ISO/IEC 11989

Edition 1.0 2010-03

INTERNATIONAL STANDARD

colour inside

Information technology – iSCSI management API

ISO/IEC 11989

Edition 1.0 2010-03

INTERNATIONAL STANDARD

colour inside

Information technology – iSCSI management API

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE XD

ICS 35.200

ISBN 2-8318-1082-5

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INFORMATION TECHNOLOGY –

iSCSI management API

FOREWORD

International Standard ISO/IEC 11989 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

International Standard ISO/IEC 11989 was prepared by ANSI, was adopted, under the fast track procedure, by joint technical committee 1: Information technology and has been assigned to subcommittee 25: Interconnection of information technology equipment.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INFORMATION TECHNOLOGY –

iSCSI management API

1 Scope

This International Standard specifies an Application Programming Interface (API) that provides interfaces to discover and manage iSCSI resources on a system. This International Standard is applicable to vendors who deliver drivers that provide iSCSI resources to a system.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document, including any amendments, applies.

The provisions of the referenced specifications other than ISO/IEC, IEC, ISO and ITU documents, as identified in this clause, are valid within the context of this International Standard. The reference to such a specification within this International Standard does not give it any further status within ISO/IEC. In particular, it does not give the referenced specification the status of an International Standard.

ISO/IEC 14776-453, *Information technology – Small Computer System Interface – Part 453: SCSI Primary Commands-3 (SPC-3)*

ISO/IEC 19501, *Unified Modeling Language (UML)*

ISO/IEC 8802-3:2000, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*[1]

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

IETF RFC 2945, *The SRP Authentication and Key Exchange System*

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

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

3 Document conventions

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**

[1] IEEE 802.1AX, IEEE Standard for Information technology

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 states whether 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, definitions and abbreviations

For the purposes of this document the following terms, definitions and abbreviations apply.

4.1.1
Alias
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

NOTE Definition taken from IETF RFC 3720.

4.1.2
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

4.1.3
Host
a compute node connected to the SAN

4.1.4
iSCSI Node
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

NOTE Definition taken from IETF RFC 3720.

4.1.5
Logical HBA
a representation of a parallel SCSI HBA to the operating system

4.1.6
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 link aggregation (see IEEE Std 802.1AX, 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.

4.1.7
Network Entity
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

NOTE Definition taken from IETF RFC 3720.

4.1.8
Network Portal
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

NOTE Definition taken from IETF RFC 3720.

4.1.9
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

4.1.10
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 is also 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

4.1.11
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

4.1.12
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

4.1.13
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

4.1.14
Portal Groups
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

NOTE Definition taken from IETF RFC 3720.

4.1.15
Portal Group Tag
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

NOTE Definition taken from IETF RFC 3720.

4.1.16
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

4.1.17
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

4.1.18
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

4.1.19
Unicode
unicode is a system of uniquely identifying (numbering) characters such that nearly any character in any language is identified

4.1.20
UTF-8
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

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

4.1.21 Abbreviations

API	**A**pplication **P**rogramming **I**nterface
FC	**F**ibre **C**hannel
ID	**ID**entifier
iSCSI	**i**nternet **S**mall **C**omputer **S**ystem **I**nterface
iSNS	**i**nternet **S**torage **N**ame **S**ervice
HBA	**H**ost **B**us **A**daptor
IMA	**i**SCSI **M**anagement **API**

LHBA	**L**ogical **H**ost **B**us **A**daptor
LNP	**L**ogical **N**etwork **P**ort
LUN	**L**ogical **U**nit **N**umber
RAM	**R**andom-**A**ccess **M**emory
OID	**O**bject **ID**
PHBA	**P**hysical **H**ost **B**us **A**daptor
PNP	**P**hysical **N**etwork **P**ort
SLP	**S**ervice **L**ocation **P**rotocol
SNIA	**S**torage **N**etworking **I**ndustry **A**ssociation
UTF	**U**nicode **T**ransformation **F**ormat

4.2 Concepts

4.2.1 Library and plugins

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

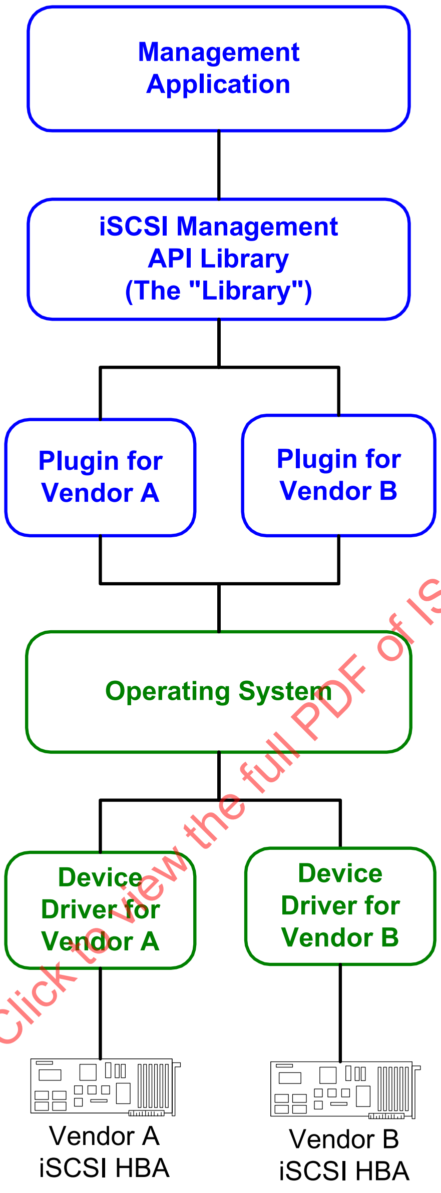
- a library;
- a library in combination with plugins.

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.



4.2.2 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.

4.2.3 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.

4.2.4 The Shared Node versus 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 reflects some limitations of existing HBA implementations.

4.2.5 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.

4.2.6 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.

4.2.7 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.

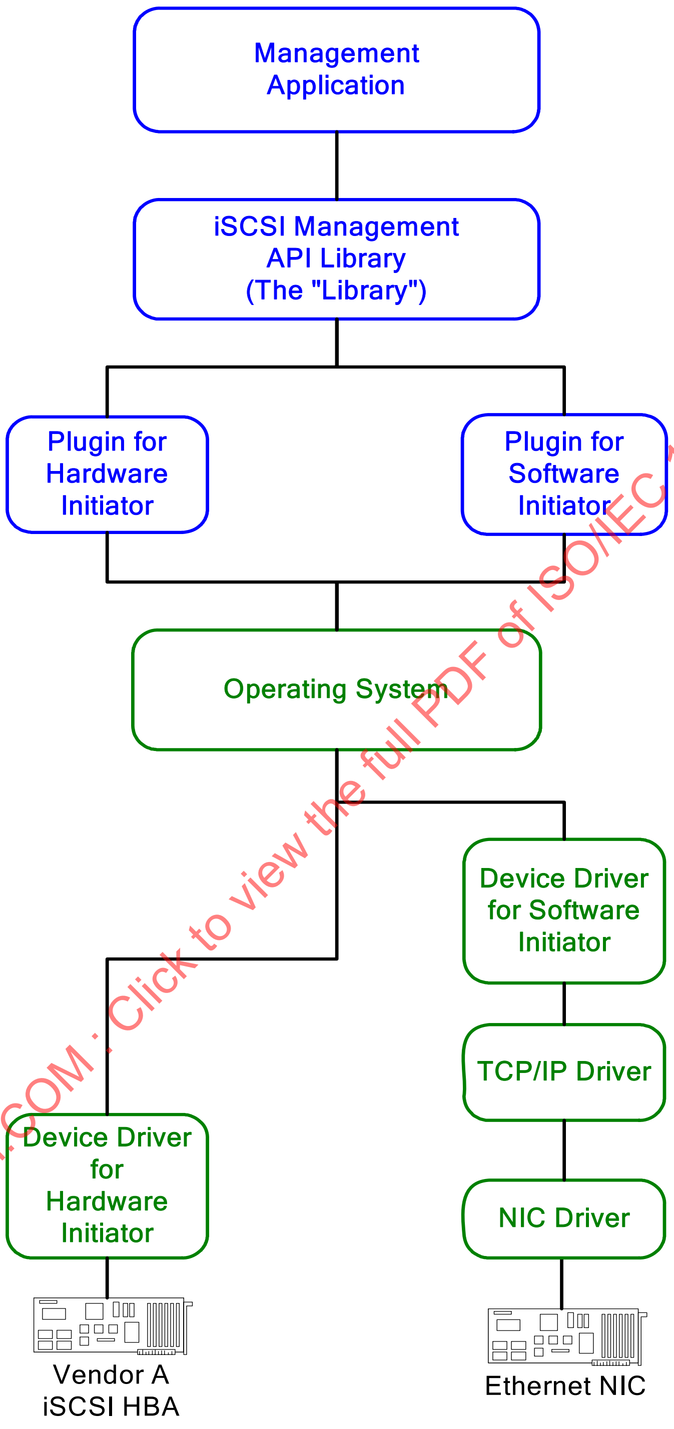
These terms are imprecise as both types of initiators use software and hardware. However, these terms have come into common use and so this specification uses them as well.

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.


Management Application
iSCSI Management API Library (The "Library")
Plugin for Hardware Initiator
Plugin for Software Initiator
Operating System
Device Driver for Software Initiator
TCP/IP Driver
NIC Driver
Device Driver for Hardware Initiator
Vendor A iSCSI HBA
Ethernet NIC

4.2.8 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, then 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.

