retrieved.

pStats

A pointer to an IMA_TARGET_ERROR_STATISTICS structure allocated by the caller. On successful return it will contain the error statistics of the specified target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pStats* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *targetOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *targetOid* does not specify a target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *targetOid* does not specify a target that is currently known to the system.

IMA_ERROR_STATS_COLLECTION_NOT_ENABLED

Returned if statistics collection is not enabled for the specified target.

Remarks

Statistics are only collected when statistics collection is enabled for the target. Determining if statistics collection can be enabled and is enabled is done using the IMA_GetStatisticsProperties API. Statistics collection is enabled using the IMA_SetStatisticsCollection API.

Support

Mandatory if the *statisticsCollectionEnabled* field of the IMA_STATISTICS_PROPERTIES structure returned by the IMA_GetStatisticsProperties API for the specified target has the value IMA_TRUE.

See Also

IMA_STATISTICS_PROPERTIES

IMA_GetStatisticsProperties

IMA_GetTargetOidList

IMA_SetStatisticsCollection

6.2.76 IMA_GetTargetOidList

Synopsis

Gets a list of the object IDs of all the targets that have been discovered by the specified logical HBA or that are reachable via the specified logical network port.

Prototype

```
IMA_STATUS IMA_GetTargetOidList(
    /* in */      IMA_OID oid,
    /* out */     IMA_OID_LIST **ppList
);
```

Parameters

oid

The object ID of the object to get the known targets of. This shall be a logical HBA object or a logical network port object.

ppList

A pointer to a pointer to an IMA_OID_LIST structure. On successful return this will contain a pointer to an IMA_OID_LIST that contains the object IDs of all of the targets associated with the specified object.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *ppList* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or an LNP object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an object that is currently known to the system.

Remarks

The returned list is the union of all targets that have been found via the various discovery methods: static discovery, SLP, iSNS, and SendTargets.

The returned list is guaranteed to not contain any duplicate entries. Therefore, if *oid* specifies an LHBA and an iSCSI target is discovered using more than one discovery method it will appear only once in the returned list. However, if *oid* specifies an LNP then a single iSCSI target may be accessible via multiple LHBAs, in which case a single iSCSI target would be referred to be mulitple OIDs – one for each LHBA. In this case, all of the OIDs that refererd to the iSCSI target would be in the returned list.

When a client is finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

Support

Mandatory

See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_FreeMemory

IMA_GetTargetProperties

6.2.77 IMA_GetTargetProperties

Synopsis

Gets the properties of the specified target.

Prototype

```
IMA_STATUS IMA_GetTargetProperties(
    /* in */     IMA_OID targetOid,
    /* out */    IMA_TARGET_PROPERTIES *pProps
);
```

Parameters

targetOid

The object ID of the target whose properties are being retrieved.

pProps

A pointer to an IMA_TARGET_PROPERTIES structure allocated by the caller. On successful return it will contain the properties of the specified target.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pProps* is NULL or specifies a memory area to which data cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *targetOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *targetOid* does not specify a target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *targetOid* does not specify a target that is currently known to the system.

IMA_ERROR_TARGET_TIMEOUT

Returned if the target failed to respond to an iSCSI login request from an initiator.

IMA_ERROR_LOGIN_REJECTED

Returned if the target rejected an iSCSI login request from an initiator.

Remarks

This call can force an iSCSI login to the specified target.

Support

Mandatory

See Also

IMA_GetTargetOidList

6.2.78 IMA_IsPhbaDownloadFile

Synopsis

Determines if a file is suitable for download to a physical HBA.

Prototype

```
IMA_STATUS IMA_IsPhbaDownloadFile(
    /* in */      IMA_OID phbaOid,
    /* in */      const IMA_WCHAR *pFileName,
    /* out */     IMA_PHBA_DOWNLOAD_IMAGE_PROPERTIES *pProps
);
```

Parameters

phbaOid

The object ID of the PHBA whose download properties are being queried.

pFileName

A pointer to the name, and if necessary the path, of a file that is to be examined to determine if it contains a downloadable image for the specified PHBA.

pProps

A pointer to an IMA_PHBA_DOWNLOAD_IMAGE_PROPERTIES structure allocated by the caller. On successful return this structure will contain the properties of the specified image file.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if this API is not supported by the specified PHBA.

IMA_ERROR_INVALID_PARAMETER

Returned if *pFileName* is NULL or specifies a memory area from which data cannot be read.

Returned if *pFileName* specifies a file that cannot be opened for reading or that does not contain a download image that can be download to the specified PHBA.

Returned if *pProps* is NULL or specifies a memory to which data cannot written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *phbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *phbaOid* does not specify a PHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *phbaOid* does not specify a PHBA that is currently known to the system.

Remarks

A client can use this API to validate that a file contains a download image before attempting to download a file using IMA_PhbaDownload.

Support

Mandatory if the *isPhbaDownloadFileSupported* field of the IMA_PHBA_DOWNLOAD_PROPERTIES structure returned by the IMA_GetPhbaDownloadProperties API for the same PHBA has the value IMA_TRUE.

See Also

IMA_GetPhbaDownloadProperties

IMA_PhbaDownload

6.2.79 IMA_LuInquiry

Synopsis

Gets the specified SCSI INQUIRY data for the specified logical unit.

Prototype

```
IMA_STATUS IMA_LuInquiry(
    /* in */        IMA_OID deviceOid,
    /* in */        IMA_BOOL evpd,
    /* in */        IMA_BOOL cmddt,
    /* in */        IMA_BYTE pageCode,

    /* out */       IMA_BYTE *pOutputBuffer,
    /* in, out */   IMA_UINT *pOutputBufferLength,

    /* out */       IMA_BYTE *pSenseBuffer,
    /* in, out */   IMA_UINT *pSenseBufferLength
);
```

Parameters

deviceOid

The object ID of the target or LU whose INQUIRY data is to be retrieved. If the object ID specifies a target then the command will be sent to LUN 0 of the target.

evpd

A boolean indicating if the EVPD bit shall be set.

cmddt

A boolean indicating if the CMDDT bit shall be set

pageCode

The value to be placed in the page or operation code byte.

pOutputBuffer

A pointer to the memory location which on successful completion will contain the data returned by the logical unit.

pOutputBufferLength

On entry this shall point to the length, in bytes, of the memory area specified by *pOutputBuffer*. On successful return this shall point to the number of bytes that were placed into the output buffer.

pSenseBuffer

A pointer to a memory location which upon this command failing with a CHECK CONDITION status will contain the sense data received from the logical unit. This parameter may be NULL, in which case no sense data will be transferred to the client if the INQUIRY command fails with a CHECK CONDITION status.

pSenseBufferLength

A pointer to the sense buffer length. On entry to the function this shall point to a variable containing the maximum length, in bytes, of *pSenseBuffer*.

If IMA_ERROR_SCSI_STATUS_CHECK_CONDITION is returned by the API then on exit it will contain the number of sense data bytes returned in *pSenseBuffer*. If *pSenseBuffer* is NULL then this value shall be 0. If any other value is returned by the API then on exit this value will be unchanged.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pOutputBuffer* is NULL or specifies a memory area to which *pOutputBufferLength* bytes cannot be written.

Returned if *outputBufferLength* is 0.

Returned if **pSenseBufferLength* is greater than zero and *pSenseBuffer* specifies a memory area to which *pSenseBufferLength* bytes cannot be written.

Returned if *evpd* or *cmddt* have a value other than IMA_TRUE or IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *deviceOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *deviceOid* does not specify a target or LU object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *deviceOid* does not specify a target or LU that is currently known to the system.

IMA_ERROR_SCSI_STATUS_CHECK_CONDITION

Returned if the SCSI INQUIRY command failed with a CHECK CONDITION status. If this value is returned then if *pSenseBuffer* is not NULL it will contain the sense data returned by the logical unit.

IMA_ERROR_TARGET_TIMEOUT

Returned if the target associated with the specified LU failed to respond to an iSCSI login request from an initiator.

IMA_ERROR_LOGIN_REJECTED

Returned if the target associated with the specified LU rejected an iSCSI login request from an initiator.

Remarks

None

Support

Mandatory

See Also

IMA_LuReadCapacity

IMA_LuReportLuns

6.2.80 IMA_LuReadCapacity

Synopsis

Gets the SCSI READ CAPACITY data for the specified logical unit.

Prototype

```
IMA_STATUS IMA_LuReadCapacity(
    /* in */        IMA_OID deviceOid,
    /* in */        IMA_UINT cdbLength,

    /* out */       IMA_BYTE *pOutputBuffer,
    /* in, out */   IMA_UINT *pOutputBufferLength,

    /* out */       IMA_BYTE *pSenseBuffer,
    /* in, out */   IMA_UINT *pSenseBufferLength
);
```

Parameters

deviceOid

The object ID of the target or LU whose READ CAPACITY data is being retrieved. If the object ID specifies a target then the command will be sent to LUN 0 of the target.

cdbLength

The length in bytes of the READ CAPACITY CDB that shall be sent to the target or LU. Valid values are 10 and 16. Support of the 10 byte CDB is mandatory, while support for the 16 byte CDB is optional.

pOutputBuffer

A pointer to the memory location that on successful completion will contain the data returned by the logical unit.

pOutputBufferLength

On entry this shall point to the length, in bytes, of the memory area specified by *pOutputBuffer*. On successful return this shall point to the number of bytes that were actually placed into the output buffer.

pSenseBuffer

A pointer to a memory location that upon this command failing with a CHECK CONDITION status will contain the sense data received from the logical unit. This parameter may be NULL, in which case no sense data will be transferred to the client if the READ CAPACITY command fails with a CHECK CONDITION status.

pSenseBufferLength

A pointer to the sense buffer length. On entry to the function this shall point to a variable containing the maximum length, in bytes, of *pSenseBuffer*.

If IMA_ERROR_SCSI_STATUS_CHECK_CONDITION is returned by the API then on exit it will contain the number of sense data bytes returned in *pSenseBuffer*. If *pSenseBuffer* is NULL then this value shall be 0. If any other value is returned by the API then on exit this value will be unchanged.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pOutputBuffer* is NULL or specifies a memory area to which *pOutputBufferLength* bytes cannot be written.

Returned if *outputBufferLength* is 0.

Returned if *pSenseBufferLength* is greater than zero and *pSenseBuffer* specifies a memory area to which *pSenseBufferLength* bytes cannot be written.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *deviceOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *deviceOid* does not specify a target or LU object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *deviceOid* does not specify a target or LU that is currently known to the system.

IMA_ERROR_SCSI_STATUS_CHECK_CONDITION

Returned if the SCSI READ CAPACITY command failed with a CHECK CONDITION status. If this value is returned then if *pSenseBuffer* is not NULL it will contain the sense data returned by the logical unit.

IMA_ERROR_TARGET_TIMEOUT

Returned if the target associated with the specified LU failed to respond to an iSCSI login request from an initiator.

IMA_ERROR_LOGIN_REJECTED

Returned if the target associated with the specified LU rejected an iSCSI login request from an initiator.

Remarks

The length of the CDB affects both the length of the data buffer used and the format of the data in *pOutputBuffer* on successful return from this API. It is up to the caller to properly specify the length of the data buffer and to interpret the format of the returned data.

Support

Mandatory

See Also

IMA_LuInquiry

IMA_LuReportLuns

6.2.81 IMA_LuReportLuns

Synopsis

Gets the SCSI REPORT LUNS data for the specified logical unit.

Prototype

```
IMA_STATUS IMA_LuReportLuns(
    /* in */        IMA_OID deviceOid,
    /* in */        IMA_BOOL sendToWellKnownLu,
    /* in */        IMA_BYTE selectReport,

    /* out */       IMA_BYTE *pOutputBuffer,
    /* in, out */   IMA_UINT *pOutputBufferLength,

    /* out */       IMA_BYTE *pSenseBuffer,
    /* in, out */   IMA_UINT *pSenseBufferLength
);
```

Parameters

deviceOid

The object ID of the target or LU whose REPORT LUNS data is being retrieved. If the object ID identifies a target then the command will be sent to either LUN 0 or the well known REPORT LUNS LUN of the target, depending upon the value of *wellKnownLun*.

sendToWellKnownLu

If *oid* specifies a target then *sendToWellKnownLu* indicates if the REPORT LUNS command shall be sent to the REPORT LUNS well known LU or to the LU at LUN 0.

