Storage Management Technical Specification, Part 1 Common Architecture

Version 1.3.0, Rev 6

SNIA Technical Position

21 April, 2009

Revision History

Revison 1

Date:

4 January, 2007

SCRs Incorporated:

Comments

Editorial comments displayed.

Revison 2

Date

14 April 2007

SCRs Incorporated and other changes

Installation Clause

- Added text for Multi-tenant CIMOM & Multiple CIMOM Coexistence support (MP-SMIS-SCR00039) (6-1-0)

Deploying SMI-S with DMTF Profiles Annex

- Added this new annex (SMIS-130-Draft-SCR00010) (4-0-0)

Comments

Only minor editorial work for this revision.

Revison 3

Date

19 June 2007

SCRs Incorporated and other changes

Front Matter: Typographical Conventions

- Updated the description of Experimental (SMIS-120-Errata-SCR00061) (12-1-1)

Clause 9: Standard Messages

- Updated the list of Common Messages, 9.3.1 (CORE-SMIS-SCR-00028) (1-0-2)

- Added FC Messages in Message Registry, 9.3.3 (FC-SMIS- SCR00034) (2-0-0)

Comments

Editorial notes displayed.

Responses to INCITS editor queries re SMI-S 1.1.0 incorporated as applicable.

Typographical Conventions revised in all books: Revised explanation of Experimental text (per SMIS-120-Errata-SCR00061 - Typographical Conventions), added explanations of Draft and Editorial text.

Revision 4

Date

20 July 2007

SCRs Incorporated and other changes

Clause 9: Standard Messages

- Updated & **promoted to Experimental** the list of Common Messages, 9.3.1 (CORE-SMIS-SCR-00028) (2-1-0)

- **Promoted to Experimental** the Fabric Message Registry (SMIS-130-Draft-SCR00016) (3-0-0)

Added an annex describing the WQL and CQL filter string patterns used in SMI-S.

Comments

Editorial notes displayed, but the DRAFT material is not.

Revision 5

Date

14 November 2007

SCRs Incorporated and other changes

None

Comments

Editorial notes and DRAFT material are not displayed.

Revision 6

Date

14 January 2009

SCRs Incorporated and other changes

References to *Storage Management Technical Specification, Part 7 Information Lifecycle Management*, deleted.

Fixed Incorrect HealthState values (SMIS-130-Errata-SCR00004)

CQL mandatory for new indication filters (SMIS-130-Errata-SCR00006)

Clarified logic for addressing the SLP overflow bit in Discovery Clause (SMIS-130-Errata-SCR00015)

Comments

Editorial notes and DRAFT material are not displayed.

Suggestion for changes or modifications to this document should be sent to the SNIA Storage Management Initiative Technical Steering Group (SMI-TSG) at http://www.snia.org/feedback/.

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

1)   Any text, diagram, chart, table or definition reproduced must be reproduced in its entirety with no alteration, and,

2)   Any document, printed or electronic, in which material from this document (or any portion hereof) is repro-duced must acknowledge the SNIA copyright on that material, and must credit the SNIA for granting permis-sion for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing tcmd@snia.org please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

INTENDED AUDIENCE

This document is intended for use by individuals and companies engaged in developing, deploying, and promoting interoperable multi-vendor SANs through the SNIA organization.

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to http://www.snia.org/feedback/.

Portions of the CIM Schema are used in this document with the permission of the Distributed Management Task Force (DMTF). The CIM classes that are documented have been developed and reviewed by both the Storage Networking Industry Association (SNIA) and DMTF Technical Working Groups. However, the schema is still in development and review in the DMTF Working Groups and Technical Committee, and subject to change.

CHANGES TO THE SPECIFICATION

Each publication of this specification is uniquely identified by a three-level identifier, comprised of a version number, a release number and an update number. The current identifier for this specification is version 1.2.0. Future publications of this specification are subject to specific constraints on the scope of change that is permissible from one publication to the next and the degree of interoperability and backward compatibility that should be assumed between products designed to different publications of this standard. The SNIA has defined three levels of change to a specification:

- Major Revision: A major revision of the specification represents a substantial change to the underlying scope or architecture of the SMI-S API. A major revision results in an increase in the version number of the version identifier (e.g., from version 1.x.x to version 2.x x). There is no assurance of interoperability or backward compatibility between releases with different version numbers.

- Minor Revision: A minor revision of the specification represents a technical change to existing content or an adjustment to the scope of the SMI-S API. A minor revision results in an increase in the release number of the specification's identifier (e.g., from x.1.x to x.2.x). Minor revisions with the same version number preserve interoperability and backward compatibility.

- Update: An update to the specification is limited to minor corrections or clarifications of existing specification content. An update will result in an increase in the third component of the release identifier (e.g., from x.x.1 to x.x.2). Updates with the same version and minor release levels preserve interoperability and backward compatibility.

TYPOGRAPHICAL CONVENTIONS

This specification has been structured to convey both the formal requirements and assumptions of the SMI-S API and its emerging implementation and deployment lifecycle. Over time, the intent is that all content in the specification will represent a mature and stable design, be verified by extensive implementation experience, assure consistent support for backward compatibility, and rely solely on content material that has reached a similar level of maturity. Unless explicitly labeled with one of the subordinate maturity levels defined for this specification, content is assumed to satisfy these requirements and is referred to as "Finalized". Since much of the evolving specification

content in any given release will not have matured to that level, this specification defines three subordinate levels of implementation maturity that identify important aspects of the content's increasing maturity and stability. Each subordinate maturity level is defined by its level of implementation experience, its stability and its reliance on other

emerging standards. Each subordinate maturity level is identified by a unique typographical tagging convention that clearly distinguishes content at one maturity model from content at another level.

Experimental Maturity Level

No material is included in this specification unless its initial architecture has been completed and reviewed. Some content included in this specification has complete and reviewed design, but lacks implementation experience and the maturity gained through implementation experience. This content is included in order to gain wider review and to gain implementation experience. This material is referred to as "Experimental". It is presented here as an aid to implementers who are interested in likely future developments within the SMI specification. The contents of an Experimental profile may change as implementation experience is gained. There is a high likelihood that the changed content will be included in an upcoming revision of the specification. Experimental material can advance to a higher maturity level as soon as implementations are available. Figure 1 is a sample of the typographical convention for Experimental content.
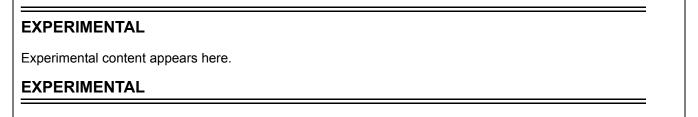

EXPERIMENTAL

Experimental content appears here.

EXPERIMENTAL

Figure 1 - Experimental Maturity Level Tag

Implemented Maturity Level

Profiles for which initial implementations have been completed are classified as "Implemented". This indicates that at least two different vendors have implemented the profile, including at least one provider implementation. At this maturity level, the underlying architecture and modeling are stable, and changes in future revisions will be limited to the correction of deficiencies identified through additional implementation experience. Should the material become obsolete in the future, it must be deprecated in a minor revision of the specification prior to its removal from subsequent releases. Figure 2 is a sample of the typographical convention for Implemented content.
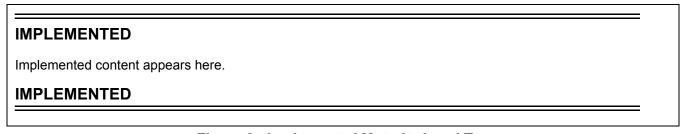

IMPLEMENTED

Implemented content appears here.

IMPLEMENTED

Figure 2 - Implemented Maturity Level Tag

Stable Maturity Level

Once content at the Implemented maturity level has garnered additional implementation experience, it can be tagged at the Stable maturity level. Material at this maturity level has been implemented by three different vendors, including both a provider and a client. Should material that has reached this maturity level become obsolete, it may only be deprecated as part of a minor revision to the specification. Material at this maturity level that has been deprecated may only be removed from the specification as part of a major revision. A profile that has reached this maturity level is guaranteed to preserve backward compatibility from one minor specification revision to the next. As a result, Profiles at or above the Stable maturity level shall not rely on any content that is Experimental. Figure 3 is a sample of the typographical convention for Implemented content.
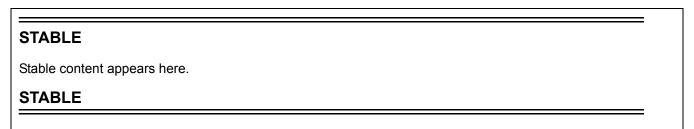

STABLE

Stable content appears here.

STABLE

Figure 3 - Stable Maturity Level Tag

Finalized Maturity Level

Content that has reached the highest maturity level is referred to as "Finalized." In addition to satisfying the requirements for the Stable maturity level, content at the Finalized maturity level must solely depend upon or refine material that has also reached the Finalized level. If specification content depends upon material that is not under the control of the SNIA, and therefore not subject to its maturity level definitions, then the external content is evaluated by the SNIA to assure that it has achieved a comparable level of completion, stability, and implementation experience. Should material that has reached this maturity level become obsolete, it may only be deprecated as part of a major revision to the specification. A profile that has reached this maturity level is guaranteed to preserve backward compatibility from one minor specification revision to the next. Over time, it is hoped that all specification content will attain this maturity level. Accordingly, there is no special typographical convention, as there is with the other, subordinate maturity levels. Unless content in the specification is marked with one of the typographical conventions defined for the subordinate maturity levels, it should be assumed to have reached the Finalized maturity level.

Deprecated Material

Non-Experimental material can be deprecated in a subsequent revision of the specification. Sections identified as "Deprecated" contain material that is obsolete and not recommended for use in new development efforts. Existing and new implementations may still use this material, but shall move to the newer approach as soon as possible. The maturity level of the material being deprecated determines how long it will continue to appear in the specification. Implemented content shall be retained at least until the next revision of the specialization, while Stable and Finalized material shall be retained until the next major revision of the specification. Providers shall implement the deprecated elements as long as it appears in the specification in order to achieve backward compatibility. Clients may rely on deprecated elements, but are encouraged to use non-deprecated alternatives when possible.

Deprecated sections are documented with a reference to the last published version to include the deprecated section as normative material and to the section in the current specification with the replacement. Figure 4 contains a sample of the typographical convention for deprecated content.

DEPRECATED

Content that has been deprecated appears here.

DEPRECATED

Figure 4 - Deprecated Tag

USAGE

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

3)   Any text, diagram, chart, table or definition reproduced shall be reproduced in its entirety with no alteration.

4)   Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing tcmd@snia.org please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

Contents

List of Tables

List of Figures

Foreword

Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6 defines the core architecture of SMI-S. This includes the protocols (WBEM, SLP,...); the model is defined in the other specification parts.

Parts of this Standard

This standard is subdivided in the following parts:

• *Storage Management Technical Specification, Overview, 1.3.0 Rev 6*

• *Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6*

• *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6*

• *Storage Management Technical Specification, Part 3 Block Devices, 1.3.0 Rev 6*

• *Storage Management Technical Specification, Part 4 File Systems, 1.3.0 Rev 6*

• *Storage Management Technical Specification, Part 5 Fabric, 1.3.0 Rev 6*

• *Storage Management Technical Specification, Part 6 Host Elements, 1.3.0 Rev 6*

• *Storage Management Technical Specification, Part 7 Media Libraries, 1.3.0 Rev 6*

SNIA Web Site

Current SNIA practice is to make updates and other information available through their web site at http://www.snia.org

SNIA Address

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent via the SNIA Feedback Portal at http://www.snia.org/feedback/ or by mail to the Storage Networking Industry Association, 500 Sansome Street, Suite #504, San Francisco, CA 94111, U.S.A.

Acknowledgments

The SNIA SMI Technical Steering Group, which developed and reviewed this standard, would like to recognize the significant contributions made by the following members:

Organization Represented	*Name of Representative*
Brocade	John Crandall
Dell	Vance Corn
EMC	Mike Thompson
Hewlett Packard	Alex Lenart
	Steve Peters
Hitachi Data Systems	Steve Quinn
Individual member	Tom West
IBM	Krishna Harathi
	Mike Walker
	Martine Wedlake
Olocity	Scott Baker
Pillar	Gary Steffens
Symantec	Steve Hand
	Paul von Behren

xx

Clause 1: Scope

Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6 defines the core architecture and protocols in SMI-S. The components of SMI-S architecture include:

- Transport - communicating management information between constituents of the management system

- Health and fault management - detecting failures through monitoring the state of storage components

- General information about the object model

- Names - how SMI-S uses names to allow applications to correlate across SMI-S and to other standards

- Policy - the expression of management behavior such that administrators and other management software can control that behavior, tailoring it to accomplish specific goals.

- Standard messages - how exceptions are presented to client applications

- Service discovery - techniques clients use to discover SMI-S services

- Installation and upgrade - recommendations for implementations

- Compliance - requirement for compliance to the standard

Clause 2: Normative references

2.1   General

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.

2.2   Approved references

ISO/IEC 14776-413, SCSI Architecture Model - 3 (SAM-3) [ANSI INCITS 402-200x]

ISO/IEC 14776-452, SCSI Primary Commands - 3 (SPC-3) [ANSI INCITS.351-2005]

ANSI/INCITS 374:2003, Information technology - Fibre Channel Single - Byte Command Set-3 (FC-SB-3)

2.3   DMTF references (Final)

DMTF Final documents are accepted as standards. For DMTF Draft or Preliminary documents, see 2.5.

DMTF DSP0004 CIM Infrastructure Specification 2.3.0
http://www.dmtf.org/standards/published_documents/DSP0004V2.3_final.pdf

DMTF DSP0200 CIM Operations over HTTP 1.1.0
http://www.dmtf.org/standards/documents/WBEM/DSP200.html

DMTF DSP0201 Representation of CIM in XML 2.2.0
http://www.dmtf.org/standards/documents/WBEM/DSP201.htm

DMTF DSP0202 CIM Query Language Specification 1.0.0
http://www.dmtf.org/standards/published_documents/DSP0202_1.0.0.pdf

DMTF CIM Schema 2.15.0
http://www.dmtf.org/standards/cim/cim_schema_v215

2.4   IETF references (standards or draft standards)

For IETF Informational documents and proposed standards, see 2.5.

IETF RFC 2045, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
http://www.ietf.org/rfc/rfc2045.txt

IETF RFC 2246, The TLS Protocol Version 1.0
http://www.ietf.org/rfc/rfc2246.txt

IETF RFC 4291, IP Version 6 Addressing Architecture

IETF RFC 2396, Uniform Resource Identifiers (URI)
http://www.ietf.org/rfc/rfc2396.txt

IETF RFC 2608, Service Location Protocol, Version 2
http://www.ietf.org/rfc/rfc2608.txt

IETF RFC 2609, Service Templates and Service: Schemes
http://www.ietf.org/rfc/rfc2609.txt

IETF RFC 2610, DHCP Options for Service Location Protocol
http://www.ietf.org/rfc/rfc2610.txt

IETF RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1
http://www.ietf.org/rfc/rfc2616.txt

IETF RFC 2617, HTTP Authentication: Basic ad Digest Access Authentication
http://www.ietf.org/rfc/rfc2617.txt

IETF RFC 2445, Internet Calendaring and Scheduling Core Object Specification (iCalendar)
http://www.ietf.org/rfc/rfc2445.txt

IETF RFC 3280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
http://www.ietf.org/rfc/rfc3280.txt

IETF RFC 3723, Securing Block Storage Protocols over IP
http://www.ietf.org/rfc/rfc3723.txt

IETF RFC 3986, Definitions of Managed Objects for the DS3/E3 Interface Type
http://www.ietf.org/rfc/rfc3986.txt

IETF RFC 4291, IP Version 6 Addressing Architecture
http://www.ietf.org/rfc/rfc4291.txt

IETF RFC 4346, The Transport Layer Security (TLS) Protocol Version 1.1
http://www.ietf.org/rfc/rfc4346.txt

IETF RFC 4514, Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names
http://www.ietf.org/rfc/rfc4514.txt

2.5   References under development

The following documents (and their web addresses) are subject to change.

DMTF DSP0200 CIM Operations over HTTP 1.2
http://www.dmtf.org/standards/published_documents/DSP200.pdf

DMTF DSP0201, Representation of CIM in XML 2.2
http://www.dmtf.org/standards/wbem/DSP201.html

DMTF DSP0225, URI Format for DMTF Published XML Schema
http://www.dmtf.org/standards/published_documents/DSP0225.pdf

DMTF DSP0226, WS-Management Protocol Specification

DMTF DSP0230, WS-CIM Mapping Specification

Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6

2.6   Other references

IETF RFC 1945 Hypertext Transfer Protocol -- HTTP/1.0
http://www.ietf.org/rfc/rfc1945.txt

IETF RFC 2614 An API for Service Location
http://www.ietf.org/rfc/rfc2614.txt

SSL 3.0 Draft Specification
http://wp.netscape.com/eng/ssl3/

UML (Universal Modeling Language) Specifications
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

4

ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework

PKCS #12, Personal Information Exchange Syntax
http://www.rsasecurity.com/rsalabs/node.asp?id=2138

Clause 3: Definitions, symbols, abbreviations, and conventions

3.1   Definitions

3.1.1   agent
An Object Manager that includes the provider service for a limited set of resources. An Agent may be embedded or hosted and can be an aggregator for multiple devices.

3.1.2   aggregation
A strong form of an association. For example, the containment relationship between a system and the components that make up the system can be called an aggregation. An aggregation is expressed as a Qualifier on the association class. Aggregation often implies, but does not require, that the aggregated objects have mutual dependencies.

3.1.3   CIM Server
A Server that provides support for CIM requests and provides CIM responses.

3.1.4   Common Information Model (CIM)
An object-oriented description of the entities and relationships in a business' management environment maintained by the Distributed Management Task Force. CIM is divided into a Core Model and Common Models. The Core Model addresses high-level concepts (such as systems and devices), as well as fundamental relationships (such as dependencies). The Common Models describe specific problem domains such as computer system, network, user or device management. The Common Models are subclasses of the Core Model and may also be subclasses of each other.

3.1.5   client
A process that issues requests for service. Formulating and issuing requests may involve multiple client processes distributed over one or more computer systems.

3.1.6   dedicated SMI-S Server
A CIM Server that is dedicated to supporting a single device or subsystem.

3.1.7   dynamic host control protocol (DHCP)
An Internet protocol that allows nodes to dynamically acquire ("lease") network addresses for periods of time rather than having to pre-configure them.

3.1.8   discovery
Discovery provides information about what physical and logical storage entities have been found within the management domain.

3.1.9   Distributed Management Task Force (DMTF)
An industry organization that develops management standards for computer system and enterprise environments. DMTF standards include WBEM, CIM, DMI, DEN and ARM.

3.1.10   embedded SMI-S Server
A CIM Server that is embedded in the device or subsystem for which it provides management.

3.1.11   enclosure
A box or cabinet.

3.1.12   enumerate

This operation is used to enumerate subclasses, subclass names, instances and instance names in the target Namespace. If successful, the method returns zero or more requested elements that meet the required criteria.

3.1.13   event

An occurrence of a phenomenon of interest.

3.1.14   extent

A set of consecutively addressed disk blocks.

3.1.15   extrinsic method

A method defined as part of CIM Schema.

3.1.16   fabric

Any interconnect between two or more Fibre Channel N_Ports, including point-to-point, loop, and Switched Fabric.

3.1.17   FICON™[1]

Fibre Channel storage protocol used in IBM mainframe computers and peripheral devices such as ECKD storage arrays and tape drives.

3.1.18   general purpose SMI-S Server

An SMI-S Server that is not dedicated to supporting a single device or subsystem, and may support multiple devices or subsystems.

3.1.19   grammar

A formal definition of the syntactic structure of a language (see 3.1.50), normally given in terms of production rules that specify the order of constituents and their sub-constituents in a sentence (a well-formed string in the language).

3.1.20   host bus adapter (HBA)

Card that contains ports for host systems.

3.1.21   Hypertext Transfer Protocol (HTTP)

Request-reply protocol used for internet communications.

3.1.22   interconnect element

Non terminal network elements (Switches, hubs, routers, directors).

3.1.23   interface definition language (IDL)

High-level declarative language that provides the syntax for interface declarations.

3.1.24   intrinsic method

Operations made against a CIM server and a CIM namespace independent of the implementation of the schema defined in the server.

3.1.25   logical unit number (LUN)

Abbreviation for a SCSI logical unit or logical unit number.

1.FICON™ is an example of a suitable product available commercially. This information is given for the convenience of users of this standard and does not constitute an endorsement of this product by SNIA or any standards organization.

3.1.26   Network Address Authority (NAA)

A four bit identifier to denote a network address authority (i.e., an organization such as CCITT or IEEE that administers network addresses).

3.1.27   out-of-band

Transmission of management information for storage components outside of the data path, typically over Ethernet.

3.1.28   path

combination of initiator and target ports and logical unit

3.1.29   partition

collection of contiguous block on a disk or virtual disk

3.1.30   policy based

An SMI-S compliant implementation that supports one or more policy profiles directly.

3.1.31   policy client

A CIM Client that creates or manipulates instances of policy classes in a CIM Server.

3.1.32   policy enabled

An SMI-S profile, subprofile or package that includes properties and methods that are used in one or more policy profiles.

3.1.33   policy implementation

The implementation of policy class instances in a CIM Server (i.e. providers).

3.1.34   proxy SMI-S Server

An SMI-S Server that does not run on the device or subsystem which it supports. For example, a proxy SMI-S Server might run on a host system, but support a storage array.

3.1.35   public key infrastructure (PKI)

A framework established to issue, maintain, and revoke public key certificates accommodating a variety of security technologies.

3.1.36   protocol

A set of rules that define and constrain data, operations, or both. For example, xmlCIM uses XML as its transfer syntax, and HTTP as the request-reply protocol HTTP is layered over the TCP/IP network protocol.

3.1.37   SAN

A group of fabrics that have common leaf elements.

3.1.38   SB

Single-Byte Command Code Sets.

3.1.39   Service Access Point

The network address and port number of a process offering a service.

3.1.40   Storage Networking Industry Association (SNIA)

An association of producers and consumers of storage networking products whose goal is to further storage networking technology and applications.

3.1.41 Simple Network Management Protocol (SNMP)

An IETF protocol for monitoring and managing systems and devices in a network. The data being monitored and managed is defined by a MIB. The functions supported by the protocol are the request and retrieval of data, the setting or writing of data, and traps that signal the occurrence of events.

3.1.42 SMI-S Server

A CIM Server that supports SMI-S Profiles for management of a device or subsystem.

3.1.43 SNMP Trap

A type of SNMP message used to signal that an event has occurred.

3.1.44 Soft Zone

 A Zone consisting of Zone Members that are made visible to each other through Client Service requests. Typically, Soft Zones contain Zone Members that are visible to devices via Name Server exposure of Zone Members. The Fabric does not enforce a Soft Zone. Note that well known addresses are implicitly included in every Zone.

3.1.45 SCSI Parallel Interface.(SPI)

The family of SCSI standards that define the characteristics of the parallel version of the SCSI interface.

3.1.46 storage resource management (SRM)

Management of physical and logical storage resources, including storage elements, storage devices, appliances, virtual devices, disk volume and file resources.

3.1.47 Secure Sockets Layer (SSL)

A suite of cryptographic algorithms, protocols and procedures used to provide security for communications used to access the world wide web. More recent versions of SSL are known as TLS (Transport Level Security) and are standardized by the Internet Engineering Task Force (IETF)

3.1.48 Switch

Fibre channel interconnect element that supports a mesh topology.

3.1.49 Switched Fabric

A fabric comprised of one or more Switches

3.1.50 Syntax

The structure of strings in some language. A language's syntax is described by a grammar.

3.1.51 User Datagram Protocol (UDP)

An Internet protocol that provides connectionless datagram delivery service to applications

3.1.52 Web Based Enterprise Management (WBEM)

Web-Based Enterprise Management is an initiative in the DMTF. It is a set of technologies that enables interoperable management of an enterprise. WBEM consists of CIM, an XML DTD defining the tags (XML encodings) to describe the CIM Schema and its data, and a set of HTTP operations for exchanging the XML-based information.

3.1.53 eXtensible Markup Language (XML)

A universal format for structured documents and data on the World Wide Web.

3.1.54 Zone

A group of ports and switches that allow access. Defined by a zone definition.

3.1.55   Zone Set

One or more Zones that may be activated or deactivated as a group.

3.2   Acronyms and abbreviations

API	application programming interface
DHCP	dynamic host control protocol
FC	Fibre Channel
HBA	host bus adapter
IDL	interface definition language
IETF	Internet Engineering Task Force
IMA	iSCSI Management API
IP	Internet Protocol
iSCSI	Internet SCSI
OS	operating system
RFC	Request for Comments
SAM-3	SCSI Architecture Model
SAN	storage area network
SB	Single Byte (command set)
SCSI	Small Computer System Interface
SES	SCSI Enclosure Services
SLP	Service Location Protocol
SPC-3	SCSI Primary Commands-3
SSL	Secure Socket Layer
SSP	Storage Service Provider
TC	Technical Committee
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URL	Uniform Resource Locator.

3.3   Keywords

3.3.1   expected

A keyword used to describe the behavior of the hardware or software in the design models presumed by this standard. Other hardware and software design models may also be implemented.

3.3.2   invalid

A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.3.3   mandatory

A keyword indicating an item that is required to be implemented as defined in this standard to claim compliance with this standard.

3.3.4   may

A keyword that indicates flexibility of choice with no implied preference.

3.3.5   may not

Keywords that indicates flexibility of choice with no implied preference.

3.3.6   obsolete

A keyword indicating that an item was defined in prior standards but has been removed from this standard.

3.3.7   opaque:

A keyword indicating that value has no semantics or internal structure.

3.3.8   optional

A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined by this standard is implemented, it shall be implemented as defined in this standard.

3.3.9   reserved

A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. Their use and interpretation may be specified by future extensions to this or other standards. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

3.3.10   shall

A keyword indicating a mandatory requirement. Designers are required to implement all such requirements to ensure interoperability with other products that conform to this standard.

3.3.11   should

A keyword indicating flexibility of choice with a preferred alternative; equivalent to the phrase "it is recommended".

3.4   Conventions

Certain words and terms used in this American National Standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in Clause 3: Definitions, symbols, abbreviations, and conventions or in the text where they first appear.

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (xxb) are binary values.

Numbers immediately followed by lower-case h (xxh) are hexadecimal values.

Hexadecimal digits that are alphabetic characters are upper case (i.e., ABCDEF, not abcdef).

Hexadecimal numbers may be separated into groups of four digits by spaces. If the number is not a multiple of four digits, the first group may have fewer than four digits (e.g., AB CDEF 1234 5678h)

Decimal fractions are initiated with a comma (e.g., two and one half is represented as 2,5).

Decimal numbers having a value exceeding 999 are separated with a space(s) (e.g., 24 255).

Clause 4: Transport and Reference Model

4.1   Introduction

4.1.1   Overview

The interoperable management of storage devices and network elements in a distributed storage network requires a common transport for communicating management information between constituents of the management system. This section of the specification details the design of this transport, as well as the roles and responsibilities of constituents that use the common transport (i.e., a reference model).

4.1.2   Language Requirements

To express management information across the interface, a language is needed that:

•   Can contain platform independent data structures,

•   Is self describing and easy to debug,

•   Can be extended easily for future needs.

4.1.3   Communications Requirements

Communications protocols to carry the XML based management information are needed that:

•   Can take advantage of the existing ubiquitous IP protocol infrastructures,

•   Can be made to traverse inter- and intra-organizational firewalls,

•   Can easily be embedded in low cost devices.