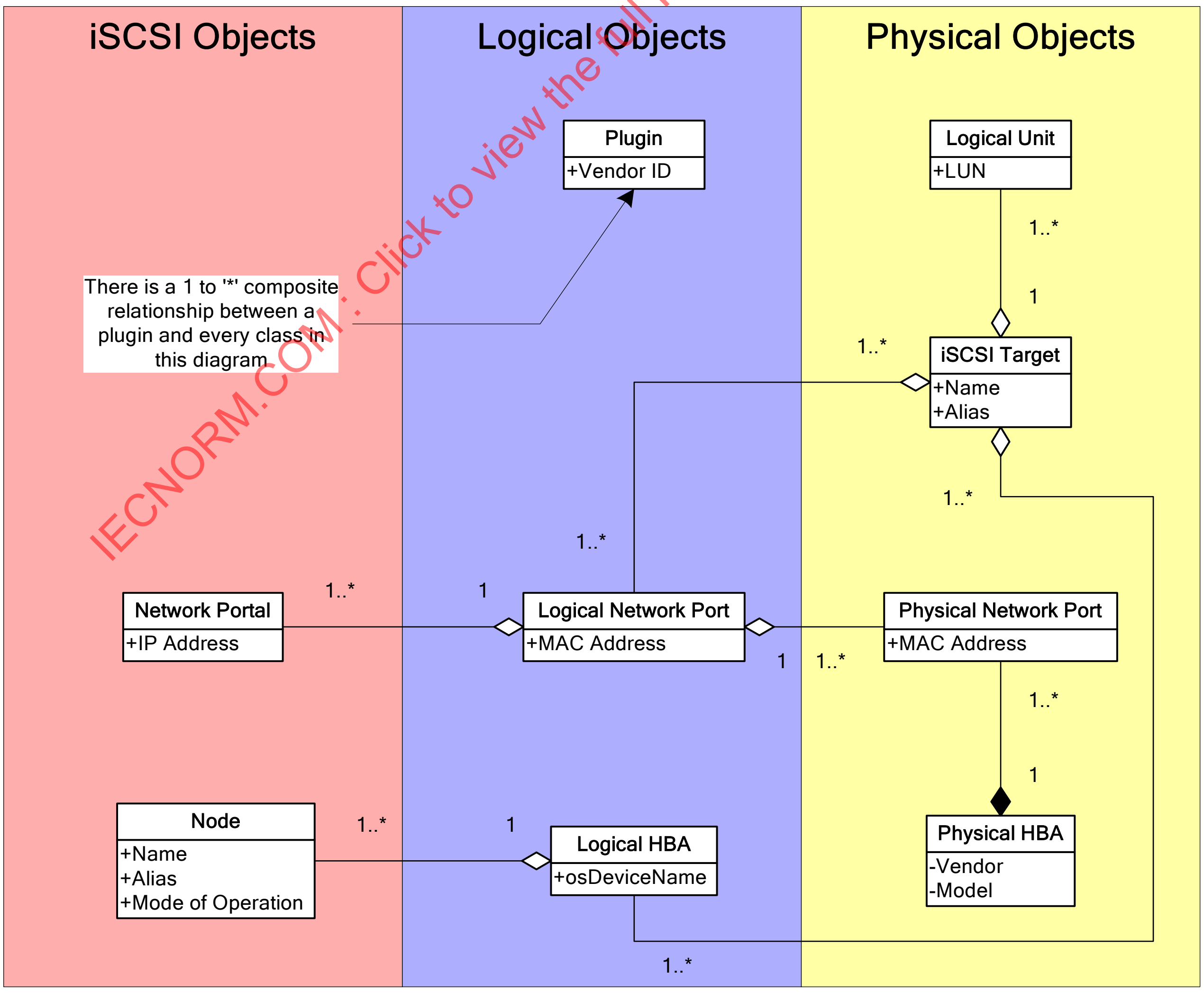
If a client does not set a value for a parameter at either the target or the LHBA level then the driver level value, i.e., a default value specified by the driver implementation is used.

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.

4.2.9 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

5.10.1.1 Format

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

5.10.1.2 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.

5.10.1.3 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

5.11.1.1 Format

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

5.11.1.2 Parameters

oid

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

5.11.1.3 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

5.12.1 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;
```

5.12.2 Fields

5.12.2.1 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.

5.12.2.2 IMA_OBJECT_TYPE_PLUGIN

The object represents a plugin to the IMA library.

This symbol has the value 1.

5.12.2.3 IMA_OBJECT_TYPE_NODE

The object represents an iSCSI node.

This symbol has the value 2.

5.12.2.4 IMA_OBJECT_TYPE_LHBA

The object represents a logical HBA.

This symbol has the value 3.

5.12.2.5 IMA_OBJECT_TYPE_PHBA

The object represents a physical HBA.

This symbol has the value 4.

5.12.2.6 IMA_OBJECT_TYPE_NETWORK_PORTAL

The object represents an iSCSI network portal.

This symbol has the value 5.

5.12.2.7 IMA_OBJECT_TYPE_PORTAL_GROUP

The object represents an iSCSI portal group.

This symbol has the value 6.

5.12.2.8 IMA_OBJECT_TYPE_LNP

The object represents a logical network port.

This symbol has the value 7.

5.12.2.9 IMA_OBJECT_TYPE_PNP

The object represents a physical network port.

This symbol has the value 8.

5.12.2.10 IMA_OBJECT_TYPE_TARGET

The object represents an iSCSI target relative to an LHBA.

This symbol has the value 9.

5.12.2.11 IMA_OBJECT_TYPE_LU

The object represents a logical unit relative to a target.

This symbol has the value 10.

5.12.2.12 IMA_OBJECT_TYPE_DISCOVERY_ADDRESS

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

This symbol has the value 11.

5.12.2.13 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.

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.

5.13.1 Macros

5.13.1.1 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.

5.13.1.2 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.

5.13.2 Non-error statuses

5.13.2.1 IMA_STATUS_SUCCESS

This status indicates that the API call succeeded.

This symbol has the value 0x00000000.

5.13.2.2 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.

5.13.2.3 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.

5.13.2.4 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.

5.13.3 Error statuses

5.13.3.1 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.

5.13.3.2 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.

5.13.3.3 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.

5.13.3.4 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.

5.13.3.5 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.

5.13.3.6 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.

5.13.3.7 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.

5.13.3.8 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.

5.13.3.9 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.

5.13.3.10 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.

5.13.3.11 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.

5.13.3.12 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.

5.13.3.13 IMA_ERROR_SCSI_STATUS_BUSY

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

This symbol has the value 0x800000101.

5.13.3.14 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.

5.13.3.15 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.

5.13.3.16 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.

5.13.3.17 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.

5.13.3.18 IMA_ERROR_PLUGINS_NOT_SUPPORTED

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

This symbol has the value 0x80000106.

5.13.3.19 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.

5.13.3.20 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.

5.13.3.21 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.

5.13.3.22 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.

5.13.3.23 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.

5.13.3.24 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

5.14.1 Format

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

5.14.2 Fields

5.14.2.1 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.

5.14.2.2 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.

5.14.2.3 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.

5.14.3 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

5.15.1 Format

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

5.15.2 Fields

5.15.2.1 oidCount

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

5.15.2.2 oids

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

5.15.3 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

5.16.1 Format

```
typedef IMA_WCHAR IMA_NODE_NAME[224];
```

5.16.2 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

5.17.1 Format

```
typedef IMA_WCHAR IMA_NODE_ALIAS[256];
```

5.17.2 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

5.18.1 Format

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

} IMA_IP_ADDRESS;
```

5.18.2 Fields

5.18.2.1 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.

5.18.2.2 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

5.19.1 Format

```
typedef IMA_WCHAR IMA_HOST_NAME[256];
```

5.19.2 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

5.20.1 Format

```
typedef struct IMA_host_id
{
    IMA_BOOL            hostnameInUse;

    union
    {
            IMA_HOST_NAME                   hostname;
            IMA_IP_ADDRESS                  ipAddress;
    } id;

} IMA_HOST_ID;
```

5.20.2 Fields

5.20.2.1 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.

5.20.2.2 hostname

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

5.20.2.3 ipAddress

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

5.20.3 Remarks

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

5.21 IMA_TARGET_ADDRESS

5.21.1 Format

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

5.21.2 Fields

5.21.2.1 hostnameIpAddress

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

5.21.2.2 portNumber

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

5.21.3 Remarks

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

5.22 IMA_ADDRESS_KEY

5.22.1 Format

```
typedef struct IMA_address_key
{
    IMA_IP_ADDRESS      ipAddress;

    IMA_BOOL            portalNumberValid;
    IMA_UINT16          portNumber;

    IMA_BOOL            portalGroupTagValid;
    IMA_UINT16          portalGroupTag;

} IMA_ADDRESS_KEY;
```

5.22.2 Fields

5.22.2.1 ipAddress

An IP address at which a target can be contacted.

5.22.2.2 portNumberValid

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

5.22.2.3 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 undefined.

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

5.22.2.4 portalGroupTagValid

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

5.22.2.5 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.

5.22.3 Remarks

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

5.23 IMA_ADDRESS_KEYS

5.23.1 Format

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

} IMA_ADDRESS_KEYS;
```

5.23.2 Fields

5.23.2.1 addressKeyCount

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

5.23.2.2 addressKeys

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

5.23.3 Remarks

This structure is returned by the IMA_GetAddressKeys API.

5.24 IMA_STATIC_DISCOVERY_TARGET

5.24.1 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;
```

5.24.2 Fields

5.24.2.1 targetName

The iSCSI node name of the statically discovered target.

5.24.2.2 targetAddress

A target address at which the statically discovered target resides.

5.24.2.3 portalGroupTagValid

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

5.24.2.4 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.

5.24.3 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

5.25.1 Format

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

5.25.2 Fields

5.25.2.1 associatedNodeOid

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

5.25.2.2 associatedLhbaOid

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

5.25.2.3 discoveryAddress

A target address representing the discovery address.

5.25.3 Remarks

This structure is returned by the IMA_GetDiscoveryAddressProperties API.

5.26 IMA_STATIC_DISCOVERY_TARGET_PROPERTIES

5.26.1 Format

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

} IMA_STATIC_DISCOVERY_TARGET_PROPERTIES;
```

5.26.2 Fields

5.26.2.1 associatedNodeOid

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

5.26.2.2 associatedLhbaOid

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

5.26.2.3 staticConfigTarget

The statically configured target.

5.26.3 Remarks

This structure is returned by the IMA_GetStaticDiscoveryTargetProperties API.

5.27 IMA_IP_PROPERTIES

5.27.1 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;
```

5.27.2 Fields

5.27.2.1 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.

5.27.2.2 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.