- If *oid* specifies a target and *sendToWellKnownLu* has the value IMA_TRUE then the REPORT LUNS command will be sent to the REPORT LUNS well known LU of the target.
- If *oid* specifies a target and *sendToWellKnownLu* has the value IMA_FALSE then the REPORT LUNS command will be sent to LUN 0 of the target.
- If *oid* specifies a LU then the REPORT LUNS command is sent to the specified LU and the value of *sendToWellKnownLu* is ignored.

See section 9 of *ANSI INCITS 408-2005* for more information regarding well known LUs.

selectReport

The select report value as defined for the REPORT LUNS command. See the T10 document *ANSI INCITS 408-2005* or your device's SCSI interface documentation for more information on the values this field can contain.

pOutputBuffer

A pointer to the memory location that on successful completion will contain the data returned by the logical unit.

pOutputBufferLength

A pointers to the output buffer length. On entry to the function this shall point to a variable containing the maximum length, in bytes, of *pOutputBuffer*.

On successful return it will contain the number of bytes that were actually returned in *pOutputBuffer*. On failed return this value will be unchanged.

pSenseBuffer

A pointer to a memory location that upon this command failing with a CHECK CONDITION status will contain the sense data received from the logical unit. This parameter may be NULL, in which case no sense data will be transferred to the client if the REPORT LUNS command fails with a CHECK CONDITION status.

pSenseBufferLength

A pointer to the sense buffer length. On entry to the function this shall point to a variable containing the maximum length, in bytes, of *pSenseBuffer*.

If IMA_ERROR_SCSI_STATUS_CHECK_CONDITION is returned by the API then on exit it will contain the number of sense data bytes returned in *pSenseBuffer*. If *pSenseBuffer* is NULL then this value shall be 0. If any other value is returned by the API then on exit this value will be unchanged.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if *deviceOid* specifies a target and *sendToWellKnownLu* has the value IMA_TRUE and sending to the REPORT LUNS well known LUN is not supported.

IMA_ERROR_INVALID_PARAMETER

Returned if *pOutputBuffer* is NULL or specifies a memory area to which *pOutputBufferLength* bytes cannot be written.

Returned if *pOutputBufferLength* is NULL or specifies a memory which cannot be written.

Returned if **pOutputBufferLength* is 0.

Returned if *pSenseBufferLength* is greater than zero and *pSenseBuffer* specifies a memory area to which *pSenseBufferLength* bytes cannot be written.

Returned if *sendToWellKnownLu* has a value other than IMA_TRUE and IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *deviceOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *deviceOid* does not specify a LU object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *deviceOid* does not specify a LU that is currently known to the system.

IMA_ERROR_SCSI_STATUS_CHECK_CONDITION

Returned if the SCSI REPORT LUNS command failed with a CHECK CONDITION status. If this value is returned then if *pSenseBuffer* is not NULL it will contain the sense data returned by the logical unit.

IMA_ERROR_TARGET_TIMEOUT

Returned if the target associated with the specified LU failed to respond to an iSCSI login request from an initiator.

IMA_ERROR_LOGIN_REJECTED

Returned if the target associated with the specified LU rejected an iSCSI login request from an initiator.

Remarks

None

Support

Mandatory in all situations except when *oid* specifies a target and *sendToWellKnownLu* has the value IMA_TRUE. Supporting this situation is optional.

See Also

IMA_LuInquiry

IMA_LuReadCapacity

6.2.82 IMA_PersistHbaParameters

Synopsis

Persists parameters applied through IMA set operations.

Prototype

```
IMA_STATUS IMA_PersjstHbaParameters(
    /* in */        IMA_OID oid
);
```

Parameters

hbaOid

The object ID of the PHBA or LHBA whose settings are being persisted.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if this API is not supported by the specified PHBA or LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *phbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *phbaOid* does not specify a PHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *phbaOid* does not specify a PHBA that is currently known to the system.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary in order for the settings to be made persistent.

Remarks

When this call is successful, any current connections to which set operations have been applied may be disconnected and an attempt to re-establish those connections will be made.

Support

Mandatory if the autoPersistenceSupported property on the PHBA or LHBA is set to IMA_TRUE.

See Also

6.2.83 IMA_PhbaDownload

Synopsis

Downloads the image in the specified file to the specified physical HBA.

Prototype

```
IMA_STATUS IMA_PhbaDownload(
    /* in */    IMA_OID phbaOid,
    /* in */    IMA_PHBA_DOWNLOAD_IMAGE_TYPE imageType,
    /* in */    const IMA_WCHAR *pFileName
);
```

Parameters

phbaOid

The object ID of the PHBA whose to which the image file is being downloaded.

imageType

The type of a image that is to be downloaded.

pFileName

A pointer to the name, and if necessary the path, of a file that contains the image to download to the PHBA.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if this API is not supported by the specified PHBA.

IMA_ERROR_INVALID_PARAMETER

Returned if *imageType* does not specify a valid IMA_PHBA_DOWNLOAD_IMAGE_TYPE value.

Returned if *pFileName* is NULL or specifies a memory area from which data cannot be read.

Returned if *pFileName* specifies a file that cannot be opened for reading or that does not contain a download image of the specified image type.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *phbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *phbaOid* does not specify a PHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *phbaOid* does not specify a PHBA that is currently known to the system.

Remarks

None

Support

If IMA_GetPhbaDownloadProperties is not supported then this API shall not be supported as well.

Mandatory if any of the fields in the IMA_PHBA_DOWNLOAD_PROPERTIES structure as would be returned by the IMA_GetPhbaDownloadProperties API for the specified PHBA can be set to IMA_TRUE.

See Also

IMA_PHBA_DOWNLOAD_IMAGE_TYPE

IMA_GetPhbaDownloadProperties

IMA_IsPhbaDownloadFile

6.2.84 IMA_PluginIOCtl

Synopsis

Sends a vendor specific command to a specified plugin.

Prototype

```
IMA_STATUS IMA_PluginIOCtl(
    /* in */        IMA_OID pluginOid,
    /* in */        IMA_UINT command,
    /* in */        const void *pInputBuffer,
    /* in */        IMA_UINT inputBufferLength,
    /* out */       void *pOutputBuffer,
    /* in, out */   IMA_UINT *pOutputBufferLength,
);
```

Parameters

pluginOid

The object ID of the plugin to which the command is being set.

command

The command to be sent to the plugin.

pInputBuffer

A pointer to a buffer allocated by the caller that contains any input parameters the specified command needs. A value of NULL for this parameter indicates that there are no input parameters being provided by the caller. In this case the value of *inputBufferLength* shall be 0.

inputBufferLength

The length, in bytes, of the input buffer. The plugin shall not attempt to read more data than is specified by the caller.

pOutputBuffer

A pointer to a buffer allocated by the caller that contains any output of the specified command. A value of NULL for this parameter indicates that the caller expects to receive no output data from the plugin. In this case the value of *pOutputBufferLength* shall be NULL.

pOutputBufferLength

A pointer to a value that on entry contains the length, in bytes, of the output buffer. On successful return that value will be the number of bytes returned in *pOutputBuffer*. If this value is NULL then both *pInputBuffer* and *pOutputBuffer* shall be NULL and *inputBufferLength* shall be 0.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the plugin does not support this API call.

IMA_ERROR_INVALID_PARAMETER

Returned if:

- *pInputBuffer* is NULL and *inputBufferLength* is not 0

- *pOutputBuffer* is NULL and *pInputBufferLength* points to a value that is not 0.
- *pOutputBuffer* is not NULL and *pOutputBufferLength* is NULL or points to a value that is 0.
- if the plugin does not support the specified command or if the input and/or output buffers and lengths are not correct for the given command.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *pluginOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *pluginOid* does not specify a plugin object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *pluginOid* does not specify a plugin that is currently known to the system.

Remarks

This purpose of this API call is to allow a client to send a plugin a command that is vendor unique, i.e., a command that is specific to a particular plugin. The command values and the format of both the input and output buffers is entirely plugin specific. It is up to a client to determine if a particular plugin supports particular plugin IOCtls that the client wants to send. Clients can do this by using the vendor and version fields of the IMA_PLUGIN_PROPERTIES structure as returned by the IMA_GetPluginProperties API.

Support

Optional

See Also

IMA_PLUGIN_PROPERTIES

IMA_GetPluginOidList

IMA_GetPluginProperties

6.2.85 IMA_RegisterForObjectPropertyChanges

Synopsis

Registers a client function to be called whenever the property of an object changes.

Prototype

```
IMA_STATUS IMA_RegisterForObjectPropertyChanges (
    /* in */        IMA_OBJECT_PROPERTY_FN pClientFn
);
```

Parameters

pClientFn

A pointer to an IMA_OBJECT_PROPERTY_FN function defined by the client. On successful return this function will be called to inform the client of objects that have had one or more properties change.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pClientFn* is NULL or specifies a memory area that is not executable.

Support

Mandatory

Remarks

The function specified by *pClientFn* will be called whenever the property of an object changes. For the purposes of this function a property is defined to be a field in an object's property structure and the object's status. Therefore, the client function will not be called if a statistic of the associated object changes. But, it will be called when the status changes (e.g. from working to failed) or when a name or other field in a property structure changes.

It is not an error to re-register a client function. However, a client function has only one registration. The first call to deregister a client function will deregister it no matter how many calls to register the function have been made.

If multiple properties of an object change simultaneously a client function may be called only once to be notified that the changes have occurred.

See Also

IMA_DeregisterForObjectPropertyChanges

6.2.86 IMA_RegisterForObjectVisibilityChanges

Synopsis

Registers a client function to be called whenever a high level object appears or disappears.

Prototype

```
IMA_STATUS IMA_RegisterForObjectVisibilityChanges (
    /* in */        IMA_OBJECT_VISIBILITY_FN pClientFn
);
```

Parameters

pClientFn

A pointer to an IMA_OBJECT_VISIBILITY_FN function defined by the client. On successful return this function will be called to inform the client of objects whose visibility has changed.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *pClientFn* is NULL or specifies a memory area that is not executable.

Support

Mandatory

Remarks

The function specified by *pClientFn* will be called whenever high level objects appear or disappear. The following are considered high level objects:

- Nodes (IMA_OBJECT_TYPE_NODE)
- Logical HBAs (IMA_OBJECT_TYPE_LHBA)
- Physical HBAs (IMA_OBJECT_TYPE_PHBA)
- Targets (IMA_OBJECT_TYPE_TARGET)
- Sessions (IMA_OBJECT_TYPE_SESSION)
- Connections (IMA_OBJECT_TYPE_CONNECTION)

All other objects are considered lower level objects and the function specified by *pClientFn* will not be called for their appearance or disappearance. Lower level object visibility can be determined from high level object visibility.

It is not an error to re-register a client function. However, a client function has only one registration. The first call to deregister a client function will deregister it no matter how many calls to register the function have been made.

See Also

IMA_DeregisterForObjectVisibilityChanges

6.2.87 IMA_RemoveDiscoveryAddress

Synopsis

Removes a discovery address being used for send targets discovery by a physical network port or logical HBA.

Prototype

```
IMA_STATUS IMA_RemoveDiscoveryAddress(
    /* in */        IMA_OID discoveryAddressOid
);
```

Parameters

discoveryAddressOid

The object ID of the discovery address that is to no longer be used for send targets discovery.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the specified discovery address is no longer used for send targets discovery.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *discoveryAddressOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *discoveryAddressOid* does not specify a discovery address object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *discoveryAddressOid* does not specify a discovery address that is currently known to the system.

IMA_ERROR_LU_EXPOSED

Returned if *discoveryAddressOid* specifies an iSCSI target that currently has a LU exposed to the operating system.

Remarks

This change is persistent. The specified discovery address will no longer be used by the associated PNP or LHBA for send targets discovery.

Support

Mandatory

See Also

IMA_AddDiscoveryAddress

IMA_GetDiscoveryAddressProperties

IMA_GetDiscoveryProperties

IMA_SetSendTargetsDiscovery

6.2.88 IMA_RemoveLhbaMutualAuthParms

Synopsis

Removes a mutual auth parm from the list for a specified LHBA.