The Hyper Text Transport Protocol (HTTP) was chosen for the messaging protocol and TCP was chosen for the base transfer protocol to carry the XML management information for this interface as they meet the requirements in 4.1.3.

4.1.4   XML Message Syntax and Semantics

In order to be successful, the expression of XML management information (messages) across this interface needs to follow consistent rules for semantics and syntax. These rules are detailed in this specification. They are of sufficient quality, extensibility, and completeness to allow their wide adoption by storage vendors and management software vendors in the industry. In addition, to facilitate rapid adoption, existing software that can parse, marshal, un-marshal, and interpret these XML messages should be widely available in the market such that vendor implementations of the interface are accelerated. The message syntax and semantics selected should:

•   Be available on multiple platforms,

•   Have software implementations that are Open source (i.e., collaborative code base),

•   Have software implementations available in Java and C++,

•   Leverage industry standards where applicable,

•   Conform with W3C standards for XML use.

•   Be object model independent (i.e., be able to express any object model).

Virtually the only existing industry standard in this area is the WBEM standards as developed and maintained by the DMTF.

4.2   Transport Stack

The complete transport stack for this interface is illustrated in Figure 5. It is the primary objective of this interface to drive seamless interoperability across vendors as communications technology and the object model underlying this interface evolves. Accordingly, the transport stack has been layered such that (if required) other protocols can be added as technology evolves. For example, should SOAP or IIOP become prominent, the content in the stack could be expanded with minimal changes to existing product implementations in the market.

EXPERIMENTAL
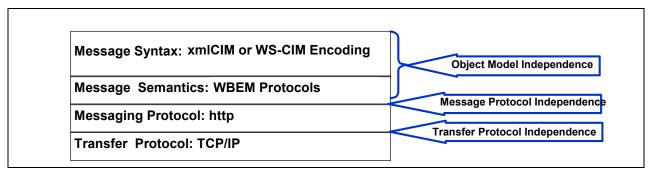
Figure 5 - Transport Stack

Implementers should be aware that an announced plan for converging web services standards is expected to cause changes to WS-Management, WSDM, and related protocols. SNIA intends to specify the resulting converged protocols for use with SMI-S, and hence use of web services protocols with SMI-S may remain Experimental until stable versions of the converged protocol specifications are available. Implementers are encouraged to experiment with web services protocols for SMI-S in the interim, but should consult the convergence plan to understand the potential protocol changes and possible impacts.

EXPERIMENTAL

This specification relies on the DMTF WBEM Protocol Specifications. Please refer to the DMTF WBEM Specification page for details on these specifications.

To be compliant with this specification ,CIM-XML shall be supported.

EXPERIMENTAL

Optionally, other protocols, such as WS-Management may also be supported.

EXPERIMENTAL

It should be noted that this specification places no restriction on the physical network selected to carry this transport stack. For example, a vendor can choose to use in-band communication over Fibre-channel as the backbone for this interface. Another vendor could exclusively (and wisely) choose out-of-band communication over Ethernet to implement this management interface. Additionally, select vendors could choose a mix of in-band and out-of-band physical network to carry this transport stack.

4.3   Reference Model

4.3.1   Overview

As shown in Figure 6, the Reference Model shows all possible constituents of the management environment in the presence of the transport stack for this interface.
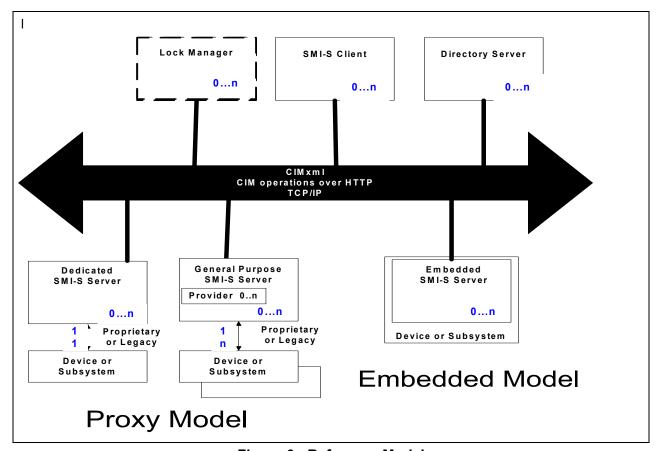
Figure 6 - Reference Model

Figure 6 illustrates that the transport for this interface uses a WBEM Protocol and HTTP/TCP/IP to execute intrinsic and extrinsic methods against the schema for this interface.

> **Note:** It is envisioned that a more complete version of this reference model would include a Lock Manager. However, in this version of SMI-S, a Lock Manager is not specified. As a result, it is shown as a dotted box to illustrate where the role would fit.

4.3.2   Roles for Interface Constituents

4.3.2.1   Client

A Client is the consumer of the management information in the environment. It provides an API (language binding in Java or C++ for example) for overlying management applications (like backup engines, graphical presentation frameworks, and volume managers) to use.

4.3.2.2   SMI-S Server

An SMI-S Server is a CIM Server. It shall implement those functional profiles, as defined in the DMTF specifications, necessary to satisfy the SMI-S profile with which it conforms. Often, an SMI-S Server controls only one device or subsystem, and is incapable of providing support for complex intrinsic methods like schema

traversal. An SMI-S Server can be embedded in a device (like a Fibre Channel Switch) or provide a proxy on a host that communicates to a device over a legacy or proprietary interconnect (like a SCSI based array controller).

Embedding an SMI-S Server directly in a device or subsystem reduces the management overhead seen by a customer and eliminates the requirement for a stand-alone host (running the proxy agent) to support the device.

Embedded SMI-S Servers are the desired implementation for "plug and play" support in an SMI-S managed environment. However, proxy SMI-S Servers are a practical concession to the legacy devices that are already deployed in storage networked environments. In either case, the minimum CIM support for SMI-S Servers applies to either SMI-S Server deployments.

4.3.2.3   General Purpose SMI-S Server

A General Purpose SMI-S Server is CIM Server that serves management information from one or more devices or underlying subsystems through providers. As such a General Purpose SMI-S Server is an aggregator that enables proxy access to devices/subsystems and can perform more complex operations like schema traversals. A General Purpose SMI-S Server typically includes a standard provider interface to which device vendors adapt legacy or proprietary product implementations.

4.3.2.4   Provider

A provider expresses management information for a given resource such as a storage device or subsystem exclusively to a CIM Server. The resource may be local to the host that runs the Object Manager or may be remotely accessed through a distributed systems interconnect.

4.3.2.5   Lock Manager

This version of the specification does not support a lock manager.

4.3.2.6   Directory Server (SLP Directory Agent)

A directory server provides a common service for use by clients for locating services in the management environment.

4.3.3   Cascaded Agents

This specification discusses constituents in the SMI-S environment in the context of Clients and Servers. This version of the specification also allows constituents in a SMI-S management environment to function as both client and server.

Clause 5: Health and Fault Management

5.1   Objectives

Health and Fault Management is the activity of anticipating or detecting failures through monitoring the state of the storage network and its components and intervening before services can be interrupted. A service in this case is the realization of storage through several interconnected devices connected, configured for a dedicated purpose. The purpose is the delivery of software application functionality in support of some business function.

5.2   Overview

•   Express states and statuses with standard meanings.

•   Define the use of comprehensive error reporting in determining the type, category, and source of failures.

•   Define the quality associated with errors rather than qualities.

•   Define explicit failure scopes rather than requiring HFM enabled application to construct them.

5.3   Terms

5.3.1   error

An unexpected condition, result, signal or datum. An error is usually caused by an underlying problem in the system such as a hardware fault or software defect. Errors can be classified as correctable (recoverable) or uncorrectable, detectable or undetectable.

5.3.2   fault

A problem that occurs when something is broken and therefore not functioning in the manner it was intended to function. A fault may cause an error to occur.

5.3.3   fault region

Many devices or applications can attempt to fix themselves upon encountering some adverse condition. The set of components which the device or application can attempt to fix is called the Fault Region. The set may include part or all of other devices or applications. Having the Fault Regions declared helps a HFM application, acting as a doctor, to do no harm by attempting to interfere and thereby adversely affect the corrective action being attempted.

5.3.4   Health and Fault Management (HFM)

Health and Fault Management is the activity of anticipating or detecting debilitating failures through monitoring the state of the storage network and its components and intervening in before services can be interrupted. A service in this case is the realization of storage utilization through several interconnected devices connected, configured for a dedicated purpose. The purpose is the delivery of software application functionality in support of some business function.

5.3.5   operational status

These values indicate the current status(es) of the element. Various operational statuses are defined (e.g., OK, starting, stopping, stopped, In Service, No Contact).

5.3.6   health state

These values indicate the current health of the element. This attribute expresses the health of this element but not necessarily that of its subcomponents.

5.4   Description of Health and Fault Management

The goal of effective administration requires devices and applications that comprise storage services to report their status and the nature of their errors in standard terms. These terms need to be understandable by a client without device-specific knowledge.

Fault Detection

| Indication | CIM_Error | Poll for Health State | SubComponentInError Association |
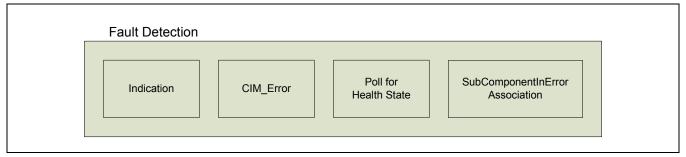
Figure 7 - Basic Fault Detection

There are four basic ways for a SMI-S client to detect an error or fault condition. Figure 7 lists the four basic methods for fault detection. These are:

• Health state and Operational status - Polling.

• Error - Standard errors returned from CIM operations.

• Indications - Subscribe for and receive asynchronous Indications.

• Fault Regions (experimental) - Walk the CIM model looking for RelatedElementCausingError associations.

5.4.1   Operational Status and Health State (Polling)

Operational Status and Health State are the two properties that will be used to monitor health. These two properties could convey very different statuses and may at times be related or independent of each other. For example, you may have a disk drive with the Operational Status of "Stopped" and the HealthState of 30 (Non-recoverable error) or 5 (OK). Now the reason the disk drive is stopped could vary from the fact that it had a head crash (HealthState = 30) to the situation where it was stopped for the routine maintenance (HealthState = 5).

Table 1 is an example of how HealthState can disambiguate health for a disk drive, various values for OperationalStatus and HealthState:

Table 1 shows, for a disk drive, various possible values for OperationalStatus and HealthState. Note that there are many cases not shown.:

Table 1 - OperationalStatus for Disk Drive

OperationalStatus	Description	HealthState	Description	Comment
2	OK	5	OK	Everything is fine
2	OK	10	Degraded/ Warning	Some soft errors
3 or 2	Degraded or Predicted Failure	15	Minor Failure	Many soft errors

Table 1 - OperationalStatus for Disk Drive (Continued)

3 or 2	Degraded or Predicted Failure	20	Major Failure	Some hard errors
3	Degraded	10	Good	A subcomponent has failed (no data loss)
10	Stopped	5	OK	Drive spun down normally
10	Stopped	30	Non-recoverable Error	Head crash
8	Starting	10	Degraded/ Warning	Will update HealthState once fully started
4	Stressed	5	OK	Too many I/O in progress, but the drive is fine.
15	Dormant	5	OK	The drive is not needed currently

The property OperationalStatus is multi-valued and more dynamic. It tends to emphasize the current status and potentially the immediate status leading to the current status; whereas, the property HealthState is less dynamic and tends to imply the health over a longer period of time. Again, in the disk drive example, the disk drive's operational status may change many times in a given time period. However, in the same time period, the health of the same drive may not change at all.

5.4.2   Standard Errors and Events

Standardization of error and events are required so that the meaning is unambiguous and is given to comparisons.

Error and Alert indications

HFM clients shall not be required to be embodied with specific knowledge of the devices and applications in order to derive the quality of the error from the datum. The device and application shall express the quality of the error rather than the quantity interpreted with *a priori* knowledge to determine that error condition is present. For example, a device needs to express that it is too hot rather than requiring the HFM enabled application to determine this from the temperature datum and device specific knowledge of acceptable operating conditions.

Standard errors are defined for each Profile/Subprofile. The definitions will be contained in the profiles/subprofiles. Standard errors are not the only error codes that can be returned, but are the only codes that a generic client will understand.

5.4.3   Indications

Indications are asynchronous messages from CIM servers to clients. A client must register for them. Each SMI-S profile/subprofile contains lists of indication filters that clients use to indicate the information it is interested in. The message itself is defined in the SMI-S indication subprofile.

EXPERIMENTAL

5.4.4   Event Correlation and Fault Containment

Automation will require that an error arising through control and configuration activities, as a side effect of them, or by failures caused by defects can be directly correlatable. Error categories like network cabling failures or network

transmission errors will help organize the types of error that can be produced. Standard errors, like impending disk media failure, will be required as well.

Once the errors have been collected and correlated, the HFM enabled application can produce an impact list sorted by likelihood. Some of the error correlation can be determined by the common affect through the manifestation of the RelatedElementCausingError association to be described later. The alerts themselves can report its correlation with other alerts.

Potential faults can then be derived from errors for each component. Deriving such a list may require a dialog between the HFM enabled application and the device or application in question such that the HFM enabled application is assisted in the production of the list.

If permitted, then control and configuration operations may be executed to contain the fault. The pallet of these operations will be those operations already available through SMI-S. However, special operations may arise from the HFM design work as well. Fault containment will include the reconfiguration of the storage service with alternative components, leaving failing components or interconnections isolated.

Much like a physician, the HFM enabled application is notified or consulted when symptoms appear. The HFM enabled application then develops a prognosis based on the manifestation of the ailment. At times, the HFM enabled application will perform diagnostic procedures. The end result of the process is to produce a list of possible causes, ranked by probability, and associated recommended procedures.

Also like a doctor, the HFM enabled application will settle for enabling the patients to heal themselves. That is the HFM enabled applications cannot be expected to heal the device in all cases. A significant portion of all possible corrective actions will require the intervention of people or device unique knowledge.

The simplified state diagram shown in Figure 8 follows the fault mitigation life cycle for HFM.

The device or application manifests an event, either by a state change, error returned from a WBEM operation, or an alert indication.

The event is recognized by the HFM enabled application and accessed by the HFM enabled application. It may be that the event indication does the represent the existence of an error. An error condition may be heralded by a single or multiple events occurring in some order. The process of examining and characterizing event as errors is called error handling.

Once it is determined that an error condition is present, then possible causes are sought and ranked by likelihood. The causes themselves describe a potential problem or fault with the component in question. Alternatively, the device or application may report the fault directly, through an alert indication, optionally with recommended actions.
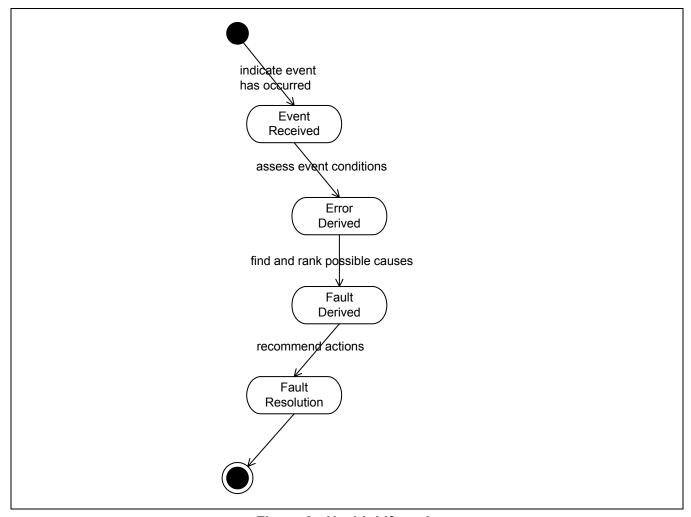
Figure 8 - Health Lifecycle

Fault resolution may not require the intervention of an operator or field technician. It is these faults that can be handled entirely by the HFM enabled application. Otherwise, the HFM enabled application can not actively participate in whole fault resolution life cycle. In this case, the HFM enabled application would wait for the end state of fault resolution to come to being before ending its fault mitigation exercise.

Faults are contained and components repaired or replaced. The instructions to the HFM enabled application for what can be done to repair the fault are the recommended actions. Fault Containment includes fencing off the faulty component and maintaining the service. To be minimally effective, the HFM enabled application contains the fault. The repair may or may not be done with human intervention.

The devices and application that comprise a storage system have themselves some level of self diagnostics and report functionality.
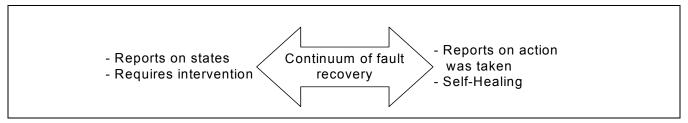
Figure 9 - Continuum

There is a range of ability of devices and applications to recover from failures and to report on the error recovery actions taken. See Figure 9 The variance of capabilities for device and applications can be plotted on a continuum. At one end of continuum, the device or application recognizes a fault condition and takes action, reporting on the action taken and any further action required to service it. At the other end of the continuum, the device can only report on that states and requires intervention both in the detection of fault conditions and taking corrective action.

There are limits to what an HFM enabled application can do.   Obviously, if the device or application can not report states, errors and alerts in a standard way or can not report this data at all, then there is little an external implementation can do.

However, few, if any, of these devices and applications can monitor and correct the service as a whole. It is for this reason, the HFM implementation is needed to augment the effectiveness of the administrator.

5.4.5   Fault Regions

A scope can be applied to the effect of errors and the associated fault. A fault may affect a component, a device or application, storage service, or all the above. This scope defines the area of influence for fault containment. For example, the device itself may monitor its components and perform fault mitigation on its own.   The plot of components whose errors are handled by a given fault mitigation entity is the fault region. The scope of effect of this fault region shall be defined.

Figure 10 illustrates the scope of fault regions in a simplified SAN example and how the may be recursive in nature. AN HFM application has the widest scope of concern in this example.
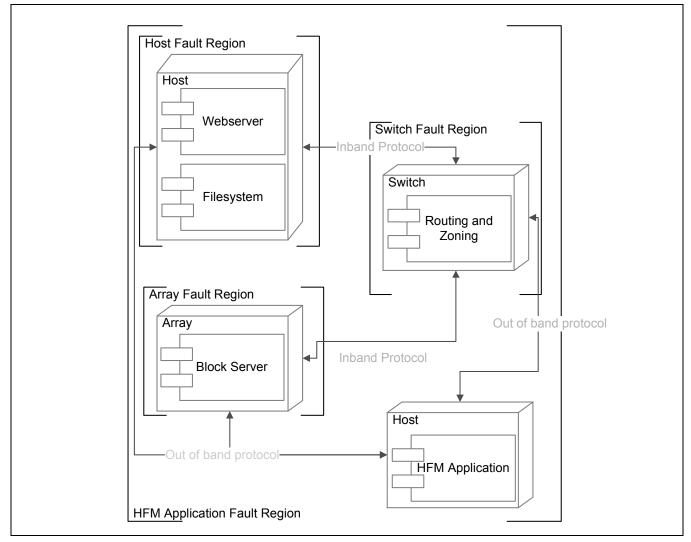
Figure 10 - Application Fault Region

Error handling is initiated by the interception of error events. For example, a switch may recognize the failure of one it ports and reroute traffic to a working port. In this case, the fault region is defined as the switch itself. If the failure event is publicly consumable, other fault mitigation entities can also handle the error as well. The failure of a drive may be mitigated one way in the array fault region and mitigated differently in the HFM enabled application fault region. For example, the array fault mitigation entity can bring a volume off line if the failure of the disk brings the set of disks below the minimum required for quorum. At the same time, the HFM enabled application can reconfigure the storage service to create a replacement volume and then restore the failed volume's data from backup.

The HFM enabled application is one of the several possible storage network scope fault mitigation entities. As discussed previously, this broad scope is necessary to mitigate faults where the faults cannot be entirely mitigated by the storage device or application alone. It is necessary that fault mitigation entities like the HFM enabled application can observe the activities of the fault mitigation entities contained within their fault regions such that they do no harm. Device or application should express what error conditions are to be handled inside their own fault domain and how an HFM enabled application can detect that such fault containment is occurring. State changes on components may BE sufficient representation of these activities.

In general, the HFM enabled application fault region mitigation may not necessarily include the same actions that the host, switch, or array may take to fix them.

EXPERIMENTAL

5.4.6   Examples

5.4.6.1   Array Example

The scenario presented is related to a storage array that contains one or many ports. See Figure 11. A port is off-line. This port effects the serving of a volume to a host.
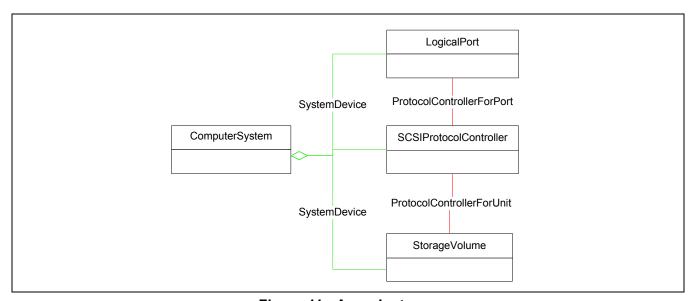
Figure 11 - Array Instance

5.4.6.1.1   Indication

An AlertIndication is produced by the array notifying the HFM enabled application of the failure. The indication reports the Object Name of the ProtocolController that has failed through its AlertingManagedElement property. When storage capacity configuration operations are attempted on storage related to the failed ProtocolController, an Error is reported.   The error reports the Object Name of the ProtocolController that has failed through the ErrorSource property.   Error is a class introduced in CIM 2.9 that provides a mechanism to express error number, category, recommended actions and the like.

5.4.6.1.2   Standard Errors

It is mandatory to report error conditions through both AlertIndication and Error in those cases where Error is returned when the method call failed for reasons other than the method call itself. For example, if the device port is down then a method call can fail because of this condition. It is expected that the device will report a port error AlertIndication to listening clients as well.

5.4.6.1.3   Operational status and Health State (Polling)

A client that gets the top Computer system instance should see an operational status of degraded and a health state of good if the data wasn't lost. At the same time, reading the instance of Computer system for the broken controller would see an operational status of "stopped" and a health state of "non-recoverable Error".

EXPERIMENTAL

5.4.6.1.4   Fault Region

The RelatedElementCausingError association defines the relationship between a CIM Instance that is reporting an error status and the component that is the cause of the reported status.   The Port and a Volume using the port both report error status and the RelatedElementCausingError association reports that the ProtocolController through which the Volume is exposed has failed and at least some of the volumes are no longer visible externally to the array. The array itself would be thereby degraded.

The RelatedElementCausingError association is independent of all other associations. It is only use to report error associations and comes into existence only when necessary. Once the error has been handled, the association is removed from the model.

EXPERIMENTAL

5.4.6.2   Switch Example

The scenario presented is related to a FC Switch that contains many ports. See Figure 12. One of the ports is off-line.
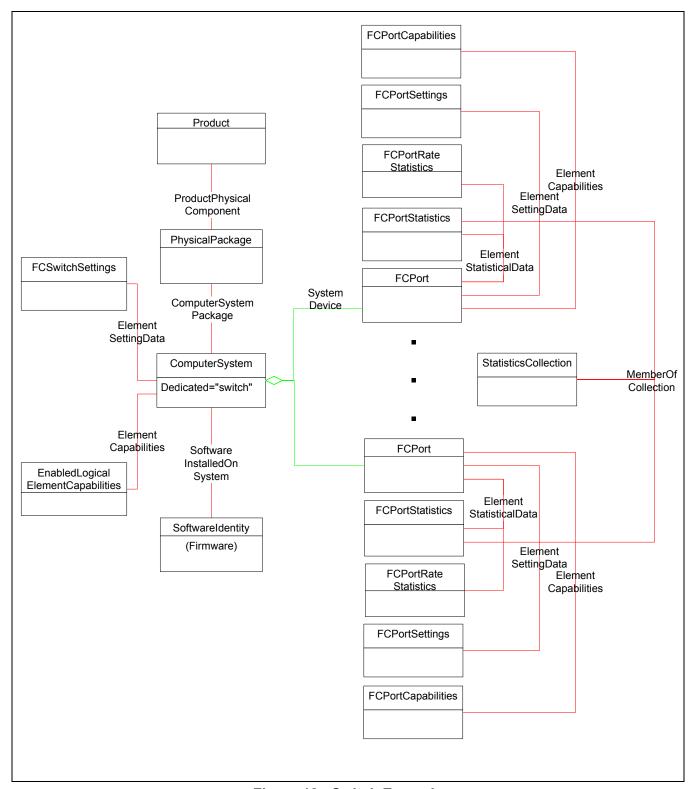
Figure 12 - Switch Example

5.4.6.2.1   Indication

An AlertIndication is produced by the switch notifying the HFM enabled client of the failure. The indication reports the Object Name of the FC port (FCPort) that has failed through its AlertingManagedElement property.

5.4.6.2.2   Standard Errors

A call to Port settings, port capabilities, or statistics cause an Error to be reported.   The error reports the Object Name of the FCPort that has failed through the ErrorSource property.

It is mandatory to report error conditions through both AlertIndication and Error in those cases where Error is returned when the method call failed for reasons other than the method call itself. For example, if the device is over heat, then a method call can fail because of this condition. It is expected that the device will report an over heat AlertIndication to listening clients as well.

EXPERIMENTAL

5.4.6.2.3   Fault Region

The RelatedElementCausingError association defines the relationship between a CIM Instance that is reporting an error status and the component that is the cause of the reported status.   The failed port would report error status and the RelatedElementCausingError association reports that the PortStatistics and PortSettings are effected. The switch itself would be thereby degraded.

The RelatedElementCausingError association is independent of all other associations. It is only use to report error associations and comes into existence only when necessary. Once the error has been handled, the association is removed from the model.

EXPERIMENTAL

Clause 6: Object Model General Information

6.1   Model Overview (Key Resources)

6.1.1   Overview

The SMI-S object model is based on the Common Information Model (CIM), developed by the DMTF. For a more complete discussion of the full functionality of CIM and its modeling approach, see http://www.dmtf.org/standards/standard_cim.php.

Readers seeking a more complete understanding of the assumptions, standards and tools that assisted in the creation of the SMI-S object model are encouraged to review the following:

•   CIM Tutorial (http://www.wbemsolutions.com/tutorials/CIM/index.html)

•   CIM UML Diagrams and MOFs (http://www.dmtf.org/standards/standard_cim.php)

Managed Object File (MOF) is a way to describe CIM object definitions in a textual form. A MOF can be encoded in either Unicode of UTF-8. A MOF can be used as input into an MOF editor, parser or compiler for use in an application.

The SMI-S model is divided into several *profiles*, each of which describes a particular class of SAN entity (such as disk arrays or FibreChannel Switches). These profiles allow for differences in implementations but provide a consistent approach for clients to discover and manage SAN resources. IN DMTF parlance, a *provider* is the instrumentation logic for a profile. In many implementations, providers operate in the context of a *CIM Server* that is the infrastructure for a collection of providers. A WBEM *client* interacts with one or more WBEM Servers.

6.1.2   Introduction to CIM UML Notation

CIM diagrams use a subset of Unified Modeling Language (UML) notation.