5.27.2.3 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.

5.27.2.4 subnetMaskValid

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

5.27.2.5 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.

5.27.2.6 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.

5.27.2.7 defaultGatewayValid

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

5.27.2.8 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.

5.27.2.9 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.

5.27.2.10 primaryDnsServerAddressValid

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

5.27.2.11 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.

5.27.2.12 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.

5.27.2.13 alternateDnsServerAddressValid

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

5.27.2.14 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.

5.27.2.15 reserved

This field is reserved.

5.27.3 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

5.28.1 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;
```

5.28.2 Fields

5.28.2.1 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 one.

5.28.2.2 vendor

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

5.28.2.3 implementationVersion

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

5.28.2.4 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.

5.28.2.5 buildTime

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

5.28.2.6 reserved

This field is reserved.

5.28.3 Remarks

This structure is returned by the IMA_GetLibraryProperties API.

5.29 IMA_PLUGIN_PROPERTIES

5.29.1 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_BYTE          reserved[64];

} IMA_PLUGIN_PROPERTIES;
```

5.29.2 Fields

5.29.2.1 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 one.

5.29.2.2 vendor

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

5.29.2.3 implementationVersion

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

5.29.2.4 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.

5.29.2.5 buildTime

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

5.29.2.6 lhbasCanBeCreatedAndDestroyed

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

5.29.2.7 reserved

This field is reserved.

5.29.3 Remarks

This structure is returned by the IMA_GetPluginProperties API.

5.30 IMA_NODE_PROPERTIES

5.30.1 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;
```

5.30.2 Fields

5.30.2.1 runningInInitiatorMode

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

5.30.2.2 runningInTargetMode

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

5.30.2.3 nameValid

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

5.30.2.4 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.

5.30.2.5 aliasValid

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

5.30.2.6 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.

5.30.2.7 nameAndAliasSettable

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

5.30.2.8 reserved

This field is reserved.

5.30.3 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

5.31.1 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_BYTE            reserved[128];

} IMA_LHBA_PROPERTIES;
```

5.31.2 Fields

5.31.2.1 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 for details on the value(s) of this field for each operating system.

5.31.2.2 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.

5.31.2.3 isDestroyable

A boolean indicating if the LHBA can be destroyed.

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

5.31.2.4 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.

5.31.2.5 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.

5.31.2.6 initiatorAuthMethodsSettable

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

5.31.2.7 targetAuthMethodsSettable

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

5.31.2.8 reserved

This field is reserved.

5.31.3 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.

5.32.1.1 IMA_ULP_TCP

This flag indicates that TCP is supported.

This symbol has the value 0x01.

5.32.1.2 IMA_ULP_SCTP

This flag indicates that SCTP is supported.

This symbol has the value 0x02.

5.32.1.3 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

5.33.1 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;
```

5.33.2 Fields

5.33.2.1 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.

5.33.2.2 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.

5.33.2.3 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.

5.33.2.4 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.

5.33.2.5 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.

5.33.2.6 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.

5.33.2.7 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.

5.33.2.8 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.

5.33.2.9 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.

5.33.2.10 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.

5.33.2.11 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

5.33.2.12 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.

5.33.2.13 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.

5.33.2.14 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.

5.33.2.15 isNic

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

5.33.2.16 isInitiator

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

5.33.2.17 isTarget

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

5.33.2.18 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.

5.33.2.19 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.

5.33.2.20 reserved

This field is reserved.

5.33.3 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

5.34.1 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;
```

5.34.2 Fields

5.34.2.1 iSnsDiscoverySettable

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

5.34.2.2 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.

5.34.2.3 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.

5.34.2.4 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.

5.34.2.5 slpDiscoverySettable

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

5.34.2.6 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.

5.34.2.7 staticDiscoverySettable

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

5.34.2.8 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.

5.34.2.9 sendTargetsDiscoverySettable

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

5.34.2.10 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.

5.34.2.11 reserved

This field is reserved.

5.34.3 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

5.35.1 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_PHBA_DOWNLOAD_IMAGE_TYPE;
```

5.35.2 Fields

5.35.2.1 IMA_IMAGE_TYPE_FIRMWARE

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

This symbol has the value 0.

5.35.2.2 IMA_IMAGE_TYPE_OPTION_ROM

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

This symbol has the value 1.

5.35.2.3 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.

5.35.3 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

5.36.1 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;
```

5.36.2 Fields

5.36.2.1 imageType

This field indicates the type of the associated download image.

5.36.2.2 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.

5.36.2.3 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.

5.36.2.4 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.

5.36.3 Remarks

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

5.37 IMA_ISNS_DISCOVERY_METHOD

5.37.1 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;
```

5.37.2 Fields

5.37.2.1 IMA_ISNS_DISCOVERY_METHOD_STATIC

This value indicates that the discovery method is static.

This symbol has the value 0.

5.37.2.2 IMA_ISNS_DISCOVERY_METHOD_DHCP

This value indicates that the discovery method is DHCP.

This symbol has the value 1.

5.37.2.3 IMA_ISNS_DISCOVERY_METHOD_SLP

This value indicates that the discovery method is SLP.

This symbol has the value 2.

5.37.3 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

5.38.1 Format

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

        IMA_BYTE            reserved[32];

    } IMA_PHBA_DOWNLOAD_PROPERTIES;
```

5.38.2 Fields

5.38.2.1 isPhbaDownloadFileSupported

A boolean indicating if the PHBA supports the IMA_IsPhbaDownloadFile API.

5.38.2.2 optionRomDownloadSupported

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

5.38.2.3 firmwareDownloadSupported

A boolean indicating if the PHBA supports downloading firmware code.

5.38.2.4 reserved

This field is reserved.

5.38.3 Remarks

This structure is returned by the IMA_GetPhbaDownloadProperties API.

5.39 IMA_IPSEC_PROPERTIES

5.39.1 Format

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

    IMA_BYTE            reserved[32];

} IMA_IPSEC_PROPERTIES;
```

5.39.2 Fields

5.39.2.1 ipsecSupported

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

5.39.2.2 implementedInHardware

An boolean indicating if IPsec is provided in hardware.

5.39.2.3 implementedInSoftware

An boolean indicating if IPsec is provided in software.

5.39.2.4 reserved

This field is reserved.

5.39.3 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

5.40.1 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;
```

5.40.2 Fields

5.40.2.1 currentValueValid

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

5.40.2.2 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.

5.40.2.3 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.

5.40.2.4 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.

5.40.2.5 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.

5.40.2.6 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.

5.40.2.7 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.

5.40.3 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

5.41.1 Format

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

    IMA_BOOL            currentValue;
    IMA_BOOL            defaultValue;

} IMA_BOOL_VALUE;
```

5.41.2 Fields

5.41.2.1 currentValueValid

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

5.41.2.2 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.

5.41.2.3 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.

5.41.2.4 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

5.42.1 Format

```
typedef IMA_BYTE IMA_MAC_ADDRESS[6];
```

5.42.2 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

5.43.1 Format

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

    IMA_BYTE            reserved[32];

} IMA_LNP_PROPERTIES;
```

5.43.2 Fields

5.43.2.1 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.

5.43.2.2 macAddressSettable

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

5.43.2.3 reserved

This field is reserved.

5.43.3 Remarks

This structure is returned by the IMA_GetLnpProperties API.

5.44 IMA_PNP_PROPERTIES

5.44.1 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;
```

5.44.2 Fields

5.44.2.1 associatedPhbaOid

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

5.44.2.2 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.

5.44.2.3 macAddressSettable

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

5.44.2.4 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.

5.44.2.5 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.

5.44.2.6 maximumFrameSize

The maximum size of a frame, in bytes.

5.44.2.7 reserved

This field is reserved.

5.44.3 Remarks

This structure is returned by the IMA_GetPnpProperties API.

5.45 IMA_PNP_STATISTICS

5.45.1 Format

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

} IMA_PNP_STATISTICS;
```

5.45.2 Fields

5.45.2.1 bytesSent

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

5.45.2.2 bytesReceived

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

5.45.2.3 pdusSent

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

5.45.2.4 pdusReceived

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

5.45.3 Remarks

This structure is returned by the IMA_GetPnpStatistics API.

5.46 IMA_NETWORK_PORTAL_PROPERTIES

5.46.1 Format

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

    IMA_BYTE            reserved[32];

} IMA_NETWORK_PORTAL_PROPERTIES;
```

5.46.2 Fields

5.46.2.1 ipAddress

The IP address of the network portal.

5.46.2.2 associatedLnp

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

5.46.2.3 reserved

This field is reserved.

5.46.3 Remarks

This structure is returned by the IMA_GetNetworkPortalProperties API.

5.47 IMA_PHBA_STATUS

5.47.1 Format

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

5.47.2 Values

5.47.2.1 IMA_PHBA_STATUS_WORKING

This status indicates that the PHBA is working properly.

This symbol has the value 0.

5.47.2.2 IMA_PHBA_STATUS_FAILED

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

This symbol has the value 1.

5.47.3 Remarks

This status is returned by the IMA_GetPhbaStatus API.

5.48 IMA_ NETWORK_PORT_STATUS

5.48.1 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;
```

5.48.2 Fields

5.48.2.1 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.

5.48.2.2 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.

5.48.2.3 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.

5.48.2.4 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.

5.48.2.5 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.

5.48.3 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 in 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

5.49.1 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;
```

5.49.2 Fields

5.49.2.1 IMA_TARGET_DISCOVERY_METHOD_STATIC

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

This symbol has the value 1.

5.49.2.2 IMA_TARGET_DISCOVERY_METHOD_SLP

This value indicates that a target was discovered using SLP.

This symbol has the value 2.

5.49.2.3 IMA_TARGET_DISCOVERY_METHOD_ISNS

This value indicates that a target was discovered using iSNS.

This symbol has the value 4.

5.49.2.4 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

5.50.1 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;
```

5.50.2 Fields

5.50.2.1 associatedNodeOid

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

5.50.2.2 associatedLhbaOid

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

5.50.2.3 name

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

5.50.2.4 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.

5.50.2.5 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.

5.50.2.6 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.

5.50.2.7 sendTargetsDiscoveryEnabled

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

5.50.2.8 reserved

This field is reserved.