Prototype

```
IMA_STATUS IMA_RemoveLhbaMutualAuthParms(
    /* in */    IMA_OID oid,
    /* in */    IMA_AUTHMETHOD method,
    /* in */    IMA_TARGET_AUTHPARMS pParms
);
```

Parameters

oid

The object ID of an LHBA to which the authentication parameters will be removed.

method

The authentication method of the object ID whose authentication parameters are to be removed.

pParms

An IMA_INITIATOR_AUTHPARMS structure which contains the auth parms to remove from the list for the specified LHBA.

Typical Return Values

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area to which data cannot be written.

Returned if the authentication parameter to be removed can not be found.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA.

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting of the mutual authentication parameters list.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *o*id does not specify an LHBA that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory if the *mutualLhbaAuthParmsListSupported* field of the IMA_LHBA_PROPERTIES returned by the IMA_GetLhbaProperties API true for the LHBA associated with this target.

See Also

IMA_GetLhbaMutualAuthParmsList

IMA_AddLhbaMutualAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.89 IMA_RemoveStaleData

Synopsis

Removes all of the stale persistent data of the specified LHBA.

Prototype

```
IMA_STATUS IMA_RemoveStaleData(
    /* in */        IMA_OID lhbaOid,
);
```

Parameters

lhbaOid

The object ID of the LHBA whose stale data is to be removed.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support removing stale data.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lhbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lhbaOid* does not specify a LHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lhbaOid* does not specify a LHBA that is currently known to the system.

Remarks

Stale data includes the following:

- Any data required to expose LUs that belong to targets that are not present. This could occur by calling the IMA_ExposeLu API.
- Any iSCSI login parameter applied specifically to targets that are not present. This could occur by calling the IMA_SetMaxBurstLength, IMA_SetMaxRecvDataSegmentLength, or other similar APIs.

Support

Mandatory if the *staleDataRemovable* field of the IMA_LHBA_PROPERTIES structure returned by the IMA_GetLhbaProperties API has the value IMA_TRUE for the LHBA specified by *lhbaOid*. Otherwise not supported, i.e., this call shall return IMA_ERROR_NOT_SUPPORTED.

See Also

IMA_GetLhbaProperties

6.2.90 IMA_RemoveStaticDiscoveryTarget

Synopsis

Removes a target being statically discovered by a physical network port or logical HBA.

Prototype

```
IMA_STATUS IMA_RemoveStaticDiscoveryTarget(
    /* in */        IMA_OID staticDiscoveryTargetOid
);
```

Parameters

staticDiscoveryTargetOid

The object ID of the static discovery target that is to no longer be discovered.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the specified static discovery target is no longer discovered.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *staticDiscoveryTargetOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *staticDiscoveryTargetOid* does not specify a static discovery target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *staticDisocveryTargetOid* does not specify a static discovery target that is currently known to the system.

IMA_ERROR_LU_EXPOSED

Returned if *staticDiscoveryTargetOid* specifies an iSCSI target that currently has a LU exposed to the operating system.

Remarks

This change is persistent. The specified target will no longer be discovered by the associated PHBA or LHBA.

Support

Mandatory

See Also

IMA_AddStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetProperties

IMA_GetDiscoveryProperties

IMA_SetStaticDiscovery

6.2.91 IMA_SetDataDigestValues

Synopsis

Sets the list of checksums that can be negotiated for data digests.

Prototype

```
IMA_STATUS IMA_SetDataDigestValues(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT digestValueCount,
    /* in */    const IMA_DIGEST_TYPE *pDigestValueList;
);
```

Parameters

oid

The object ID of an LHBA or target whose data digest values are to be set.

digestValueCount

The number of digest values in *pDigestValueList*. There shall be at least one entry in the list.

pDigestValueList

A list of one or more digest values that the LHBA or target shall use.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the specified digest values will take affect.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA or target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an LHBA or target that is currently known to the system.

IMA_ERROR_INVALID_PARAMETER

Returned if *digestValueCount* is zero.

Returned if *pDigestValueList* is NULL, or *pDigestValueList* specifies a memory area from which data cannot be read.

Returned if the contents of *pDigestValueList* contain duplicate values

Returned if the contents of *pDigestValueList* contain an invalid value.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin associated with the specified target and applies only to

subsequent sessions and connections created by the initiator; existing sessions and connections are not affected.

Support

Mandatory if the *dataDigestsSettable* field of the IMA_DIGEST_PROPERTIES structure returned by the IMA_GetDigestProperties API for the same *oid* has the value IMA_TRUE.

See Also

6.2.92 IMA_SetDataPduInOrder

Synopsis

Sets the `DataPDUInOrder` iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetDataPduInOrder(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL dataPduInOrder
);
```

Parameters

oid

The object ID of the LHBA or target whose `DataPDUInOrder` value is being set.

dataPduInOrder

The new value for the `DataPDUInOrder` value for the specified LHBA or target.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the `DataPDUInOrder` takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `DataPDUInOrder` value.

IMA_ERROR_INVALID_PARAMETER

Returned if *dataPduInOrder* does not contain the value IMA_TRUE or IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory if the *settable* field in the IMA_BOOL_VALUE structure returned by IMA_GetDataPduInOrderProperties for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetDataPduInOrderProperties

iSCSI Session and Connection Parameters

6.2.93 IMA_SetDataSequenceInOrder

Synopsis

Sets the `DataSequenceInOrder` iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetDataSequenceInOrder(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL dataSequenceInOrder
);
```

Parameters

oid

The object ID of the LHBA or target whose `DataSequenceInOrder` value is being set.

dataSequenceInOrder

The new `DataSequenceInOrder` value for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `DataSequenceInOrder` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the `DataSequenceInOrder` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *dataSequenceInOrder* does not contain the value IMA_TRUE or IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory if the *settable* field in the IMA_BOOL_VALUE structure returned by IMA_GetDataSequenceInOrderProperties for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetDataSequenceInOrderProperties

iSCSI Session and Connection Parameters

6.2.94 IMA_SetDefaultGateway

Synopsis

Sets the default gateway for the the specified physical network port.

Prototype

```
IMA_STATUS IMA_SetDefaultGateway(
    /* in */    IMA_OID oid,
    /* in */    IMA_IP_ADDRESS defaultGateway
);
```

Parameters

oid

The object ID of the PNP whose default gateway is to be set.

defaultGateway

The new default gateway for the PNP.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the default gateway takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if setting the default gateway is not supported by the specified PNP.

IMA_ERROR_INVALID_PARAMETER

Returned if any fields in *defaultGateway* contains any invalid values.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an PNP.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an PNP that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin associated with the specified physical network port.

Support

Mandatory if the *defaultGatewaySettable* field in the IMA_IP_PROPERTIES structure returned by the IMA_GetIpProperties API for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetIpProperties

6.2.95 IMA_SetDefaultTime2Retain

Synopsis

Sets the `DefaultTime2Retain` iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetDefaultTime2Retain(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT defaultTime2Retain
);
```

Parameters

oid

The object ID of the LHBA or target whose `DefaultTime2Retain` value is being set.

defaultTime2Retain

The new value for the new `DefaultTime2Retain` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `DefaultTime2Retain` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the `DefaultTime2Retain` value actually takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *defaultTime2Retain* is out of range for the LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `DefaultTime2Retain` values can be determined by calling IMA_GetDefaultTime2RetainProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetDefaultTime2RetainProperties for the same *oid* is true.

See Also

IMA_GetDefaultTime2RetainProperties

iSCSI Session and Connection Parameters

6.2.96 IMA_SetDefaultTime2Wait

Synopsis

Sets the `DefaultTime2Wait` iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetDefaultTime2Wait(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT defaultTime2Wait
);
```

Parameters

oid

The object ID of the LHBA or target whose `DefaultTime2Wait` value is being set.

defaultTime2Wait

The new value for the new `DefaultTime2Wait` for the specified LHBA.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `DefaultTime2Wait` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the `DefaultTime2Wait` value takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *defaultTime2Wait* is out of range for the LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `DefaultTime2Wait` values can be determined by calling IMA_GetDefaultTime2WaitProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetDefaultTime2WaitProperties for the same *oid* is true.

See Also

IMA_GetDefaultTime2WaitProperties

iSCSI Session and Connection Parameters

6.2.97 IMA_SetDnsServerAddress

Synopsis

Sets the address of the primary and alternate DNS servers for the specified physical network port.

Prototype

```
IMA_STATUS IMA_SetDnsServerAddress(
    /* in */    IMA_OID oid,
    /* in */    const IMA_IP_ADDRESS *pPrimaryDnsServerAddress,
    /* in */    const IMA_IP_ADDRESS *pAlternateDnsServerAddress
);
```

Parameters

oid

The object ID of the PNP whose DNS servers are to be set.

primaryDnsServerAddress

A pointer to the IP address of the primary DNS server. This parameter can be NULL in which case the primary DNS server address is being cleared. In this case *alternateDnsServerAddress* shall also be NULL.

alternateDnsServerAddress

A pointer to the IP address of the alternate DNS server. This parameter can be NULL in which case the alternate DNS server address is being cleared.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the DNS servers takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if setting the primary DNS server address is not supported by the specified PNP.

Returned if setting the alternate DNS server address is not supported by the specified PNP and *alternateDnsServerAddress* is not NULL.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an PNP.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an PNP that is currently known to the system.

IMA_ERROR_INVALID_PARAMETER

Returned if *primaryDnsServerAddress* or *alternateDnsServerAddress* are not NULL and specify a memory area from which data cannot be read.

Returned if *primaryDnsServerAddress* is NULL and *alternateDnsServerAddress* is not NULL. It is not valid to have an alternate DNS server without having a primary DNS server.

Returned if *primaryDnsServerAddress* and *alternateDnsServerAddress* are both not NULL and both specify the same IP address.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory if the *primaryDnsServerAddressSettable* field in the IMA_IP_PROPERTIES structure returned by the IMA_GetIpProperties API for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetIpProperties

6.2.98 IMA_SetErrorRecoveryLevel

Synopsis

Sets the `ErrorRecoveryLevel` iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetErrorRecoveryLevel(
    /* in */        IMA_OID oid,
    [in[            IMA_UINT errorRecoveryLevel
);
```

Parameters

oid

The object ID of the LHBA or target whose `ErrorRecoveryLevel` value is being set.

errorRecoveryLevel

The new value for the `ErrorRecoveryLevel` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `ErrorRecoveryLevel` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the `ErrorRecoveryLevel` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *errorRecoveryLevel* is out of range for the LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `ErrorRecoverLevel` values can be determined by calling IMA_GetErrorRecoveryLevelProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_BOOL_VALUE structured returned by IMA_GetErrorRecoveryLevelProperties for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetErrorRecoveryLevelProperties

iSCSI Session and Connection Parameters

6.2.99 IMA_SetFirstBurstLength

Synopsis

Sets the `FirstBurstLength` iSCSI login parameter of the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetFirstBurstLength(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT firstBurstLength
);
```

Parameters

oid

The object ID of the LHBA or target whose `FirstBurstLength` value is being set.

firstBurstLength

The value for the new `FirstBurstLength` for the LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `FirstBurstLength`.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `FirstBurstLength` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *firstBurstLength* is out of range for the LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `FirstBurstLength` values can be determined by calling IMA_GetFirstBurstLengthProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetFirstBurstLengthProperties for the same *oid* is true.

See Also

IMA_GetFirstBurstLengthProperties

6.2.100 IMA_SetHeaderDigestValues

Synopsis

Sets the list of checksums that can be negotiated for header digests.

Prototype

```
IMA_STATUS IMA_SetDataDigestValues(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT digestValueCount,
    /* in */    const IMA_DIGEST_TYPE *pDigestValueList;
);
```

Parameters

oid

The object ID of an LHBA or target whose header digest values are to be set.

digestValueCount

The number of digest values in *pDigestValueList*. There shall be at least one entry in the list.

pDigestValueList

A list of one or more digest values that the LHBA or target shall use.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the specified digest values will take affect.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify an LHBA or target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify an LHBA or target that is currently known to the system.