Most classes are depicted in rectangles.

```
┌──────────────────┐
│ PhysicalPackage  │
├──────────────────┤
│                  │
└──────────────────┘
```

The class name is in the upper part and *properties* (also known as *attributes* or *fields*) are listed in the lower part. A third subdivision added for *methods, if they are included*. A special type of class, called an *association*, is used to describe the relationship between two or more CIM classes

Three types of lines connect classes, as shown in Figure 13.
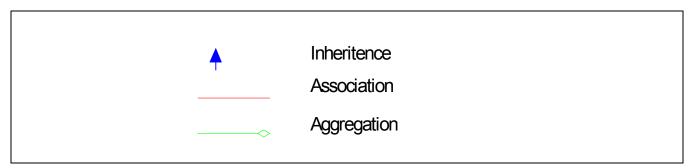
Figure 13 - Lines that Connect Classes

The CIM documents generally follow the convention of using blue arrows for inheritance, red lines for associations and green lines for aggregation. The color-coding makes large diagrams much easier to read but is not a part of the UML standard.

The ends of some associations have numbers (cardinality) indicating the valid count of object instances. Cardinality is expressed either as a single value (such as 1), or a range of values (0..1 or 1..4); "*" is shorthand for 0..n.

Some associations and aggregations are marked with a "W" at one end indicating that the identity of this class depends on the class at the other end of the association. For example, fans may not have worldwide unique identifiers; they are typically identified relative to a chassis.

This document uses two other UML conventions.

The UML Package symbol [Disk1] is used as a shortcut representing a group of classes that work together as an entity. For example, several classes model different aspects of a disk drive. After the initial explanation of these objects, a single disk package symbol is used to represent the entire group of objects.

Schema diagrams include all of a profile's classes and associations; the class hierarchy is included and each class is depicted one time in the schema diagram. Instance diagrams also contain classes and associations but represent a particular configuration; multiple instances of an object may be depicted in an instance diagram. An instance may be named with an instance name followed by a colon and a class name (underlined). For example,

Array: ComputerSystem

Switch: ComputerSystem

represents an array and a switch – two instances of <COMPUTER SYSTEM> objects.

6.2   Techniques

6.2.1   CIM Fundamentals

This section provides a rudimentary introduction to some of the modeling techniques used in CIM, and is intended to speed understanding of the SMI-S object model.

6.2.1.1   Associations as Classes

CIM presents relationships between objects with specialized classes called *associations* and *aggregations.* In addition to references to the related objects, the association or aggregations may also contain domain-related properties. For example, ControlledBy associates a controller and a device. There is a many-to-many cardinality between controllers and devices (i.e., a controller may control multiple devices and multi-path devices connect to multiple controllers); each controller/device connection has a separate activity state. This state corresponds to the AcccessState property of ControlledBy association linking the device and the controller.

6.2.1.2   Logical and Physical Views

CIM separates physical and logical views of a system component, and represents them as different objects – the "realizes" association ties these logical and physical objects together.

6.2.1.3   Identity

Different agents may each have information about the same organic object and may need to instantiate different model objects representing the same thing. Access control is one example: a switch zone defines which host device ports may access a device port. The switch agent creates partially populated port objects that are also created by the HBA and storage system agents. The ConcreteIdentity association is used to indicate the associated object instances are the same thing. ConcreteIdentity is also used as a language-independent alternative to multiple inheritance. For example, a FibreChannel port inherits from a generic port and also has properties of a SCSI controller. CIM models this as FCPort and ProtocolController objects associated by ConcreteIdentity.

6.2.1.4   Extensibility

CIM makes allowances for additional values in enumerations that were not specified in the class Derivation by adding a property to hold arbitrary additional values for an enumeration. This property is usually named OtherXXXX (where XXXX is the name of the enumeration property) and specifying "other" as the value in the enumeration property indicates its use. For an example see the ConnectorType and OtherTypeDescription properties of Slot object in the CIM_Physical MOF.

6.2.1.5   Value/ValueMap Arrays

CIM uses a pair of arrays to represent enumerated types. ValueMap is an array of integers; Values is an array of strings that map to the equivalent entry in ValueMap. For example, PrinterStatus (in the CIM_Device MOF) is defined as follows:

```
ValueMap {"1", "2", "3", "4", "5", "6", "7"},
Values {"Other", "Unknown", "Idle", "Printing", "Warm-up,
"Stopped Printing", "Offline"},
```

A status value of 6 means "Stopped Printing". A client application can automatically convert the integer status value to a human-readable message using this information from the MOF.

6.2.1.6   Return Codes

When a class definition includes a method, the MOF includes Value/ValueMap arrays representing the possible return codes. These values are partitioned into ranges of values; values from 0 to 0x1000 are used for return codes that may be common to various methods. Interoperable values that are specific to a method start at 0x1001; and vendor-specific values may be defined starting at 0x8000. Here's an example of return codes for starting a storage volume.

```
 ValueMap {"0", "1", "2", "4", "5", ".", "0x1000",
"0x1001", "…", "0x8000.."},
 Values {"Success", "Not Supported", "Unknown", "Time-out,
"Failed", "Invalid Parameter", "DMTF_Reserved",
"Method parameters checked - job started",
"Size not supported",
"Method_Reserved", "Vendor_Specific"}]
```

6.2.1.7   Model Conventions

This is a summary of objects and associations that are common to multiple profiles.

> **PhysicalPackage** represents the physical storage product. PhysicalPackage may be sub-classed to ChangerDevice, but PhysicalPackage accommodates products deployed in multiple chassis.

> **Product** models asset information including vendor and product names. Product is associated with PhysicalPackage.

> **SoftwareIdentity** models firmware and optional software packages. InstalledSoftwareIdentity associates SoftwareIdentity and ComputerSystem, ElementSoftwareIdentity associates SoftwareIdentity and LogicalDevices (a superclass of devices and ports).

> **Service** models a configuration interface (for example, a switch zoning service or an array access control service). Services typically have methods and properties describing the capabilities of the service. A storage system may have multiple services; for example, an array may have separate services for LUN Masking and LUN creation. A client can test for the existence of a named service to see if the agent is providing this capability.

> **LogicalDevice** (for example, FCPort) is a superclass with device subclasses (like and DiskDrive and TapeDrive) and also intermediate nodes like Controller and FCPort. Each LogicalDevice subclass shall

be associated to a ComputerSystem with a SystemDevice aggregation. Due to the large number of LogicalDevice subclasses, SystemDevice aggregations are often omitted in instance diagrams in this specification.

This specification covers many common storage models and management interfaces, but some implementations include other objects and associations not detailed in the specification. In some cases, these are modeled by CIM schema elements not covered by this document. When vendor-specific capabilities are needed, they should be modeled in subclasses of CIM objects. These subclasses may contain vendor-specific properties and methods and vendor-specific associations to other classes.

6.2.2   Modeling Profiles

In addition to modeling SAN components, SMI-S servers shall model the profiles they provide. This information is used two ways:

• Clients can quickly determine which profiles are available.

• An SLP component can query the SMI-S Server and automatically determine the appropriate SLP Service Template information (see Clause 10: Service Discovery).

A client can traverse the Server Profile in each SMI-S server to see which profiles (and objects) claim SMI-S compliance. RegisteredProfile describes the profiles that a CIM server claims are supported. The RegisteredSubprofile is used to define the optional features supported by the system being modeled. A client can traverse the associations in the Server Profile to see which profiles claim SMI-S compliance.

6.2.3   CIM Naming

There may be multiple SMI-S servers in any given storage network environment. It is not sufficient to think of the name of an object as just the combination of its key values. The name also serves to identify the Server that is responsible for the object. The name of an object (instance) consists of the namespace and the model. The namespace provides access to a specific SMI-S server implementation and is used to locate a particular namespace within a server. The model provides full navigation within the CIM Schema and is the concatenation of the class name and key-qualified properties and values.

The namespace has special rules. It should uniquely identify a SMI-S server. However, a SMI-S server may support multiple namespaces. How an implementation defines Namespaces within a SMI-S server is not restricted. However, to ease interoperability, SMI-S implementations should manage all objects within a profile in one namespace.

Clause 7: Correlatable and Durable Names

7.1   Overview

Management applications often read and write information about managed objects in multiple CIM namespaces or between CIM and some other storage management namespace. When an object in one namespace is associated with an object in another namespace, each namespace may represent some amount of information about the same managed resource using different objects. A management application understands when objects in different namespaces represent the same managed resource by the use of a unique common identifier, referred to as a "correlatable name". A correlatable name is designated as a mandatory property for any objects representing managed resources that may be seen from multiple points of view. These durable names are used by management applications for object coordination.

A related concept is referred to as "durability". Some names may be correlatable at a particular point in time, but may change over time (e.g., a durable name is a hardware-assigned port or volume name and a correlatable, non-durable ID is a DHCP IP address). No name is permanently durable (e.g., even a name derived from hardware may change due to FRU replacement). A client application should assume that a stored durable name remains valid over time where a non-durable may not remain valid over time.

Correlatable names are unique within a defined namespace. In some cases, that namespace is world-wide; requiring compliance to standards defined by a naming authority. In other cases, the namespace is the hosting system or some set of connected systems (e.g., operating system device names are unique to the containing host).

A name may be expressed in different formats (e.g., numeric value are sometimes displayed as decimal or hexadecimal, the hexadecimal value sometimes has a leading "0x" or a trailing "h"). To assure interoperability, mandatory formats are specified by this standard.

A necessary technique associated with correlatable names involves the use of CIM properties that describe the format or namespace from which the name is derived. CIM key-value combinations are unique across instances of a class, but CIM does not fully address cases where different types of identifiers are possible on different instances of an object. It is therefore necessary to ensure that multiple sources of information about managed resources use the same approach for forming correlatable names whenever different types of identifiers are possible.

When different types of identifiers are possible, the profile specifies the possible name formats and namespaces for durable and correlatable IDS, the preferred order that each implementation should use if multiple namespaces are available, and the related properties that a client uses to determine the namespace.

Correlatable, durable names are mandatory for these objects:

- SCSI logical units or (such as storage volumes or tape drives) that are exported from storage systems; also SB (Single Byte Command Code Sets)

- SB control unit issues

- External Ports on hosts and storage devices

- Fibre Channel ports on interconnect elements

- Fibre Channel fabric (modeled as AdminDomain)

- ComputerSystem objects that server as top-level systems for all SMI-S profiles

- Operating System Device Names

CIM keys and correlatable names are not tightly coupled. For some classes, they may be the same, but this is not mandatory as long as all correlatable names are unique and management applications are able to determine when objects in different namespaces are providing information about the same managed resource.

The common types of information used for names include the SCSI Device Identifiers from the Identification Vital Product Data page (i.e., VPD page 83h), SB Node Element Descriptors from Read-Configuration Data, the response form ATA IDENTIFY commands, Fibre Channel Name_Identifiers (i.e. World Wide Names), Fully Qualified Domain Names, and IP Address information. See 7.2, 7.3, 7.4, and 7.5 for general information on the advantages and disadvantages of certain types of names. The details for each class requiring durable correlatable names are provided in the profiles subclauses of this document.

If the name used in the instrumentation in binary, the CIM representation is an upper case hexadecimal-encoded representation of the value returned. For example, decimal 27 is hexadecimal 1b and will be represented by the string "1B". Note that each binary byte requires two ASCII characters using this representation. If the name used in the instrumentation is ASCII text, the case of the characters is preserved in the CIM property.

7.2   Guidelines for SCSI Logical Unit Names

The preferred logical unit identifier is returned from a SCSI INQUIRY command in VPD page 83h.

> **Note:** Legacy systems may lack correlatable names as SCSI standards prior to SAM-3 and SPC-3 did not clearly define logical unit names, however this has been clarified to be logical unit names and recent systems have converged in compliance.

The Unit Serial Number VPD page (i.e., SCSI Inquiry VPD Page 80h) returns a serial number, but the SPC-3 standard allows this either be a serial number for a single logical unit or a serial number of the target device. There's no mechanism to discover which approach the device is using. If a client is not coded to understand which products provide per-logical unit or per-target serial numbers, then it should not use the Unit Serial Number VPD page as a logical unit name.

The Identification Vital Product Data page (i.e., VPD page 83h) returns a list of identifiers with metadata describing each identifier.   The metadata includes:

•   Code Set (binary versus ASCII)

•   Association (indicates the SCSI object to which the identifier applies, e.g., for a logical unit, port, or target device)

•   Type (the naming authority for identifiers of the structure of information about target ports)

•   Protocol Identifier (indicates the SCSI transport protocol to which the identifier applies)

To identify a logical unit name the Association shall be set to zero. The preferred Types for logical units are 3 (NAA), 2 (EUI), and 8 (SCSI Name). However type 1 (T10) is allowed. If the code set in the inquiry response indicates the identifier is binary, the CIM representation is hexadecimal-encoded.

7.3   Guidelines for FC-SB-2 Device Names

FC-SB-2 devices and control unit images use the node-element descriptor (NED) name format. NEDs are retrieved within a configuration record retrieved by the READ-CONFIGURATION DATA command. A configuration record contains information describes the internal configuration of the device, where the information retrieved describes the corresponding node elements that are accessed when an I/O operation is performed.

NEDs are 32 bytes and contain these fields:

•   4 bytes (flags, type, class, reserved) - binary

•   6 byte "type number" - string

•   3 byte "model number" - string

•   3 byte "manufacturer" - string

- 2 byte "plant of manufacture"- string

- 12 byte sequence number" - string

- 2 byte tag - binary

The I/O-Device NED is used for identifying devices. The Token NED is used for identifying control-unit images.

The Name property for LogicalDevices representing SB devices is world-wide unique value formed by composing these fields.

7.4   Guidelines for Port Names

The following is a list of optimal names for ports based on the transport type:

1)   Fibre Channel ports use Port World Wide Names (i.e. FC Name_Identifier)

2)   iSCSI has three types of ports

- the combination of IP address and TCP port number serve as the primary correlatable name for iSCSI target ports. Note that this information is stored in two separate properties and hence there is no single correlatable name.

- the logical element (iSCSIProrotolEndpoint) that represents the SCSI port The SCSI logical port shall be named with an iSCSI name.

- the underlying physical ports (typically Ethernet ports). Ethernet ports names shall use the MAC address.

3)   Parallel SCSI (SPI) and ATA ports typically do not have names, they are identified by a bus-relative address typically set with jumpers. In configurations where these drives are not shared by multiple hosts, the host-relative name acts as the name.

4)   CIM port classes do no include NameFormat; the appropriate format is determined by the transport implied by the port subclass.

SCSIProtocolEndpoint represents SCSI protocol running through a port. In many cases, there is one-to-one mapping between SCSIProtocolEndpoint and some subclass of LogicalPort and the name requirements are identical. For iSCSI, there many be multiple Ethernet ports per SCSIProtocolEndpoint instance. The IP address and TCP port number are modeled in IPProtocolEndpoint and TCPProtocolEndpoint. iSCSIProtocolEndpoint Name holds the iSCSI initiator or target name.

SBProtocolEndpoint represents SB protocol running through a port. In many cases, there is a one to-one mapping between SBProtocolEndpoint and some subclass of LogicalPort and the name requirements are identical.

7.5   Guidelines for Storage System Names

Each profile has a ComputerSystem or AdminDomain instance that represents the entire system. There are a variety of standard and proprietary names used to name storage systems. Unlike SCSI logical units and ports, there is no particular name format in common use. There are advantages and disadvantages to certain types of names.

IP addresses have an advantage in human recognition; (e.g., administrators are accustomed to referring to systems by their IP addresses). The downsides are that IP addresses are not necessarily durable (e.g., DHCP) are not necessarily system-wide (e.g., some storage systems have multiple network interfaces), and are not necessarily unique (e.g., NAT allows the same IP address to be used in multiple network zones).

Full Qualified Domain Names are friendlier than IP addresses and may fix the durability issue of IP addresses (e.g., a host name may be constant even when the IP address changes). But storage systems do not necessarily have access to their network names. Network names are typically handled through a central service such as DNS.

When a client application opens a connection to a remote system, it asks the local system to resolve the name to an IP address, the local system redirects the request to the DNS server, the IP address is returned and the client application opens the connection. If the remote system is the storage system, this sequence requires the DNS server to know about the storage system, but not vice-versa. A storage system is only required to know about DNS if software on the storage system acts as a network client using host names. And, like IP addresses, a storage system may have several network interfaces with different FQDNs.

Transport-specific names are specific to a particular storage transport (e.g., Fibre Channel or iSCSI). There are some good standard names (e.g., FC platform names or iSCSI Network Entity names). The disadvantage of transport-specific names is that they are not able to be consistently used on storage systems supporting multiple transports or in configurations with transport bridges (e.g., a client may have no mechanism to issue FC commands to an FC device behind an FC/iSCSI bridge).

SCSI target names solve the transport-specific issue. Before the SAM-3 and SPC-3 standards there was not a standard SCSI system name, however with SPC-3, the Identification Vital Product Data page association value 2 was defined for a target name. At this time, the SPC-3 standard is too new to be in common use. Most storage systems include some vendor-specific way to get a target name, but client is not able to use these names without specific knowledge of the vendor-specific interface.

At this time, no single storage system name format is in common use. The best approach is for implementations to expose several names, along with information that tells the client how to interpret the name. The OtherIdentifyingInfo and IdentifyingDescriptions array properties of ComputerSystem provide the list of names and interpretations. However, IdentifyingDescriptions is not an enumerated type; and as a result, any string is valid from a CIM perspective.

7.6   Standard Formats for Correlatable Names

7.6.1   General

Correlatable names shall be used and formatted consistently. Storage volume names are more complex that other element names (i.e., the same format may be used in different namespaces). For example several common INQUIRY Vital Product Data page names use the IEEE NAA format and as a result a client is not able to correlate names from different namespaces.

7.6.2   Standard Formats for Logical Unit Names

For disks and arrays, multiple name formats are in common use. Table 2 specifies standard formats for storage volume names.

Table 2 - Standard Formats for StorageVolume Names

Description	Format property and value(valuemap)	Format of Name
SCSI VPD page 83 type 3, Association 0, NAA 0101b	NameFormat = NAA(9), NameNamespace = VPD83Type3(1)	NAA name with first nibble of 5. Recommended format (8 bytes long) when the ID is directly associated with a hardware component. Formatted as 16 un-separated upper case hex digits (e.g., '21000020372D3C73')
SCSI VPD page 83, type 3h, Association=0, NAA 0110b	NameFormat = NAA(9), NameNamespace= VPD83Type3(1)	NAA name with first nibble of 6. Recommended format (16 bytes long) when IDs are generated dynamically. Formatted as 32 un-separated upper case hex digits.

Table 2 - Standard Formats for StorageVolume Names

SCSI VPD page 83, type 3h, Association=0, NAA 0010b	NameFormat = NAA(9), NameNamespace = VPD83Type3(1)	NAA name with first nibble of 2. Formatted as 16 un-separated upper case hex digits
SCSI VPD page 83, type 3h, Association=0, NAA 0001b	NameFormat = NAA(9), NameNamespace = VPD83Type3(2)	NAA name with first nibble of 1. Formatted as 16 un-separated upper case hex digits
SCSI VPD page 83, type 2h, Association=0	NameFormat = EUI64(10), NameNamespace = VPD83Type2(3)	Formatted as 16, 24, or 32 un-separated upper case hex digits
SCSI VPD page 83, type 1h, Association=0	NameFormat = T10VID(11), NameNamespace = VPD83Type1(4)	Formatted as 1 to 252 bytes of ASCII.
SCSI VPD page 80, serial number	NameFormat = Other(1), NameNamespace = VPD80(5)	Only if serial number refers to logical units rather than the enclosure. 1-252 ASCII characters
SB I/O Device NED	NameFormat=SBDevice (13), NameNamespace=SB	64 un-separated upper case hex digits. The tag subfield contains CU_image+device_address
SB Token NED	NameFormat=SBToken(14), NameNamespace=SB	64 un-separated upper case hex digits. The tag sub-field contains the CU_image
SCSI Concatenation of Vendor,Product , SerialNumber	NameFormat = SNVM(7), NameNamespace = SNVM(7)	A concatenation of three strings representing the vendor name, product name within the vendor namespace, and serial number within the model namespace. These strings come from SCSI standard INQUIRY response data. Strings are delimited with a '+' and spaces are included. Vendor and Product are fixed length: Vendor ID is 8 bytes, Product is 16 bytes. SerialNumber is variable length and may be up to 252 bytes in length. If one of these fields contains a plus sign, it shall be escaped with a backslash ('\+'). The concatenation is done to provide world-wide uniqueness; clients should not parse this name.

Table 2 - Standard Formats for StorageVolume Names

ATA Concatenation of, Model, SerialNumber	NameFormat=ATA, NameNamespace=ATA	A concatenation of three strings representing the vendor and model names and serial number within the model namespace. The manufacturer name is not based on a specific standard. The model name and serial number strings come from ATA IDENTIFY DEVICE response data. Strings are delimited with a '+' and spaces are included. The vendor is 20 characters, model is 40 characters, and serial number is 20 characters. If one of these fields contains a plus sign, it shall be escaped with a backslash ('\+'). The concatenation is done to provide uniqueness; clients should not parse this name. Note that ATA standards do not require any interface to return a manufacturer ID; many implementations put a manufacturer name in the model string.
FC Node WWN	NameFormat = NodeWWN(8) NameNamespace = NodeWWN(6)	16 un-separated upper case hex digits (e.g., '21000020372D3C73')

Storage volumes may have multiple standard names. A page 83 logical unit identifier shall be placed in the Name property with NameFormat and Namespace set as specified in Table 2. Each additional name should be placed in an element of OtherIdentifyingInfo. The corresponding element in IdentifyingDescriptions shall contain a string from the Values lists from NameFormat and NameNamespace, separated by a semi-colon. For example, an identifier from SCSI VPD page 83 with type 3, association 0, and NAA 0101b - the corresponding entry in IdentifyingDescriptions[] shall be "NAA;VPD83Type3".

For other types of devices, the logical unit name shall be in the Name property; NameFormat and NameNamespace are not valid properties of these other device classes.

7.6.3   Standard Formats for Port Names

Table 3 specifies standard formats for port names.

Table 3 - Standard Formats for Port Names

An IP interface's MAC	Network Port Permanent Address property; no corresponding format property	Six upper case hex bytes, bytes are delimited by colons ':'
World Wide Name (i.e. FC Name_Identifier)	FCPort Permanent Address property; no corresponding format property	16 un-separated upper case hex digits (e.g., '21000020372D3C73')
	ProtocolEndpoint Name property; ConnectionType = 2 (Fibre Channel)	16 un-separated upper case hex digits (e.g., '21000020372D3C73')

Table 3 - Standard Formats for Port Names

Parallel SCSI Name	SPI Port Name property; no corresponding format property	String - platform-specific name representing the name. Note that this name is only correlatable relative to the system containing the port.
	SCSIProtocolEndpoint Name property; ConnectionType = 3 (Parallel SCSI)	String - platform-specific name representing the name.
iSCSI Port Name	iSCSIProtocolEndpoint Name	< iSCSI node name > + ' i, ' + ISID for initiators, < iSCSI node name > + ' t, ' + TPGT for target ports, where < iSCSI node name> may be any of the standard iSCSI name namespaces (e.g., iqn, eui); and includes the namespace prefix.
SAS Port Names	SASPort Name property; no corresponding format property	SAS Address, 16 un-separated upper case hex digits
	SCSIProtocolEndpoint Name property; ConnectionType = 8 (SAS)	SAS Address, 16 un-separated upper case hex digits
ATA Port Name	ATAPort or SASSATAPort Name property; no corresponding format property	String - platform-specific name representing the name. Note that this name is only correlatable relative to the system containing the port.
	ATAProtocolEndpoint Nameproperty	String - platform-specific name representing the name.

Note that iSCSI Network Portals do not have a single correlable name. The combination of IPProtocolEndpoint IPv4Address or IPv6Address and TCPProtocolEndpoint PortNumber uniquely identifies the network portal, but since these are two properties, they do not form a correlatable name.

7.6.4   Standard Formats for Fabric Names

A fabric is modeled as AdminDomain. AdminDomain.Name shall hold the fabric name (i.e., WWN) and AdminDomain.NameFormat shall be set to "WWN". AdminDomain.Name shall be formatted as 16 unseparated upper case hex digits.

7.6.5   Standard Formats for Storage System Names

Due to the limited list of possible formats, the Name property is not considered an essential identifier for SMI-S. SMI-S clients should use OtherIdenfyiingInfo property as described in Table 4.

Providers shall supply at least one Durable or Correlatable Name as an element in the IdentifyingDescriptions[] array. The corresponding array elements of OtherIdentifyingInfo[] shall include a value from Table 4 for all elements of IdentifyingDescriptions[]. The elements in the IdentifyingDescriptions array are strings and may contain white space between words. Whenever white-space appears, it shall consist of a single blank; other white-space characters and multiple consecutive blanks shall not be used.

At least one of the values in IdentifyingDescriptions[] shall be something other than "SCSI Vendor Specific Name" or "Other Vendor Specific Name".

OtherIdentifyingInfo[0] should be assigned the most preferable name by the instrumentation.

In all cases, if the name is returned to the instrumentation in binary, the corresponding entry in OtherIdentifyingInfo holds an upper-case hexadecimal-encoded representation of the value returned. Standard names defined in binary are called out in Table 4.

Other ComputerSystem properties should be set as follows:

Name is a CIM key and shall be unique for ComputerSystem instances within the CIM namespace. SMI-S clients should not assume Name is either durable or correlatable.

NameFormat is an enumerated type describing the Name property. Only a few of the defined values are appropriate for storage systems. Use "IP" if Name is derived from an IP address of Fully Qualified Domain Name. Use "HID" if Name is derived from a hardware ID. Use "OID" if Name is a unique ID determined by some unique ID generating logic.

ElementName is a friendly name; SMI-S clients should not assume that ElementName is unique, correlatable, or durable since a customer may provide the same info for multiple systems.

Table 4 - Standard Formats for Storage System Names

IdentifyingDescriptions [x] value	Description		Format of OtherIdentifyinginfo[x]
T10 Target Name Type 1	An identifier from a Identification Vital Product Data page response with Association equal to 2	Type 1 (T10)	1 to 252 bytes of ASCII
T10 Target Name Type 2		Type 2 (EUI)	16, 24, or 32 un-separated upper case hex digits (e.g., '21000020372D3C73')
T10 Target Name Type 3		Type 3 (NAA)	16 or 32 un-separated upper case hex digits (e.g., '21000020372D3C73')
T10 Target Name Type 8		Type 8 (SCSI Names)	iSCSI Names (see 7.8)
T11 FC-GS-4 Platform Name	A platform name as defined in T11 FC-GS-4 standard		Up to 508 hex digits (254 bytes) as specified by T11 FC-GS-4 subclause on Platform Name. Format as unseparated as hex digits. Platform Name Format Byte shall be included.
T11 RNID Name	The sixteen byte Vendor Unique name from the General Topology Discovery format RNID response as defined in T11 FC LS standard. This name format should only be used if the storage system supports RNID General Topology Discovery and provides a meaning system identifier in the Vendor Unique field.		32 unseparated hex digits.
iSCSI Network Entity Name	An iSCSI Network Entity name.		iSCSI Names (see 7.8)
Ipv4 Address	An IP V4 name		Four decimal bytes delimited with dots ('.')