5.50.3 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

5.51.1 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;
```

5.51.2 Fields

5.51.2.1 loginFailedCountValid

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

5.51.2.2 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.

5.51.2.3 sessionFailedCountValid

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

5.51.2.4 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.

5.51.2.5 headerOrDigestSessionFailedCountValid

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

5.51.2.6 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.

5.51.2.7 timeLimitExceededSessionFailedCountValid

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

5.51.2.8 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.

5.51.2.9 formatErrorSessionFailedCountValid

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

5.51.2.10 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.

5.51.2.11 closedConnectionDueToTimeoutCountValid

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

5.51.2.12 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.

5.51.2.13 lastLoginFailureTimeValid

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

5.51.2.14 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.

5.51.2.15 reserved

This field is reserved.

5.51.3 Remarks

This structure is returned by the IMA_GetTargetErrorStatistics API.

5.52 IMA_LU_PROPERTIES

5.52.1 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;
```

5.52.2 Fields

5.52.2.1 associatedTargetOid

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

5.52.2.2 targetLun

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

5.52.2.3 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.

5.52.2.4 timeExposedToOs

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

5.52.2.5 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.

5.52.2.6 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.

5.52.2.7 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.

5.52.2.8 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.

5.52.2.9 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.

5.52.2.10 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.

5.52.2.11 reserved

This field is reserved.

5.52.3 Remarks

This structure is returned by the IMA_GetLuProperties API.

5.53 IMA_DEVICE_STATISTICS

5.53.1 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;
```

5.53.2 Fields

5.53.2.1 scsiPayloadBytesSent

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

5.53.2.2 scsiPayloadBytesReceived

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

5.53.2.3 iScsiPduBytesSent

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

5.53.2.4 iScsiPduBytesReceived

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

5.53.2.5 iScsiPdusSent

This value contains the number of iSCSI PDUs sent.

5.53.2.6 iScsiPdusReceived

This value contains the number of iSCSI PDUs received.

5.53.2.7 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.

5.53.2.8 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.

5.53.3 Remarks

This structure is returned by the IMA_GetDeviceStatistics API.

5.54 IMA_STATISTICS_PROPERTIES

5.54.1 Format

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

} IMA_STATISTICS_PROPERTIES;
```

5.54.2 Fields

5.54.2.1 statisticsCollectionSettable

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

5.54.2.2 statisticsCollectionEnabled

A boolean indicating if statistics collection is enabled.

5.54.3 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

5.55.1 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;
```

5.55.2 Fields

5.55.2.1 IMA_AUTHMETHOD_NONE

This indicates that no authentication is performed.

This symbol has the value 0.

5.55.2.2 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.

5.55.2.3 IMA_AUTHMETHOD_SRP

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

This symbol has the value 2.

5.55.2.4 IMA_AUTHMETHOD_KRB5

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

This symbol has the value 3.

5.55.2.5 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.

5.55.2.6 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

5.56.1 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;
```

5.56.2 Fields

5.56.2.1 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.

5.56.2.2 name

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

5.56.2.3 nameLength

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

5.56.2.4 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.

5.56.2.5 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.

5.56.2.6 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.

5.56.2.7 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*.

5.56.2.8 reserved

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

5.56.3 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

5.57.1 Format

```
typedef struct IMA_srp_initiator_authparms
{

    IMA_BYTE              userName[512];
    IMA_UINT              userNameLength;

    IMA_BYTE              reserved[512];

} IMA_SRP_INITIATOR_AUTHPARMS;
```

5.57.2 Fields

5.57.2.1 userName

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

5.57.2.2 userNameLength

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

5.57.2.3 reserved

This field is reserved and shall be set to zero.

5.57.3 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

5.58.1 Format

```
typedef struct IMA_krb5_initiator_authparms
{

    IMA_BYTE            clientKey[1024];
    IMA_UINT            clientKeyLength;

    IMA_BYTE            reserved[2048];

} IMA_KRB5_INITIATOR_AUTHPARMS;
```

5.58.2 Fields

5.58.2.1 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.

5.58.2.2 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*.

5.58.2.3 reserved

This field is reserved and shall be set to 0.

5.58.3 Remarks

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

Only target authentication is supported; mutual authentication, where the initiator authenticates the target and the target authenticates the initiator is not currently supported.

This structure is included in the IMA_INITIATOR_AUTHPARMS union.

5.59 IMA_SPKM_INITIATOR_AUTHPARMS

5.59.1 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;
```

5.59.2 Fields

5.59.2.1 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.

5.59.2.2 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*.

5.59.2.3 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.

5.59.2.4 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*.

5.59.3 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

5.60.1 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;
```

5.60.2 Fields

5.60.2.1 chapParms

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

5.60.2.2 srpParms

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

5.60.2.3 kerberosParms

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

5.60.2.4 spkmParms

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

5.60.3 Remarks

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

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

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 port 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

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_AddStaticDiscoveryTarget
3. IMA_DeregisterForObjectPropertyChanges
4. IMA_DeregisterForObjectVisibilityChanges
5. IMA_ExposeLu
6. IMA_FreeMemory
7. IMA_GenerateNodeName
8. IMA_GetAddressKeyProperties
9. IMA_GetAssociatedPluginOid
10. IMA_GetDataPduInOrderProperties
11. IMA_GetDataSequenceInOrderProperties
12. IMA_GetDefaultTime2RetainProperties
13. IMA_GetDefaultTime2WaitProperties
14. IMA_GetDeviceStatistics
15. IMA_GetDiscoveryAddressOidList
16. IMA_GetDiscoveryAddressProperties
17. IMA_GetDiscoveryProperties
18. IMA_GetErrorRecoveryLevelProperties
19. IMA_GetFirstBurstLengthProperties
20. IMA_GetImmediateDataProperties
21. IMA_GetInitialR2TProperties
22. IMA_GetInitiatorAuthParms
23. IMA_GetInUseInitiatorAuthMethods
24. IMA_GetIpProperties
25. IMA_GetIpsecProperties
26. IMA_GetLhbaOidList
27. IMA_GetLhbaProperties
28. IMA_GetLibraryProperties
29. IMA_GetLnpOidList
30. IMA_GetLnpProperties
31. IMA_GetLuOid
32. IMA_GetLuOidList
33. IMA_GetLuProperties
34. IMA_GetMaxBurstLengthProperties
35. IMA_GetMaxConnectionsProperties
36. IMA_GetMaxOutstandingRT2Properties
37. IMA_GetMaxRecvDataSegmentLength Properties
38. IMA_GetNetworkPortalOidList
39. IMA_GetNetworkPortalProperties
40. IMA_GetNetworkPortStatus
41. IMA_GetNodeProperties
42. IMA_GetNonSharedNodeOidList
43. IMA_GetObjectType
44. IMA_GetPhbaDownloadProperties
45. IMA_GetPhbaOidList
46. IMA_GetPhbaProperties
47. IMA_GetPhbaStatus
48. IMA_GetPluginOidList
49. IMA_GetPluginProperties
50. IMA_GetPnpOidList
51. IMA_GetPnpProperties
52. IMA_GetPnpStatistics
53. IMA_GetSharedNodeOid
54. IMA_GetStaticDiscoveryTargetOidList
55. IMA_GetStaticDiscoveryTargetProperties
56. IMA_GetStatisticsProperties
57. IMA_GetSupportedAuthMethods
58. IMA_GetTargetErrorStatistics
59. IMA_GetTargetOidList
60. IMA_GetTargetProperties
61. IMA_IsPhbaDownloadFile
62. IMA_LuInquiry

63. IMA_LuReadCapacity
64. IMA_LuReportLuns
65. IMA_PhbaDownload
66. IMA_PluginIOCtl
67. IMA_RegisterForObjectPropertyChanges
68. IMA_RegisterForObjectVisibilityChanges
69. IMA_RemoveDiscoveryAddress
70. IMA_RemoveStaleData
71. IMA_RemoveStaticDiscoveryTarget
72. IMA_SetDataPduInOrder
73. IMA_SetDataSequenceInOrder
74. IMA_SetDefaultGateway
75. IMA_SetDefaultTime2Retain
76. IMA_SetDefaultTime2Wait
77. IMA_SetDnsServerAddress
78. IMA_SetErrorRecoveryLevel
79. IMA_SetFirstBurstLength
80. IMA_SetImmediateData
81. IMA_SetInitialRT2
82. IMA_SetInitiatorAuthMethods
83. IMA_SetInitiatorAuthParms
84. IMA_SetIpConfigMethod
85. IMA_SetIsnsDiscovery
86. IMA_SetMaxBurstLength
87. IMA_SetMaxConnections
88. IMA_SetMaxOutstandingR2T
89. IMA_SetMaxRecvDataSegmentLength
90. IMA_SetNetworkPortalIpAddress
91. IMA_SetNodeAlias
92. IMA_SetNodeName
93. IMA_SetSendTargetsDiscovery
94. IMA_SetSlpDiscovery
95. IMA_SetStaticDiscovery
96. IMA_SetStatisticsCollection
97. IMA_SetSubnetMask
98. IMA_UnexposeLu

6.2.1 IMA_AddDiscoveryAddress

6.2.1.1 Synopsis

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

6.2.1.2 Prototype

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

6.2.1.3 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.

6.2.1.4 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.

6.2.1.5 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.

6.2.1.6 Support

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

6.2.1.7 See Also

IMA_RemoveDiscoveryAddress

IMA_GetDiscoveryAddressProperties

IMA_RegisterForObjectVisibilityChanges

IMA_GetDiscoveryProperties

6.2.2 IMA_AddStaticDiscoveryTarget

6.2.2.1 Synopsis

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

6.2.2.2 Prototype

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

6.2.2.3 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.

6.2.2.4 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.

6.2.2.5 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.

6.2.2.6 Support

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

6.2.2.7 See Also

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetProperties

IMA_RegisterForObjectVisibilityChanges

IMA_GetDiscoveryProperties

IMA_SetStaticDiscovery

6.2.3 IMA_DeregisterForObjectPropertyChanges

6.2.3.1 Synopsis

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

6.2.3.2 Prototype

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

6.2.3.3 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.