IMA_ERROR_INVALID_PARAMETER

Returned if *digestValueCount* is zero.

Returned if *pDigestValueList* is NULL, or *pDigestValueList* specifies a memory area from which data cannot be read.

Returned if the contents of *pDigestValueList* contain duplicate values

Returned if the contents of *pDigestValueList* contain an invalid value.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions and connections created by the initiator; existing sessions and connections are not affected.

Support

Mandatory if the *headerDigestsSettable* field of the IMA_DIGEST_PROPERTIES structure returned by the IMA_GetDigestProperties API for the same *oid* has the value IMA_TRUE.

See Also

6.2.101 IMA_SetIFMarkerProperties

Synopsis

Sets the `IFMarker` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetIFMarkerProperties(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL ifMarker
);
```

Parameters

oid

The object ID of the LHBA or target whose `IFMarker` value is being set.

ifMarker

The new value for `IFMarker` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `IFMarker` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `IFMarker` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *ifMarker* does not contain the value IMA_TRUE or IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory for both LHBAs and targets

See Also

6.2.102 IMA_SetIFMarkIntProperties

Synopsis

Sets the `IFMarkInt` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetIFMarkIntProperties(
    /* in */    IMA_OID oid,
    /* in */    IMA_MARKER_INT ifMarkInt
);
```

Parameters

oid

The object ID of the LHBA or target whose `IFMarkInt` value is being set.

ifMarkInt

The new value for `IFMarkInt` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `IFMarkInt` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `IFMarkInt` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *ifMarkInt* contains a value outside of the range of the LHBA or target.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory for both LHBAs and targets

See Also

6.2.103 IMA_SetImmediateData

Synopsis

Sets the `ImmediateData` iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetImmediateData(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL immediateData
);
```

Parameters

oid

The object ID of the LHBA or target whose `ImmediateData` value is being set.

immediateData

The new value for `ImmediateData` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `ImmediateData` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `ImmediateData` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *immediateData* does not contain the value IMA_TRUE or IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory if the *settable* field in the IMA_BOOL_VALUE structure returned by IMA_GetImmediateDataProperties for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetImmediateDataProperties

iSCSI Session and Connection Parameters

6.2.104 IMA_SetInitialR2T

Synopsis

Sets the `InitialR2T` iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetInitialR2T(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL initialR2T
);
```

Parameters

oid

The object ID of the LHBA or target whose `InitialR2T` value is being set.

initialR2T

The new value for the `InitialR2T` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `InitialR2T` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `InitialR2T` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *initialR2T* does not contain the value IMA_TRUE or IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory if the *settable* field in the IMA_BOOL_VALUE structure returned by IMA_GetInitialR2TProperties for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetInitialR2TProperties

6.2.105 IMA_SetInitiatorAuthMethods

Synopsis

Sets the authentication methods for the specified logical HBA when used as an initiator.

Prototype

```
IMA_STATUS IMA_SetInitiatorAuthMethods(
    /* in */    IMA_OID lhbaOid,
    /* in */    IMA_UINT methodCount,
    /* in */    const IMA_AUTHMETHOD *pMethodList
);
```

Parameters

lhbaOid

The object ID of an LHBA whose authentication methods are to be set.

methodCount

The number of authentication methods in *pMethodList*. There shall be at least one entry in the list.

pMethodList

A list of one or more authentication methods that the LHBA shall use for all targets with which the LHBA communicates.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the specified authentication methods will take affect.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lhbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lhbaOid* does not specify an LHBA.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lhbaOid* does not specify an LHBA that is currently known to the system.

IMA_ERROR_INVALID_PARAMETER

Returned if *methodCount* is zero.

Returned if *pMethodList* is NULL, or *pMethodList* specifies a memory area from which data cannot be read.

Returned if the contents of *pMethodList* contain duplicate authentication methods

Returned if the contents of *pMethodList* contain an authentication method which is not supported..

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory if the *initiatorAuthMethodsSettable* field of the IMA_LHBA_PROPERTIES structure returned by the IMA_GetLhbaProperties API for the same *lhbaOid* has the value IMA_TRUE.

See Also

IMA_GetSupportedAuthMethods

IMA_GetInUseInitiatorAuthMethods

6.2.106 IMA_SetInitiatorAuthParms

Synopsis

Sets the parameters for the specified authentication method for the specified LHBA.

Prototype

```
IMA_STATUS IMA_SetInitiatorAuthParms(
    /* in */      IMA_OID lhbaOid,
    [in[          IMA_AUTHMETHOD method,
    /* in */      const IMA_INITIATOR_AUTHPARMS *pParms
);
```

Parameters

lhbaOid

The object ID of the LHBA whose authentication parameters are to be retrieved.

method

The authentication method of the LHBA whose authentication parameters are to be retrieved.

pParms

A pointer to an IMA_INITIATOR_AUTHPARMS structure that contains the parameters to be associated with the specified authentication method.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the authentication parameters takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if the setting of authentication parameters is not supported by the specified LHBA. In this case, it is likely that LHBA does not support any authentication methods.

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *lhbaOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *lhbaOid* does not specify an LHBA.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *lhbaOid* does not specify an LHBA that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin..

This API does not cause the specified authentication method to be used. To do that a client shall call the IMA_SetInitiatorAuthMethods API.

The setting of authentication parameters does not affect any established sessions. If the authentication method is enabled, or is subsequently enabled then the authentication parameters specified as a parameter to this API will be used.

Support

Optional

See Also

IMA_GetInitiatorAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.107 IMA_SetInitiatorLocalAuthParms

Synopsis

Sets the parameters for the specified authentication method for the specified object ID to be used for one-way authentication.

Prototype

```
IMA_STATUS IMA_SetInitiatorLocalAuthParms(
    /* in */    IMA_OID oid,
    /* in */    IMA_AUTHMETHOD method,
    /* in */    const IMA_INITIATOR_AUTHPARMS *pParms,
    /* in */    IMA_BOOL enabled
);
```

Parameters

oid

The object ID of the target, static target or discovery address whose authentication parameters are to be set.

method

The authentication method of the target whose authentication parameters are to be set.

pParms

A pointer to an IMA_INITIATOR_AUTHPARMS structure that contains the parameters to be associated with the specified authentication method.

enabled

A boolean indicating whether use of the target specific initiator authentication parameters is enabled or disabled. If this parameter is set to IMA_TRUE the authentication parameters set for this object ID will be used for initiator authentication.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the authentication parameters takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a target.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin. This API can be used to effectively override the authentication parameters that have been set using IMA_SetInitiatorAuthParms but only for the specified object ID.

Support

Mandatory if the *initiatorLocalAuthParmsSupported* field in the IMA_LHBA_PROPERTIES structure returned by IMA_GetLhbaProperties is true for the LHBA associated with this target oid.

See Also

IMA_GetInitiatorLocalAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.108 IMA_SetIpConfigMethod

Synopsis

Sets the IP configuration method for the specified physical network port.

Prototype

```
IMA_STATUS IMA_SetIpConfigMethod(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL enableDhcpIpConfiguration
);
```

Parameters

oid

The object ID of the PNP whose IP configuration method is to be set.

enableDhcpI[Configuration

A boolean indicating if DHCP configuration of IP is being enabled. If this parameter has the value IMA_TRUE then DHCP configuration of IP is enabled. If this parameter has the value IMA_FALSE then DHCP configuration of IP is disabled, thereby enabling static configuration of IP.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before this call will take effect.

IMA_ERROR_INVALID_PARAMETER

Returned if *enableDhcpIpConfiguration* does not contain the value IMA_TRUE or IMA_FALSE.

IMA_ERROR_NOT_SUPPORTED

Returned if setting the IP configuration method is not supported by the specified PHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PNP object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PNP that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin associated with the specified physical network port.

It is not an error to enable DHCP IP configuration when it is already enabled, nor is it an error to disable DHCP IP configuration when it is already disabled.

Neither enabling or disabling DHCP IP configuration causes the removal of any stored static IP configuration information, e.g. DNS server addresses or the gateway address. Thus if static IP configuration is enabled and static IP configuration parameters are set, e.g. DNS server addresses, then DHCP IP configuration is

enabled, and then static IP configuration is enabled once again the previously set static IP configuration parameters will be in effect.

Support

Mandatory if the *ipConfigurationMethodSettable* field of the IMA_IP_PROPERTIES structure returned by the IMA_GetIpProperties API for the same PNP has the value IMA_TRUE.

See Also

IMA_GetIpProperties

IMA_SetDefaultGateway

IMA_SetDnsServerAddress

IMA_SetSubnetMask

6.2.109 IMA_SetIsnsDiscovery

Synopsis

Enables/disables iSNS target discovery for a physical HBA, logical HBA, or network portal.

Prototype

```
IMA_STATUS IMA_SetIsnsDiscovery(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL enableIsnsDiscovery,
    /* in */    IMA_ISNS_DISCOVERY_METHOD discoveryMethod,
    /* in */    const IMA_HOST_ID *iSnsHost
);
```

Parameters

oid

The object ID of the PHBA, LHBA, or network portal whose iSNS target discovery properties are being set.

enableIsnsDiscovery

Set to IMA_TRUE if the PHBA, LHBA, or network portal shall discover targets using iSNS, set to IMA_FALSE if the PHBA,LHBA, or network portal shall not discover targets using iSNS.

discoveryMethod

If *enableIsnsDiscovery* has the value IMA_TRUE then this parameter specifies the method that shall be used to discover the iSNS server.

If this parameter has the value IMA_ISNS_DISCOVERY_METHOD_STATIC then the iSNS server will be discovered statically. In this case the iSnsHost parameter shall contain the name or address of the iSNS server and the iSNS server at that location will be used.

If this parameter has the value IMA_ISNS_DISCOVERY_METHOD_DHCP then the iSNS server will be discovered using DHCP.

If this parameter has the value IMA_ISNS_DISCOVERY_METHOD_SLP then the iSNS server will be discovered using SLP.

iSnsHost

If *enableIsnsDiscovery* has the value IMA_TRUE and *discoveryMethod* has the value IMA_ISNS_DISCOVERY_METHOD_STATIC then this is a pointer to the host name of the iSNS server to be used.

If *enableIsnsDiscovery* has the value IMA_FALSE or *enableIsnsDiscovery* has the value IMA_TRUE, but *discoveryMethod* does not have the value IMA_ISNS_DISCOVERY_METHOD_STATIC then this parameter shall be NULL.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before this call will take effect.

IMA_ERROR_NOT_SUPPORTED

Returned if the enabling/disabling of iSNS target discovery is not supported by the specified PHBA/LHBA/network portal.

IMA_ERROR_INVALID_PARAMETER

Returned if *enableIsnsDiscovery* has a value other than IMA_TRUE and IMA_FALSE.

Returned if *discoveryMethod* has a value other than IMA_ISNS_DISCOVERY_METHOD_STATIC, IMA_ISNS_DISCOVERY_METHOD_DHCP, and IMA_iSNS_DISCOVERY_METHOD_SLP.

Returnd if *enabledIsnsDiscovery* has the value IMA_TRUE and *domainName* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA, LHBA, or network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA, LHBA, or network portal that is currently known to the system.

IMA_ERROR_LAST_PRIMARY_DISCOVERY_METHOD

Returned if *enableIsnsDiscovery* is set to IMA_FALSE and iSNS target discovery is the last primary discovery method for the PHBA or LHBA, i.e., SLP target discovery and static target discovery are both either disabled or not supported.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Disabling iSNS discovery does not “undiscover” any targets that were previously discovered using iSNS discovery.

Support

Mandatory if the *iSnsDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by IMA_GetDiscoveryProperties is set to IMA_TRUE for the same *oid*.

See Also

IMA_GetDiscoveryProperties

IMA_SetSlpDiscovery

IMA_SetStaticDiscovery

IMA_SetSendTargetsDiscovery

6.2.110 IMA_SetMaxBurstLength

Synopsis

Sets the `MaxBurstLength` iSCSI login parameter of the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetMaxBurstLength(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT maxBurstLength
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxBurstLength` value is to be set

maxBurstLength

The value for the new `MaxBurstLength` for the LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `MaxBurstLength`.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `MaxBurstLength` actually takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *maxBurstLength* is out of range for the LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `MaxBurstLength` values can be determined by calling IMA_GetMaxBurstLengthProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetMaxBurstLengthProperties for the same *oid* is true.