Table 4 - Standard Formats for Storage System Names (Continued)

IdentifyingDescriptions [x] value	Description		Format of OtherIdentifyinginfo[x]
Ipv6 Address	An IP V6 name		'x:x:x:x:x:x:x:x', where the 'x's are the uppercase hexadecimal values of the eight 16-bit pieces of the address. Examples: 'FEDC:BA98:7654:3210:FEDC: BA98:7654:3210', '1080:0:0:0:8:800:200C:417A' Leading zeros in individual fields should not be included and there shall be at least one numeral in every field. (This format is compliant with RFC 4291.) In addition, omitting groups of zeros or using dotted decimal format for an embedded IPv4 address is prohibited.
Fully Qualified Domain Name	A fully qualified domain name.		A legal DNS name (fully qualified) consisting of strings delimited by periods.
Node WWN	The Fibre Channel Node WWN. The provider shall assure that the same Node WWN shall be available through all FC ports within a target device.		16 un-separated upper case hex digits (e.g., '21000020372D3C73')
T10 Unit Serial Number VPD page	SCSI Inquiry VPD page 80 response is a serial number This name may be unique for a specific logical unit or for the target (e.g., storage system). These names are only valid if the instrumentation is certain that all logical units in a system return the same value. Since there is no mechanism to test whether the value is unique per target or per logical unit, this value is not interoperably correlatable and should not be used		1-252 ASCII characters
SCSI Vendor Specific Name	This is a name accessible through a vendor-specific SCSI command	A client with a priori knowledge may be able to correlate this based on vendor and Product IDs.	unknown
Other Vendor Specific Name	This is a name accessible through some non-SCSI vendor-specific interface.		unknown

7.6.6 Operating System Device Names

Each operating system has different conventions for naming devices. Many operating systems provide multiple names for the same device instance. In this version of the specification, operating system device name formats are recommended.

The case of names specified by operating system interfaces shall be preserved.

Operating system device names are unique within the namespace of the scoping system and are not unique between systems.

Table 5 specifies the format for names of tape devices.

Table 5 - Standard Operating System Names for Tape Devices

Operating System	Format	Notes
AIX	/dev/rmtX	X represents a hexadecimal number and may be more than one character
HP-UX	/dev/rmn/Xm	X represents a hexadecimal number and may be more than one character
Linux	/dev/stX	X represents one or two lower case alphabetic characters
Solaris	/dev/rmt/Xn	X represents a hexadecimal number and may be more than one character
WIndows	\\.\\TAPEX	X represents a decimal number

Some operating systems treat disk partitions as virtual devices; applications operate on partitions as if they were disks. The model requires two classes for each partition, LogicalDisk and GenericDiskPartition. Other operating systems allow applications to operate on the entire disk without partitions. Linux allows both.

Table 6 specifies the format for LogicalDisk.Name of disk partitions

Table 6 - LogicalDisk.Name for disk partitions

Operating System	Format	Notes
Linux	dev/sdXY or /dev/hdXY	where X represents one or two lower case alphabetic characters and Y represents an integer between 1 and 15
Solaris	/dev/dsk/cXtXdXsX	X represents one or two lower case alphabetic characters
WIndows	C: or the file name of mount point	C represents an uppercase letter
zSeries	CC:SS:DDDD or CC:DDDD	CC represents a Channel Subsystem Identifier, SS is a subchannel set (within the channel subsystem), and DDDD is the device number. SS is optional for subchannel set zero.

Table 7 specifies the format for GerneicDiskParition.Name and DeviceId properties for disk partitions

Table 7 - GenericDiskParittion.Name for disk partitions

Operating System	Format	Notes
Linux	sdXY or hdXY	X represents one or two lower case alphabetic characters
Solaris	/dev/dsk/cXtXdXsX	where X represents one or two lower case alphabetic characters and Y represents an integer between 1 and 15
WIndows	Disk #X, Partition #X	X represents a decimal digit

Table 8 specifies the format for LogicalDisk.Name for unpartitioned disks.

Table 8 - Standard Operating System Names for Unpartitioned DIsks

Operating System	Format	Notes
AIX	/dev/hdiskX	X represents a hexadecimal number and may be more than one character
HP-UX	/dev/dsk/cXtYdZ	X, Y, and Z represents hexadecimal number and may be more than one character in length
Linux	/dev/sdX or /dev/hdX	X represents one or two lower case alphabetic characters
Windows	\.\PHYSICALDRIVEx	x represents a a decimal number and may be more than one character

7.6.7   Case Sensitivity

Names and NameFormats are case sensitive and the cases provided in Table 8 shall be used If not otherwise specified, uppercase should be used.

7.7   Testing Equality of correlatable Names

The implementation shall only compare objects of the same class or parent class. For objects that do not require the use of additional properties, a simple direct comparison is sufficient, providing the format for the mandatory correlatable name as identified in this section or the specific profile is adhered to.

For objects that do require the use of additional properties (e.g., NameFormat), the correlatable names of objects representing the same entity should compare positively, negatively, or indicate clearly when a comparison is ambiguous:

•   If the two objects have the same NameFormat and Name, then they refer to the same resource.

•   If the two objects have the same NameFormat and different Names, then they refer to different resources.

•   If the two objects have different NameFormats, whether the Names are the same or different, then it is unknown whether they refer to the same resource.

This reduces the possibility that a match is missed by a string equals comparison simply because of an incompatibility of formats rather than non-equality of the data.

7.8   iSCSI Names

The iSCSI standards define three text formats for names that apply to various iSCSI elements. The three formats are: iSCSI qualified name (iqn), IEEE Extended Unique Identifier (eui), and ANSI T10 NAA. The format is included in the name as a three-letter prefix. The three formats are explained in more detail.

The iSCSI qualified name (iqn) format is defined in [iSCSI] and contains (in order):

1)   1 - The string "iqn."

2)   2 - A date code specifying the year and month in which the organization registered the domain or sub-domain name used as the naming authority string.

3)   3 - The organizational naming authority string, which consists of a valid, reversed domain or subdomain name.

Optionally, a ':', followed by a string of the assigning organization's choosing, which shall make each assigned iSCSI name unique.

Figure 14 contains examples of iSCSI-qualified names that may be generated by "EXAMPLE Storage, Inc."

```
          Organizational      Subgroup Naming Authority
                  Naming      and/or string Defined by
Type  Date      Auth        Org. or Local Naming Authority
+--++-----+ +---------+ +------------------------------+
|  ||     | |         | |                              |
iqn.2001-04.com.example:diskarrays-sn-a8675309
iqn.2001-04.com.example
iqn.2001-04.com.example:storage.tape1.sys1.xyz
iqn.2001-04.com.example:storage.disk2.sys1.xyz
```

Figure 14 - iSCSI Qualified Names (iqn) Examples

The IEEE Registration Authority provides a service for assigning globally unique identifiers [EUI]. The EUI-64 format is used to build a global identifier in other network protocols.

The format is "eui." followed by an EUI-64 identifier. Figure 15 contains an example.

```
        Type  EUI-64 identifier (ASCII-encoded hexadecimal)
        +--++--------------+
        |  ||              |
        eui.02004567A425678D
```

Figure 15 - iSCSI EUI Name Example

Type "naa." - Network Address Authority

The ANSI T10 FC-FS standard defines a format for constructing globally unique identifiers [FC-FS] referred to as an Network Address Authority (NAA) format. The iSCSI name format is "naa." followed by an NAA identifier (ASCII-encoded hexadecimal digits).

44

Figure 16 contains an example of an iSCSI name with a 64-bit NAA value: type NAA identifier (ASCII-encoded hexadecimal).

```
              +--++--------------+
              |  ||              |
              naa.52004567BA64678D
```

Figure 16 - iSCSI 64-bit NAA Name Example

Figure 17 contains an example of an iSCSI name with a 128-bit NAA value: type NAA identifier (ASCII-encoded hexadecimal).

```
              +--++------------------------+
              |  ||                         |
              naa.62004567BA64678D0123456789ABCDEF
```
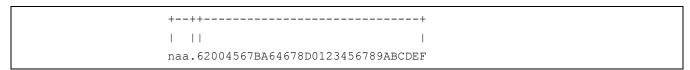
Figure 17 - iSCSI 128-bit NAA Name Example

iSCSI names are composed only of displayable characters. iSCSI names allow the use of international character sets but are not case sensitive. No whitespace characters are used in iSCSI names.

Clause 8: Policy

8.1   Objectives

Policy in the context of SMI-S refers to the common expression of policy in the management of storage. The specific objectives to be addressed by policy include:

1)   Provide for the exposure of element specific policies which control the behavior of management for the element. This includes:

- Native behavior currently unexposed through a standard interface;

- Behavior that can be implemented on behalf of the element by the SMI-S implementation;

2)   Provide a common expression that can be extended for vendor specific behavior;

3)   Provide for a mechanism that allows for embedded policy implementations;

4)   Provide for the implementation of policy external to individual elements;

5)   Provide for policies that work across multiple profiles and implementations of those profiles;

6)   Provide a policy mechanism that scales to enterprise environments;

7)   Extend existing DMTF standard policy models that are used for network and security;

8)   Provide a mechanism that allows SMI-S clients to determine the level of (SMI-S) policy support.

8.2   Overview

Policy is the expression of management behavior such that administrators and other management software can control that behavior, tailoring it to accomplish specific goals. Policy based storage management holds the promise of reducing the cost and complexity of the mostly manual management of storage resources today. Policies provide for a level of automation while allowing control over the behavior of that automation. Policies are envisioned as being implemented by management software and device vendors and manipulated and extended by administrators in order to achieve specific results in their environment.

Any good IT organization has specific policies and procedures as well as best practices that are followed (largely through manual tasks) by the IT personnel in managing the IT environment. The best organizations have documented these policies and have a process for updating and revising them. Policy based management allows for the creation and maintenance of management policies that automate the management of IT environments to achieve the desired goals of the business. These policies can themselves be managed just as the best organizations manage their written policies and procedures.

> **Note:** Management of policy implementations including policy services and management of policies themselves is left for a future version of SMI-S.

8.3   Policy Terms

A number of concepts have required new terms to be defined as follows:

- Policy Client - A CIM Client that creates or manipulates instances of policy classes in a CIM Server.

- Policy Implementation - The implementation of policy class instances in a CIM Server (i.e. providers).

- Policy Based - An SMI-S compliant implementation that supports one or more policy profiles directly.

- Policy Enabled - An SMI-S profile, subprofile or package that includes properties and methods that are used in one or more policy profiles.

8.4   Policy Definition

The expression of Policy in the CIM Model takes the form of policy rules that aggregate conditions and actions whereby upon successful evaluation of the conditions the actions are taken. Up until recently, the specialization of these base classes was along the lines of specific extensions for domains such as Networking and Security. Rather than create domain specific extensions for storage, a more general approach wa s taken.

The new extensions to the CIM Policy Model are meant to enable policies that act on anything that is itself modeled in CIM, testing conditions on instances in the model and invoking methods to manipulate those instances.

> **Note:** More information on the CIM Policy model and its application can be found in the DMTF Policy whitepaper. This clause does not attempt to duplicate that material.

8.4.1   Query Condition

The base PolicyCondition class is extended in such a way as to allow a condition to query any state that exists in implementations of the model. The new class QueryCondition allows a query string of unrestricted complexity to be used, just as any other CIM Client, to interrogate the model and create results. The presence of these results indicates that the condition evaluated to true. The absence means that the condition is false.

8.4.2   Method Action

The base PolicyAction class is extended so that any arbitrary method (extrinsic or intrinsic) can be invoked and appropriate parameters can be passed. This is also accomplished by a query string that, in this case, specifies how to use the results from the QueryCondition(s).

8.4.3   Query Condition Result

The result of a QueryCondition when it evaluates to TRUE (FALSE produces no results by definition) is one or more "rows" of embedded objects each with a predefined classname of QueryResultInstance whose properties match (both name and type) the query select criteria. This result is used possibly by other query conditions and method actions.

8.4.4   Method Action Result

The result of a QueryCondition or MethodAction is an instance indication that has scope and life only within an enclosing PolicyRule. This result is used by other conditions or methods in the Policy Rule. This is illustrated in Figure 18.

The implementation of policy can be limited to static instances of a policy model, specifically due to the cost in resources to implement full query language support. This is similar to the situation with Indication Filters in that static instances of QueryCondition and MethodActions and their associated query strings are available when a CIM Client interrogates the model for the device. Attempts to create new instances of these classes will fail because the logic embodied in the query strings is hard coded by the implementation.
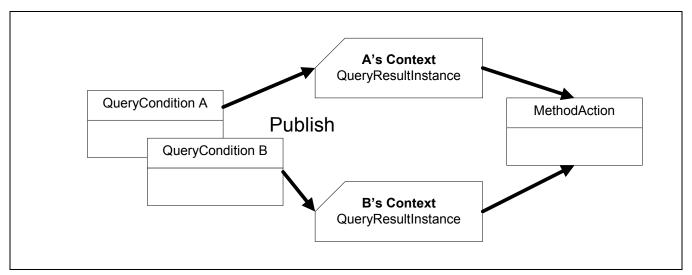
Figure 18 - Use of Results as Context in the Execution of a Policy Rule

There are basically three levels of support:

- Full Dynamic Policy Rules – a CIM Client can create new instances of PolicyRule, QueryCondition and MethodAction. Full support for a query language is implied in this level of support

- Dynamic Rules with Static Components – a CIM Client may discover existing QueryConditions and MethodActions, but may not create new ones. PolicyRule instances may be created to combine them in unlimited ways.

- Static Rules and Components – the full logic of the rule instances that already exist is hard coded by the implementation. The CIM Client only has control of what the policy applies to (via, for example. PolicySetAppliesToElement).

Any level of policy implementation may use the association PolicySetAppliesToElement to apply the policy to one or more elements in the same object manager, but the query strings need to already specify how the association is used.

8.4.5   Capabilities

The level of policy based support is driven by a capabilities class described in the policy subprofile. Normative text for implementing this capability class is covered there.

8.5   Policy Recipes

The principal recipe for policy is the creation of policies in a CIM Server as follows:

```
// DESCRIPTION
// This recipe describes how to create a policy. The assumption is made that
// there is only one policy implementation present in the system.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. The rule name is known as the #PolicyRuleName variable.
// 2. The condition name is known as the #PolicyConditionName variable.
// 3. The condition query is known as the #Query variable.
```

```
// MAIN
// Step 1. Create a Policy Rule
$PolicyRule = newInstance("CIM_PolicyRule")
$PolicyRule.setProperty("PolicyRuleName", #PolicyRuleName)
$PolicyRule.setProperty("Enabled", 2)// disabled
$PolicyRule.setProperty("SequencedActions", 1)// manditory
$PolicyRule.setProperty("ExecutionStrategy", 3)// do until failure
$PolicyRule-> = CreateInstance($PolicyRule)

// Step 2. Create a Policy Condition
$QueryCondition = newInstance("CIM_QueryCondition")
$QueryCondition.setProperty("PolicyConditionName", #PolicyConditionName)
$QueryCondition.setProperty("Query", #Query)
$QueryCondition.setProperty("QueryLanguage", "CQL")
$QueryCondition-> = CreateInstance($QueryCondition)

// Step 3. Associate Condition to the Rule
$PolicyConditionInRule = newInstance("CIM_PolicyConditionInPolicyRule")
$PolicyConditionInRule.setProperty("GroupComponent", $PolicyRule->)
$PolicyConditionInRule.setProperty("PartComponent", $QueryCondition->)
$PolicyConditionInRule-> = CreateInstance($PolicyConditionInRule)

// Step 4. Create the Action
$MethodAction = newInstance("CIM_MethodAction")
<Assign values to MethodAction attributes>
$PolicyAction-> = CreateInstance($MethodAction)

// Step 5. Associate the Action to the Rule
$PolicyActionStruct = newInstance("CIM_PolicyActionStructure")
$PolicyActionStruct.setProperty("GroupComponent", $PolicyRule->)
$PolicyActionStruct.setProperty("PartComponent", $PolicyAction->)
$PolicyActionStruct.setProperty("ActionOrder", 1)// Group 1
$PolicyActionStruct-> = CreateInstance($PolicyActionStruct)

// Step 6. Enable the Rule
$PolicyRule.Enabled = 1// Enabled
ModifyInstance($PolicyRule->,
     $PolicyRule,
     false,
     {"Enabled"})
```

EXPERIMENTAL

Clause 9: Standard Messages

9.1   Overview

Management of computer resources is, at times, fraught with exceptional conditions. SMI-S provides the means by which storage related computing resources can be controlled, configured, and, to some extent, monitored. This clause defines standard messages used in reporting the nature of these exceptional condition. Standard Messages are the expression of exceptional conditions in a managed device or application in a standard form. In other words, the indication of this condition as a standard message enables a client application that relies solely on SMI-S for instrumentation to take meaningful action in response.

There are two types of SMI-S enabled client applications supported by standard messages. The first type actively configures and controls. It requires the details why these types of operations failed to complete successfully.   The second type of client application is a passive observer of state changes from the SMI-S Agent. It is solely an observer.

Failures in active management may arise for three reasons. The first type of failure is caused by invalid parameters or an invalid combination of parameters to an extrinsic or intrinsic CIM Operation. The second type of failure may also be caused by reasons other than the way in which the operation was requested of the SMI-S agent.   The third type of failure may be result from an exception condition in the WBEM Infrastructure itself.

The monitoring client waits for indications of exception condition on the device or application it is monitoring.

A CIM Operations may be successful and return a response or they may be unnecessarily and return an error. The error is the combination of a standard CIM status code, like CIM_ERR_FAILED, a description, and Error instance. This clause uses the term *Error* for the Error instance returned.

A particular combination of state changes within the computer resource may arise from a single condition. The profile, subprofile, or package designers may choose to indicate the condition directly. This indication can be sent to the client, asynchronously, as a AlertIndication instance. This clause uses the term *Alert* for the AlertIndication instance. The combination of the standard message and the enclosing vehicle is called a standard event.

See *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 25:, "Health Package" for further details on this mechanism.

The Errors and Alerts produced need to be interoperability interpreted by the client application that receives them. Without such interoperability, the client developer would behavior details of the computer resource in question from other sources than SMI-S. This situation is undesirable for functionality specified in SMI-S because it means that the functionality specification is incomplete.

Some types of exceptional conditions may be both the Error resulting from some CIM Operation and an Alert, like 'system is shutting down'.   The same standard message should be conveyed either an Error or an Alert such that both types of clients can interpret the indication in the same manner.   Additionally, these types of exceptional conditions may be indicated from a read or write CIM Operation.

9.2   Required Characteristics of Standard Messages

9.2.1   Declaring and Producing Standard Messages

Standard Messages are defined in registries. Each registry is the collection of standard messages defined by a particular working group. In the case of SNIA, the registry is defined by particular working groups. Each working group works on a part or domain of the storage management problem. Each message as a unique id within the content of an owning organization, SNIA in this case, and working group.

Each message in the registry shall define values for the five message properties, OwningEntity, MessageID, Message, MessageArguments, and MessageFormatString. Since registries are a collection of messages and each registry is defined within the context of a owning entity, the owning entity is implied.

The message, as conveyed in an Error or Alert, and received by a client, shall contain the OwningEntity, MessageID, Message, and MessageArguments.   See *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* 25.1.3, "Standard Events" in the Health Package.

When the Message is produced, the variables defined in the MessageFormatString are replaced with the values from the MessageArguments array in the order in which the variables are defined. The MessageArguments array is an array of strings. So the implementation shall coerce the value in its native CIM data type to a string before adding that value to the MessageArguments. A client may coerce that value back to its native data type using the string coercion rules for each CIM data type.

An argument present in the MessageArguments array may itself be an array. The coercion of this array argument to a string element in the MessageArgument shall result in each value of the array argument to be delimited in the resulting string by a comma. If a value within the array argument contains a whitespace, then the value of that element shall appear in the MessageArgument element contained within matching double quotes in the resulting common delimited list of array argument elements. The resulting comma delimited list of array arguments elements shall contain no whitespace characters other that those that are part of a element value.

Neither the Message nor the MessageArguments shall contain non-printable characters other than the whitespace.

The Message shall be localized in the language requested by the client. See the CIM Operations specification, version 1.2, for details on internationalization with WBEM.

A Standard Message may be conveyed with an Error or an Alert. The omission of specific values for the other properties in the Error or Alert instance does not imply that this message may not be conveyed in the omitted form.

Table 9 - Example Standard Message Declaration

Message Property	Value
OwningEntity	SNIA
MessageID	MP5
MessageFormatString"	Parameter <Position> of the <Method Type> method, <Method Name>, is invalid producing <Status Code>. <Additional Status>
MessageArguments	Position: The position the errant argument appears in the declaration of the method, from left to right. Method Type: intrinsic or extrinsic Method Name Status Code: CIM Status Code <status code> Additional Status: Additional circumstances describing the error (ex. Parameter out of range).

Given the following method declaration:

```
uint32 RequestStateChange(
        [IN, Description ("…"),
         ValueMap { "2", "3", "4", "5", "6", "7..32767",
            "32768..65535" },
         Values { "Start", "Suspend", "Terminate", "Kill", "Service",
            "DMTF Reserved", "Vendor Reserved" }]
```

```
                        uint16 RequestedState,
                            [IN, Description ("…")]
                        datetime TimeoutPeriod);
```

A client makes the following call

```
                RequestStateChange("1", null);
```

"1" is an invalid RequestedState. Therefore, the target of the CIM Operations will produce a Error.

Table 10 - Example Standard Message Values

Message Property	Value
OwningEntity	SNIA
MessageID	MP5
Message	Parameter 0 of the extrinsic method, RequestStateChange, is invalid producing CIM_ERR_INVALID_PARAMETER CIM Error. Parameter out of range.
MessageArguments	"0" "extrinsic" RequestedStateChange "CIM_ERR_INVALID_PARAMETER" "Parameter out of range"

9.3   Message Registry

9.3.1   Common Element Messages

9.3.1.0.1   Message: Redundancy

Owning Entity: SNIA

Message ID: Core1

Message Format String: <Device Type> <Device Unique Identifier> had redundancy failure for <Component Type> at <Component Location Or Identifier>

Table 11 describes the message arguments.

Table 11: Redundancy Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	

Table 11: Redundancy Message Arguments

Message Argument	Data Type	Description	Possible Values
Component Type	string	The type of component a redundancy failure has occurred for. Typically the string would contain one of the following: "Power Component", "Fan Component", "Board Component", Cross Bar", "System Clock'	Power Component
			Fan Component
			Board Component
			Cross Bar
			System Clock
Component Location Or Identifier	string	Location or identifier of the component	

Table 12 describes the alerts that are associated with this message.

Table 12 - Redundancy Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	Y		The system containing the device.
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		Minor

9.3.1.0.2   Message: Environmental

Owning Entity: SNIA

Message ID: Core2

Message Format String: <Device Type> <Device Unique Identifier> had an environmental problem of type <SensorType> of <Environmental Issue> <Sensor Location Or Identifier>

Table 13 describes the message arguments.

Table 13 - Environmental Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array

Table 13 - Environmental Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
SensorType	string	The sensor type.	temperature
			humidity
Environmental Issue	string	The enivornmental issue.	
Sensor Location Or Identifier	string	Location or identifier of the Sensor	

Table 14 describes the alerts that are associated with this message.

Table 14 - Environmental Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		
ALERT_TYPE	Y		Environmental Alert
PERCEIVED_SEVERITY	Y		Minor

9.3.1.0.3   Message: FRU Operation

Owning Entity: SNIA

Message ID: Core3

Message Format String: <Device Type> <Device Unique Identifier> had a Field Replaceable Unit (FRU) <The FRU Operation> on <FRU Type> at <FRU Location Or Identifier>

Table 15 describes the message arguments.

Table 15 - FRU Operation Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
The FRU Operation	string	The operation on the FRU that is the basis of the message	removed
			added

Table 15 - FRU Operation Message Arguments

Message Argument	Data Type	Description	Possible Values
			replaced
			incompatible
FRU Type	string	The Type of FRU	
FRU Location Or Identifier	string	Location or the Identifier of the FRU	

Table 16 describes the alerts that are associated with this message.

Table 16 - FRU Operation Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	Y		
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		High

9.3.1.0.4   Message: Password change

Owning Entity: SNIA

Message ID: Core4

Message Format String: <Device Type> <Device Unique Identifier> password has change for user <User Identification>

Table 17 describes the message arguments.

Table 17 - Password change Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
User Identification	string	User or Account Identification	

56

Table 18 describes the alerts that are associated with this message.

Table 18 - Password change Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		
ALERT_TYPE	Y		
PERCEIVED_SEVERITY	Y		Informational

9.3.1.0.5   Message: User or Account Operation

Owning Entity: SNIA

Message ID: Core5

Message Format String: <Device Type> <Device Unique Identifier> user <User Identification> <User Operation>

Table 19 describes the message arguments.

Table 19 - User or Account Operation Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
User Identification	string	User or Account Identification	
User Operation	string	Operation on User	removed
			disabled
			added

Table 20 describes the alerts that are associated with this message.

Table 20 - User or Account Operation Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		
ALERT_TYPE	Y		Security Alert
PERCEIVED_SEVERITY	Y		Informational

9.3.1.0.6   Message: User Login

Owning Entity: SNIA

Message ID: Core6

Message Format String: <Device Type> <Device Unique Identifier> user <User> <Login Operation>

Table 21 describes the message arguments.

Table 21 - User Login Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
User	string	The user or account name	
Login Operation	string	Operation on User	logged in
			logged out

Table 22 describes the alerts that are associated with this message.

Table 22 - User Login Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		The object name must reference the top-most computer system that is shutting down. If the computer system is cluster, then the cluster computer system must be referenced.
ALERT_TYPE	Y		Security Alert
PERCEIVED_SEVERITY	Y		Informational

9.3.1.0.7   Message: Proxy Agent Device Communication

Owning Entity: SNIA

Message ID: Core7

Message Format String: Agent <Agent Identifier> <Agent Connectivity> communication with <Device Type> <Device Unique Identifier>

Table 23 describes the message arguments.

Table 23 - Proxy Agent Device Communication Message Arguments

Message Argument	Data Type	Description	Possible Values
Agent Identifier	string	An indentifier for the SMI Agent	
Agent Connectivity	string	A description for the connectivity	lost
			regained
Device Type	string	A description of the type of element	Switch
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	

Table 24 describes the alerts that are associated with this message.

Table 24 - Proxy Agent Device Communication Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		Major

9.3.1.0.8   Message: Port Status Changed

Owning Entity: SNIA

Message ID: Core9

Message Format String: FC Port <Port Identifier> in <Device Type> <Device Unique Identifier> status changed to <Port Status>

Table 25 describes the message arguments.

Table 25 - Port Status Changed Message Arguments

Message Argument	Data Type	Description	Possible Values
Port Identifier	string	The Fibre CHannel Port Name (WWN).	
Device Type	string		Switch
			HBA
			Array
			Fabric

Table 25 - Port Status Changed Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
Port Status	string	Fibre Channel Port Status. This should be the same value as the OperationalStatus for the FCPort.	

Table 26 describes the alerts that are associated with this message.

Table 26 - Port Status Changed Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	Y		
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		High

9.3.1.0.9   Message: Datacheck Error

Owning Entity: SNIA

Message ID: Core9

Message Format String: <Device Type> <Device Unique Identifier> data check (<Data Check Type>

Table 27 describes the message arguments.

Table 27 - Datacheck Error Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
Data Check Type	string	The type of data check that occured.	Switch
			CRC

Table 28 describes the alerts that are associated with this message.

Table 28 - Datacheck Error Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		High

9.3.1.0.10   Message: User Login Failure

Owning Entity: SNIA

Message ID: Core10

Message Format String: <Device Type> <Device Unique Identifier> user <User> had login failure at

Table 29 describes the message arguments.

Table 29 - User Login Failure Message Arguments

Message Argument	Data Type	Description	Possible Values
Device Type	string	A description of the type of element	Switch
			HBA
			Array
Device Unique Identifier	string	An identifier for the device (host, array, switch, etc.).	
User	string	The user or account name	

Table 30 describes the alerts that are associated with this message.