6.2.3.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.3.5 Support

Mandatory

6.2.3.6 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.

6.2.3.7 See Also

IMA_RegisterForObjectPropertyChanges

6.2.4 IMA_DeregisterForObjectVisibilityChanges

6.2.4.1 Synopsis

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

6.2.4.2 Prototype

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

6.2.4.3 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.

6.2.4.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.4.5 Support

Mandatory

6.2.4.6 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.

6.2.4.7 See Also

IMA_RegisterForObjectVisibilityChanges

6.2.5 IMA_ExposeLu

6.2.5.1 Synopsis

Exposes the specified logical unit to the operating system.

6.2.5.2 Prototype

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

6.2.5.3 Parameters

luOid

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

6.2.5.4 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.

6.2.5.5 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.

6.2.5.6 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.

6.2.5.7 See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_UnexposeLu

IMA_GetLuId

IMA_GetLuOidList

6.2.6 IMA_FreeMemory

6.2.6.1 Synopsis

Frees memory returned by an IMA API.

6.2.6.2 Prototype

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

6.2.6.3 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.

6.2.6.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.6.5 Support

Mandatory

6.2.6.6 Remarks

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

6.2.6.7 See Also

IMA_GetLhbaOidList

IMA_GetLnpOidList

IMA_GetLuOidList

IMA_GetNonSharedNodeOidList

IMA_GetPhbaOidList

IMA_GetPnpOidList

IMA_GetTargetOidList

6.2.7 IMA_GenerateNodeName

6.2.7.1 Synopsis

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

6.2.7.2 Prototype

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

6.2.7.3 Parameters

generatedName

On successful return contains the generated node name.

6.2.7.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.7.5 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.

6.2.7.6 Support

Mandatory

6.2.7.7 See Also

IMA_GetSharedNodeOid

IMA_GetNonSharedNodeOidList

IMA_SetNodeName

Example of Setting a Node Name

6.2.8 IMA_GetAddressKeys

6.2.8.1 Synopsis

Gets the address keys of a target.

6.2.8.2 Prototype

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

6.2.8.3 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.

6.2.8.4 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.

6.2.8.5 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.

6.2.8.6 Support

Mandatory

6.2.8.7 See Also

IMA_FreeMemory

6.2.9 IMA_GetAssociatedPluginOid

6.2.9.1 Synopsis

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

6.2.9.2 Prototype

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

6.2.9.3 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.

6.2.9.4 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.

6.2.9.5 Remarks

None

6.2.9.6 Support

Mandatory

6.2.9.7 See Also

Example of Getting an Associated Plugin ID

IMA_PluginIOCtl

6.2.10 IMA_GetDataPduInOrderProperties

6.2.10.1 Synopsis

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

6.2.10.2 Prototype

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

6.2.10.3 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.

6.2.10.4 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.

6.2.10.5 Remarks

None

6.2.10.6 Support

Mandatory for both LHBAs and targets

6.2.10.7 See Also

IMA_SetDataPduInOrder
iSCSI Session and Connection Parameters

6.2.11 IMA_GetDataSequenceInOrderProperties

6.2.11.1 Synopsis

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

6.2.11.2 Prototype

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

6.2.11.3 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.

6.2.11.4 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.

6.2.11.5 Remarks

None

6.2.11.6 Support

Mandatory for both LHBAs and targets

6.2.11.7 See Also

IMA_SetDataSequenceInOrder

iSCSI Session and Connection Parameters

6.2.12 IMA_GetDefaultTime2RetainProperties

6.2.12.1 Synopsis

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

6.2.12.2 Prototype

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

6.2.12.3 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.

6.2.12.4 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.

6.2.12.5 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.

6.2.12.6 Support

Mandatory for both LHBAs and targets

6.2.12.7 See Also

IMA_SetDefaultTime2Retain

iSCSI Session and Connection Parameters

6.2.13 IMA_GetDefaultTime2WaitProperties

6.2.13.1 Synopsis

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

6.2.13.2 Prototype

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

6.2.13.3 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.

6.2.13.4 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.

6.2.13.5 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.

6.2.13.6 Support

Mandatory for both LHBAs and targets

6.2.13.7 See Also

IMA_SetDefaultTime2Wait

iSCSI Session and Connection Parameters

6.2.14 IMA_GetDeviceStatistics

6.2.14.1 Synopsis

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

6.2.14.2 Prototype

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

6.2.14.3 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.

6.2.14.4 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 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*.

6.2.14.5 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.

6.2.14.6 Support

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

6.2.14.7 See Also

IMA_STATISTICS_PROPERTIES

IMA_GetLuOidList

IMA_SetStatisticsCollection

6.2.15 IMA_GetDiscoveryAddressOidList

6.2.15.1 Synopsis

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

6.2.15.2 Prototype

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

6.2.15.3 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.

6.2.15.4 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.

6.2.15.5 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.

6.2.15.6 Support

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

6.2.15.7 See Also

IMA_FreeMemory

IMA_AddDiscoveryAddress

IMA_RemoveDiscoveryAddress

IMA_GetDiscoveryAddressProperties

6.2.16 IMA_GetDiscoveryAddressProperties

6.2.16.1 Synopsis

Gets the properties of the specified discovery address.

6.2.16.2 Prototype

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

6.2.16.3 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*.

6.2.16.4 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.

6.2.16.5 Remarks

None

6.2.16.6 Support

Mandatory

6.2.16.7 See Also

IMA_AddStaticDiscoveryTarget

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetOidList

6.2.17 IMA_GetDiscoveryProperties

6.2.17.1 Synopsis

Gets the discovery properties of the specified physical or logical HBA.

6.2.17.2 Prototype

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

6.2.17.3 Parameters

oid

The object ID of the PHBA or LHBA 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 or LHBA.

6.2.17.4 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 or LHBA object.

IMA_ERROR_OBJECT_NOT_FOUND

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

6.2.17.5 Remarks

None

6.2.17.6 Support

Mandatory for PHBAs.

Optional for LHBAs.

6.2.17.7 See Also

IMA_GetPhbaOidList

IMA_GetLhbaOidList

IMA_SetIsnsDiscovery

IMA_SetSlpDiscovery

IMA_SetStaticDiscovery

IMA_SetSendTargetsDiscovery

Example of Getting PHBA Discovery Properties

6.2.18 IMA_GetErrorRecoveryLevelProperties

6.2.18.1 Synopsis

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

6.2.18.2 Prototype

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

6.2.18.3 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.

6.2.18.4 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.

6.2.18.5 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.

6.2.18.6 Support

Mandatory for both LHBAs and targets.

6.2.18.7 See Also

IMA_SetErrorRecoveryLevel
iSCSI Session and Connection Parameters

6.2.19 IMA_GetFirstBurstLengthProperties

6.2.19.1 Synopsis

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

6.2.19.2 Prototype

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

6.2.19.3 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.

6.2.19.4 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.

6.2.19.5 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.

6.2.19.6 Support

Mandatory for both LHBAs and targets.

6.2.19.7 See Also

IMA_MIN_MAX_VALUE

IMA_SetFirstBurstLength

6.2.20 IMA_GetImmediateDataProperties

6.2.20.1 Synopsis

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

6.2.20.2 Prototype

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

6.2.20.3 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.

6.2.20.4 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.

6.2.20.5 Remarks

None

6.2.20.6 Support

Mandatory for both LHBAs and targets.

6.2.20.7 See Also

IMA_SetImmediateData
iSCSI Session and Connection Parameters

6.2.21 IMA_GetInitialR2TProperties

6.2.21.1 Synopsis

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

6.2.21.2 Prototype

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

6.2.21.3 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.

6.2.21.4 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.

6.2.21.5 Remarks

None

6.2.21.6 Support

Mandatory for both LHBAs and targets.

6.2.21.7 See Also

IMA_SetInitialRT2

iSCSI Session and Connection Parameters

6.2.22 IMA_GetInitiatorAuthParms

6.2.22.1 Synopsis

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

6.2.22.2 Prototype

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

6.2.22.3 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.

6.2.22.4 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.

6.2.22.5 Remarks

None

6.2.22.6 Support

Optional

6.2.22.7 See Also

IMA_SetInitiatorAuthParms

Client Implementation Notes: Transmission of Authorization Parameters

6.2.23 IMA_GetInUseInitiatorAuthMethods

6.2.23.1 Synopsis

Gets the authentication methods currently in use by the specified logical HBA.

6.2.23.2 Prototype

```
IMA_STATUS IMA_GetInUseInitiatorAuthMethods(
    /* in */         IMA_OID  lhbaOid,
    [/* in, out */]    IMA_UINT *pMethodCount,
    /* out */        IMA_AUTHMETHOD *pMethodList;
);
```

6.2.23.3 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.

6.2.23.4 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.

6.2.23.5 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.

6.2.23.6 Support

Mandatory

6.2.23.7 See Also

IMA_GetSupportedAuthMethods

IMA_SetInitiatorAuthMethods

6.2.24 IMA_GetIpProperties

6.2.24.1 Synopsis

Gets the IP properties of the specified physical network port.

6.2.24.2 Prototype

```
IMA_STATUS IMA_GetIpProperties(
    /* in */     IMA_OID oid,
    /* out */    IMA_IP_PROPERTIES *pProps
);
```

6.2.24.3 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*.

6.2.24.4 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.

6.2.24.5 Remarks

None

6.2.24.6 Support

Mandatory

6.2.24.7 See Also

IMA_SetDefaultGateway

IMA_SetDnsServerAddress

IMA_SetIpConfigMethod

IMA_SetSubnetMask

6.2.25 IMA_GetIpsecProperties

6.2.25.1 Synopsis

Gets the IPsec properties of the specified physical network port or logical HBA.

6.2.25.2 Prototype

```
IMA_STATUS IMA_GetIpsecProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_IPSEC_PROPERTIES *pProps
);
```

6.2.25.3 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.

6.2.25.4 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.

6.2.25.5 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.

6.2.25.6 Support

Mandatory

6.2.25.7 See Also

IMA_GetLhbaOidList

IMA_GetPnpOidList

IPsec Security in Client Usage Notes

6.2.26 IMA_GetLhbaOidList

6.2.26.1 Synopsis

Gets a list of the object IDs of all the logical HBAs in the system.