See Also

IMA_GetMaxBurstLengthProperties

iSCSI Session and Connection Parameters

Example of Getting/Setting LHBA Max Burst Length

6.2.111 IMA_SetMaxConnections

Synopsis

Sets the `MaxConnections` for sessions iSCSI login parameter for the specified logical HBA or target

Prototype

```
IMA_STATUS IMA_SetMaxConnections(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT maxConnections
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxConnections` value is to be set.

maxConnections

The new value for `MaxConnections` of the specified object.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `MaxConnections` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `MaxConnections` value actually takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *maxConnections* is out of range for the LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `MaxConnections` values can be determined by calling IMA_GetMaxConnectionsProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetMaxConnectionsProperties for the same *oid* is true.

See Also

IMA_GetMaxConnectionsProperties

iSCSI Session and Connection Parameters

6.2.112 IMA_SetMaxRecvDataSegmentLength

Synopsis

Sets the `MaxRecvDataSegmentLength` iSCSI login parameter of the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetMaxRecvDataSegmentLength(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT maxRecvDataSegmentLength
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxRecvDataSegmentLength` value is to be set.

maxRecvDataSegmentLength

The value for the new `MaxRecvDataSegmentLength`.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `MaxRecvDataSegmentLength`.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `MaxRecvDataSegmentLength` actually takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *maxRecvDataSegmentLength* is out of range for the LHBA.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `MaxRecvDataSegmentLength` values can be determined by calling IMA_GetMaxRecvDataSegmentLengthProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetMaxRecvDataSegmentLengthProperties for the same *oid* is true.

See Also

IMA_GetMaxRecvDataSegmentLengthProperties

iSCSI Session and Connection Parameters

6.2.113 IMA_SetMaxOutstandingR2T

Synopsis

Sets the `MaxOutstandingR2T` per task iSCSI login parameter value for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetMaxOutstandingR2T(
    /* in */    IMA_OID oid,
    /* in */    IMA_UINT maxOutstandingR2T
);
```

Parameters

oid

The object ID of the LHBA or target whose `MaxOutstandingR2T` value is being set.

maxOutstandingR2T

The new value for the `MaxOutstandingR2T` per task for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA or target does not support setting the `MaxOutstandingR2T` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `MaxOutstandingR2T` per task takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *maxOutstandingR2T* is out of range for the LHBA .

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

The valid range of `MaxOutstandingR2T` values can be determined by calling IMA_GetMaxOutstandingR2TProperties and examining the minimum and maximum values returned in the IMA_MIN_MAX_VALUE structure.

Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetMaxOutstandingR2TProperties for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetMaxOutstandingR2TProperties

iSCSI Session and Connection Parameters

6.2.114 IMA_SetMutualLocalAuth

Synopsis

Sets whether mutual authentication is performed for the specified authentication method for the specified object ID.

Prototype

```
IMA_STATUS IMA_SetMutualAuth (
    /* in */    IMA_OID oid,
    /* in */    IMA_AUTHMETHOD method,
    /* in */    IMA_BOOL mutualAuthEnabled
);
```

Parameters

oid

The object ID of the target, static target or discovery address whose mutual authentication behavior is being set.

method

The authentication method of the object ID for which mutual authentication behavior is being set.

mutualAuthEnabled

A boolean indicating whether mutual authentication will be performed for the specified object ID for the specified authentication method. If this parameter is set to IMA_TRUE the initiator will perform mutual authentication for the specified object ID for the specified authentication method.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the authentication parameters takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a target, static target or discovery address.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *o*id does not specify an object ID that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory if the *mutualAuthSupported* field of the IMA_LHBA_PROPERTIES returned by the IMA_GetLhbaProperties API is true for the LHBA associated with this target.

See Also

IMA_GetMutualLocalAuthParms

IMA_SetMutualLocalAuthParms

IMA_GetMutualLocalAuth

IMA_AddLhbaMutualAuthParms

IMA_RemoveLhbaMutualAuthParms

IMA_GetLhbaMutualAuthParmsList

Client Implementation Notes: Transmission of Authorization Parameters

6.2.115 IMA_SetMutualLocalAuthParms

Synopsis

Sets the parameters for the specified authentication method for the specified object ID to be used for mutual authentication.

Prototype

```
IMA_STATUS IMA_SetMutualAuthParms(
    /* in */    IMA_OID oid,
    /* in */    IMA_AUTHMETHOD method,
    /* in */    const IMA_TARGET_AUTHPARMS *pParms,
    /* in */    IMA_BOOL mutualAuthEnabled
);
```

Parameters

oid

The object ID of the target, static target or discovery address whose authentication parameters are to be set.

method

The authentication method of the object ID whose authentication parameters are to be set.

pParms

A pointer to an IMA_TARGET_AUTHPARMS structure that contains the parameters to associated with the specified authentication method.

mutualAuthEnabled

A boolean indicating whether mutual authentication will be performed for the specified object ID for the specified authentication method. If this parameter is set to IMA_TRUE the initiator will perform mutual authentication for the specified object ID for the specified authentication method.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the authentication parameters takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *method* does not specify a valid authentication method or does not specify a supported authentication method.

Returned if *method* has the value IMA_AUTHMETHOD_NONE.

Returned if *pParms* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a target, static target or discovery address.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *o*id does not specify an object ID that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory if the *mutualLocalAuthParmsSupported* field of the IMA_LHBA_PROPERTIES returned by the IMA_GetLhbaProperties API is true for the LHBA associated with this target.

See Also

IMA_GetMutualLocalAuthParms

IMA_GetMutualLocalAuth

IMA_SetMutualLocalAuth

Client Implementation Notes: Transmission of Authorization Parameters

6.2.116 IMA_SetNetworkPortalIpAddress

Synopsis

Sets the IP address of the specified network portal.

Prototype

```
IMA_STATUS IMA_SetNetworkPortalIpAddress(
    /* in */    IMA_OID networkPortalOid,
    /* in */    const IMA_IP_ADDRESS *pNewIpAddress
);
```

Parameters

networkPortalOid

The object ID of the network portal to set the IP address of.

pNewIpAddress

A pointer to an IMA_IP_ADDRESS structure allocated and initialized by the caller that contains the new IP address of the network portal specified by *networkPortalOid*.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the IP address actually takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if the specified network portal does not support having its IP address being set.

IMA_ERROR_INVALID_PARAMETER

Returned if *pNewIpAddress* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *networkPortalOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *networkPortalOid* does not specify a network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *networkPortalOid* does not specify a network portal that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

As noted above a reboot may be required to cause the new IP address to be used. Also, as noted above, this API call may or may not fail if there is an IP address conflict on the network portal's fabric. If it fails is up to the implementer(s) of the plugin and the underlying software, firmware, and hardware.

Support

Optional

See Also

IMA_GetNetworkPortalProperties

6.2.117 IMA_SetNodeAlias

Synopsis

Sets the alias of the specified node.

Prototype

```
IMA_STATUS IMA_SetNodeAlias
    /* in */    IMA_OID nodeOid,
    /* in */    const IMA_NODE_ALIAS newAlias
);
```

Parameters

nodeOid

The object ID of the node whose alias is being set.

newAlias

A pointer to a Unicode string that contains the new node alias. If this parameter is NULL then the current alias is deleted, in which case the specified node no longer has an alias.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

A reboot is necessary before the setting of the alias actually takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *newAlias* is NULL or specifies a memory area from which data cannot be read.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *nodeOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *nodeOid* does not specify a node object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *nodeOid* does not specify a node that is currently known to the system.

IMA_ERROR_NAME_TOO_LONG

Returned if *newAlias* contains more than 255 bytes when encoded in UTF-8.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

iSCSI node aliases are transferred to initiators and targets in UTF-8 format. A client of this API does not need to deal with UTF-8 strings, however underneath the covers the alias will be encoded as a UTF-8 string.

If this call returns a status of IMA_STATUS_REBOOT_NECESSARY then any subsequent calls made to IMA_GetNodeProperties for the same node may return IMA_STATUS_INCONSISTENT_NODE_PROPERTIES when, in fact, there is no

inconsistency. Rebooting the system, as indicated by the IMA_STATUS_REBOOT_NECESSARY status, will rectify the situation.

Support

Mandatory if *nameAndAliasSettable* field of the IMA_NODE_PROPERTIES structure returned for the same node has the value IMA_TRUE.

See Also

IMA_GetSharedNodeOid

IMA_GetNodeProperties

IMA_SetNodeName

6.2.118 IMA_SetNodeName

Synopsis

Sets the name of the specified node.

Prototype

```
IMA_STATUS IMA_SetNodeName(
    /* in */    IMA_OID nodeOid,
    /* in */    const IMA_NODE_NAME newName
);
```

Parameters

nodeOid

The object ID of the node whose name is being set.

newName

The new name of the specified node.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the name actually takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *newName* is NULL, or specifies a memory area from which data cannot be read, or has a length of 0.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *nodeOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *nodeOid* does not specify a node object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *nodeOid* does not specify a node that is currently known to the system.

IMA_ERROR_NAME_TOO_LONG

Returned if *newName* contains more than 223 bytes when encoded in UTF-8.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

There are several important points to note regarding this API:

- iSCSI node names are transferred to initiators and targets in UTF-8 format. A client of this API does not need to deal with UTF-8 strings, this is done internally by the IMA library or plugins.

- This API does not examine the iSCSI network to ensure that the new name is not already in use. It is the client's responsibility to do that *before* calling this API.
- If this call returns a status of IMA_STATUS_REBOOT_NECESSARY then any subsequent calls made to IMA_GetNodeProperties for the same node may return IMA_STATUS_INCONSISTENT_NODE_PROPERTIES when, in fact, there is no inconsistency. Rebooting the system, as indicated by the IMA_STATUS_REBOOT_NECESSARY status, will rectify the situation.

Support

Mandatory if *nameAndAliasSettable* field of the IMA_NODE_PROPERTIES structure returned for the same node has the value IMA_TRUE.

See Also

IMA_GetSharedNodeOid

IMA_GetNonSharedNodeOidList

IMA_GetNodeProperties

IMA_SetNodeAlias

Example of Setting a Node Name

6.2.119 IMA_SetOFMarkerProperties

Synopsis

Sets the `OFMarker` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetOFMarkerProperties(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL ofMarker
);
```

Parameters

oid

The object ID of the LHBA or target whose `OFMarker` value is being set.

ofMarker

The new value for `OFMarker` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `OFMarker` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `OFMarker` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *ofMarker* does not contain the value IMA_TRUE or IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory for both LHBAs and targets

See Also

6.2.120 IMA_SetOFMarkIntProperties

Synopsis

Sets the `OFMarkInt` iSCSI connection parameter properties for the specified logical HBA or target.

Prototype

```
IMA_STATUS IMA_SetOFMarkIntProperties(
    /* in */    IMA_OID oid,
    /* in */    IMA_MARKER_INT ofMarkInt
);
```

Parameters

oid

The object ID of the LHBA or target whose `OFMarkInt` value is being set.

ifMarkInt

The new value for `OFMarkInt` for the specified LHBA or target.

Typical Return Values

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA does not support setting the `OFMarkInt` value.

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of `OFMarkInt` takes affect.

IMA_ERROR_INVALID_PARAMETER

Returned if *ofMarkInt* contains a value outside of the range of the LHBA or target.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a LHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a LHBA or target that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

Support

Mandatory for both LHBAs and targets

See Also

6.2.121 IMA_SetRadiusAccess

Synopsis

Configures RADIUS access for the specified LHBA, PHBA or network portal.

Prototype