Table 30 - User Login Failure Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		The object name must reference the top-most computer system that is shutting down. If the computer system is cluster, then the cluster computer system must be referenced.
ALERT_TYPE	Y		Security Alert
PERCEIVED_SEVERITY	Y		Warning

9.3.1.0.11

9.3.2   Common Protocol Messages

9.3.2.0.1   Message: Authorization Failure

Owning Entity: SNIA

Message ID: MP1

Message Format String: <Type of Operation> Access is Denied

Table 31 describes the message arguments.

Table 31: Authorization Failure Message Arguments

Message Argument	Data Type	Description	Possible Values
Type of Operation	string	Type of operation attempted.	Creation
			Modification
			Deletion
			Execution

Table 32 describes the error properties.

Table 32 - Authorization Failure Error Properties

Property	Value	Description
CIMSTATUSCODE	2 (CIM_ERR_ACCESS_DENIED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(A reference to the object to whom access is requested.)	Existence is required
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.2.0.2   Message: Operation Not Supported

Owning Entity: SNIA

Message ID: MP2

Message Format String: <CIM Operation> is not supported.

Table 33 describes the message arguments.

Table 33 - Operation Not Supported Message Arguments

Message Argument	Data Type	Description	Possible Values
CIM Operation	string		GetClass
			GetInstance

Table 33 - Operation Not Supported Message Arguments

Message Argument	Data Type	Description	Possible Values
			DeleteClass
			DeleteInstance
			CreateClass
			CreateInstance
			ModifyClass
			ModifyInstance
			EnumerateClasses
			EnumerateInstances
			EnumerateInstanceNames
			ExecQuery
			Associators
			AssociatorNames
			References
			ReferenceNames
			GetProperty
			SetProperty
			GetQualifier
			SetQualifier
			DeleteQualifier
			EnumerateQualifier

9.3.2.0.3   Message: Property Not Found

Owning Entity: SNIA

Message ID: MP3

Message Format String: <Property Name> property was not found in the <Class name> class.

Table 34 describes the message arguments.

Table 34 - Property Not Found Message Arguments

Message Argument	Data Type	Description	Possible Values
Property Name	string	The property name is specified as it was passed by the client.	
Class name	string	The property name is specified as it was passed by the client.	

9.3.2.0.4   Message: Invalid Query

Owning Entity: SNIA

Message ID: MP4

Message Format String: Query language is not supported. The query language supported are <Supported Query Languages>

Table 35 describes the message arguments.

Table 35 - Invalid Query Message Arguments

Message Argument	Data Type	Description	Possible Values
Supported Query Languages	string		

9.3.2.0.5   Message: Parmeter Error

Owning Entity: SNIA

Message ID: MP5

Message Format String: Parameter <Position> of the <Method Type> method, <Method Name> , is invalid producing <Status Code> . <Additional Status>

Table 36 describes the message arguments.

Table 36 - Parmeter Error Message Arguments

Message Argument	Data Type	Description	Possible Values
Position	uint16	The position the errant argument appears in the declaration of the method, from left to right.	
Method Type	string		extrinsic
			intrinsic
Method Name	string		
Status Code	string		no

Table 36 - Parmeter Error Message Arguments

Message Argument	Data Type	Description	Possible Values
			CIM Status Code: Add status code number after the above
Additional Status	string		parameter value out of range
			invalid combination
			null parameter is not permitted
			non-null value is not permitted
			empty string is not permitted
			empty array is not permitted

Table 37 describes the error properties.

Table 37 - Parmeter Error Error Properties

Property	Value	Description
CIMSTATUSCODE	4 (CIM_ERR_INVALID_PARAMETER)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(It is discouraged from specifying any reference here.)	Existence is discouraged
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.2.0.6   Message: Query Syntax Error

Owning Entity: SNIA

Message ID: MP6

Message Format String: Syntactical error on query: <Errant Query Components> <Syntax Errors>

Table 38 describes the message arguments.

Table 38 - Query Syntax Error Message Arguments

Message Argument	Data Type	Description	Possible Values
Errant Query Components	string	The parts of the query that are in error with a carrot '^' in front of text that is in error	
Syntax Errors	string	The syntax errors for each of the query components in the previous argument. The two arrays are to match element to element.	

Table 39 describes the error properties.

Table 39 - Query Syntax Error Error Properties

Property	Value	Description
CIMSTATUSCODE	4 (CIM_ERR_INVALID_QUERY)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(It is discouraged from specifying any reference here.)	Existence is discouraged
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.2.0.7   Message: Query Too Expensive

Owning Entity: SNIA

Message ID: MP7

Message Format String: Query is too expensive because the <Rejection Reason>

Table 40 describes the message arguments.

Table 40 - Query Too Expensive Message Arguments

Message Argument	Data Type	Description	Possible Values
Rejection Reason	string		result set will be too big
			query will take too must computing resources to process

Table 41 describes the error properties.

Table 41 - Query Too Expensive Error Properties

Property	Value	Description
CIMSTATUSCODE	4 (CIM_ERR_INVALID_QUERY)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(It is discouraged from specifying any reference here.)	Existence is discouraged
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.2.0.8   Message: Class or Property Invalid in Query

Owning Entity: SNIA

Message ID: MP8

Message Format String: Invalid <Invalid Query Component>

66

Table 42 describes the message arguments.

Table 42 - Class or Property Invalid in Query Message Arguments

Message Argument	Data Type	Description	Possible Values
Invalid Query Component	string	This argument shall contain the 'class name' or 'class name'.'property name'	

Table 43 describes the error properties.

Table 43 - Class or Property Invalid in Query Error Properties

Property	Value	Description
CIMSTATUSCODE	4 (CIM_ERR_INVALID_QUERY)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(It is discouraged from specifying any reference here.)	Existence is discouraged
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.2.0.9   Message: Invalid Join in Query

Owning Entity: SNIA

Message ID: MP9

Message Format String: Invalid join clause: <Invalid Join Clause>

Table 44 describes the message arguments.

Table 44 - Invalid Join in Query Message Arguments

Message Argument	Data Type	Description	Possible Values
Invalid Join Clause	string	This argument shall contain the entire join clause that is in error.	

Table 45 describes the error properties.

Table 45 - Invalid Join in Query Error Properties

Property	Value	Description
CIMSTATUSCODE	4 (CIM_ERR_INVALID_QUERY)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required

Table 45 - Invalid Join in Query Error Properties

Property	Value	Description
ERROR_SOURCE	(It is discouraged from specifying any reference here.)	Existence is discouraged
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.2.0.10   Message: Unexpected Hardware Fault

Owning Entity: SNIA

Message ID: MP10

Message Format String: Call technical support and report the following error number has occurred, <Hardware Error>

Table 46 describes the message arguments.

Table 46 - Unexpected Hardware Fault Message Arguments

Message Argument	Data Type	Description	Possible Values
Hardware Error	sint32	Vendor specific hardware error. Use this error, only when all other standard messages can not cover this condition.	

Table 47 describes the error properties.

Table 47 - Unexpected Hardware Fault Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	(It is discouraged from specifying any reference here.)	Existence is discouraged
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.2.0.11   Message: Too busy to respond

Owning Entity: SNIA

Message ID: MP11

Message Format String: WBEM Server is <Adverse Condition> to respond.

Table 48 describes the message arguments.

Table 48 - Too busy to respond Message Arguments

Message Argument	Data Type	Description	Possible Values
Adverse Condition	string		too busy
			initializing

9.3.2.0.12   Message: Shutdown Started

Owning Entity: SNIA

Message ID: MP12

Message Format String: The computer system is shutting down in <seconds to shutdown> seconds.

Table 49 describes the message arguments.

Table 49 - Shutdown Started Message Arguments

Message Argument	Data Type	Description	Possible Values
seconds to shutdown	uint32	The number of seconds before the system is shutdown.	

Table 50 describes the alerts that are associated with this message.

Table 50 - Shutdown Started Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	Y		The object name must reference the top-most computer system that is shutting down. If the computer system is cluster, then the cluster computer system must be referenced.
ALERT_TYPE	Y	5	Device Alert
PERCEIVED_SEVERITY	Y	4	High

9.3.2.0.13   Message: Component overheat

Owning Entity: SNIA

Message ID: MP13

Message Format String: A component has overheated. <Component Type>

Table 51 describes the message arguments.

Table 51 - Component overheat Message Arguments

Message Argument	Data Type	Description	Possible Values
Component Type	string		The entire device is affected. Device wide failure has already or can be expected shortly.
			Only a single component is affected. Corrective action may be taken.

Table 52 describes the error properties.

Table 52 - Component overheat Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	6 (Environment Error)	Existence is required
ERROR_SOURCE	(The object name must reference the physical element most affected by the over temperature message.)	Existence is required
PERCEIVED_SEVERITY	4 (High)	Existence is required

Table 53 describes the alerts that are associated with this message.

Table 53 - Component overheat Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	Y		The object name must reference the physical element most affected by the over temperature message.
ALERT_TYPE	Y	6	Environmental Alert
PERCEIVED_SEVERITY	Y	4	High

9.3.2.0.14   Message: WBEM Management Interface is not available

Owning Entity: SNIA

Message ID: MP14

Message Format String: The management interface for the device is not available.

9.3.2.0.15   Message: Device Failover

Owning Entity: SNIA

Message ID: MP15

Message Format String: Management interface is active on different device at the following URI, <URI>

Table 54 describes the message arguments.

Table 54 - Device Failover Message Arguments

Message Argument	Data Type	Description	Possible Values
URI	string		

9.3.2.0.16   Message: Functionality is not licensed

Owning Entity: SNIA

Message ID: MP16

Message Format String: Functionality requested is not licensed. The following license is required, <Required License Name>

Table 55 describes the message arguments.

Table 55 - Functionality is not licensed Message Arguments

Message Argument	Data Type	Description	Possible Values
Required License Name	string		

Table 56 describes the error properties.

Table 56 - Functionality is not licensed Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Reference to top most Computer System.)	Existence is required
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

9.3.2.0.17   Message: Invalid Property Combination during instance creation or modification

Owning Entity: SNIA

Message ID: MP17

Message Format String: The instance contains an invalid combination of properties. The <Errant Property Name> property may not have the value, <Errant Property Value> , when the <Existing Property Name> property has value, <Existing Property Value>

Table 57 describes the message arguments.

Table 57 - Invalid Property Combination during instance creation or modification Message Arguments

Message Argument	Data Type	Description	Possible Values
Errant Property Name	string	The name of the property is primary reason for the rejection of this instance.	
Errant Property Value	string	The invalid property value, coersed as a string.	
Existing Property Name	string	The property whose value has to be set in some way before or regardless of the "Errant Property Name" property. For example, property A of value X may be compatible with property B with value Y. But, property B may have had value Y prior to property A having a value or value X. Or, property B may be a key and must logically have a value before any other property set operation is considered.	
Existing Property Value	string		

Table 58 describes the error properties.

Table 58 - Invalid Property Combination during instance creation or modification Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Nothing to reference.)	Existence is discouraged
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

9.3.2.0.18   Message: Property Not Found

Owning Entity: SNIA

Message ID: MP18

Message Format String: <Errant Property Name> property was not found in class <Class Name used in Operation>

Table 59 describes the message arguments.

Table 59 - Property Not Found Message Arguments

Message Argument	Data Type	Description	Possible Values
Errant Property Name	string	The name of the property provided in a instance related CIM Operation that simply does not exist in the class as indicated by the class name.	
Class Name used in Operation	string	The class name used in the CIM Operation as stated directly as a method parameters or as part of a CIM Object Name (CIM Object Path).	

Table 60 describes the error properties.

Table 60 - Property Not Found Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Reference the class in question.)	Existence is required
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

9.3.2.0.19   Message: Proxy Can Not Connect

Owning Entity: SNIA

Message ID: MP19

Message Format String: Proxy CIM provider can not connect. <Reason for Connection Failure>

Table 61 describes the message arguments.

Table 61 - Proxy Can Not Connect Message Arguments

Message Argument	Data Type	Description	Possible Values
Reason for Connection Failure	string	The reason for the connection failure.	Authentication Failure
			Authorization Failure
			Communications Failure

Table 62 describes the error properties.

Table 62 - Proxy Can Not Connect Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Nothing to reference.)	Existence is discouraged
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

9.3.2.0.20   Message: Not Enough Memory

Owning Entity: SNIA

Message ID: MP20

Message Format String: <Method Type> method <Method Name> can not be complted because of lack of memory.

Table 63 describes the message arguments.

Table 63 - Not Enough Memory Message Arguments

Message Argument	Data Type	Description	Possible Values
Method Type	string		intrinsic
			extrinsic
Method Name	string	The method name. If the method is an intrinsic method, provide the CIM Operation Name, e.g. EnumerateInstances. If the method is an extrinswic method, i.e. InvokeMethod, then provide the method name in the class that was invoked.	

Table 64 describes the error properties.

Table 64 - Not Enough Memory Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Nothing to reference.)	Existence is discouraged
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

9.3.2.0.21   Message: Object Already Exists

Owning Entity: SNIA

Message ID: MP21

Message Format String: Object already exists.

Table 65 describes the error properties.

Table 65 - Object Already Exists Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	Reference to the already existing zone element. ()	Existence is required
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.3   Storage Messages

9.3.3.0.1   Message: Device Not ready

Owning Entity: SNIA

Message ID: DRM1

Message Format String: Device <Device ID> not ready because of <StatusOrStatus> state or status.

Table 31 describes the message arguments.

Table 66: Device Not ready Message Arguments

Message Argument	Data Type	Description	Possible Values
Device ID	string	LogicalDevice.DeviceID, PhysicalElement.Tag, or ComputerSystem.Name	
StatusOrStatus	string	Relavent State or Status the most explains the reason for the production of this message.	

Table 32 describes the error properties.

Table 67 - Device Not ready Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	(Object Name for the top-level object for the device, which is typically the computer system instance)	Existence is required
PERCEIVED_SEVE RITY	4 (High)	Existence is required

9.3.3.0.2   Message: Internal Bus Error

Owning Entity: SNIA

Message ID: DRM2

Message Format String: Internal Bus Error

Table 68 describes the error properties.

Table 68 - Internal Bus Error Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	(Object Name for the top-level object for the device, which is typically the computer system instance)	Existence is required
PERCEIVED_SEVE RITY	4 (High)	Existence is required

9.3.3.0.3   Message: DMA Overflow

Owning Entity: SNIA

Message ID: DRM3

Message Format String: DMA Overflow

Table 69 describes the error properties.

Table 69 - DMA Overflow Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required

Table 69 - DMA Overflow Error Properties

Property	Value	Description
ERROR_SOURCE	Object Name for the top-level object for the device, which is typically the computer system instance ()	Existence is required
PERCEIVED_SEVERITY	4 (High)	Existence is required

9.3.3.0.4   Message: Firmware Logic Error

Owning Entity: SNIA

Message ID: DRM4

Message Format String: Firmware Logic Error

Table 70 describes the error properties.

Table 70 - Firmware Logic Error Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	Object Name for the top-level object for the device, which is typically the computer system instance ()	Existence is required
PERCEIVED_SEVERITY	4 (High)	Existence is required

9.3.3.0.5   Message: Front End Port Error

Owning Entity: SNIA

Message ID: DRM5

Message Format String: Front End Port Error on Device identified by <Device ID>

Table 71describes the message arguments.

Table 71 - Front End Port Error Message Arguments

Message Argument	Data Type	Description	Possible Values
Device ID	string	LogicalDevice.DeviceID	

Table 72 describes the alerts that are associated with this message.

Table 72 - Front End Port Error Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		Object Name for the top-level object for the device, which is typically the computer system instance
ALERT_TYPE	Y	2	Communications Alert
PERCEIVED_SEVERITY	Y	4	High

9.3.3.0.6   Message: Back End Port Error

Owning Entity: SNIA

Message ID: DRM6

Message Format String: Back End Port Error on Device identified by <Device ID>

Table 73 describes the message arguments.

Table 73 - Back End Port Error Message Arguments

Message Argument	Data Type	Description	Possible Values
Device ID	string	LogicalDevice.DeviceID	

Table 113 describes the alerts that are associated with this message.

Table 74 - Back End Port Error Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		Object Name for the top-level object for the device, which is typically the computer system instance
ALERT_TYPE	Y	2	Communications Alert
PERCEIVED_SEVERITY	Y	4	High

9.3.3.0.7   Message: Remote Mirror Error

Owning Entity: SNIA

Message ID: DRM7

Message Format String: Error detected associated with remote volume, <Remote Volume Name>

Table 75 describes the message arguments.

Table 75 - Remote Mirror Error Message Arguments

Message Argument	Data Type	Description	Possible Values
Remote Volume Name	string	StorageVolume.Name	

Table 76 describes the error properties.

Table 76 - Remote Mirror Error Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	(Object Name for the top-level object for the remote block server, which is typically the computer system instance. The implementation will have to implement the Cascading Subprofile.)	Existence is optional
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

Table 77 describes the alerts that are associated with this message.

Table 77 - Remote Mirror Error Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	N		Object Name for the top-level object for the remote block server, which is typically the computer system instance. The implementation will have to implement the Cascading Subprofile.
ALERT_TYPE	Y		
PERCEIVED_SEVERITY	Y	3	Medium

9.3.3.0.8   Message: Cache Memory Error

Owning Entity: SNIA

Message ID: DRM8

Message Format String: Cache Memory Error

Table 78describes the error properties.

Table 78 - Cache Memory Error Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	(Object Name for the top-level object for the device, which is typically the computer system instance)	Existence is required
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

9.3.3.0.9   Message: Unable to Access Remote Device

Owning Entity: SNIA

Message ID: DRM9

Message Format String: Unable to Access Remote Device

Table 79 describes the error properties.

Table 79 - Unable to Access Remote Device Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	(Object Name for the top-level object for the remote block server, which is typically the computer system instance. The implementation will have to implement the Cascading Subprofile.)	Existence is optional
PERCEIVED_SEVERITY	3 (Medium)	Existence is required

9.3.3.0.10   Message: Error Reading Data

Owning Entity: SNIA

Message ID: DRM10

Message Format String: Error Reading Data

Table 80 describes the alerts that are associated with this message.

Table 80 - Error Reading Data Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	Y		Object Name for the top-level object for the device, which is typically the computer system instance
ALERT_TYPE	Y	2	Communications Alert
PERCEIVED_SEVERITY	Y	3	Medium

9.3.3.0.11   Message: Error Writing Data

Owning Entity: SNIA

Message ID: DRM11

Message Format String: Error Writing Data

Table 81 describes the alerts that are associated with this message.

Table 81 - Error Writing Data Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		Object Name for the top-level object for the device, which is typically the computer system instance
ALERT_TYPE	Y	2	Communications Alert
PERCEIVED_SEVERITY	Y	3	Medium

9.3.3.0.12   Message: Error Validating Write (CRC)

Owning Entity: SNIA

Message ID: DRM12

Message Format String: Error Validating Write

Table 82 describes the alerts that are associated with this message.

Table 82 - Error Validating Write (CRC) Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		Object Name for the top-level object for the device, which is typically the computer system instance
ALERT_TYPE	Y	2	Communications Alert
PERCEIVED_SEVERITY	Y	3	Medium

9.3.3.0.13   Message: Copy Operation Failed

Owning Entity: SNIA

Message ID: DRM13

Message Format String: Copy Operation Failed

Table 83 describes the error properties.

Table 83 - Copy Operation Failed Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	()	Existence is discouraged
PERCEIVED_SEVE RITY	3 (Medium)	Existence is required

9.3.3.0.14   Message: RAID Operation Failed

Owning Entity: SNIA

Message ID: DRM14

Message Format String: RAID Operation Failed

Table 84 describes the error properties.

Table 84 - RAID Operation Failed Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	5 (Hardware Error)	Existence is required
ERROR_SOURCE	()	Existence is discouraged
PERCEIVED_SEVE RITY	3 (Medium)	Existence is required

9.3.3.0.15   Message: Invalid RAID Type
Owning Entity: SNIA
Message ID: DRM15
Message Format String: Invalid RAID Type

Table 85 describes the error properties.

Table 85 - Invalid RAID Type Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	10 (Unsupported Operation Error)	Existence is required
ERROR_SOURCE	()	Existence is discouraged
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.3.0.16   Message: Invalid Storage Element Type
Owning Entity: SNIA
Message ID: DRM16
Message Format String: Invalid Device Type

Table 86 describes the error properties.

Table 86 - Invalid Storage Element Type Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	10 (Unsupported Operation Error)	Existence is required

Table 86 - Invalid Storage Element Type Error Properties

Property	Value	Description
ERROR_SOURCE	()	Existence is discouraged
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.3.0.17   Message: Configuration Change Failed

Owning Entity: SNIA

Message ID: DRM17

Message Format String: Configuration Change Failed

Table 87describes the error properties.

Table 87 - Configuration Change Failed Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	Object Name for the top-level object for the device, which is typically the computer system instance ()	Existence is required
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.3.0.18   Message: Buffer Overrun

Owning Entity: SNIA

Message ID: DRM18

Message Format String: Buffer Overrun

Table 60 describes the error properties.

Table 88 - Buffer Overrun Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	Object Name for the top-level object for the device, which is typically the computer system instance ()	Existence is required
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.3.0.19   Message: Stolen Capacity

Owning Entity: SNIA

Message ID: DRM19

Message Format String: The capacity requested, <Requested Capacity> , that was requested is no longer available.

Table 89 describes the message arguments.

Table 89 - Stolen Capacity Message Arguments

Message Argument	Data Type	Description	Possible Values
Requested Capacity	sint64	Capacity requested in bytes expressed in powers of 10.	

Table 90 describes the error properties.

Table 90 - Stolen Capacity Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(The pool, volume, or logical disk being modified, or, in the case of element creation the parent pool from which capacity is being drawn.)	Existence is required
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.3.0.20   Message: Invalid Extent passed

Owning Entity: SNIA

Message ID: DRM20

Message Format String: One or more of the extents passed can not be used to create or modify storage elements. <Invalid Extents Array>

Table 91 describes the message arguments.

Table 91 - Invalid Extent passed Message Arguments

Message Argument	Data Type	Description	Possible Values
Invalid Extents Array	reference	Array of references to the all Extents that can not be used in the specified manner (ex. CreateOrModifyStroragePool or CreateOrModifyElementsFromElements).	

Table 92 describes the error properties.

Table 92 - Invalid Extent passed Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(A reference to the storage configuration service instance on which the method was called that caused this error.)	Existence is required
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.3.0.21   Message: Invalid Deletion Attempted

Owning Entity: SNIA

Message ID: DRM21

Message Format String: Existing pool or storage element (StorageVolume or LogicalDisk) may not be deleted because there are existing Storage Extents which relay on it.

Table 93describes the error properties.

Table 93 - Invalid Deletion Attempted Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(A reference to one of the dependent StorageExtents.)	Existence is required
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.3.0.22   Message: Job Failed to Start

Owning Entity: SNIA

Message ID: DRM22

Message Format String: Job failed to start because resources required for method execution are no longer available.

Table 94 describes the error properties.

Table 94 - Job Failed to Start Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	8 (Oversubscription Error)	Existence is required

Table 94 - Job Failed to Start Error Properties

Property	Value	Description
ERROR_SOURCE	(Reference to Job instance which failed to start for this reason if a Job instance was created because of the time required to make this resource assessment. If a Job instance was not created, because the assessment was fast enough, then this property must be NULL.)	Existence is required
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.3.0.23   Message: Job was Halted

Owning Entity: SNIA

Message ID: DRM23

Message Format String: Job was <Reason for Job halt>

Table 95 describes the message arguments.

Table 95 - Job was Halted Message Arguments

Message Argument	Data Type	Description	Possible Values
Reason for Job halt	string	A Job may be stopped by a client using the RequestedStateChange method. If the job stopped executing for other reasons, then use a different message.	killed
			terminated

9.3.3.0.24   Message: Invalid State Transition

Owning Entity: SNIA

Message ID: DRM24

Message Format String: An invalid state transition, <Invalid Sync State> , was requested given current state, <Current Sync State>

Table 96 describes the message arguments.

Table 96 - Invalid State Transition Message Arguments

Message Argument	Data Type	Description	Possible Values
Invalid Sync State	string	The textual equivalent (Value) for StorageSynchronized.SyncState value requested.	
Current Sync State	string	The textual equivalent (Value) for the current StorageSynchronized.SyncState value	

Table 97 describes the error properties.

Table 97 - Invalid State Transition Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Reference to the StorageSynchronized instance in question.)	Existence is required
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.3.0.25   Message: Invalid SAP for Method

Owning Entity: SNIA

Message ID: DRM25

Message Format String: Invalid type of copy services host. The host must be a <Host Type>

Table 98 describes the message arguments.

Table 98 - Invalid SAP for Method Message Arguments

Message Argument	Data Type	Description	Possible Values
Host Type	string	The type of copy services on which the method was invoked.	source
			target

Table 99 describes the error properties.

Table 99 - Invalid SAP for Method Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Reference to the Computer System host which is of the wrong type.)	Existence is required
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.3.0.26   Message: Resource Not Available

Owning Entity: SNIA

Message ID: DRM26

Message Format String: <Resource Needed>

Table 100 describes the message arguments.

Table 100 - Resource Not Available Message Arguments

Message Argument	Data Type	Description	Possible Values
Resource Needed	string		No replication log available.
			Special replica pool required.

Table 101 describes the error properties.

Table 101 - Resource Not Available Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Nothing to reference.)	Existence is discouraged
PERCEIVED_SEVE RITY	2 (Low)	Existence is required

9.3.3.0.27   Message: Resource Limit Exceeded

Owning Entity: SNIA

Message ID: DRM27

Message Format String: <Reason>

Table 102 describes the message arguments.

Table 102 - Resource Limit Exceeded Message Arguments

Message Argument	Data Type	Description	Possible Values
Reason	string	The reasons for the lack of resources for copy services operation.	Insufficient pool space.
			Maximum replication depth exceeded.
			Maxium replicas exceeded for source element.

Table 103 describes the error properties.

Table 103 - Resource Limit Exceeded Error Properties

Property	Value	Description
CIMSTATUSCODE	1 (CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	4 (Software Error)	Existence is required
ERROR_SOURCE	(Nothing to reference.)	Existence is discouraged
PERCEIVED_SEVERITY	2 (Low)	Existence is required

9.3.4   Fabric Messages

9.3.4.0.1   Message: Zone Database Changed
Owning Entity: SNIA
Message ID: FC1
Message Format String: Zone database changed for <Fabric Identity Type> <WWN>

Table 104 describes the message arguments.

Table 104: Zone Database Changed Message Arguments

Message Argument	Data Type	Description	Possible Values
Fabric Identity Type	string	Defines the type of fabric entity names by the following WWN.	fabric

Table 104: Zone Database Changed Message Arguments

Message Argument	Data Type	Description	Possible Values
			switch
WWN	string	World Wide name identifier. The required form of the WWN is defined by this regular expression, "^[0123456789ABCDEF]{16}$"	

Table 105 describes the alerts that are associated with this message.

Table 105 - Zone Database Changed Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		A reference to the switch or fabric which is named by the WWN.
ALERT_TYPE	Y		Environmental Alert
PERCEIVED_SEVERITY	Y		Informational

9.3.4.0.2   Message: ZoneSet Activated

Owning Entity: SNIA

Message ID: FC2

Message Format String: ZoneSet <ZoneSet Name> was activated for fabric <WWN>

Table 106 describes the message arguments.

Table 106 - ZoneSet Activated Message Arguments

Message Argument	Data Type	Description	Possible Values
ZoneSet Name	string	CIM_ZoneSet.ElementName attribute	
WWN	string	World Wide name identifier. The required form of the WWN is defined by this regular expression, "^[0123456789ABCDEF]{16}$"	

Table 107 describes the alerts that are associated with this message.

Table 107 - ZoneSet Activated Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		A reference to the fabric which is named by the WWN.
ALERT_TYPE	Y		Environmental Error
PERCEIVED_SEVERITY	Y		Minor

9.3.4.0.3   Message: Session Locked

Owning Entity: SNIA

Message ID: FC3

Message Format String: Operation blocked by session lock.