6.2.26.2 Prototype

```
IMA_STATUS IMA_GetLhbaOidList(
    /* out */      IMA_OID_LIST **ppList
);
```

6.2.26.3 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.

6.2.26.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.26.5 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.

6.2.26.6 Support

Mandatory

6.2.26.7 See Also

IMA_FreeMemory

Example of Getting LHBA Properties

6.2.27 IMA_GetLhbaProperties

6.2.27.1 Synopsis

Gets the properties of the specified logical HBA.

6.2.27.2 Prototype

```
IMA_STATUS IMA_GetLhbaProperties(
     /* in */       IMA_OID lhbaOid,
     /* out */      IMA_LHBA_PROPERTIES *pProps
);
```

6.2.27.3 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*.

6.2.27.4 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.

6.2.27.5 Remarks

None

6.2.27.6 Support

Mandatory

6.2.27.7 See Also

Example of Getting LHBA Properties

6.2.28 IMA_GetLibraryProperties

6.2.28.1 Synopsis

Gets the properties of the IMA library that is being used.

6.2.28.2 Prototype

```
IMA_STATUS IMA_GetLibraryProperties(
    /* out */    IMA_LIBRARY_PROPERTIES *pProps
);
```

6.2.28.3 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.

6.2.28.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.28.5 Remarks

None

6.2.28.6 Support

Mandatory

6.2.28.7 See Also

Example of Getting Library Properties

6.2.29 IMA_GetLnpOidList

6.2.29.1 Synopsis

Gets a list of the object IDs of all the iSCSI usable logical network ports in the system.

6.2.29.2 Prototype

```
IMA_STATUS IMA_GetLnpOidList(
    /* out */    IMA_OID_LIST **ppList
);
```

6.2.29.3 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.

6.2.29.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.29.5 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.

6.2.29.6 Support

Mandatory

6.2.29.7 See Also

IMA_FreeMemory

6.2.30 IMA_GetLnpProperties

6.2.30.1 Synopsis

Gets the properties of the specified logical network port.

6.2.30.2 Prototype

```
IMA_STATUS IMA_GetLnpProperties(
    /* in */     IMA_OID lnpOid,
    /* out */    IMA_LNP_PROPERTIES *pProps
);
```

6.2.30.3 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*.

6.2.30.4 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.

6.2.30.5 Remarks

None

6.2.30.6 Support

Mandatory

6.2.31 IMA_GetLuOid

6.2.31.1 Synopsis

Gets a logical unit object ID for the specified LUN connected to the specified target.

6.2.31.2 Prototype

```
IMA_STATUS IMA_GetLuOid(
    /* in */     IMA_OID targetOid,
    /* in */     IMA_BYTE lun[8],
    /* out */    IMA_OID *pluOid
);
```

6.2.31.3 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.

6.2.31.4 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.

6.2.31.5 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.

6.2.31.6 Support

Mandatory

6.2.31.7 See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_GetLuOidList

IMA_GetLuProperties

IMA_LuReportLuns

6.2.32 IMA_GetLuOidList

6.2.32.1 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.

6.2.32.2 Prototype

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

6.2.32.3 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.

6.2.32.4 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.

6.2.32.5 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 has finished using the returned list it shall free the memory used by the list by calling IMA_FreeMemory.

6.2.32.6 Support

Mandatory

6.2.32.7 See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_FreeMemory

IMA_GetLuOid

IMA_GetLuProperties

6.2.33 IMA_GetLuProperties

6.2.33.1 Synopsis

Gets the properties of the specified logical unit.

6.2.33.2 Prototype

```
IMA_STATUS IMA_GetLuProperties(
    /* in */     IMA_OID luOid,
    /* out */    IMA_LU_PROPERTIES *pProps
);
```

6.2.33.3 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.

6.2.33.4 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.

6.2.33.5 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.

6.2.33.6 Support

Mandatory

6.2.33.7 See Also

IMA_GetLuOid

IMA_GetLuOidList

6.2.34 IMA_GetMaxBurstLengthProperties

6.2.34.1 Synopsis

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

6.2.34.2 Prototype

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

6.2.34.3 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.

6.2.34.4 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.

6.2.34.5 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.

6.2.34.6 Support

Mandatory for both LHBAs and targets.

6.2.34.7 See Also

IMA_SetMaxBurstLength

iSCSI Session and Connection Parameters

Example of Getting/Setting LHBA Max Burst Length

6.2.35 IMA_GetMaxConnectionsProperties

6.2.35.1 Synopsis

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

6.2.35.2 Prototype

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

6.2.35.3 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.

6.2.35.4 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.

6.2.35.5 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.

6.2.35.6 Support

Mandatory for both LHBAs and targets.

6.2.35.7 See Also

IMA_SetMaxConnections

iSCSI Session and Connection Parameters

6.2.36 IMA_GetMaxOutstandingR2TProperties

6.2.36.1 Synopsis

Gets the `MaxOutstandingR2T` per task iSCSI login parameter properties for the specified logical HBA or target.

6.2.36.2 Prototype

```
IMA_STATUS IMA_GetMaxOutstandingR2TProperties(
    /* in */     IMA_OID oid,
    /* out */    IMA_MIN_MAX_VALUE *pProps
);
```

6.2.36.3 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.

6.2.36.4 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.

6.2.36.5 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.

6.2.36.6 Support

Mandatory for both LHBAs and targets.

6.2.36.7 See Also

IMA_SetMaxOutstandingR2T

iSCSI Session and Connection Parameters

6.2.37 IMA_GetMaxRecvDataSegmentLengthProperties

6.2.37.1 Synopsis

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

6.2.37.2 Prototype

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

6.2.37.3 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.

6.2.37.4 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.

6.2.37.5 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.

6.2.37.6 Support

Mandatory for both LHBAs and targets.

6.2.37.7 See Also

IMA_SetMaxRecvDataSegmentLength

iSCSI Session and Connection Parameters

6.2.38 IMA_GetNetworkPortalOidList

6.2.38.1 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.

6.2.38.2 Prototype

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

6.2.38.3 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.

6.2.38.4 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.

6.2.38.5 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.

6.2.38.6 SupportMandatory

6.2.38.7

6.2.38.8 See Also

IMA_FreeMemory

6.2.39 IMA_GetNetworkPortalProperties

6.2.39.1 Synopsis

Gets the properties of the specified network portal.

6.2.39.2 Prototype

```
IMA_STATUS IMA_GetNetworkPortalProperties(
     /* in */      IMA_OID networkPortalOid,
     /* out */     IMA_NETWORK_PORTAL_PROPERTIES *pProps
);
```

6.2.39.3 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*.

6.2.39.4 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.

6.2.39.5 Remarks

None

6.2.39.6 Support

Mandatory

6.2.40 IMA_GetNetworkPortStatus

6.2.40.1 Synopsis

Gets the status of a specified logical or physical network port.

6.2.40.2 Prototype

```
IMA_STATUS IMA_GetNetworkPortStatus(
    /* in */    IMA_OID portOid,
    /* out */   IMA_NETWORK_PORT_STATUS *pStatus
);
```

6.2.40.3 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.

6.2.40.4 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.

6.2.40.5 Remarks

None

6.2.40.6 Support

Mandatory for both LNPs and PNPs.

6.2.41 IMA_GetNodeProperties

6.2.41.1 Synopsis

Gets the properties of the specified iSCSI node.

6.2.41.2 Prototype

```
IMA_STATUS IMA_GetNodeProperties(
    /* in */    IMA_OID nodeOid,
    /* out */   IMA_NODE_PROPERTIES *pProps
);
```

6.2.41.3 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*.

6.2.41.4 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.

6.2.41.5 Remarks

None

6.2.41.6 Support

Mandatory

6.2.41.7 See Also

IMA_GetSharedNodeOid
IMA_GetNonSharedNodeOidList
Example of Getting Node Properties

6.2.42 IMA_GetNonSharedNodeOidList

6.2.42.1 Synopsis

Gets a list of the object IDs for the non-shared nodes of the currently executing operating system image.

6.2.42.2 Prototype

```
IMA_STATUS IMA_GetNonSharedNodeOidList(
    /* out */      IMA_OID_LIST **ppList
);
```

6.2.42.3 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.

6.2.42.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.42.5 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.

6.2.42.6 Support

Mandatory

6.2.42.7 See Also

Concepts: The Shared Node versus Non-shared Nodes

IMA_FreeMemory

IMA_GetSharedNodeOid

IMA_GetNodeProperties

Example of Getting an Associated Plugin ID

6.2.43 IMA_GetObjectType

6.2.43.1 Synopsis

Gets the object type of an initialized object ID.

6.2.43.2 Prototype

```
IMA_STATUS IMA_GetObjectType(
    /* in */     IMA_OID oid,
    /* out */    IMA_OBJECT_TYPE *pObjectType
);
```

6.2.43.3 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*.

6.2.43.4 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.

6.2.43.5 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.

6.2.43.6 Support

Mandatory

6.2.43.7 See Also

IMA_RegisterForObjectVisibilityChanges

6.2.44 IMA_GetPhbaDownloadProperties

6.2.44.1 Synopsis

Gets the download properties for the specified PHBA.

6.2.44.2 Prototype

```
IMA_STATUS IMA_GetPhbaDownloadProperties(
    /* in */     IMA_OID phbaOid,
    /* out */    IMA_PHBA_DOWNLOAD_PROPERTIES *pProps
);
```

6.2.44.3 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.

6.2.44.4 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.

6.2.44.5 Remarks

None

6.2.44.6 Support

Mandatory

6.2.44.7 See Also

IMA_GetPhbaOidList
IMA_IsPhbaDownloadFile
IMA_PhbaDownload

6.2.45 IMA_GetPhbaOidList

6.2.45.1 Synopsis

Gets a list of the object IDs of all the physical HBAs in the system.

6.2.45.2 Prototype

```
IMA_STATUS IMA_GetPhbaOidList(
    /* out */      IMA_OID_LIST **ppList
);
```

6.2.45.3 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.

6.2.45.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.45.5 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.