```
IMA_STATUS IMA_SetRadiusAccess(
    /* in */    IMA_OID oid,
    /* in */    IMA_RADIUS_PROPERTIES *radiusProps
);
```

Parameters

oid

The object ID of the LHBA, PHBA or network portal whose RADIUS access properties are being set.

radiusProps

A pointer to an IMA_RADIUS_PROPERTIES structure allocated by the caller containing the values which are to be set.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before this call will take effect.

IMA_ERROR_NOT_SUPPORTED

Returned if the enabling/disabling of RADIUS server access is unsupported by the LHBA, PHBA or network portal.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA, LHBA, or network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA, LHBA, or network portal that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Optional

See Also

IMA_GetRadiusAccess

6.2.122 IMA_SetSendTargetsDiscovery

Synopsis

Enables/disables `SendTargets` target discovery for a physical HBA, logical HBA, network portal, or target.

Prototype

```
IMA_STATUS IMA_SetSendTargetsDiscovery(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL enableSendTargetsDiscovery
);
```

Parameters

oid

The object ID of the PHBA, LHBA, network portal, or target whose `SendTargets` target discovery property is being set.

enableSendTargetsDiscovery

Set to IMA_TRUE if `SendTargets` discovery of targets is being enabled for the specified PHBA, LHBA, network portal, or target. Set to IMA_FALSE if `SendTargets` discovery of targets is being disabled.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before this call will take effect.

IMA_ERROR_NOT_SUPPORTED

Returned if enabling/disabling `SendTargets` discovery is not supported by the specified PHBA, LHBA, network portal, or target.

IMA_ERROR_INVALID_PARAMETER

Returned if *enableSendTargetsDiscovery* has a value other than IMA_TRUE and IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA, LHBA, network portal, or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA, LHBA, network portal, or target that is currently known to the system.

Remarks

The setting of this value is persistent.

Disabling `SendTargets` discovery does not “undiscover” any targets that were previously discovered using `SendTargets` discovery.

Support

For a PHBA or LHBA mandatory if the *sendTargetsDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by the IMA_GetDiscoveryProperties API has the value IMA_TRUE for the same *oid*.

For a target mandatory if the *sendTargetsDiscoverySettable* field in the IMA_TARGET_PROPERTIES structure returned by the IMA_GetTargetProperties API has the value IMA_TRUE for the same *oid*.

See Also

IMA_GetDiscoveryProperties

IMA_SetIsnsDiscovery

IMA_SetSlpDiscovery

6.2.123 IMA_SetSlpDiscovery

Synopsis

Enables/disables SLP target discovery for a physical HBA, logical HBA, or network portal.

Prototype

```
IMA_STATUS IMA_SetSlpDiscovery(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL enableSlpDiscovery
);
```

Parameters

phbaOid

The object ID of the PHBA, LHBA, or network portal whose SLP target discovery property is being set.

enableSlpDiscovery

Set to IMA_TRUE if SLP discovery of targets is being enabled for the specified PHBA or LHBA. Set to IMA_FALSE if SLP discovery of targets is being disabled.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before this change will take effect.

IMA_ERROR_NOT_SUPPORTED

Returned if enabling/disabling SLP discovery is not supported by the specified PHBA, LHBA, or network portal.

IMA_ERROR_INVALID_PARAMETER

Returned if *enableSlpDiscovery* has a value other than IMA_TRUE and IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA, LHBA, or network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA, LHBA, or network portal that is currently known to the system.

IMA_ERROR_LAST_PRIMARY_DISCOVERY_METHOD

Returned if *enableSlpDiscovery* is set to IMA_FALSE and SLP target discovery is the last primary discovery method for the PHBA/LHBA/network portal, i.e., SNS target discovery and static target discovery are both either disabled or not supported.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Disabling SLP discovery does not “undiscover” any targets that were previously discovered using SLP discovery.

Support

Mandatory if the *slpDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by IMA_GetDiscoveryProperties is set to IMA_TRUE for the same *oid*.

See Also

IMA_GetDiscoveryProperties

IMA_SetIsnsDiscovery

IMA_SetStaticDiscovery

IMA_SetSendTargetsDiscovery

6.2.124 IMA_SetStaticDiscovery

Synopsis

Enables/disables static target discovery for a physical HBA, logical HBA, or network portal.

Prototype

```
IMA_STATUS IMA_SetStaticDiscovery(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL enableStaticDiscovery
);
```

Parameters

oid

The object ID of the PHBA, LHBA, or network portal whose static target discovery property is being set.

enableStaticDiscovery

Set to IMA_TRUE if static target discovery is being enabled for the specified PHBA or LHBA. Set to IMA_FALSE if static target discovery is being disabled.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before this call will take effect.

IMA_ERROR_NOT_SUPPORTED

Returned if enabling/disabling static discovery is not supported by the specified PHBA or LHBA.

IMA_ERROR_INVALID_PARAMETER

Returned if *enableStaticDiscovery* has a value other than IMA_TRUE and IMA_FALSE.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA, LHBA, or network portal object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA, LHBA, or network portal that is currently known to the system.

IMA_ERROR_LAST_PRIMARY_DISCOVERY_METHOD

Returned if *enableStaticDiscovery* is set to IMA_FALSE and static target discovery is the last primary discovery method for a PHBA, LHBA, or network portal i.e., iSNS target discovery and SLP target discovery are both either disabled or not supported.

Remarks

This setting is persistent.

Disabling static discovery does not “undiscover” any targets that were previously discovered using static discovery. Such targets can be “undiscovered” by calling the IMA_RemoveStaticDiscoveryTarget API.

Support

Mandatory if the *staticDiscoverySettable* field in the IMA_DISCOVERY_PROPERTIES structure returned by IMA_GetDiscoveryProperties is set to IMA_TRUE for the same *oid*.

See Also

IMA_GetDiscoveryProperties

IMA_SetIsnsDiscovery

IMA_SetSlpDiscovery

IMA_SetSendTargetsDiscovery

6.2.125 IMA_SetStatisticsCollection

Synopsis

Enables/disables statistics collection for a physical HBA's physical network ports, or a target, or a target's LUs.

Prototype

```
IMA_STATUS IMA_SetStatisticsCollection(
    /* in */    IMA_OID oid,
    /* in */    IMA_BOOL enableStatisticsCollection
);
```

Parameters

oid

The object ID of the PHBA, target, or LU whose statistics collection setting is being modified.

enableStatisticsCollection

Set to IMA_TRUE if statistics collection is being enabled for the specified object. Set to IMA_FALSE if statistics collection is being disabled for the specified object.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of statistics collection takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if *oid* does not support enabling/disabling of statistics collection.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PHBA or target object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PHBA or target that is currently known to the system.

Remarks

If *oid* specifies a PHBA then this call affects the statistics collection of the PNPs of the PHBA.

If *oid* specifies a target then this call affects the statistics collection of the target and/or the target's LUs. A plugin shall support the collection of either target statistics or LU statistics and may support both.

The setting of this value is persistent.

Support

Mandatory if the *statisticsCollectionSettable* field of the IMA_STATISTICS_PROPERTIES structure as returned by IMA_GetStatisticsProperties for the same oid has the value IMA_TRUE.

See Also

IMA_GetDeviceStatistics

IMA_GetStatisticsProperties

IMA_GetTargetErrorStatistics

6.2.126 IMA_SetSubnetMask

Synopsis

Sets the subnet mask for the specified physical network port.

Prototype

```
IMA_STATUS IMA_SetSubnetMask(
    /* in */    IMA_OID oid,
    /* in */    IMA_IP_ADDRESS subnetMask
);
```

Parameters

oid

The object ID of the PNP whose subnet mask is to be set.

subnetMask

The new subnet mask for the PNP.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the setting of the subnet mask takes affect.

IMA_ERROR_NOT_SUPPORTED

Returned if setting the subnet mask is not supported by the specified PNP.

IMA_ERROR_INVALID_PARAMETER

Returned if any fields in *subnetMask* contain any invalid values.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *oid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *oid* does not specify a PNP.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *oid* does not specify a PNP that is currently known to the system.

Remarks

The persistence of this value is dependent upon the autoPersistenceSupported setting on the Plugin.

Support

Mandatory if the *subnetMaskSettable* field in the IMA_IP_PROPERTIES structure returned by the IMA_GetIpProperties API for the same *oid* has the value IMA_TRUE.

See Also

IMA_GetIpProperties

6.2.127 IMA_UnexposeLu

Synopsis

Unexposes the specified logical unit from the operating system.

Prototype