Table 108 describes the error properties.

Table 108 - Session Locked Error Properties

Property	Value	Description
CIMSTATUSCODE	(CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	(Software Error)	Existence is required
ERROR_SOURCE	()	Existence is required
PERCEIVED_SEVE RITY	(Degraded/Warning)	Existence is required

9.3.4.0.4   Message: Session Aborted

Owning Entity: SNIA

Message ID: FC4

Message Format String: Operation by another client caused the session to be aborted.

Table 109 describes the error properties.

Table 109 - Session Aborted Error Properties

Property	Value	Description
CIMSTATUSCODE	(CIM_ERR_FAILED)	Existence is required
ERROR_TYPE	(Software Error)	Existence is required
ERROR_SOURCE	()	Existence is required
PERCEIVED_SEVE RITY	(Degraded/Warning)	Existence is required

9.3.4.0.5   Message: Switch Status Changed

Owning Entity: SNIA

Message ID: FC5

Message Format String: Switch <Switch Unique Identifier> in Fabric <Fabric Name> status changed to <Switch OperationalStatus>

Table 110 describes the message arguments.

Table 110 - Switch Status Changed Message Arguments

Message Argument	Data Type	Description	Possible Values
Switch Unique Identifier	string	The Switch Name (WWN).	
Fabric Name	string	Fabric name. Typically the principal switch's WWN	
Switch OperationalStatus	string	Switch Status	

Table 111 describes the alerts that are associated with this message.

Table 111 - Switch Status Changed Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELEMENT	Y		The Fabric
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		Major

9.3.4.0.6   Message: Fabric Merge/Segmentation

Owning Entity: SNIA

Message ID: FC6

Message Format String: <Fabric Name> has detected a <Fabric Change>

Table 38 describes the message arguments.

Table 112 - Fabric Merge/Segmentation Message Arguments

Message Argument	Data Type	Description	Possible Values
Fabric Name	string	Fabric name. Typically the principal switch's WWN	
Fabric Change	string	A value of merge or segmentation	merge
			segmentation

Table 113 describes the alerts that are associated with this message.

Table 113 - Fabric Merge/Segmentation Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		The SAN
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		Minor

9.3.4.0.7   Message: Switch Added/Removed

Owning Entity: SNIA

Message ID: FC7

Message Format String: The fabric <Fabric Name> has detected switch <Switch Unique Identifier> has been <Fabric Change Type>

Table 40 describes the message arguments.

Table 114 - Switch Added/Removed Message Arguments

Message Argument	Data Type	Description	Possible Values
Fabric Name	string	Fabric name. Typically the principal switch's WWN	
Switch Unique Identifier	string	The Switch Name (WWN).	
Fabric Change Type	string	A value of added or removed	added
			removed

Table 115 describes the alerts that are associated with this message.

Table 115 - Switch Added/Removed Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		The Fabric
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		Minor

9.3.4.0.8   Message: Fabric Added/Removed

Owning Entity: SNIA

Message ID: FC8

Message Format String: Fabric <Fabric Identifier> was <Change Type>

Table 116 describes the message arguments.

Table 116 - Fabric Added/Removed Message Arguments

Message Argument	Data Type	Description	Possible Values
Fabric Identifier	string	The Fabric Identity	
Change Type	string	A value of Added or Removed	added
			removed

Table 117 describes the alerts that are associated with this message.

Table 117 - Fabric Added/Removed Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		The SAN
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		Major

9.3.4.0.9   Message: Security Policy change

Owning Entity: SNIA

Message ID: FC9

Message Format String: Fabric Security Policy changed in <Fabric Name>

Table 44 describes the message arguments.

Table 118 - Security Policy change Message Arguments

Message Argument	Data Type	Description	Possible Values
Fabric Name	string	Fabric name. Typically the principal switch's WWN	

Table 119 describes the alerts that are associated with this message.

Table 119 - Security Policy change Alert Information

Name	Req	Value	Description
ALERTING_MANAGED_ELE MENT	Y		The Fabric
ALERT_TYPE	Y		Device Alert
PERCEIVED_SEVERITY	Y		Minor

EXPERIMENTAL

EXPERIMENTAL

Clause 10: Service Discovery

10.1   Objectives

Service discovery in the context of SMI-S refers to the discovery of dedicated SMI-S servers, general purpose SMI-S servers, and directory servers, and the functions they offer in an SMI-S managed environment. The specific objectives to be addressed by the discovery architecture are:

1)   Provide a mechanism that allows SMI-S clients to discover the SMI-S constituents in a storage network environment so that they may communicate with these constituents using CIM Operations over HTTP protocol. This includes:

   · Finding the address for the SMI-S constituent;

   · Finding the capabilities of the server, including communications capabilities, security capabilities, CIM operational capabilities and the functional capabilities (CQL, Batch operations support, etc.);

2)   Provide a mechanism that is efficient in the amount of information exchanged with minimal exchanges to acquire the information;

3)   Provide a mechanism that accurately defines the services in the network, independent of whether or not those services are currently available;

4)   Provide a mechanism that provides information on namespaces provided and the CIM Schema supported;

5)   Provide a mechanism that allows SMI-S clients the profile(s) supported by agents and object managers;

6)   Provide a mechanism that scales to enterprise environments;

7)   Utilize existing standard mechanisms to effect the SMI-S service discovery to enable rapid deployment;

8)   Provide a mechanism that allows SMI-S clients to determine the level of (SMI-S) support provided by the constituents (e.g., R1, R2, etc.)

10.2   Overview

SMI-S Release 1 uses the Service Location Protocol Version 2 (SLPv2), as defined by IETF RFC 2608, for its *basic* discovery mechanism. SLPv2 is used to locate constituents (agents, object managers, etc.), but complete discovery of all the services offered involves traversing the interoperability model for the SMI-S profile supported. This clause of the SMI-S specification deals primarily with the information discovered using SLPv2. There are references to information discovered by traversing the interoperability model, but details on this are provided in 11.3.

   Note: SLPv1 is not supported in SMI-S as discovery mechanism. SMI-S requires capabilities that were introduced in SLPv2 in order to support the discovery of SMI-S agents and object managers. SLPv2 defines discovery protocols among three constituents:

User Agent (UA): A process that attempts to establish contact with one or more services. A User Agent retrieves service information from Service Agents or Directory Agents. In SMI-S, a "user agent" would be part of an SMI-S Client.

Service Agent (SA): A process working on behalf of one or more services to advertise the services. In SMI-S, a "service agent" would be supported by SMI-S dedicated or general purpose servers.

Directory Agent (DA): A process that caches SLP service advertisements registered by Service Agents and forwards the service advertisements to User Agents on demand. In SMI-S, the SLP "Directory agent" is defined as the main function of the "directory server" role in the SMI-S Reference Model. SMI-S allows multiple Directory

Agents to be used for purposes including load sharing and availability. These Directory Agents may have the same scope, as allowed by SLPv2.

SLPv2 provides a framework for client applications, represented by User Agents, to find and utilize services, represented by Service Agents. The Directory Agents represent an optional part that enhances the performance and scalability of the protocol by acting as a cache for all services that have been advertised. Directory Agents also reduce the load on Service Agents, making simpler implementations of Service Agents possible. User Agents can then query the Directory Agents for services. Service Agents register with Directory Agents and are required to re-register as the registrations expire. If no Directory Agents are present, User Agents may request service information directly from the Service Agents.

Using SLPv2, a client can discover SMI-S servers and SLPv2 Directory Agents in the storage network. In the case of SMI-S servers, the basic information discovered is the profiles supported and the URL of the service. Details on the specific services provided with the profile are then found by traversing the service structure modeled for the profile.

Using SLPv2, a "service agent" advertises its services. These advertisements have an expiration time period. To avoid getting an advertisement deleted, a service agent shall reregister before the time period expires. SMI-S servers may deregister as part of a graceful shutdown.

A service advertisement consists of file components:

- Service type name – describes the general type of service being advertised (ex. Printing, faxing, etc.). The working assumption is that DMTF wants "WBEM Servers" advertised with the service type WBEM. This is used by SMI-S servers (both dedicated and general purpose servers).;

- Attributes – The collection of attributes describes the particular instance of the service in more detail. For SMI-S, these would be the attributes defined by the service type template for WBEM. The attributes are defined in 10.5.2;

- Service Access point – the service access point defines the point of connection that the software client of the UA uses to connect to the service over the network.;

- Scopes – These are administrative groupings of services. The default value ("default") should be used for SMI-S servers. Other scopes may be defined by the customer, but care must be taken when this is done. The administrator shall do this correctly or SMI-S servers will not be visible. All the SMI-S recipes assume that DEFAULT is set for scopes;

- Language – Services advertisements contain human readable strings. These are provided in English, but may also be in other languages.

IMPLEMENTED

SLPv2 provides for authentication of service URLs and service attributes. This provides user agents (UAs) and directory agents (DAs) with assurances of the integrity of service URLs and attributes included in SLP messages. The only systems which can generate digital signatures are those which have been configured by administrators in advance. Agents that verify signed data may assume it is trustworthy inasmuch as administrators have assured trustworthiness through the cryptographic keying of SAs and DAs. The SLPv2 security model assumes that service information is public, and therefore does not require confidentiality.

Section 2.5 of RFC 3723, *Securing Block Storage Protocols over IP*, states that the SA advertisements as well as UA requests and/or responses are vulnerable to these security threats:

1) An attacker could insert or alter service agent (SA) advertisements or responses to a UA requests in order to masquerade as the real peer or launch a denial of service attack.

2)   An attacker could gain knowledge about an SA or a UA through sniffing, and launch an attack against the peer.

3)   An attacker could spoof DA advertisements and thereby cause UAs and SAs to use a rogue DA.

Section 2.5 of RFC 3723 also outlines the capabilities required to address these threats, but notes that SLP (as defined in RFC 2608) does not satisfy these security requirements. SLPv2 only provides end-to-end authentication (i.e., does not support confidentiality), but with this authentication, there is no way to authenticate zero result responses. Thus an attacker could mount a denial of service attack by sending UAs a zero results Service Reply (SrvRply) or Attribute Reply (AttrRply) with a source address corresponding to a legitimate DA advertisement.

The RFC 3723 mitigation strategies include reliance on digital signatures for authentication of service URLs and attributes as well as IPsec. For SMI-S environments that require security in conjunction with the use of SLPv2, the major RFC 3723 recommendations are not necessary as long as the SLP messages are not fully trusted and SSL or TLS with server certificates are used. Additional security guidance is provided in the sections associated with UAs and SAs.

IMPLEMENTED

10.3   SLP Messages

SLP v2 divides the base set of SLP messages into required and optional subsets.

Note: SLP v2 also includes a new feature, an extension format. Extension messages are attached to base messages. SMI-S does not use extensions. The discussion of messages introduces terms that define the SLP services:

・ Attribute Reply (AttrRply): A reply to an Attribute Request. (optional)

・ Attribute Request (AttrRqst): A request for attributes of a given type of service or attributes of a given service. (optional)

・ DA Advertisements (DAAdvert): A solicited (unicast) or unsolicited (multicast) advertisement of Directory Agent availability.

・ SA Advertisement (SAAdvert): Information describing a service that consists of the Service Type, Service Access Point, lifetime, and Attributes.

・ Service Acknowledgement (SrvAck): A reply to a SrvReg request.

・ Service Deregister (SrvDereg): A request to deregister a service or some attributes of a service. (optional)

・ Service Register (SrvReg): A request to register a service or some attributes of a service.

・ Service Reply (SrvRply): A reply to a Service Request.

・ Service Request (SrvRqst): A request for a service on the network.

・ Service Type Reply (SrvTypeRply): A reply to a Service Type Request. (optional)

・ Service Type Request (SrvTypeRqst): A request for all types of service on the network. (optional)

Service Agents (SAs) and User Agents (UAs) shall support Service Request, Service Reply, and DAAdvertisement message types. Service Agents shall additionally support Service Registration, SA Advertisement, and Service Acknowledgement message types. The remaining message types may be supported by Service Agents and User

Agents. Directory Agents (DAs) shall support all message types with the exception of SA Advertisement. Table 120 lists each base message type, its abbreviation, function code, and required/optional status.

Table 120 - Message Types

Message Type	Abbreviation	Function Code	Required (R)/ Optional (O)		
			DAs	SAs	UAs
Service Request	SrvRqst	1	R	R	R
Service Reply	SrvRply	2	R	R	R
Service Registration	SrvReg	3	R	R	O
Service Deregistration	SrvDereg	4	R	O	O
Service Acknowledgement	SrvAck	5	R	R	O
Attribute Request	AttrRqst	6	R	R	R
Attribute Reply	AttrRply	7	R	R	R
DA Advertisement	DAAdvert	8	R	R	R
Service Type Request	SrvTypeRqst	9	R	O	O
Service Type Reply	SrvTypeRply	10	R	O	O
SA Advertisement	SAAdvert	11	N/A	R	O

Note: The requirements in this table extend the requirements defined for SLP V2. SMI-S adds additional requirements for AttrRqst and AttrRply beyond those defined by the RFC.

10.4   Scopes

SLPv2 defines a scope as follows:

Scope: A set of services, typically making up a logical administrative group.

Scopes are sets of service instances. The primary use of Scopes is to provide the ability to create administrative groupings of services. A set of services may be assigned a scope by network administrators. A User Agent (UA) seeking services is configured to use one or more scopes. The UA only discovers those services that have been configured for it to use. By configuring UAs and Service Agents with scopes, administrators may make services available. Scopes strings are case insensitive. The default SCOPE string is "DEFAULT".

SMI-S does not dictate how Scopes are set. That is, scopes can be set by customers to match their needs. However, SMI-S requires that SMI-S servers use the "default" scope as a means of making SMI-S advertisements visible to SMI-S clients.

To be compliant with SMI-S, User Agents (SMI-S clients) and Service Agents (SMI-S servers) shall not require scope settings that interfere with administrative use of scopes. Specifically, this means:

•   SMI-S clients and servers shall allow an administrator to set scopes to define what is to be searched, and,

- SMI-S clients and servers shall allow an administrator to configure scopes, including turning off the "default" scope.

10.5   Services Definition

Services definition uses these terms defined in SLPv2:

- Service Type Template: A formalized, computer-readable description of a Service Type. The template defines the format of the service URL and attributes supported by the service type.

- Service URL: A Uniform Resource Locator for a service containing the service type name, network family, Service Access Point, and any other information needed to contact the service.

Services are defined by two components: the Service URL and the Service Type Template. The Service URL defines an access point for the service and identifies a unique resource in the network. Service URLs may be either existing generic URLs or URLs from the service: URL scheme.

The second component in a Service definition is a Service Type Template. Service Type Templates define the attributes associated with a service. These attributes, through inclusion in registrations and queries, allow clients to differentiate between similar services.

SMI-S servers use a Service Type Template defined by DMTF for advertising "WBEM Servers" (e.g., CIMOMs). The template name for WBEM Servers is "WBEM".

10.5.1   Service Type

Service Type: The class of a network service represented by a unique string (for example a namespace assigned by IANA).

The service type describes a class of services that share the same attributes (e.g., the service printer or the service "WBEM"). DMTF is considering an SLP-based discovery mechanism that locates "WBEM" (e.g., CIMOMs). The SMI-S design builds on the DMTF proposal.

The basic function of SLP discovery is the identification of the service offered by a constituent. In the case of SMI-S, the service type advertised by all constituents is "WBEM." This follows a DMTF proposal for advertising WBEM Servers. The only exception to this is the Directory Server, which advertises itself as a "directory-agent." That is, SMI-S uses a standard SLP directory service. SMI-S does not require a unique SMI-S directory server.

For other roles (SMI-S servers) the role advertises its services as a WBEM services (e.g., "WBEM").

10.5.2   Service Attributes

Attributes: A collection of tags and values describing the characteristics of a service.

SMI-S servers shall advertise a standard set of attributes:

- Service-hi-name – This is the name of the service for use in human interfaces.

- Service-hi-description – This is a description of the CIM service that is suitable for use in human interfaces.

- Service-id – A unique id for the CIM Server that is providing the service.

- Service-location-tcp – This is a list of TCP addresses that can be used to reach the service. NOTE: This need only be one (for CIM-XML). But it could hold others (for other communications protocols).

- CommunicationMechanism – "cim-xml" (at least). The SMI-S server could support others, but "cim-xml" is mandatory for SMI-S servers.

- OtherCommunicationMechanismDescription – used only if "other" is also specified for CommunicationMechanism.

- InteropSchemaNamespace – The Namespace within the SMI-S server where the CIM Interop Schema can be accessed. Each namespace provided shall contain the complete information and if multiple namespaces are provided they shall contain the same information. Even though multiple InteropSchemaNamespaces may be provided, an SMI-S client may rely on the first namespace as the definitive namespace for accessing the Interop Schema (including the class instances of the Server Profile).

- ProtocolVersion – The Version of the cim-xml protocol if this is the defined. This is mandatory for SMI-S servers.

- FunctionalProfilesSupported: Permissible values are "Unknown", "Other", "Basic Read", "Basic Write", "Schema Manipulation", "Instance Manipulation", "Association Traversal", "Query Execution", "Qualifier Declaration", "Indications". This defines the CIM Operation Profiles supported by the SMI-S server. Can return multiple values.

- FunctionalProfileDescriptions - If the "other" value is used in the FunctionalProfilesSupported attribute, this shall be populated. If provided it shall be derived from the CommunicationMechanism.FunctionalProfileDescriptions property. Use of this attribute is not specified by SMI-S.

- MultipleOperationsSupported – A Boolean that defines whether the SMI-S server supports batch operations.

- AuthenticationMechanismsSupported – Permissible values are "Unknown", "None", "Other", "Basic", "Digest". Defines the authentication mechanism supported by the SMI-S server. Can return multiple values.

- AuthenticationMechanismDescriptions - Defines other Authentication mechanism supported by the SMI-S server. The value shall be supplied if the "Other" value is set in the AuthenticationMechanismSupported attribute. This attribute is optional. It is to be provided only when the AuthenticationMechanismSupported attribute is "other".

- Namespace - Namespace(s) supported on the SMI-S server. This attribute may have multiple values (one for each namespace defined in the SMI-S server), and is literal (L) because the namespace names may not be translated into other languages.

- Classinfo - The values are taken from the interop schema Namespace.classinfo property. The values represent the classinfo (CIM Schema version, etc.) for the namespaces defined in the corresponding namespace listed in the namespace attribute. Each entry in this attribute shall correspond to the namespace defined in the same position of the namespace attribute. There shall be one entry in this attribute for each entry in the namespace attribute.

- RegisteredProfilesSupported – The SMI-S profile(s) supported by the server, prefixed by "SNIA" (at least). An SMI-S server may also support other RegisteredProfiles, but it shall support at least one "SNIA" profile. In addition, this attributed can also be used to advertise subprofiles, when subprofiles are to be advertised. The RegisteredProfilesSupported is an array. Each entry includes a RegisteredOrganization (i.e., SNIA), a Profile name and an optional subprofile name. Each name is separated by a colon.

Note that a single SMI-S server can support multiple profiles. As a result, the profile attribute is an array of values.

Additional attributes, such as specific profile services supported, model subprofiles supported and the SMI-S release level are not discovered via SLP. They would be found by traversing the model presented by the SMI-S server.

10.6   User Agents (UA)

A User Agent is a Client process working on the user's behalf to establish contact with some service. A User Agent retrieves service information from Service Agents (10.7) or Directory Agents (10.8). Further description of a Client and its role may be found in 11.2, "SMI-S Client".

The only required feature of a User Agent is that it can issue SrvRqsts and interpret DAAdverts, SAAdverts and SrvRply messages. If Directory Agents exist, User Agents shall issue requests as Directory Agents are discovered.

An SMI-S Client should act as an SLP user agent (UA) using the query functions of SLP V2 to determine location and other attributes of the "WBEM" SLP Service Type Template defined in 10.11, "'Standard WBEM' Service Type Templates".

The basic search methodology for SMI-S clients is to search for directory agents and service agents within their scope. If all SMI-S servers are supported by a directory agent, then the search yields nothing but directory agents. The client can then obtain a list of services (and their URLs) for management of the SMI-S servers.

If any Service agents are not covered by a directory agent (i.e., are not within its scope), then the client obtains service replies from those service agents.

An client would typically search for all service types available in their scope(s). This returns a list of service types available in the network. However, an SMI-S client can be assumed to be searching for "WBEM" service types. If a client only manages selected devices (e.g., switches or arrays), the SMI-S client can issue a request for the specific services by using predicates on the "RegisteredProfilesSupported" attribute.

IMPLEMENTED

When a SMI-S client uses SLPv2 and security is an issue, the following should be considered:

- SSL and TLS should be used with a certificate-based cipher suite along with a certificate installed on each SMI-S server (SA) for communications with discovered SAs (SMI-S servers).

- SLPv2 Service Agents (SA) and Directory Agents (DA) may advertise (SAAdverts and DAAdverts, respectively) their presence on the network, using multicast; however, SMI-S clients should treat these advertisements as advisory (i.e., identity shall be verified as described in 10.7 and 10.8).

- SMI-S clients should maintain and use a negative authentication cache to avoid repeatedly contacting an SMI-S server that fails to authenticate as part of the SSL or TLS handshake.

IMPLEMENTED

10.7   Service Agents (SAs)

A Service Agent supports an SMI-S server process working on behalf of one or more services to advertise the services.

See Clause 11: SMI-S Roles for further description of SMI-S servers.

Service Agents shall accept multicast service requests and unicast service requests. SAs may accept other requests (Attribute and Service Type Requests). An SA shall reply to appropriate SrvRqsts with SrvRply or SAAdvert messages. The SA shall also register with all DAs as they are discovered.

To provide for SMI-S Client discovery of SMI-S servers, a CIM Server shall act as a Service agent (SA) for the IETF Service Level Protocol (SLP) V2 as defined in IETF RFC 2608. The service shall correspond to V2 of SLP (IETF RFC 2608 and 2609) and shall use the Service Templates defined in 10.11 of this specification for advertisements. An SMI-S server acting as an SA shall provide a separate SLP advertisement for each address/port that the CIM Server advertises.

IMPLEMENTED

When a SMI-S server uses SLPv2 and security is an issue, the following should be considered:

- SMI-S servers should accept SSL and TLS unicast connections from SMI-S clients as well as selecting a certificate-based cipher suite.

- SMI-S servers that advertise their existence as SLPv2 SAs (SAAdverts) should minimize leakage of information, by minimizing the information that is contained in the multicast advertisements.

- SMI-S servers, functioning as SAs, should register with all discovered DAs, which advertise any of its configured scopes and establish connections with these DAs over unicast.

- When SMI-S servers are also functioning as clients (e.g., cascading), they should follow the security guidance provided in 10.6 User Agents (UA).

IMPLEMENTED

10.8   Directory Agents (DAs)

SMI-S supports existing SLPv2 Directory Agents (without modification). That is, SMI-S makes no assumptions on Directory Agents that are not made by SLPv2. Note that this cannot quite be said for User Agents, which are looking for SMI-S specific services, or Service Agents, which are advertising SMI-S specific services.

10.9   Service Agent Server (SA Server)

10.9.1   General Information

The reserved listening port for SLP is 427, the destination port for all SLP messages. Service Agents (SAs) are required to listen for both unicast and multicast requests. A Directory Agent (DA) shall listen for unicast request and specific multicast DA discovery service requests. SAs and User Agents (UAs) that perform passive DA discovery shall listen for multicast DA Advertisements (DAAdverts).

TCP/IP requires that a single server process per network interface control all incoming messages to a port. That requirement necessitates a mechanism to share the SLP port (427).

Sharing the SLP port (427) is accomplished with a Service Agent Server (SA Server) process that 'owns' the port on behalf of all SAs, UAs and optional DA that are listening for SLP messages. The SA Server listens for incoming messages that request advertisement information and either answer each request or forward it to the appropriate SA. The SA Server also performs passive DA discovery and distribute the DA addresses and scopes to the SAs and UAs that it serves.

A SA Server may also function as a DA if the SA Server is implemented so that it answers requests for advertisement information rather than forwarding each request to the appropriate SA. The combined DA/SA Server is acting as an intermediary between a SA that registered an advertisement and a UA requesting information about the advertisement.

10.9.2   SA Server (SAS) Implementation

IETF RFC 2614 describes APIs for both the C and Java languages. Both APIs are designed for standardized access to the Service Location Protocol (SLP).

The goals of the C API are:

- Directly reflect the structure of SLP messages in API calls and return types as character buffers and other simple data structures.

- Simplify memory management to reduce API client requirements.

- Provide API coverage of just the SLP protocol operations to reduce complexity.

- Allow incremental and asynchronous access to return values, so small memory implementations are possible.

- Support multithreaded library calls on platforms where thread packages are available.

The Java API goals are:

- Provide complete coverage of all protocol features, including service type templates, through a programmatic interface.

- Encourage modularity so that implementations can omit parts of the protocol that are not needed.

- In conformance with Java's object-oriented nature, reflect the important SLP entities as objects and make the API itself object-oriented.

- Use flexible collection data types consistently in the API to simplify construction of parameters and analysis of results.

- Designed for small code size to help reduce download time in networked computers.

10.9.3 SA Server (SAS) Clients

10.9.3.1 Description

An SAS Client is a Service Agent (SA), User Agent (UA), or Directory Agent (DA) that is associated with a SA Server. The SA Server listens on the SLP port (427) and appropriately handle all incoming messages for each SAS Client. A DA acting as a SAS Client is separately configured on the same host as the SA Server.

10.9.3.2 SAS Client Requests – SA Server Responses

A SA Server responds when appropriate, to incoming unicast and multicast messages from SAS Clients. The SA Server may answer with the appropriate advertisement, if available, or forward the request on to the appropriate SAS Client. If the SA Server is also functioning as a DA, it discards a multicast SrvRqst of "service:directory-agent" that has either a missing scope list or the scope list does not contain a scope the Service Agent Server/DA is configured with.

10.9.4 SA Server Configuration

10.9.4.1 Overview

SA Servers may be configured via an individual SLP configuration file, programmatically, or a combination of the two. DHCP may also be used obtain the scope list for a SA Server. Figure 19 illustrates the various means of configuring a SA Server.

10.9.4.2 SLP Configuration File

If a SA Server is also functioning as a DA, the DA configuration properties shown in Table 121 shall be set:

Table 121 - Required Configuration Properties for SA as DA

Keyword	Data Type	Value
net.slp.isDA	boolean	true
net.slp.DAAttributes	string	(SA-Server=true)

The DA attribute/value pair of "SA-Server=true" allows a query to be used when a SA Server/DA needs to be identified. In addition, when the SA Server/DA responds to a SrvRqst message with a DAAdvert message, the DA attribute/value pair is included.

The remaining DA configuration property, net.slp.DAHeartBeat, with a default of 10,800 seconds, may be set as appropriate. If a SA Server is not functioning as a DA, the SA configuration property in Table 122 shall be set:

Table 122 - Required Configuration Properties for SA

Keyword	Data Type	Value
net.slp.SAAttributes	string	(SA-Server=true)

10.9.4.3 Programmatic Configuration

Both the C and Java language API's provide access to SLP properties contained in the SLP configuration file. The actual SLP configuration file is not accessed or modified via the interfaces. Once the file is loaded into memory at the start of execution, the configuration property accessors work on the in-memory representation.

The C language API provides the `SLPGetProperty`() and `SLPSetProperty`() functions. The `SLPGetProperty`() function allows read access to the SLP configuration properties while the `SLPSetProperty`() function allows modification of the configuration properties.