6.2.45.6 Support

Mandatory

6.2.45.7 See Also

IMA_FreeMemory

6.2.46 IMA_GetPhbaProperties

6.2.46.1 Synopsis

Gets the general properties of a physical HBA.

6.2.46.2 Prototype

```
IMA_STATUS IMA_GetPhbaProperties(
    /* in */     IMA_OID phbaOid,
    /* out */    IMA_PHBA_PROPERTIES *pProps
);
```

6.2.46.3 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*.

6.2.46.4 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.

6.2.46.5 Remarks

None

6.2.46.6 Support

Mandatory

6.2.46.7 See Also

IMA_GetPhbaOidList

Example of Getting PHBA Properties

6.2.47 IMA_GetPhbaStatus

6.2.47.1 Synopsis

Gets the status of a specified physical HBA.

6.2.47.2 Prototype

```
IMA_STATUS IMA_GetPhbaStatus(
    /* in */     IMA_OID hbaOid,
    /* out */    IMA_PHBA_STATUS *pStatus
);
```

6.2.47.3 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.

6.2.47.4 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.

6.2.47.5 Remarks

None

6.2.47.6 Support

Mandatory for PHBAs.

6.2.48 IMA_GetPluginOidList

6.2.48.1 Synopsis

Gets a list of the object IDs of all currently loaded plugins.

6.2.48.2 Prototype

```
IMA_STATUS IMA_GetPluginOidList(
    /* out */    IMA_OID_LIST **ppList
);
```

6.2.48.3 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.

6.2.48.4 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.

6.2.48.5 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.

6.2.48.6 Support

Mandatory

6.2.48.7 See Also

IMA_FreeMemory

IMA_GetPluginProperties

Example of Getting Plugin Properties

6.2.49 IMA_GetPluginProperties

6.2.49.1 Synopsis

Gets the properties of the specified vendor plugin.

6.2.49.2 Prototype

```
IMA_STATUS IMA_GetPluginProperties(
    /* in */     IMA_OID pluginOid
    /* out */    IMA_PLUGIN_PROPERTIES *pProps
);
```

6.2.49.3 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*.

6.2.49.4 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.

6.2.49.5 Remarks

None

6.2.49.6 Support

Mandatory

6.2.49.7 See Also

IMA_GetAssociatedPluginOid
IMA_GetPluginOidList
Example of Getting Plugin Properties

6.2.50 IMA_GetPnpOidList

6.2.50.1 Synopsis

Gets a list of the object IDs of all the physical network ports associated with an object.

6.2.50.2 Prototype

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

6.2.50.3 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.

6.2.50.4 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.

6.2.50.5 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.

6.2.50.6 Support

Mandatory

6.2.50.7 See Also

IMA_FreeMemory

IMA_GetPnpProperties

6.2.51 IMA_GetPnpProperties

6.2.51.1 Synopsis

Gets the properties of the specified physical network port.

6.2.51.2 Prototype

```
IMA_STATUS IMA_GetPnpProperties(
    /* in */    IMA_OID pnpOid,
    /* out */   IMA_PNP_PROPERTIES *pProps
);
```

6.2.51.3 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*.

6.2.51.4 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.

6.2.51.5 Remarks

None

6.2.51.6 Support

Mandatory

6.2.52 IMA_GetPnpStatistics

6.2.52.1 Synopsis

Gets the statistics related to a physical network port.

6.2.52.2 Prototype

```
IMA_STATUS IMA_GetPnpStatistics(
    /* in */     IMA_OID pnpOid,
    /* out */    IMA_PNP_STATISTICS *pStats
);
```

6.2.52.3 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.

6.2.52.4 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*.

6.2.52.5 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.

6.2.52.6 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.

6.2.52.7 See Also

IMA_STATISTICS_PROPERTIES

IMA_GetStatisticsProperties

IMA_SetStatisticsCollection

6.2.53 IMA_GetSharedNodeOid

6.2.53.1 Synopsis

Gets the object ID of the shared node of the currently executing operating system image.

6.2.53.2 Prototype

```
IMA_STATUS IMA_GetSharedNodeOid(
    /* out */      IMA_OID *pSharedNodeOid
);
```

6.2.53.3 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.

6.2.53.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.53.5 Remarks

None

6.2.53.6 Support

Mandatory

6.2.53.7 See Also

Concepts: The Shared Node versus Non-shared Nodes

IMA_GetNonSharedNodeOidList

IMA_GetNodeProperties

Example of Getting Node Properties

Example of Setting a Node Name

6.2.54 IMA_GetStaticDiscoveryTargetOidList

6.2.54.1 Synopsis

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

6.2.54.2 Prototype

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

6.2.54.3 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.

6.2.54.4 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.

6.2.54.5 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.

6.2.54.6 Support

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

6.2.54.7 See Also

IMA_FreeMemory

IMA_AddStaticDiscoveryTarget

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetProperties

6.2.55 IMA_GetStaticDiscoveryTargetProperties

6.2.55.1 Synopsis

Gets the properties of the specified static discovery target.

6.2.55.2 Prototype

```
IMA_STATUS IMA_GetStaticDiscoveryTargetProperties(
    /* in */     IMA_OID staticDiscoveryTargetOid,
    /* out */    IMA_STATIC_DISCOVERY_TARGET_PROPERTIES *pProps
);
```

6.2.55.3 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*.

6.2.55.4 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.

6.2.55.5 Remarks

None

6.2.55.6 Support

Mandatory

6.2.55.7 See Also

IMA_AddStaticDiscoveryTarget

IMA_RemoveStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetOidList

6.2.56 IMA_GetStatisticsProperties

6.2.56.1 Synopsis

Gets the statistics properties of the specified target, logical unit, or physical network port.

6.2.56.2 Prototype

```
IMA_STATUS IMA_GetStatisticsProperties(
    /* in */      IMA_OID oid,
    /* out */     IMA_STATISTICS_PROPERTIES *pProps
);
```

6.2.56.3 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.

6.2.56.4 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.

6.2.56.5 Remarks

The setting of this value is persistent.

6.2.56.6 Support

Mandatory

6.2.56.7 See Also

IMA_GetDeviceStatistics

IMA_GetTargetErrorStatistics

IMA_SetStatisticsCollection

6.2.57 IMA_GetSupportedAuthMethods

6.2.57.1 Synopsis

Gets a list of the authentication methods supported by the specified logical HBA.

6.2.57.2 Prototype

```
IMA_STATUS IMA_GetSupportedAuthMethods(
    /* in */        IMA_OID lhbaOid,
    /* in */        IMA_BOOL getSettableMethods,
    /* in, out */ IMA_UINT *pMethodCount,
    /* out */       IMA_AUTHMETHOD *pMethodList;
);
```

6.2.57.3 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.

6.2.57.4 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.

6.2.57.5 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.

6.2.57.6 Support

Mandatory

6.2.57.7 See Also

IMA_GetInUseInitiatorAuthMethods

IMA_SetInitiatorAuthMethods

6.2.58 IMA_GetTargetErrorStatistics

6.2.58.1 Synopsis

Gets the error statistics of the specified target.

6.2.58.2 Prototype

```
IMA_STATUS IMA_GetTargetErrorStatistics(
    /* in */     IMA_OID targetOid,
    /* out */    IMA_TARGET_ERROR_STATISTICS *pStats
);
```

6.2.58.3 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.

6.2.58.4 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.

6.2.58.5 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.

6.2.58.6 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.

6.2.58.7 See Also

IMA_STATISTICS_PROPERTIES

IMA_GetStatisticsProperties

IMA_GetTargetOidList

IMA_SetStatisticsCollection

6.2.59 IMA_GetTargetOidList

6.2.59.1 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.

6.2.59.2 Prototype

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

6.2.59.3 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.

6.2.59.4 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.

6.2.59.5 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.

6.2.59.6 Support

Mandatory

6.2.59.7 See Also

Concepts: Target Object IDs and Logical Unit Object IDs

IMA_FreeMemory

IMA_GetTargetProperties

6.2.60 IMA_GetTargetProperties

6.2.60.1 Synopsis

Gets the properties of the specified target.

6.2.60.2 Prototype

```
IMA_STATUS IMA_GetTargetProperties(
    /* in */     IMA_OID targetOid,
    /* out */    IMA_TARGET_PROPERTIES *pProps
);
```

6.2.60.3 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.

6.2.60.4 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.

6.2.60.5 Remarks

This call can force an iSCSI login to the specified target.

6.2.60.6 Support

Mandatory

6.2.60.7 See Also

IMA_GetTargetOidList

6.2.61 IMA_IsPhbaDownloadFile

6.2.61.1 Synopsis

Determines if a file is suitable for download to a physical HBA.

6.2.61.2 Prototype

```
IMA_STATUS IMA_IsPhbaDownloadFile(
    /* in */     IMA_OID phbaOid,
    /* in */     const IMA_WCHAR *pFileName,
    /* out */    IMA_PHBA_DOWNLOAD_IMAGE_PROPERTIES *pProps
);
```

6.2.61.3 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.

6.2.61.4 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.

6.2.61.5 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.

6.2.61.6 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.

6.2.61.7 See Also

IMA_GetPhbaDownloadProperties

IMA_PhbaDownload

6.2.62 IMA_LuInquiry

6.2.62.1 Synopsis

Gets the specified SCSI INQUIRY data for the specified logical unit.

6.2.62.2 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
);
```

6.2.62.3 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.

6.2.62.4 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.

6.2.62.5 Remarks

None

6.2.62.6 Support

Mandatory

6.2.62.7 See Also

IMA_LuReadCapacity

IMA_LuReportLuns

6.2.63 IMA_LuReadCapacity

6.2.63.1 Synopsis

Gets the SCSI READ CAPACITY data for the specified logical unit.

6.2.63.2 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
);
```

6.2.63.3 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.

6.2.63.4 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.

6.2.63.5 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.

6.2.63.6 Support

Mandatory

6.2.63.7 See Also

IMA_LuInquiry

IMA_LuReportLuns

6.2.64 IMA_LuReportLuns

6.2.64.1 Synopsis

Gets the SCSI REPORT LUNS data for the specified logical unit.