```
IMA_STATUS IMA_UnexposeLu(
    /* in */        IMA_OID luOid
);
```

Parameters

luOid

The object ID of the LU to unexpose from the operating system.

Typical Return Values

IMA_STATUS_REBOOT_NECESSARY

Returned if a reboot is necessary before the LU is unexposed to the OS.

IMA_ERROR_NOT_SUPPORTED

Returned if the LHBA associated with the LU does not support selective exposing/unexposing of logical units.

IMA_ERROR_INVALID_OBJECT_TYPE

Returned if *luOid* does not specify any valid object type.

IMA_ERROR_INCORRECT_OBJECT_TYPE

Returned if *luOid* does not specify a LU object.

IMA_ERROR_OBJECT_NOT_FOUND

Returned if *luOid* does not specify a LU that is currently known to the system.

IMA_ERROR_LU_NOT_EXPOSED

Returned if *luOid* specifies a LU that is not exposed to the operating system.

IMA_ERROR_LU_IN_USE

Returned if *luOid* specifies a LU that is currently in use by the operating system.

Remarks

The unexposing of the LU is persistent, i.e., the LU will not be exposed to the operating system until a successful IMA_ExposeLu for the same LU is made.

It is the client's responsibility to ensure that the LU is not in use, e.g. no part of the LU is part of a mounted volume, before calling this API.

Support

Mandatory if the *luExposingSupported* field of the IMA_LHBA_PROPERTIES structure returned by the IMA_GetLhbaProperties API for the LHBA that is associated with the LU has the value IMA_TRUE.

See Also

IMA_ExposeLu

IMA_GetLuId

IMA_GetLuOidList

7 Implementation Compliance

An implementation of the API described in this document shall meet the following requirements:

1. Provide an entry point for each API listed in this document.
2. Implement all APIs which are listed as mandatory to implement.
3. Attempt to perform or cause the performance of all of the actions that are specified for an API when all parameters to that API are valid.
4. Fail an API call if the implementation is aware that one of the requirements specified for that API cannot be satisfied.

It's important to note that what is compliant with this specification is not simply an implementation of the library, but an implementation of the library in combination with an implementation of a plugin.

8 Notes

8.1 Client Usage Notes

Persisted Object IDs

Because the library does not make any claims as to the persistence of object sequence numbers, a client using the library shall not use persisted object identifiers across instances of itself.

Reserved Fields

Most structures in the API contain reserved fields. Clients shall ignore the values in any reserved fields in any structures.

Event Notification Within a Single Client

The method used to deliver events within a client is specific to the library and/or plugin implementation. Therefore, when a client receives an event it shall not use the thread delivering the event for any significant amount of time. If the work needed to respond to an event is at all significant the client should somehow save the information needed to respond to the event and then have another thread perform the actual work to handle the event. If a client fails to do this it may delay the delivery of subsequent events and it may even cause events to be lost. The method a client uses to save the data of an event and causes another thread to respond to the event is entirely client specific.

Event Notification and Multi-Threading

A client that uses the event notification APIs of the library shall also be multi-thread safe. A client cannot assume that an event is delivered on the same thread that registered for the event, nor can a client assume that the client created the thread used to deliver the event. The only thing a client can assume about a thread used to deliver an event is that it was properly initialized to use the C runtime library.

IPsec Security

The API provides a function that allows a client to query the IPsec capabilities of a physical HBA (PHBA). The returned data allows a client to determine if IPsec is supported by a PHBA and what IPsec options the PHBA supports. However, the API does not provide any function to set IPsec security policies. Any client that wants to add, change or delete IPsec security policies shall use mechanisms provided by the operating system to perform these operations.

Transmission of Authorization Parameters

If a client is acting as a proxy for an application on another system and the client and the application transmit authorization parameters that the transmission be encrypted.

Target OIDs and iSCSI Targets

A single iSCSI target will have one target OID for each LHBA from which the target can be accessed. It is up to a client to determine, through whatever means it chooses, that different target OIDs refer to the same iSCSI target.

Configuration Changes and the IMA_STATUS_REBOOT_NECESSARY status

Any API that causes a configuration change can return the status IMA_STATUS_REBOOT_NECESSARY. In this case, if a client retrieves values that have been previously set the client will continue to receive values as if the values had not

been set. If a client wishes to present values to a user that indicate the future value of a setting then the client shall cache the values that it set.

For example, suppose a client retrieves the `MaxBurstLength` setting of a target and it has a value of 128K. Now, suppose the client sets the `MaxBurstLength` setting of that target to 196K and then retrieves the value of the setting again. The value that is retrieved will be the original value 128K, not the new value 196K.

8.2 Library Implementation Notes

Object IDs

Object sequence numbers shall be reused in a conservative fashion to minimize the possibility that an object ID will ever refer to two (or more) different objects in any once instance of the library. This rule for reuse only applies to a particular instance of the library. The library is not required nor expected to persist object sequence numbers across instances of itself.

Multi-threading Support

Any implementation of this API, i.e. the library, shall be multi-thread safe. That is, the library shall allow a client to safely have multiple threads calling APIs in the library simultaneously. It is the responsibility of the library to synchronize the usage of any library resources being used by different threads.

Event Notification and Multi-Threading

A client shall be able to call any API while the client is handling an event. Therefore, the library implementation shall not leave any resources locked while calling a client's event handler that would be needed if the client's event handler called an API. Otherwise, if the client's event handler did call an API, either the API would have to fail or the calling thread would deadlock waiting for a resource.

Structure Packing

In order to ensure binary compatibility between different implementations of the IMA it is necessary that each implementation provide header files and/or document compiler options so that each structure is packed such that there are no padding bytes between structure members.

Calling Conventions

In order to ensure binary compatibility between different implementations of the IMA it is necessary that each implementation provide header files and/or document compiler options so that all APIs in the IMA are called using the C calling convention.

Authentication

The IMA specification provides several different call types for handling of initiator and target authentication.

There are a few key terms that are required for description of authentication:

Global: applies to all targets associated with an LHBA.

Local: Applies to only a particular target, static target or discovery address.

One-way authentication: authentication of the initiator by the target.

Mutual authentication: authentication of the target by the initiator.A list of authentication calls and a brief description is provided here:

IMA_GetInUseInitiatorAuthMethods

Gets which authentication methods are currently in use by an LHBA.

IMA_SetInitiatorAuthMethods

Sets which authentication methods are in use for an LHBA.

IMA_GetSupportedAuthMethods

Gets which authentication methods are supported by an LHBA.

IMA_GetInitiatorAuthParms

Gets the global one-way authentication parameters for an LHBA.

IMA_SetInitiatorAuthParms

Sets the global one-way authentication parameters for an LHBA.

IMA_GetInitiatorLocalAuthParms

Gets the local one-way authentication parameters for an LHBA.

IMA_SetInitiatorLocalAuthParms

Sets the local one-way authentication parameters for an LHBA.

IMA_GetMutualLocalAuth

Gets the setting for mutual authentication for an LHBA and target.

IMA_SetMutualLocalAuth

Sets whether mutual authentication is performed for an LHBA and target.

IMA_GetMutualLocalAuthParms

Gets the local mutual authentication parameters for the specified target, static target or discovery address.

IMA_SetMutualLocalAuthParms

Sets the local mutual authentication parameters for the specified target, static target or discovery address.

IMA_GetLhbaMutualAuthParmsList

Gets the a list of mutual authentication parameters for a specified LHBA.

IMA_AddLhbaMutualAuthParms

Adds a mutual authentication parameter to the list for a specified LHBA

IMA_RemovetLhbaMutualAuthParms

Removes a mutual authentication parameter from the list for a specified LHBA

These calls are meant to handle several different requirements:

1. Authentication types such as CHAP, SRP, SPKM or KRB5:

- IMA_GetInUseInitiatorAuthMethods
- IMA_SetInitiatorAuthMethods
- IMA_GetSupportedAuthMethods

2. A global set of authentication parameters that applies to all targets:

- IMA_GetInitiatorAuthParms
- IMA_SetInitiatorAuthParms
- IMA_GetLhbaMutualAuthParmsList
- IMA_AddLhbaMutualAuthParms
- IMA_RemoveLhbaMutualAuthParms

3. A target specific settting for authentication of the initiator by the target (one-way authentication):

- IMA_GetInitiatorLocalAuthParms
- IMA_SetInitiatorLocalAuthParms

If a target specific setting is provided it shall override any global authentication setting.

4. A target specific setting for authentication of the target by the initiator (mutual authentication):

- IMA_GetMutualLocalAuthParms
- IMA_SetMutualLocalAuthParms

8.3 Plugin Implementation Notes

Object IDs

Object sequence numbers shall be reused in a conservative fashion to minimize the possibility that an object ID will ever refer to two or more different objects in any one instance of a plugin. This rule for reuse only applies to a particular instance of the plugin. A plugin is not required nor expected to persist object sequence numbers across instances of itself.

Reserved Fields

Most structures in the API contain reserved fields. Plugins shall zero out any fields that they consider reserved.

Multi-threading Support

Plugins shall also be multi-thread safe. A client shall be able to have multiple threads active at anyone time. It is the responsibility of the plugin to synchronize the usage of plugin resources being used by different threads.

Event Notification To Different Clients

Timely delivery of events to clients is necessary. Therefore, vendor implementations shall not, in any way, serialize delivery of events to plugins. It is not permissible for a vendor implementation to allow one client to significantly delay delivery of events to any other client.

Event Notification and Multi-Threading

A client shall be able to call any API while the client is handling an event. Therefore, a plugin shall not leave any resources locked while calling a client's event handler that would be needed if the client's event handler called an API. Otherwise, if the client's event handler did call an API, either the API would have to fail or the calling thread would deadlock waiting for a resource.

IPsec Security

If a plugin reports that any of its physical HBAs support IPsec it is the responsibility of the vendor providing the plugin to monitor the operating system's IPsec security policy database to get policy information. This shall be done outside of the plugin because the plugin may not be loaded when the IPsec security policy database is updated. How a vendor does this is entirely upto the vendor.

Persistence of Authorization Parameters

A plugin, or some entity below the plugin, shall persist authorization parameters. It is strongly recommended that the entity persisting the authorization parameters secure them from unauthorized access. This can be accomplished by setting the appropriate permissions on files or other storage objects. This can also be accomplished by encrypting the persisted data and then decrypting the data when retrieving it. Of course, these two techniques can be combined for maximum security of the authorization parameters.

Executing SCSI Commands and Operating System Compatibility

There are three APIs that can cause the execution of SCSI commands. These APIs are:

- IMA_LuInquiry
- IMA_LuReadCapacity
- IMA_LuReportLuns

If the calling of one of these APIs results in a SCSI command being sent to a logical unit that is exposed to the operating system, it is the responsibility of the plugin to ensure that executing the command does not interfere in any way with the operating system's use of the logical unit. In particular, important statuses, such as UNIT ATTENTION, shall be conveyed to the operating system. Typically, this is accomplished by using operating system interfaces to send the commands to the logical unit when that logical unit is exposed to the operating system. When the logical unit is not exposed to the operating system the plugin shall use a private interface to the device driver.

Executing SCSI Commands and Session Management

How a plugin or lower level software manages sessions to execute SCSI commands as a result of IMA API calls is entirely up to that implementation. If a plugin determines that it needs to send a SCSI command to a logical unit as a result of an API call and no session has been created with the logical unit's associated target then a session shall be created between the initiator and target. Once the command has been executed it is entirely up to the implementation if the session remains or is closed.

Plugin IOCtls

Plugins shall not define IOCtls that allow a client to send a SCSI command to a LU using the IMA_PluginIOCtl API. The only SCSI commands that may be sent to a LU are the ones specifically supported by the library.

Target OIDs and Logical Unit OIDs

If a plugin is providing support for two or more LHBAs and the same iSCSI target or iSCSI LUs are exposed via two or more LHBAs it shall generate different (unique) object IDs for each target or LU per each LHBA through which that the target or LU is accessible.

Annex A (informative) Device Names

This appendix contains information on how to specify the *osDeviceName* field in the IMA_LHBA_PROPERTIES and the IMA_LU_PROPERTIES structures. Whenever possible the values used in the fields of these structures are identical to the values used in similar structures in ANSI INCITS 386-2004 (FC-HBA API).

In the tables below text appearing in bold shall appear in the indicated position exactly as it appears in the sample. Text appearing in italics is a placeholder for other text as determined by the specified operating system.

A.1 osDeviceName Field of the IMA_LHBA_PROPERTIES Structure

Operating System	Value
AIX	**/dev/iscsi***n*
Linux	**/dev/***name*
Solaris	**/devices/***name*
Windows	**\\.\Scsi***n***:**

A.2 osDeviceName Field of the IMA_LU_PROPERTIES Structure

Operating System	Value			
	Disk/Optical	CD-ROM	Tape	Changer
AIX	**/dev/hdisk***n* (disk) or **/dev/omd***n* (optical)	**/dev/cd***n*	**/dev/rmt***n*	Empty string
Linux	**/dev/sd***n*	**/dev/sr***n*	**/dev/st/***n*	Empty string
Solaris	**/dev/rdsk/c***x***t***y***d***z***s2**	**/dev/rdsk/c***x***t***y***d***z***s2**	**/dev/rmt/***n***n**	Empty string
Windows	**\\.\PHYSICALDRIVE***n*	**\\.\CDROM***n*	**\\.\TAPE***n*	**\\.\CHANGER***n*

Annex B (informative) Coding Examples

This appendix contains samples of how to use the IMA. All of these examples are informative; if there is a discrepancy between these examples and anything in any of the previous sections of this document the examples should be considered incorrect and the previous sections correct.

One note about the examples: the examples will all perform error detection, however they will not perform error reporting. This is an exercise left to the reader.

There are ten coding examples. They are:

- Example of Getting Library Properties
- Example of Getting Plugin Properties
- Example of Getting an Associated Plugin ID
- Example of Getting Node Properties
- Example of Setting a Node Name
- Example of Getting LHBA Properties
- Example of Getting PHBA Properties
- Example of Getting PHBA Discovery Properties
- Example of Getting/Setting LHBA Max Burst Length
- Example of Getting all LUs of all Targets Visible to a System

B.1 Example of Getting Library Properties

```
//
// This example prints the properties of the IMA library.
//
IMA_STATUS status;
IMA_LIBRARY_PROPERTIES props;


//
// Try to get the library properties.  If this succeeds then print
// the properties.
//
status = IMA_GetLibraryProperties(&props);
if ( IMA_SUCCESS(status) )
{
   printf("Library Properties:\n");
   printf("\tIMA version: %u\n", props.supportedImaVersion);
   wprintf(L"\tVendor: %ls\n", props.vendor);
   wprintf(L"\tImplementation version: %ls\n",
                           props.implementationVersion);
   wprintf(L"\tFile name: %ls\n", props.fileName);
   printf("\tBuild date/time: %s\n", DateTime(&props.buildTime));
}
```

B.2 Example of Getting Plugin Properties

```
//
// This example gets the properties of the first plugin returned by
// the library.
//
IMA_STATUS status;
IMA_OID_LIST *pList;


//
// Get the list of plugin IDs.
//
status = IMA_GetPluginOidList(&pList);
if ( IMA_SUCCESS(status) )
{
   //
   // Make sure there's a plugin to get the properties of.
   //
   if ( pList->oidCount != 0 )
   {
      IMA_PLUGIN_PROPERTIES props;
      status = IMA_GetPluginProperties(pList->oids[0], &props);
      if ( IMA_SUCCESS(status) )
      {
         printf("Plugin Properties:\n");
         printf("\tIMA version: %u\n", props.supportedImaVersion);
         wprintf(L"\tVendor: %ls\n", props.vendor);
         wprintf(L"\tImplementation version: %ls\n",
                           props.implementationVersion);
         wprintf(L"\tFile name: %ls\n", props.fileName);
         printf("\tBuild date/time: %s\n", DateTime(&props.buildTime))
      }
   }


   //
   // Always remember to free an object ID list when it's no longer
   // needed.  Failing to do so will cause memory leaks.
   //
   IMA_FreeMemory(pList);
}
```

B.3 Example of Getting an Associated Plugin ID