The SLPSetProperty() function has the following prototype:

```
void SLPSetProperty(const char *pcName, const char *pcValue);
```

The SLPSetProperty() function takes two string parameters: pcName and pcValue. The pcName parameter contains the property name and pcValue contains the property value. The following example uses the SLPSetProperty() function to configure a SA Server that is not functioning as a DA:

```
void setSAAttributes() {
char value[80]; /* A buffer for storing the attribute string. */
value = "SA Server=true";
SLPSetProperty("net.slp.SAAttributes", value);
}
```

10.9.4.4 DHCP Configuration

If the Service Agent Server is also functioning as a DA, its scope list may be obtained via DHCP. Scopes discovered via DHCP take precedence over the net.slp.useScopes property in the SLP configuration file.

10.9.4.5 Scope

A Service Agent Server is configured with a minimum scope of DEFAULT. If a Service Agent Server is not functioning as a DA, DEFAULT is the only scope configured. If a Service Agent Server is functioning as a DA, it

may have additional scopes configured. Use of the DEFAULT scope enables the associated SAS Clients (UAs, SAs and DA) to actively discover the Service Agent Server using a well-known value for scope.
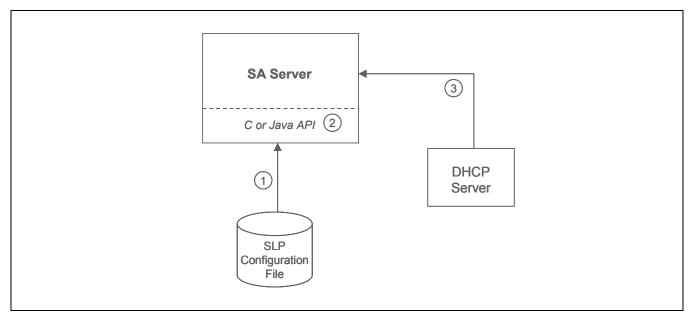

Figure 19 - SA Server Configuration

1) The SA Server may obtain specific configuration values via an individual SLP Configuration file.

2) The C or Java API provides programmatic access to the configuration file properties.

3) The SA Server may obtain its scope values from a DHCP Server.

10.9.5 SA Server Discovery

"Discovery" of a SA Server by its SAS Clients is accomplished by successfully establishing the required communication link between the two entities. There is no need for active or passive discovery as described by SLP since both the SA Server and SAS Clients reside on the same host system.

10.9.6 SAS Client Registration

Service Agents (SAs) that are SAS Clients register and deregister with the local SA Server using the SrvReg/ SrvDereg messages. The SA Server responds with a Service Acknowledgement (SrvAck) message. The SA Server store a service advertisement until either its lifetime expires or a SrvDereg message is received.

If the SA Server is also functioning as a DA, the DA registration requirement is also met. The SA server also forwards any SA registration to other DAs that have the same scope as the SA.

10.10 Configurations

There are three network configurations (10.10.1, 10.10.2, 10.10.3) showing SMI-S clients and servers. The routing of SLP's multicast messages effect the SMI-S discovery process. SMI-S clients and servers shall be able to be configured to work in these environments.

10.10.1 Multicast Configurations

This is the simplest environment and is shown in Figure 20. This network allows multicast messages to be delivered to all the components of a SMI-S management system. As defined in IETF RFC 2608 - 8.1, the client

uses multicast SLP messages to contact the SLP Service Agent (SA) associated with each SMI-S server. Then, each SA sends replies directly back to the client.

Because of the possible size of the reply, servers shall support TCP/IP (as well as UDP) to send the reply. The server shall also support the SLP overflow bit to tell the client large TCP/IP messages shall be used. When a client sees the overflow bit in a UDP response, it should retry using a TCP request.
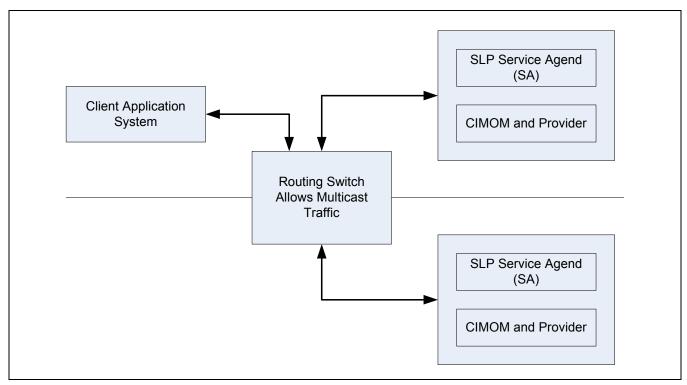

Figure 20 - Multicast Configuration

10.10.2 No Multicast configuration

In this configuration, shown in Figure 21, the network doesn't allow the use of multicast messages. All communication shall use TCP/IP point to point connections. First, a SLP directory agent should be used. Each SA shall be configurable by the user. The user will configure the SA by setting the address of the SLP directory agent (DA). At startup each SA shall use a temporary registration to tell the DA its SLP information (IETF RFC 2608 - 8.3). The SAs shall renew the registration before it expires (IETF RFC 2608 - 8.3). The registration timeout should be about 5-10 min.

The client shall also be configurable by the user. The user will configure the client by setting the DA address. The client will use this address to send SLP messages to the DA (IETF RFC 2608 - 8.1). The DA will satisfy the requests using information provided by the SAs.
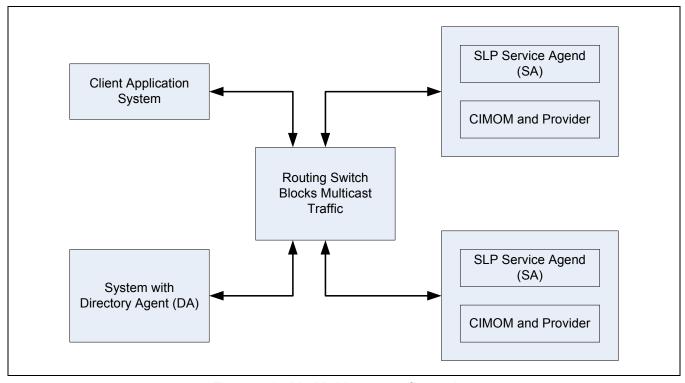

Figure 21 - No Multicast configuration

10.10.3 Multicast Islands

Networks that allow for multicast messages to reach parts of the system, require the use of both techniques described. The client should use the multicast process and the no multicast method. It should be able to combine the information found each way into a single set of discovery information. The SAs shall support both methods at the same time as shown in Figure 22.
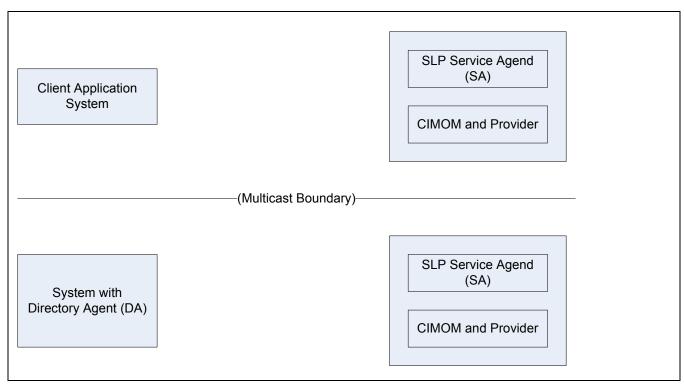
Figure 22 - Multicast Islands

10.11 'Standard WBEM' Service Type Templates

Note: For each description in the template that states the value shall be the ClassName.PropertyName value, the format/rules for these values are defined in the Interop Model of the CIM Schema and in the "Server Profile" section of this specification. This SLP Template requires a minimum Schema version of 2.7 to support the required values. Some of the optional values require CIM Schema version 2.8.

```
Name of submitter: "DMTF" <technical@dmtf.org>

Language of service template: en

Security Considerations:

Information about the specific CIM Server implementation or the
Operating System platform may be deemed a security risk in certain
environments. Therefore these attributes are optional but
recommended.

Template Text:

------------------------template begins here----------------------
template-type=wbem

template-version=1.0
```

```
template-description=
    This template describes the attributes used for advertising
    WBEM Servers.


template-url-syntax=string
#The template-url-syntax MUST be the wbem URI encoding of
#the location of one service access point offered by the WBEM Server
#over TCP transport. This attribute must provide sufficient addressing
#information so that the WBEM Server can be addressed directly using
#the url.


service-hi-name=string O
# This string is used as a name of the CIM service for human
# interfaces. This attribute MUST be the
# CIM_ObjectManager.ElementName property value.


service-hi-description=string O
# This string is used as a description of the CIM service for
# human interfaces.This attribute MUST be the
# CIM_ObjectManager.Description property value.


service-id=string L
# The ID of this WBEM Server. The value MUST be the
# CIM_ObjectManager.Name property value.


CommunicationMechanism=string L
# The communication mechanism (protocol) used by the CIM Object Manager for
# this service-location-tcp defined in this advertisement. This information
# MUST be the CIM_ObjectManagerCommunicationMechanism.CommunicationMechanism
# property value.
# CIM-XML is defined in the CIM Operations over HTTP specification which can
# be found at http://dmtf.org/
"Unknown", "Other", "cim-xml"


OtherCommunicationMechanismDescription = String L O
# The other communication mechanism defined for the CIM Server in the case
# the "Other" value is set in the CommunicationMechanism string.
# This attribute MUST be the
                        CIM_ObjectManagerCommunicationMechanism.OtherCommunicat
                        ionMechanism
# property value. This attribute is optional because it is only required if the
# "other" value is set in CommunicationMechansim. The value returned is
# a free-form string.


InteropSchemaNamespace=string L M
# Namespace within the target WBEM Server where the CIM Interop Schema can be
# accessed. Multiple namespaces may be provided. Each namespace provided
```

```
# MUST contain the same information.


ProtocolVersion=String O L
# The version of the protocol. It MUST be the
# CIM_ObjectManagerCommunicationMechanism.Version property value.


FunctionalProfilesSupported=string L M
# ProfilesSupported defines the CIM Operation profiles supported by the
# CIM Object Manager. This attribute MUST be the
# CIM_ObjectManagerCommunicationMechansim.FunctionalProfilesSupported
# property value.
"Unknown", "Other", "Basic Read", "Basic Write",
"Schema Manipulation", "Instance Manipulation",
"Association Traversal", "Query Execution",
"Qualifier Declaration", "Indications"


FunctionalProfileDescriptions=string L O M
# Other profile description if the "other" value is set in the ProfilesSupported
# attribute.  This attribute is optional because it is returned only if the "other"
# value is set in the ProfilesSupported attribute. If provided it MUST
# be equal to the
                          CIM_ObjectManagerCommunicationMechanism.FunctionalProfi
                          leDescriptions
# property value.


MultipleOperationsSupported=Boolean
# Defines whether the CIM Object Manager supports batch operations.
# This attribute MUST be the
# CIM_ObjectManagerCommunicationMechanism.MultipleOperationsSupported
# property value.


AuthenticationMechanismsSupported=String L M
# Defines the authentication mechanism supported by the CIM Object Manager.
# This attributed MUST be the
# CIM_ObjectManagerCommunicationMechanism.AuthenticationMechanismsSupported
                           property value.
"Unknown", "None", "Other", "Basic", "Digest"


AuthenticationMechansimDescriptions=String L O M
# Defines other Authentication mechanisms supported by the CIM Object Manager
# in the case where the "Other" value is set in any of the
# AuthenticationMechanismSupported attribute values. If provided, this attribute
                           MUST be the
# CIM_ObjectManagerCommunicationMechanism.AuthenticationMechansimDescriptions
# property value.


Namespace=string L M O
# Namespace(s) supported on the CIM Object Manager.
```

```
# This attribute MUST be the
# CIM_Namespace.name property value for each instance of CIM_Namespace
# that exists. This attribute is optional.
# NOTE: This value is literal (L) because
# the namespace names MUST not be translated into other languages.


Classinfo=string M O
# This attributes is optional but if used, the values MUST be the
# CIM_Namespace.classinfo property value.
# The values represent the classinfo (CIM Schema version, etc.) for
# the namespaces defined in the corresponding namespace listed in the
# Namespace attribute. Each entry in this attribute MUST correspond
# to the namespace defined in the same position of the namespace
# attribute. There must be one entry in this attribute for each
# entry in the namespace attribute.



RegisteredProfilesSupported=string L M
# RegisteredProfilesSupported defines the Profiles that
# this WBEM Server has support for. Each entry in this
# attribute MUST be in the form of
# Organization:Profile Name{:Subprofile Name}
#
# examples:
#     DMTF:CIM Server
#     DMTF:CIM Server:Protocol Adapter
#     DMTF:CIM Server:Provider Registration
# The Organization MUST be the
# CIM_RegisteredProfile.RegisteredOrganization property value.
# The Profile Name MUST be the
# CIM_RegisteredProfile.RegisteredName property value.
# The subprofile Name MUST be the
# CIM_RegisteredProfile.RegisteredName property value when it is
# used as a Dependent in the CIM_SubProfileRequiresProfile
# association for the specified Profile Name (used as the antecedent).



-------------------------template ends here-----------------------
```

Clause 11: SMI-S Roles

11.1   Introduction

As shown in Figure 23, the complete reference model shows the roles for the various entities of the management system. Any given host, network device or storage device may implement one or more of these roles as described later in this clause.
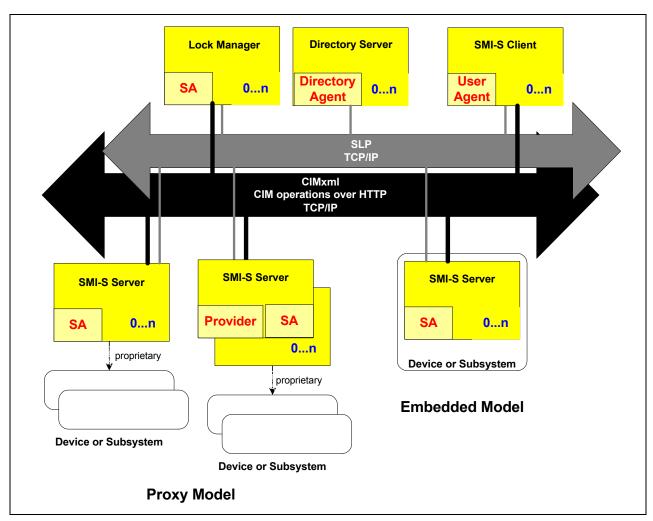

Figure 23 - Complete Reference Model

This profile presents a concise definition of each of these roles and the requirements on implementations of these roles in a management system. For each of these roles, specific functions are required to be implemented in one or more functional areas:

a)   SLP Discovery Functions – the required discovery capabilities that the role performs in the overall management system;

b)   Basic WBEM Operations – the management model operations that the role performs;

c)   Security – the security requirements that the role is expected to satisfy;

d)   Lock Management Operations – the locking operations that the role is expected to perform.

The detail of these responsibilities for each of the roles is described in this profile.

Implementers should be aware that an announced plan for converging web services standards is expected to cause changes to WS-Management, WSDM, and related protocols. SNIA intends to specify the resulting converged protocols for use with SMI-S, and hence use of web services protocols with SMI-S may remain Experimental until stable versions of the converged protocol specifications are available. Implementers are encouraged to experiment with web services protocols for SMI-S in the interim, but should consult the convergence plan to understand the potential protocol changes and possible impacts.

11.2   SMI-S Client

11.2.1   Overview

The SMI-S Client role in the overall management system is performed by software that is capable of performing management operations on the resources under management. This includes monitoring, configuration, and control of the operations on the resources. Typical clients include user interface consoles, complete management frameworks, and higher-level management applications and services such as policy based management systems.

There can be zero or more SMI-S clients in the overall management system. These clients can all coexist simultaneously and can perform independent or overlapping operations in the management system. It is outside the scope of this specification to specify client cooperation with other clients in any way. The semantics of the described management system is that the last successful client operation is valid and persists in the absence of any other client operations (last write wins).

It is expected that development kits for the management system will provide code for the required functions implemented in clients. Consoles, frameworks and management applications can then use this common code in order to comply with this specification. The specification of an API for this client code, and specific language bindings for applications is outside the scope of this specification, but is a candidate for follow-on work.

11.2.2   SLP Functions

The SMI-S Client role is required to implement SLP User Agent (UA) functionality as specified in 10.6, "User Agents (UA)". The Client discovers all SMI-S servers within its configured scope that are required for its operations by querying for service specific attributes that match the criteria for those operations.

11.2.3   WBEM Protocol Functions

The SMI-S Client role shall implement client functionality as specified by the relevant WBEM protocol standard and should implement asynchronous notification functionality as specified by that standard.

11.2.4   Security Considerations

The SMI-S Client role shall implement security as specified in *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* 42.2.2, "HTTP Security".

11.2.5   Lock Management Functions

There are no requirements for locking in this release of the specification.

11.3   Dedicated SMI-S Server

11.3.1   Overview

The intention of the SMI-S server role in a management system is to provide device management support in the absence of any other role. A simple management system could consist of just a SMI-S Client and a SMI-S Server and all management functions can be performed on the underlying resource. This means that a vendor can offer complete management for the resource by shipping a standalone client for the resource and not depend on any other management infrastructure. Although, at the same time, the SMI-S Server can participate in a more complex management environment through the use of the standard mechanisms described here.

- Embedded SMI-S Server – the SMI-S Server functions are incorporated into the resource directly and do not involve separate installation steps to become operational.

- Proxy SMI-S Server – the SMI-S Server is hosted on a system separate from the resource and communicates with the resource via either a standard or proprietary remote protocol. This typically involves an installation operation for the SMI-S Server and configuration for, or independent discovery of, the desired resource.

In order to minimize the footprint on the resource or proxy hosts, the required functions of the SMI-S Server role have purposely been scaled back from those of a typical general purpose CIM Server running on host with more significant resources. These required functions are described in 11.3.2 and 11.3.3.

11.3.2 SLP Functions

The SMI-S Server role is required to implement SLP Service Agent (SA) functionality as specified in 10.7, "Service Agents (SAs)". Optionally, it should implement Service Agent Server functionality or use an existing SA Server if one exists. The SMI-S server shall advertise service-specific attributes that allow the client to locate it based on its profile, as defined in section 10.11, "'Standard WBEM' Service Type Templates".

11.3.3 WBEM Protocol Functions

11.3.3.1 General

The SMI-S Server role shall implement the server functionality as specified by the relevant WBEM Protocol standard.

11.3.3.2 Required Intrinsic Methods

An SMI-S Server is required to implement a set of intrinsic methods as defined for each profile. The intrinsic methods are grouped by "functional profile" as specified in the CIM-XML standard and shown in Table 123.

Table 123 - Functional Profiles

Functional Group	Dependency	Methods
Basic Read	None	GetClass
		EnumerateClasses
		EnumerateClassNames
		GetInstance
		EnumerateInstances
		EnumerateInstanceNames
		GetProperty
Basic Write	Basic Read	SetProperty
Instance Manipulation	Basic Write	CreateInstance
		ModifyInstance
		DeleteInstance
Schema Manipulation	Instance Manipulation	CreateClass
		ModifyClass
		DeleteClass

Table 123 - Functional Profiles

Functional Group	Dependency	Methods
Association Traversal	Basic Read	Associators
		AssociatorNames
		References
		ReferenceNames
Query Execution	Basic Read	ExecQuery
Qualifier Declaration	Schema Manipulation	GetQualifier
		SetQualifier
		DeleteQualifier
		EnumerateQualifiers
Indication	None	

SMI-S Servers shall implement intrinsic methods as specified in the "CIM Server Requirements" section of each Profile specification.

11.3.3.3   Required Model Support

The SMI-S Server shall implement the Server Profile as detailed in *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profile.

11.3.4   Security Considerations

The SMI-S Server role shall implement security as specified in *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 48: Security Profile.

11.3.5   Lock Management Functions

There are no requirements for locking in this release of the specification.

11.4   General Purpose SMI-S Server

11.4.1   Overview

The General Purpose SMI-S Server role in an overall management system is intended to reduce the number of network connections needed by a Client to manage large numbers of resources. It is also envisioned as a convenient place to perform operations across multiple resources, further off-loading these from the Client as well.

In addition, the General Purpose SMI-S Server role can provide a hosting environment for the plug-in instrumentation of host-based resources and management proxies for resources with remote management protocols. These plug-ins are called providers and considered sub roles of the General Purpose SMI-S Server.

A General Purpose SMI-S Server is not required in a management system, but is expected to be deployed at least as a common infrastructure for host-based resources. In any large storage network, there may be several General Purpose SMI-S Servers (as many as one per host). Communication between General Purpose SMI-S Servers may be standardized in the future, but this capability is outside the scope of this specification. General Purpose SMI-S Servers may act as a point of aggregation for multiple SMI-S Profiles as described in Clause 42: Server Profile using existing standard mechanisms as specified here.

As General Purpose SMI-S Servers are expected to be deployed on hosts with more resources and less footprint concerns than other managed resources, the required functions, specified in 11.4.2, 11.4.3, and 11.4.4, are more extensive that of an Dedicated SMI-S Server.

11.4.2 SLP Functions

The General Purpose SMI-S Server role is required to implement SLP Service Agent (SA) functionality as specified in 10.7, "Service Agents (SAs)". The General Purpose SMI-S Server shall advertise service specific attributes that allow the Client to locate it based on the profiles it supports, as defined in 11.4.3.1, "General".

11.4.3 CIM-XML Protocol Functions

11.4.3.1   General

The General Purpose SMI-S Server role shall implement CIM-Server functionality as specified by the CIM-XML standard.

11.4.3.2   Required Intrinsic Methods

The General Purpose SMI-S Server is required to implement the minimum profile as specified in CIM-XML standard. In addition, it shall implement the intrinsic methods needed to support the Profiles that it supports.

11.4.3.3   Required Model Support

The General Purpose SMI-S Server shall implement the Server Profile as detailed in *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profile.

11.4.3.4   Security Considerations

The General Purpose SMI-S Server role shall implement security as specified in*Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profile.

11.4.4 Lock Management Functions

There are no requirements for locking in this release of the specification.

11.4.5 Provider Subrole

11.4.5.1   Overview

A sub-role within a General Purpose SMI-S Server that can be used to provide management support for the resource, especially useful when the resource is host-based (i.e., HBA or Host Software) and the platform provides a CIM Server as part of its operating system.

11.4.5.2   Required Model Support

The Provider shall implement the Provider Subprofile as detailed in the object model shown in *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profile).

11.5   Directory Server

The Directory Server role is used to facilitate Discovery of instances of the various roles in a management system, but may also be used by management systems to store common configurations, user credentials and management policies. Functions outside of Discovery are outside the scope of this specification. The Directory Server role is optional for a compliant management system.

11.5.1 SLP Functions

The Directory Server role is required to implement SLP Directory Agent (DA) functionality as specified in 10.8, "Directory Agents (DAs)". The Directory registers all Agents and Object Managers within its configured scope and allows queries for their respective service specific attributes.

11.5.2   CIM-XML Protocol Functions

There are no additional CIM-XML requirements for this role.

11.5.3   Security Considerations

There are no additional security requirements for this role.

11.5.4   Lock Management Functions

There are no requirements for locking in this release of the specification.

11.6   Combined Roles on a Single System

11.6.1   Overview

As mentioned previously, the various roles of the management system can be deployed in different combinations to different systems throughout the managed environment. In general, there are no restrictions on what roles can be deployed on any given system, but some examples are given to illustrate typical situations.

11.6.2   General Purpose SMI-S Server as a Profile Aggregator

11.6.2.1   SLP Functions

The General Purpose SMI-S Server role may implement SLP User Agent (UA) functionality as specified in 10.6, "User Agents (UA)". The General Purpose SMI-S Server discovers all Profiles within its configured scope that are aggregated by querying for service specific attributes that match the criteria for those aggregations.

11.6.2.2   CIM-XML Protocol Functions

The General Purpose SMI-S Server role may implement CIM-Client functionality as specified by CIM-XML standard and may implement CIM-Listener functionality as specified by CIM-XML standard. A General Purpose SMI-S Server may reflect instances and classes from the aggregated Profiles (perhaps by delegating operations to the Dedicated SMI-S Servers), but is not required to do so. The Profile's Model instances should be reflected in the advertised default namespace of the General Purpose SMI-S Server. The hierarchy of General Purpose SMI-S Servers and Dedicated SMI-S Servers in a multi-level system needs to be reflected in the model such that it can be administrated.

11.6.2.3   Security Considerations

There are no requirements for security for this role.

11.6.2.4   Lock Manager Functions

There are no requirements for locking in this release of the specification.

Clause 12: Installation and Upgrade

12.1   Introduction

The interoperability of the management communications in a storage network gives customers a choice in vendors of their management solutions, but it also can introduce ease-of-use problems when these different vendors each supply different components. In order to supply a complete management solution, many management vendors provide not only WBEM Clients, Providers and other Management Interfaces, but also software components that provide other pieces of the management infrastructure (e.g., Directory Services, WBEM Services, Database Management). Problems are possible when multiple vendors install or remove these components in the same configuration and conflicts can arise. One of the goals of creating management interoperability is to reduce the time and expense end-users apply to the management of their SANs. Thus, SAN management should be easy to install, easy to upgrade, and easy to reconfigure. Mature management products using SMI-S technology should experience seamless and almost completely automated installation, upgrade, and reconfiguration.

This clause deals with issues in installation, upgrade and uninstallation of products using SMI-S technology, and recommends some steps that vendors should take to minimize the problems, leading to better customer satisfaction with the overall management solution.

12.2   Role of the Administrator

Ultimately, a vendor's installation software cannot make perfect decisions when the conflicts referenced in Table 12.1 arise, since there may be valid reasons why a customer has deployed software of similar function from multiple vendors. In the situation where two software components are both installed that perform the same shared function, and only one can reasonably operate without conflicts, the administrator must be able to resolve these conflicts and remove or disable the redundant component(s).

Installation software should, however, make a best effort to detect any conflicts and notify the administrator of possible conflicts during its installation and initialization. A vendor's installation software should allow the administrator to install and uninstall the various infrastructure components on an individual basis should such a conflict arise. The implications of this are that vendors are motivated to support interoperation with other vendor's components. The advantage to the vendor is that a customer is more likely to install a component that can demonstrate the most interoperability with other components.

12.3   Goals

12.3.1 Non-Disruptive Installation and De-installation

WBEM Clients & Services, Providers, and Directory Services may be capable of being installed and de-installed without disrupting the operation of other constituents in a SMI-S management environment. As SANs are often deployed in mission critical environments the up-time of the solution is critical and thus, the uptime of the management backbone as a key component of the solution is equally critical. Additionally, the installation and de-installation of SMI-S interface constituents should not compromise the availability of mission critical applications.

12.3.2 Plug-and-Play

The ultimate goal of management interoperability is zero administration of the management system itself. A customer should be able to install new storage hardware and software and have the new component become part of the management system automatically. Use of the Service Discovery process (see Clause 10: Service Discovery), the discovery-related aspects of the SMI-S Role definitions (see Clause 11: SMI-S Roles), and the Server profile (see *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profile) are intended to assist in achieving this goal.

During the reconfiguration of the management system, the schema that Clients see should remain consistent (Schema forward compatibility is ensured via CIM standard).

12.4 Device Support

12.4.1 General

Manufacturers of storage hardware and software typically install their product and the accompanying management support at the same time. The SMI-S Reference model (see 4.3, "Reference Model") defines a number of different models for this management support.

Conflicts are possible between Agents if multiple vendors attempt to install support for the same device. Also, when a device vendor needs to upgrade an Agent or Provider for a device, the installation software needs to determine all of the locations of the previous installations to insure there is not duplicate management paths to the device and thus, insure reliable on-going operation of the device.