6.2.64.2 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
);
```

6.2.64.3 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 Clause 8 of ISO/IEC 14776-453 for more information regarding well known LUs.

selectReport

The select report value as defined for the REPORT LUNS command. See ISO/IEC 14776-453 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.

6.2.64.4 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.

6.2.64.5 Remarks

None

6.2.64.6 Support

Mandatory in all situations except when *oid* specifies a target and *sendToWellKnownLu* has the value IMA_TRUE. Supporting this situation is optional.

6.2.64.7 See Also

IMA_LuInquiry

IMA_LuReadCapacity

6.2.65 IMA_PhbaDownload

6.2.65.1 Synopsis

Downloads the image in the specified file to the specified physical HBA.

6.2.65.2 Prototype

```
IMA_STATUS IMA_PhbaDownload(
    /* in */     IMA_OID phbaOid,
    /* in */     IMA_PHBA_DOWNLOAD_IMAGE_TYPE imageType,
    /* in */     const IMA_WCHAR *pFileName
);
```

6.2.65.3 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.

6.2.65.4 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.

6.2.65.5 Remarks

None

6.2.65.6 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.

6.2.65.7 See Also

IMA_PHBA_DOWNLOAD_IMAGE_TYPE

IMA_GetPhbaDownloadProperties

IMA_IsPhbaDownloadFile

6.2.66 IMA_PluginIOCtl

6.2.66.1 Synopsis

Sends a vendor specific command to a specified plugin.

6.2.66.2 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,
);
```

6.2.66.3 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.

6.2.66.4 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.

6.2.66.5 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.

6.2.66.6 Support

Optional

6.2.66.7 See Also

IMA_PLUGIN_PROPERTIES

IMA_GetPluginOidList

IMA_GetPluginProperties

6.2.67 IMA_RegisterForObjectPropertyChanges

6.2.67.1 Synopsis

Registers a client function to be called whenever the property of an object changes.

6.2.67.2 Prototype

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

6.2.67.3 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.

6.2.67.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.67.5 Support

Mandatory

6.2.67.6 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.

6.2.67.7 See Also

IMA_DeregisterForObjectPropertyChanges

6.2.68 IMA_RegisterForObjectVisibilityChanges

6.2.68.1 Synopsis

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

6.2.68.2 Prototype

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

6.2.68.3 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.

6.2.68.4 Typical return values

IMA_ERROR_INVALID_PARAMETER

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

6.2.68.5 Support

Mandatory

6.2.68.6 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)

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.

6.2.68.7 See Also

IMA_DeregisterForObjectVisibilityChanges

6.2.69 IMA_RemoveDiscoveryAddress

6.2.69.1 Synopsis

Removes a discovery address being used for send targets discovery by a physical network port or logical HBA.

6.2.69.2 Prototype

```
IMA_STATUS IMA_RemoveDiscoveryAddress(
    /* in */       IMA_OID discoveryAddressOid
);
```

6.2.69.3 Parameters

discoveryAddressOid

The object ID of the discovery address that is to no longer be used for send targets discovery.

6.2.69.4 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.

6.2.69.5 Remarks

This change is persistent. The specified discovery address will no longer be used by the associated PNP or LHBA for send targets discovery.

6.2.69.6 Support

Mandatory

6.2.69.7 See Also

IMA_AddDiscoveryAddress

IMA_GetDiscoveryAddressProperties

IMA_GetDiscoveryProperties

IMA_SetSendTargetsDiscovery

6.2.70 IMA_RemoveStaleData

6.2.70.1 Synopsis

Removes all of the stale persistent data of the specified LHBA.

6.2.70.2 Prototype

```
IMA_STATUS IMA_RemoveStaleData(
    /* in */        IMA_OID lhbaOid,
);
```

6.2.70.3 Parameters

lhbaOid

The object ID of the LHBA whose stale data is to be removed.

6.2.70.4 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.

6.2.70.5 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.

6.2.70.6 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.

6.2.70.7 See Also

IMA_GetLhbaProperties

6.2.71 IMA_RemoveStaticDiscoveryTarget

6.2.71.1 Synopsis

Removes a target being statically discovered by a physical network port or logical HBA.

6.2.71.2 Prototype

```
IMA_STATUS IMA_RemoveStaticDiscoveryTarget(
    /* in */       IMA_OID staticDiscoveryTargetOid
);
```

6.2.71.3 Parameters

staticDiscoveryTargetOid

The object ID of the static discovery target that is to no longer be discovered.

6.2.71.4 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.

6.2.71.5 Remarks

This change is persistent. The specified target will no longer be discovered by the associated PHBA or LHBA.

6.2.71.6 Support

Mandatory

6.2.71.7 See Also

IMA_AddStaticDiscoveryTarget

IMA_GetStaticDiscoveryTargetProperties

IMA_GetDiscoveryProperties

IMA_SetStaticDiscovery

6.2.72 IMA_SetDataPduInOrder

6.2.72.1 Synopsis

Sets the `DataPDUInOrder` iSCSI login parameter value for the specified logical HBA or target.

6.2.72.2 Prototype

```
IMA_STATUS IMA_SetDataPduInOrder(
    /* in */      IMA_OID oid,
    /* in */      IMA_BOOL dataPduInOrder
);
```

6.2.72.3 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.

6.2.72.4 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.

6.2.72.5 Remarks

The setting of this value is persistent and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

6.2.72.6 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.

6.2.72.7 See Also

IMA_GetDataPduInOrderProperties

iSCSI Session and Connection Parameters

6.2.73 IMA_SetDataSequenceInOrder

6.2.73.1 Synopsis

Sets the `DataSequenceInOrder` iSCSI login parameter value for the specified logical HBA or target.

6.2.73.2 Prototype

```
IMA_STATUS IMA_SetDataSequenceInOrder(
    /* in */      IMA_OID oid,
    /* in */      IMA_BOOL dataSequenceInOrder
);
```

6.2.73.3 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.

6.2.73.4 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.

6.2.73.5 Remarks

The setting of this value is persistent and applies only to subsequent sessions created by the initiator; existing sessions are not affected.

6.2.73.6 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.

6.2.73.7 See Also

IMA_GetDataSequenceInOrderProperties

iSCSI Session and Connection Parameters

6.2.74 IMA_SetDefaultGateway

6.2.74.1 Synopsis

Sets the default gateway for the the specified physical network port.

6.2.74.2 Prototype

```
IMA_STATUS IMA_SetDefaultGateway(
    /* in */        IMA_OID oid,
    /* in */        IMA_IP_ADDRESS defaultGateway
);
```

6.2.74.3 Parameters

oid

The object ID of the PNP whose default gateway is to be set.

defaultGateway

The new default gateway for the PNP.

6.2.74.4 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.

6.2.74.5 Remarks

The setting of this value is persistent.

6.2.74.6 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.

6.2.74.7 See Also

IMA_GetIpProperties

6.2.75 IMA_SetDefaultTime2Retain

6.2.75.1 Synopsis

Sets the `DefaultTime2Retain` iSCSI login parameter value for the specified logical HBA or target.

6.2.75.2 Prototype

```
IMA_STATUS IMA_SetDefaultTime2Retain(
    /* in */      IMA_OID oid,
    /* in */      IMA_UINT defaultTime2Retain
);
```

6.2.75.3 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.

6.2.75.4 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.

6.2.75.5 Remarks

The setting of this value is persistent 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.

6.2.75.6 Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetDefaultTime2RetainProperties for the same *oid* is true.

6.2.75.7 See Also

IMA_GetDefaultTime2RetainProperties

iSCSI Session and Connection Parameters

6.2.76 IMA_SetDefaultTime2Wait

6.2.76.1 Synopsis

Sets the `DefaultTime2Wait` iSCSI login parameter value for the specified logical HBA or target.

6.2.76.2 Prototype

```
IMA_STATUS IMA_SetDefaultTime2Wait(
    /* in */      IMA_OID oid,
    /* in */      IMA_UINT defaultTime2Wait
);
```

6.2.76.3 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.

6.2.76.4 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.

6.2.76.5 Remarks

The setting of this value is persistent 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.

6.2.76.6 Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetDefaultTime2WaitProperties for the same *oid* is true.

6.2.76.7 See Also

IMA_GetDefaultTime2WaitProperties

iSCSI Session and Connection Parameters

6.2.77 IMA_SetDnsServerAddress

6.2.77.1 Synopsis

Sets the address of the primary and alternate DNS servers for the specified physical network port.

6.2.77.2 Prototype

```
IMA_STATUS IMA_SetDnsServerAddress(
    /* in */      IMA_OID oid,
    /* in */      const IMA_IP_ADDRESS *pPrimaryDnsServerAddress,
    /* in */      const IMA_IP_ADDRESS *pAlternateDnsServerAddress
);
```

6.2.77.3 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.

6.2.77.4 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.

6.2.77.5 Remarks

The setting of these values is persistent.

6.2.77.6 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.

6.2.77.7 See Also

IMA_GetIpProperties

6.2.78 IMA_SetErrorRecoveryLevel

6.2.78.1 Synopsis

Sets the `ErrorRecoveryLevel` iSCSI login parameter value for the specified logical HBA or target.

6.2.78.2 Prototype

```
IMA_STATUS IMA_SetErrorRecoveryLevel(
    /* in */      IMA_OID oid,
    [in[          IMA_UINT errorRecoveryLevel
);
```

6.2.78.3 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.

6.2.78.4 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.

6.2.78.5 Remarks

The setting of this value is persistent 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.

6.2.78.6 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.

6.2.78.7 See Also

IMA_GetErrorRecoveryLevelProperties

iSCSI Session and Connection Parameters

6.2.79 IMA_SetFirstBurstLength

6.2.79.1 Synopsis

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

6.2.79.2 Prototype

```
IMA_STATUS IMA_SetFirstBurstLength(
    /* in */      IMA_OID oid,
    /* in */      IMA_UINT firstBurstLength
);
```

6.2.79.3 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.

6.2.79.4 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.

6.2.79.5 Remarks

The setting of this value is persistent 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.

6.2.79.6 Support

Mandatory if the *settable* field in the IMA_MIN_MAX_VALUE structure returned by IMA_GetFirstBurstLengthProperties for the same *oid* is true.