```
//
// This example prints the name of the vendor associated with the
// first non-shared node that's in use.
//
IMA_STATUS status;
IMA_OID_LIST *pList;


//
// Get a list of the object IDs of all of the non-shared nodes in the
// system.
//
status = IMA_GetNonSharedNodeOidList(&pList);
if ( IMA_SUCCESS(status) )
{
   //
   // Make sure there's a node to get the associated plugin of.
   //
   if ( pList->oidCount > 0 )
   {
      IMA_OID pluginId;
      status = IMA_GetAssociatedPluginOid(pList->oids[0], &pluginId);
      if ( IMA_SUCCESS(status) )
      {
         IMA_PLUGIN_PROPERTIES pluginProps;
         status = IMA_GetPluginProperties(pluginId, &pluginProps);
         if ( IMA_SUCCESS(status) )
         {
            wprintf(L"Vendor: %ls\n", pluginProps.vendor);
         }
      }
   }


   //
   // Always remember to free an object ID list when it's no longer
   // needed.  Failing to do so will cause memory leaks.
   //
   IMA_FreeMemory(pList);
}
```

B.4 Example of Getting Node Properties

```
//
// This example prints the properties of the shared node.
//
IMA_STATUS status;
IMA_OID nodeOid;


//
// Get the object ID of the shared node.
//
status = IMA_GetSharedNodeOid(&nodeOid);
if ( IMA_SUCCESS(status)) )
{
   //
   // Get the properties of the shared node.  If this succeeds then
   // print the properties.
   //
   IMA_NODE_PROPERTIES props;
   status = IMA_GetNodeProperties(nodeOid, &props);
   if ( IMA_SUCCESS(status)) )
   {
      printf("Shared node properties:\n");

      if ( props.nodeNameSet == IMA_TRUE )
      {
         wprintf("\tNode Name: \"%ls\"\n", props.name);
      }
      else
      {
         printf("\tName: Not set\n");
      }

      if ( props.nodeAliasSet == IMA_TRUE )
      {

         wprintf(L"\tNode Alias: \"%ls\"\n", props.alias);
      }
      else
      {
         printf("\tAlias: Not set\n");
      }

      printf("\tInitiator: %s\n", YesNo(props.runningInInitiatorMode));
      printf("\tTarget: %s\n", YesNo(props.runningInInitiatorMode));
   }
}
```

B.5 Example of Setting a Node Name

```
//
// This example sets the name of the shared node to an IMA generated
// node name.
//
IMA_STATUS status;
IMA_OID nodeOid;


//
// Get the object ID of the shared node.
//
status = IMA_GetSharedNodeOid(&nodeOid);
if ( IMA_SUCCESS(status) )
{
   IMA_NODE_NAME nodeName;


   //
   // Generate a node name.
   //
   status = IMA_GenerateNodeName(nodeName);
   if ( IMA_SUCCESS(status) )
   {
      //
      // Set the node name to the generated name.  If this succeeds
      // then print a message to this effect.
      //
      status = IMA_SetNodeName(nodeOid, nodeName);
      if ( IMA_SUCCESS(status)) )
      {
         wprintf(L“The name of the node was set to \”%ls\”.\n”,
      nodeName);


         //
         // Check if a reboot is necessary to make the new node name
         // take affect.  If it is then print a message to this effect.
         //
         if ( status == IMA_STATUS_REBOOT_NECESSARY )
         {
            printf(“Reboot to make this change take affect.\n”);
         }
      }
   }
}
```

B.6 Example of Getting LHBA Properties

```
//
// This example prints out the OS device name for each of the LHBA's
// that are currently in the system.
//
IMA_STATUS status;
IMA_OID_LIST *pList;


//
// First, get the list of LHBA IDs.
//
status = IMA_GetLhbaOidList(&pList);
if ( IMA_SUCCESS(status) )
{
   //
   // Either print a header or print that there are no LHBAs
   // in the system.
   //
   if ( pList->oidCount == 0 )
   {
      printf("There are no logical HBAs in the system.\n");
   }
   else
   {
      printf("Logical HBA OS Device Names:\n");
   }


   //
   // Next,iterate through the list, getting the properties of each
   // one.  Print the OS device name from the properties structure.
   //
   for ( unsigned i = 0;  i < pList->oidCount;  i++ )
   {
      IMA_LHBA_PROPERTIES props;

      status = IMA_GetLhbaProperties(pList->oids[i], &props);
      if ( IMA_SUCCESS(status) )
      {
         wprintf("\t%ls\n", props.osDeviceName);
      }
   }


   //
   // Always remember to free an object ID list when it's no longer
   // needed.  Failing to do so will cause memory leaks.
   //
   IMA_FreeMemory(pList);
}
```

B.7 Example of Getting PHBA Properties

```
//
// This example prints out the description for each of the PHBA's
// that are currently in the system.
//
IMA_STATUS status;
IMA_OID_LIST *pList;


//
// First, get the list of PHBA IDs.
//
status = IMA_GetPhbaOidList(&pList);
if ( IMA_SUCCESS(status) )
{
   //
   // Either print a header or print that there are no PHBAs
   // in the system.
   //
   if ( pList->oidCount == 0 )
   {
      printf("There are no physical HBAs in the system.\n");
   }
   else
   {
      printf("Physical HBA Descriptions:\n");
   }


   //
   // Next, iterate through the list, getting the properties of each
   // one.  Print the description from the properties structure.
   //
   for ( unsigned i = 0;  i < pList->oidCount;  i++ )
   {
      IMA_PHBA_PROPERTIES props;

      status = IMA_GetPhbaProperties(pList->oids[i], &props);
      if ( IMA_SUCCESS(status) )
      {
         wprintf("\t%u. %ls\n", i + 1, props.description);
      }
   }


   //
   // Always remember to free an object ID list when it's no longer
   // needed.  Failing to do so will cause memory leaks.
   //
   IMA_FreeMemory(pList);
}
```

B.8 Example of Getting PHBA Discovery Properties

```
//
// This example prints if each of the four types of discovery methods
// is enabled or disabled.
//
IMA_STATUS status;
IMA_OID_LIST *pList;


//
// Get the list of object IDs of the PHBAs in the system.
//
status = IMA_GetPhbaOidList(&pList);
if ( IMA_SUCCESS(status) )
{
   //
   // Iterate through the list of PHBAs.
   //
   for ( unsigned i = 0;  i < pList->oidCount;  i++ )
   {
      //
      // Get the discovery properties of the “current” PHBA.
      //
      IMA_DISCOVERY_PROPERTIES props;
      printf(“PHBA %u\n”, i + 1);

      status = IMA_GetDiscoveryProperties(pList->oids[i], &props);


      //
      // If the discovery properties have been successfully gotten
      // then print them.
      //
      if ( IMA_SUCCESS(status) )
      {
         printf(“\tiSNS discovery enabled: %s\n”,
                           xbool(props.iSnsDiscoveryEnabled));
         printf(“\tSLP discovery enabled: %s\n”,
                           xbool(props.slpDiscoveryEnabled));
         printf(“\tStatic discovery enabled: %s\n”,
                           xbool(props.staticDiscoveryEnabled));
         printf(“\tSendTargets discovery enabled: %s\n\n”,
                           xbool(props.sendTargetsDiscoveryEnabled));
      }
   }


   //
   // Always remember to free an object ID list when it’s no longer
   // needed.  Failing to do so will cause memory leaks.
   //
   IMA_FreeMemory(pList);
}
```

B.9 Example of Getting/Setting LHBA Max Burst Length

```
//
// This example sets the maximum burst length of the first LHBA in a
// system to the maximum allowable value.
//
IMA_STATUS status;
IMA_OID_LIST *pList;


//
// Get the list of LHBA IDs.
//
status = IMA_GetLhbaOidList(&pList);
if ( IMA_SUCCESS(status) )
{
   //
   // Make sure there is at least one LHBA.
   //
   if ( pList->oidCount > 0 )
   {
      IMA_MIN_MAX_VALUE props;
      IMA_OID lhbaOid  = pList->oids[0];


      //
      // Get the current max burst length for the first LHBA.
      //
      status = IMA_GetMaxBurstLengthProperties(lhbaOid, &props);
      if (( IMA_SUCCESS(status) ) && ( props.currentValueValid ))
      {
         //
         // Set the maximum burst length to the maximum value.
         //
         if ( props.currentValue != props.maximumValue )
         {
            status = IMA_SetMaxBurstLength(lhbaOid,
                                           props.maximumValue);
         }
      }
   }


   //
   // Always remember to free an object ID list when it's no longer
   // needed.  Failing to do so will cause memory leaks.
   //
   IMA_FreeMemory(pList);
}
```

B.10 Example of Getting all LUs of all Targets Visible to a System

```
//
// This example shows how to go through the LUs visible to a system
// to determine what LUs should be exposed to the OS that are not
// already.
//
// This example omits an important part of doing this in a real
// application.  That is that LU IDs associated with different LHBAs
// may in fact refer to the same LU.  This is an exercise left to the
// IMA developer.  :-)
//
// Also, certain supporting functions are also omitted from this
// example.
//


//
// This function processes a “system” by getting all of the LHBAs
// in the system and processing each of those LHBAs.
//
void ProcessSystem()
{

   IMA_OID_LIST *pList;


   //
   // Get the list of LHBA IDs.
   //
   IMA_STATUS status = IMA_GetLhbaOidList(&pList);
   if ( IMA_SUCCESS(status) )
   {
      //
      // Iterate through all of the LHBAs in the system processing
      // the targets of each LHBA.
      //
      for ( unsigned i = 0;  i < pList->oidCount;  i++ )
      {
         void ProcessLhba(pList->oids[i]);
      }


      //
      // Always remember to free an object ID list when it’s no longer
      // needed.  Failing to do so will cause memory leaks.
      //
      IMA_FreeMemory(pList);
   }
}
```

```
//
// This function processes an LHBA by getting the list of targets
// associated with the LHBA and processing each of those targets.
//
void ProcessLhba(IMA_OID lhbaOid)
{

   IMA_OID_LIST *pList;


   //
   // Get the list of target OIDs for this LHBA.
   //
   IMA_STATUS status = IMA_GetTargetOidList(lhba, &pList);
   if ( IMA_SUCCESS(status) )
   {
      for ( unsigned i = 0;  i < pList->oidCount;  i++ )
      {
         void ProcessTarget(pList->oids[i]);
      }


      //
      // Always remember to free an object ID list when it's no longer
      // needed.  Failing to do so will cause memory leaks.
      //
      IMA_FreeMemory(pList);
   }
}



//
// This function processes a target by getting all of the LUNs of the
// target and then processing each of the LUs associated with a LUN.
//
void ProcessTarget(IMA_OID targetOid)
{

   IMA_BYTE reportLunsBuf[8192];
   IMA_STATUS status = IMA_LuReportLuns(targetOid, IMA_FALSE, 0,
                                 reportLunsBuf, sizeof(reportLunsBuf),
                                 NULL, 0
                        );
   if ( IMA_SUCCESS(status) )
   {
      unsigned listLength = GetReportLunsListLength(reportLunsBuf);
      for ( unsigned i = 8;  i < listLength;  i += 8 )
      {
         IMA_UINT64 lun = GetLun(reportLunsBuf, i);
         ProcessLu(targetOid, lun);
      }
   }
}
```

```
//
// This function processes a logical unit by getting the LU's
// properties and possibly other data about the LU to determine
// if it should be exposed to the OS.
//
void ProcessLu(IMA_OID targetOid, IMA_BYTE lun[8])
{

   IMA_OID luOid;
   IMA_STATUS status = IMA_GetLuOid(targetOid, lun, &luOid);
   if ( IMA_SUCCESS(status) )
   {
      IMA_LU_PROPERTIES props;
      status = IMA_GetLuProperties(luOid, &props);
      if ( IMA_SUCCESS(status) )
      {
         if ( props.exposedToOs == IMA_FALSE )
         {
            //
            // Here the application should send an INQUIRY command
            // and/or a READ CAPACITY command to see if the logical
            // unit has the desired properties.
            //
         }
      }
   }
}
```