12.4.2 Discovery and Initialization of Device Support

The SMI-S Reference Model (see 4.3, "Reference Model") defines two "Proxy Models" in which management support is provided via an Agent or through an Object Manager (with providers). In these models, the device support is expected to provide a means for establishing a reliable connection between the device itself and the Agent or Object Manager. Also, a special Client with administration/installation capability (as supplied by the vendor) is required to supply the relevant credentials for device access to the Agent or Object Manager designated to manage the device. This special Client may obtain the IP address of the device via automated means (not defined in this standard) or via manual means (e.g. by requiring a system manager to manually input the IP address of the device/subsystem from documentation supplied by the vendor).

12.4.3 Uninstallation

During the uninstallation of a device, the installation/uninstallation software (if available) should automatically detect existing management support software for the device in order to shutdown and remove it in a consistent manner. This detection process need to be cognizant that SMI-S Clients may be actively using the device and that the device may need to be disabled for new management operations and administrated through an orderly shut-down procedure prior to uninstallation. The implementation of such procedures and any order dependency is outside the scope of this specification, but may need to be considered by implementors.

12.4.4 Update

During the update of device support software, installation software should automatically detect any existing device support software in order to successfully complete the upgrade. This device support may exist on multiple hosts, but that situation is not specified in this version. If the update includes installing a new provider, the installation software needs to use the provider installation/upgrade method that is supported by the existing Object Manager.

When a software update involves a major schema version upgrade (e.g., 2.x to 3.x), the installation software needs to be cognizant of the effect of the schema upgrade on existing clients. For example, it may choose to simultaneously support both versions for some period of time.

12.4.5 Reconfiguration

When device support update requires an update of an agent or provider, the device support installation software should configure the new provider with the same subscriptions that exist in the old agent or provider before removing it, unless those subscriptions are specifically defined as being periodically cleaned up. This can be done via the instances of the subscriptions in the agent or object manager that currently exist.

12.4.6 Failure

Agents can become unavailable for several reasons. This includes the managed device being powered off and transient network failures. If a device's model becomes unavailable, it is recommended that Clients do not immediately remove that device from its visualization. If the device model reappears in another location, the old visualization should be updated to remove the previous occurrence. Also, the client can keep track of how long the device was down for purposes of availability management, etc. Clients may have to restore indication subscriptions

when the agent subsequently becomes available. In the case of the two Proxy Models in the SMI-S Reference Model, the agent (or its host, or the Object Manager) may go down, or its network connection could fail, but the device may still be available and this needs to be considered in designing availability management. In the case of a provider, the provider to device communication channel may also fail, but the device may still be available for access.

12.5   WBEM Service Support & Related Functions

12.5.1   Installation

Customers are increasingly sensitive to the size of the memory footprint for management software. The goal is to minimize the impact on hosts that are not dedicated to running management software by making appropriate choices during installation and giving the administrator control over these issues.

It is recommended that vendors take advantage of an existing Object Manager where one exists, by installing a provider that communicated with that Object Manager for device support. Additional support for such "multi-tenant" Object Managers will be included in a future version of this document.

If an object manager does not exist, or the device support does not work with the existing object manager (e.g. due to interface requirements) it is recommended that the vendor supply a Agent that is lightweight for device support. Another option is to offer to install an Object Manager that the vendor does have provider support for, allowing other vendors to further leverage that installation.

Providers that use an in-band connection to devices have an issue where zoning may alter the management path to the device from a provider or agent. In this case, the device support may need to be installed on multiple hosts in the network and the vendor needs to provide some way to coordinate which provider or agent is responsible for a particular device.

Vendors should install their providers in a unique namespace for isolation and qualification reasons. The installer should employ the Service Discovery process (see Clause 10: Service Discovery), and/or the Server profile (see *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profile) to discover the existing namespaces and insure that the one created for the new device is truly unique.

12.5.2   Multiple CIM Servers on a Single Server System

At installation and setup, a user interface should be provided by the CIM Server installation utility that allows an administrator to manually set the TCP port number in a persistent fashion.

To support discovery, the SLP Service Agent (see 10.7, "Service Agents (SAs)") associated with a newly-installed CIM Server should register its TCP port number along with all the other necessary discovery information with the Discovery Service. This requirement applies to both automated port selection as well as manually configured installations. Clients, working through their SLP User Agent (see 10.6, "User Agents (UA)"), then use this information to establish contact with the CIM Server.

12.5.3   Uninstallation/Upgrade

An Object Manager may be upgraded without needing to change the Providers that it supports. Depending on the Object Manager, the Providers may have to be reinstalled and reconfigured following such an upgrade. In this case, an administrator may need to re-run the device support installation software and such software should be able to restore the previous configuration.

12.5.4   Reconfiguration

Device Support Reconfiguration (see 12.4, "Device Support") identifies issues that may also be applicable to Object Managers.

12.5.5 Failure

Temporary failure of an object manager (for example, a host being powered off) can result in bad installation decisions for installation software. In this case, it is advisable that the installation software provide for manual input of the characteristics of additional components of the management system that the installation process needs to consider.

12.6 Client

12.6.1 Uninstallation

When Client software is removed, the uninstallation software should ensure that all client-defined information (settings, policies etc.), and any subscriptions for that client that exist in any agent or object manager, are also removed.

12.6.2 Reconfiguration

Client software can include a Listener that is configured to listen on a specific port. When this port is reconfigured, the client should redirect any Indication Handlers in existing agent and object managers as a result.

12.7 Directory Service

12.7.1 Installation

The installation of more than one Directory Agent (see 10.6, "User Agents (UA)") or Service Agent Server (see 10.7, "Service Agents (SAs)") providing a Directory Service in a management system does not impose a significant burden for management clients and adds to the overall availability. Vendors should recommend to administrators of their products that one or more SA Servers or Directory Agents should be deployed in the management system. This may also be done for network or system management reasons.

12.7.2 Uninstallation/Failure

SLP Clients are defined to handle failure and uninstallation of DAs as per the specification (see Clause 10: Service Discovery).

12.8 Issues with Discovery Mechanisms

Experience with existing SMI-S installations has indicated that some sites have policies that can impact the Service Discovery process (see Clause 10: Service Discovery). This subject will be addressed in greater detail in a future revision of this document, but two specific items of guidance are given here, as follows:

a) Where the site policy has caused multicast to be disabled, the DHCP option for SLP defined in IETF RFC 2610 is recommended as an alternate method of locating Service Agent Servers or Directory Agents. Also note that the shipping configuration of many network routers has multicast disabled.

b) Where the site policy has caused support for SLP itself to be disabled, an out of band method of providing a list of IP addresses for CIM Servers is recommended, after which the Server profile (see *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profile) should be used to obtain the information about Registered Profiles usually retrieved via SLP.

Annex A: (Informative) Mapping CIM Objects to SNMP MIB Structures

A.1   Purpose of this appendix

In order to encourage adoption of the WBEM initiative, its associated data model (CIM), protocol (xmlCIM), and profiles (described in previous sections of this specification), the Storage Media Library (SML) workgroup defined a means of mapping CIM objects to SNMP MIB objects, or "fields." SNMP (Simple Network Management Protocol) is the popular non-proprietary network management protocol used by the storage devices. This "CIM-to-MIB" mapping methodology has been successfully used by members of SNIA-SML to demonstrate—at minimal cost in development time—WBEM-based interoperability in plugfests and industry demonstrations such as Storage Networking World. The "CIM-to-MIB" mapping methodology is mentioned in this specification in order to:

•   Document that a standard path of backward compatibility is obtainable between WBEM and SNMP-based management paradigms,

•   Document one successful method of CIM-to-MIB mapping,

•   Recommend this method as *the* standard CIM-to-MIB mapping method in order to avoid a proliferation of deviant *de facto* standards, and

•   Allow companies to benefit from earlier experience and work.

A.2   CIM-to-MIB Mapping Overview

CIM is an object-based modeling schema that supports all common object-oriented principles, including abstract class objects, instance objects, inheritance, single- and multiple-association, aggregation, properties, methods, and qualifiers. In contrast, SNMPís ASN.1-based modeling schema is strictly hierarchical, involving such structures as nested parent and child nodes, and scalar and tabular fields. While unique CIM objects are typically referenced by parent class name (or Creation Class Name) and key properties, SNMP objects are typically referenced by an Object Identifier (OID) that points to their position in the SNMP Management Information Base (MIB) hierarchy or ìtree.î (In the case of tabular fields, additional indexes are appended to a base OID to identify unique instances of information.) The task of any CIM-to-MIB mapping methodology is primarily to create a one-to-one mapping between object-oriented information and tree-based hierarchical information. Naming constraints within the CIM and MIB domains must also be adhered to in a way that prevents ambiguities in uniquely identifying and referencing information, particularly in the SNMP/MIB domain. Therefore, SMLs mapping methodology provides the following:

•   A description of mapping CIM data -- classes, instances, properties, associations ñ into an SNMP format involving nodes, fields, and tables,

•   A naming convention in the SNMP/MIB domain that allows for unambiguous identification of the original CIM data,

•   A data type mapping that allows common CIM data to be represented by existing ASN.1 data types.

A.3   The SML MIB

As the CIM object model continues to change and expand, the SML MIB has also changed and expanded. As a result, it has become impractical to include the full MIB in each revision of this SMI specification.

SMI client application vendors or others interested in obtaining the latest SML MIB, or more information on the CIM-to-MIB mapping methodology in general, should contact the SNIA SML Technical Workgroup. SNIA-SML's website is: http://www.snia.org/apps/org/workgroup/sml/ or http://www.snia.org/tech_activities/workgroups/sml

Annex B: (Normative) Compliance with the SNIA SMI Specification

B.1   Compliance Statement

The declaration of SMI-S compliance of a given CIM Instance within a CIM Server also declares that any CIM Instance associated, directly or indirectly, to the first CIM Instance will also be SMIS compliant if SMIS itself declares compliance rules for either CIM Instance or instances of their superclasses.

B.2   How Compliance Is Declared

- The declaration of SMI-S compliance is made through the use of the server profile and the declaration of supported profiles.

- Direct association between CIM Instances is made through instance of a CIM Association.

- Indirect association between CIM Instance is made through more than one CIM Association.

- SMI-S Compliance is assessed against CIM Instances that are directly or indirectly associated to the CIM Instance declared as part of the declaration of supported registered profiles. These CIM Instances comprise the compliance test set.

- All CIM Instances / CIM Classes included in the compliance test set for whom compliance rules are defined in SMI-S or for superclasses thereof shall be themselves be compliant to the rules defined in SMI-S.

- Compliance tests on a superclass of a given CIM Instance are limited to the attributes and behaviors defined for the superclass.

B.3   The Server Profile and Compliance

Compliance is declared by the implementation of the Server Profile. All profiles require the Server profile. The server profile defines the means by which a SMI-S Client determines the profiles and subprofiles supported and the ComputerSystems associated. (see *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* Clause 42: Server Profilefor more details.)

B.3.1   Example

A CIM Agent for Vendor X declares compliance to the Array Profile and the Pool Manipulation Capabilities, and Setting Subprofile through the Server Profile. Once the association (via the ElementConformsToProfile association) is made to from the Array Profile declaration to the ComputerSystem that realizes the Array Profile, then compliance tests begin testing compliance. Vendor X decided to extend the StorageVolume class with additional properties. StorageVolume is associated to the ComputerSystem via SystemDevice association. ComputerSystem, StorageVolume, and SystemDevice are defined in SMI-S as required CIM elements (see *Storage Management Technical Specification, Part 3 Block Devices, 1.3.0 Rev 6* Table 2, "CIM Elements for Array").

In implementing FCPort, Vendor X decided to not provide ElementName but did provide the rest of the required properties. Vendor X decided to not use to WWN and instead used a vendor specific value for the PermanentAddress (see Clause 7: Correlatable and Durable Names) Additionally, Vendor X added FRUStatus to their subclass of FCPort. Vendor X also decided to model the back-end fibre channel, but not use an SMI-S model to do so. These back-end FCPorts are associated to the ComputerSystem via the ConsumedSystemDevice association, a subclass of SystemDevice without properties overridden. These back-end fibre channel ports where modeled using a Vendor X specific class, BackendFCPorts, that is not derived from FCPort. This BackendFCPorts were associated to the ComputerSystem with the ConsumedSystemDevice.PartComponent role.

The compliance test includes FCPort because compliance declaration identified a particular ComputerSystem the entry point into compliant CIM instantiation of the Array Profile. the compliance test includes FCPorts as part of the test set because the SystemDevice association, also defined as part of the profile, includes the FCPort realized in

that implementation. The compliance test also includes BackendFCPorts because the ConsumedSystemDevice association to the ComputerSystem for these instances is a SystemDevice association.

The compliance test locates the StorageConfigurationService, StoragePools including a Primordial StoragePool, and StorageCapabilities associated to the ComputerSystem. Vendor X's implementation supports the creation of a StoragePool. The test attempts to create a StoragePool given one of the sizes reported by the Primordial StoragePool.getSupportedSizes() method using the Primordial StoragePool reference and a StorageSetting generated from one of the StorageCapabilities.

The compliance test for Vendor X's Array Profile implementation fails because:

- FCPort.PermanentName property has a noncompliance value. Specifically, the FCPort.PermanentAddress is required to be WWN, 16 unseperated uppercase hex digits;

- ElementName property was not provided (i.e. was null);

- the SystemDevice associations contained references to BackendFCPort in the PartComponent property. CIM defined that the PartComponent is a LogicalDevice. Since BackendFCPort is not a LogicalDevice, then the test failed;

- The "Size not supported" return code was returned from CreateOrModifyStoragePool even though one of the supported sizes was used verbatim.

The compliance test for Vendor X's Array Profile implementation did not fail because:

- StorageVolume was extended;

- SystemDevice was extended.

B.4   Backward Compatibility

Backward compatibility between versions of SMI-S profiles is a requirement with very few exceptions. The goals of backwards compatibility include:

a) New profile implementations that are deployed in a customer environment work with existing SMI-S Clients. This includes:

1) SMI-S operations, including recipes and CTP, continue to work against the new profile implementation;

2) SMI-S Clients can support a given profile version and above (later minor version numbers);

b) No guarantee of backwards compatibility is implied between major version numbers (i.e. 1.x to 2.x);

c) If a profile in a newer version of SMI-S cannot maintain backward compatibility, it shall be renamed (and the old profile deprecated). Otherwise the client may assume that the newer profile is backwards compatible and that all operations in the earlier version will continue to work in this newer version.

d) It shall be possible for SMI-S provider and client implementations to support older versions of an incompatible profile.

e) Content marked experimental is not standard in this version of the specification. Future versions of the specification may not be backwards compatible to content marked experimental in this version. Content marked experimental in this version of the specification may be removed in a future version. See Figure 1.

B.4.1   Overview

SMI-S backward compatibility is necessary to ensure that customer environments are minimally disrupted by newer implementations of SMI-S. Deployment of several concurrent implementations of multiple minor versions of SMI-S shall be possible in a customer environment. Compatibility is required from both the Client side and from the provider side. Compatibility also has aspects both in the specification of newer functionality via SMI-S and in the implementation of both providers and clients.

Figure B.1 shows the interaction between a Client coded to an older minor version of SMI-S (M.m) acting against a later minor version (M.n) provider implementation.
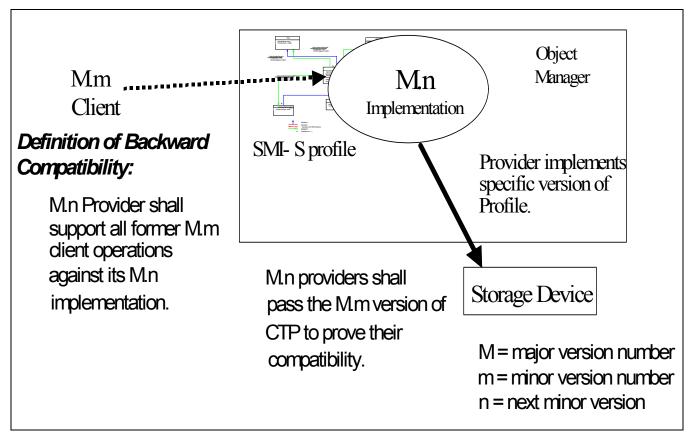

Figure B.1 - Provider Migration

As shown in Figure B.1, the newer implementation shall support all of the old operations from the previous minor version of SMI-S in order to maintain compatibility. The Client will not be able to take advantage of any newer features that have been added in the later version of the specification, but will still be able to accomplish all of the functions it was coded for in the previous version. This allows minimum disruption to the customer environment.

Clients shall be written to take advantage of the functionality of implementations that are currently shipping and that are or will soon be deployed in customer environments. This client functionality needs to be careful in how it makes use of each SMI-S version's new features. Any client code that uses a specific version's features shall also include a version check against the profile or subprofile version in the RegisteredProfile (Subprofile) instance for that functionality. This version check shall verify that the functionality is at a specific minor version and above (up to the next major release). If a client were only to check for a specific version, it would not be able to use newer implementations of that functionality. A client will, over time, contain multiple such code blocks as newer versions are supported. Each piece of code will be written to the functionality introduced in a specific version and continue to work against that functionality in later minor releases.

B.4.2   Requirements

In order to maintain backwards compatibility with older minor versions of the specification, profile authors have followed specific rules in developing the specification. The requirements that were followed in profile versioning and shall be followed by subsequent implementations include:

- **Support for required classes:** A newer minor version of an SMI-S profile shall support all required classes of the previous minor version of the profile and shall continue to require them.

- **Support for conditional classes:** A newer minor version of an SMI-S profile shall support all conditional classes of the previous minor version of the profile and shall continue to require them as specified in the conditions of the previous minor version. But the newer minor version may add other conditions under which the class will be required. In addition, conditional classes in a previous minor version may be promoted to required in a newer minor version.

- **Support for optional classes:** A newer minor version of an SMI-S profile may promote a class to Conditional or Mandatory any class that was optional in the previous minor version.

- **Deprecation of classes:** A newer minor version of an SMI-S profile may deprecate or include deprecated (via the CIM schema) classes introduced in previous minor version(s), but shall continue to require their implementation.

- **Support for required properties:** A newer minor version of an SMI-S profile shall support all required properties of classes in the previous minor version(s) of the profile and shall continue to require them.

- **Support for conditional properties:** A newer minor version of an SMI-S profile shall support all conditional properties of classes in the previous minor version(s) of the profile and shall continue to require them as specified by the conditions of the previous minor version. But the newer minor version may add other conditions under which the property will be required. In addition, conditional properties in a previous minor version may be promoted to required in a newer minor version.

- **Support for optional classes:** A newer minor version of an SMI-S profile may promote a class to Conditional or Mandatory any class that was optional in the previous minor version.

- **Deprecation of properties:** A newer minor version of an SMI-S profile may deprecate or include deprecated (via the CIM schema) properties of classes introduced in previous minor version(s), but shall continue to require their implementation.

- **Support for subprofiles:** A newer minor version of an SMI-S profile shall support the functionality of all subprofiles of the previous minor version(s) of the profile and shall continue to require them if they were required in the previous version. A newer minor version of an SMI-S profile may require a subprofile that was optional or conditional in the previous minor version, but shall not make optional or conditional a subprofile that was required in a previous minor version. If a newer minor version of an SMI-S profile does not have subprofiles by the same name as previous minor version(s), it shall still require implementation of the Registered (Sub)Profile with the previous version information such that the client will be able to find and use the subsumed functionality.

- A newer minor version of an SMI-S profile shall support all conditional subprofiles of the previous minor version of the profile and shall continue to require them as specified in the conditions of the previous minor version. But the newer minor version may add other conditions under which the subprofile will be required.

- **Profile renaming:** A newer minor version of an SMI-S profile that cannot remain backwards compatible shall either become a major revision of the profile or shall be renamed to a different profile name such that a client will not find newer, incompatible, versions of that functionality.

B.4.3   Implementation Considerations

Even in the case of a newer minor version of an SMI-S profile that was unable to retain backward compatibility, an implementation may support clients with a separate implementation of the previous minor version's functionality. Implementations shall not implement these earlier versions in such a way that a client of the previous minor version would become confused or break when accessing this functionality. This may happen if the previous version's functionality is implemented in the same namespace as the later version, but a careful evaluation needs to be done by the implementer to determine this. Particular attention should be paid to the recipes from the earlier version, but since recipes are not exhaustive, a fuller evaluation is necessary.

B.5   Rules for Combining (Autonomous) Profiles

B.5.1   General

SMI-S specifies the behavior of (autonomous) profiles. The rules for compliance and backward compatibility are defined in the context of a profile (an Autonomous Profile). This subclause defines the rules that shall be applied when a device (or program) wishes to support the behavior of multiple (autonomous) profiles.

The guiding principles in such support are:

• Maintain Compliance (see B.1 through B.3)

Combining (autonomous) profiles shall not break compliance rules for any of the combined individual profiles.

• Maintain Backward Compatibility (see B.4)

Combining (autonomous) profiles shall not break backward compatibility for any of the combined individual profiles.

B.5.2   Backward Compatibility Rules for combining profiles

The backward compatibility rules apply to combined profiles in that combined profile implementations that are deployed in a customer environment shall work with SMIS clients of any one of the profiles that were combined:

• **Support for required classes:** A combination of SMI-S profiles shall support all required classes of the individual profiles that have been combined and shall continue to require them. If a class is required in one individual profile, it shall be required in the combination profile.

• **Support for conditional classes:** A combination of SMI-S profiles shall support all conditional classes of the individual profiles that have been combined and shall continue to require them as specified in the conditions of individual profiles that have been combined. If a class is conditional in one or more of the individual profiles (and not required in any other individual profile) then it shall be conditional in the combination profile. If a class is conditional in multiple individual profiles, but with different conditions, then all conditions shall yield the existence of the class.

• **Deprecation of classes:** A combination of SMI-S profiles shall include any deprecated (via the CIM schema) classes introduced by any one of the individual profiles that are combined, and shall continue to require their implementation. Similarly, conditions for deprecated conditional classes shall apply (as stated in the support for conditional classes).

• **Support for required properties:** A combination of SMI-S profiles shall support all required properties of classes in any one of the individual profiles that are combined and shall continue to require them. If a property is required in any of the individual profiles, then the property will be required in the combined profile.

• **Support for conditional properties:** A combination of SMI-S profiles shall support all conditional properties of classes in the individual profiles that are combined and shall continue to require them as specified by the conditions of the individual profiles that are combined. If a property is conditional in one or more of the

individual profiles (and not required in any other individual profile) then it shall be conditional in the combination profile. If a property is conditional in multiple individual profiles, but with different conditions, then all conditions shall yield the existence of the class.

- **Deprecation of properties:** A combination of SMI-S profiles may include deprecated (via the CIM schema) properties of classes introduced in any one of the individual profiles that are combined, and shall continue to require their implementation. Similarly, conditions for deprecated conditional properties shall apply (as stated in the support for conditional properties).

- **Support for subprofiles:** A combination of SMI-S profiles shall support the functionality of all subprofiles of all of the individual profiles that are combined and shall continue to require them if they were required in any one of the individual profiles that are combined. If a combination of SMI-S profiles results in two references to a subprofile by the same name from multiple individual profiles that were combined, the combined profile may require multiple implementations if the subprofiles in question have different major version numbers. And if the subprofiles have different minor version numbers, then the higher version number shall be implemented (since it provides backward compatibility to the earlier subprofile).

If a subprofile is required in any one of the individual profiles then it will be required in the combined profile.

If a subprofile is not required in any of the individual profiles, but is conditional in at least one of the individual profiles, then it will be conditional in the combined profile. If a subprofile is conditional in multiple individual profiles (that are being combined) then all conditions shall yield existence of the subprofile.

B.5.3   Conditions for a New Profile

If any of the conditions outlined in section B.5.1 cannot be satisfied, then a new profile shall be defined that represents the desired semantic of the device (or program) in question.

Annex C: (Normative) Indication Filter Strings

WBEM indications are defined using filter strings. The filter strings are expressed in a query language that includes the type of indication and related CIM elements.

For versions of this standard starting with 1.3.0, new indication filters shall be defined using CQL (see DMTF DSP0202).

Prior to version 1.3.0, indication filters were defined using two query languages:

CQL was recommended but experimental since the DMTF specification was not a final standard.

WQL was a proposed query language partially described in white papers and later withdrawn in favor of CQL The subset of WQL used in this standard is also referred to as the SMI-S 1.0.x query language. This set of filters defined in this annex defines the full set of WQL functionality used in SMI-S.

Although CQL and WQL support complex filter strings, the filters used in SMI-S are very simple and may be expressed as a few patterns – literal text containing a limited number of variables representing CIM elements. The patterns are defined in the following simple grammar:

- literal text does not include curly brackets ("{" and "}")

- variables are surrounded by curly brackets; the usage of variables is explained in the "Semantic" sub-section following each filter stringS

C.1   Instance Creation

C.1.1   Filter String

The same filter string applies to CQL and WQL.

```
SELECT * FROM CIM_InstCreation WHERE
    SourceInstance ISA {class-name}
```

C.1.2   Semantic

An instance of a class is instantiated. {class-name} is the name of a class (or one of its subclasses) of the instance created.

C.2   Instance Deletion

C.2.1   Filter String

The same filter string applies to CQL and WQL.

```
SELECT * FROM CIM_InstDeletion WHERE
    SourceInstance ISA {class-name}
```

C.2.2   Semantic

An instance of a class is deleted. {class-name} is the name of a class (or one of its subclasses) of the instance deleted.

C.3 Modification of any value in an array property

C.3.1 WQL string

```
SELECT * FROM CIM_InstModification WHERE
  SourceInstance ISA {class-name} AND
  SourceInstance.{property-name} <>
  PreviousInstance.{property-name}
```

C.3.2 CQL string

```
SELECT * FROM CIM_InstModification WHERE
  SourceInstance ISA {class-name} AND
  SourceInstance.{class-name}::{property-name} <>
  PreviousInstance.{class-name}::{property-name}
```

C.3.3 Semantic

One of the values of the array property {property-name} in class {class-name} (or one of its subclasses) has been modified, or an additional value is added to {property-name} or a value is removed from {property-name}.

C.4 Modification to either of Two Specific values in an Array Property

C.4.1 WQL string

```
SELECT * FROM CIM_InstModification
  WHERE SourceInstance ISA {class-name}
  AND SourceInstance.{property-name} = {value}
  AND SourceInstance.{property-name} = {value}
```

C.4.2 CQL string

```
SELECT * FROM CIM_InstModification
  WHERE SourceInstance ISA {class-name}
  AND ANY
  SourceInstance.{class-name}::{property-name}[*] = {value1}
  AND ANY
  SourceInstance.{class-name}::{property-name}[*] = {value2}
```

C.4.3 Semantic

The array property {property-name} in class {class-name} (or one of its subclasses) has been modified resulting in one of the entries in the array having a value of {value1} and another of the entries having a value of {value2}. Either {value1} or {value2} shall be a new value for an existing entry or is the value of a newly added entry.

C.5 Alert

C.5.1 Filter String

The same filter string applies to CQL and WQL.

```
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA'
  AND MessageID='{message-id}'
```

NO_ANSI_ID

Note that WQL does not require the quotes around the value of OwningEntity. Legacy profiles may use a filter string including OwningEntity=SNIA (without quoting SNIA) as WQL filter, but CQL strings shall include the quotes.

C.5.2   Semantic

An alert indication referencing the standard message with message ID {message-id}. Note that the message ID is a concatenation of the name of the appropriate SNIA registry and message number. For example, the {message-id} for the first message in the FC registry is 'FC1'.