Storage Management Technical Specification, Part 3 Common Profiles

Version 1.6.1, Revision 5

Abstract: This SNIA Technical Position defines an interface between WBEM-capable clients and servers for the secure, extensible, and interoperable management of networked storage.

SNIA Technical Position

December 17, 2014

Revision History

Revision 1

Date

1 December 2011

SCRs Incorporated and other changes

FCoE Target Ports Profile (SMIS-160-Addenda-Draft-SCR00005)
- **Added** as a new (Draft) Profile

Date

25 May 2012

SCRs Incorporated and other changes

FCoE Target Ports Profile (CORE-SMIS-SCR-00062)
- **Promoted** FCoE Target Ports Profile to Experimental

Diagnostics Job Control Profile (SMIS-160-Addenda-Draft-SCR00004)
- **Added** a Diagnostics Job Control profile as a specialization of the DMTF Job Control profile and **promoted** it to Experimental

Comments

Editorial notes and DRAFT material are displayed.

Revision 2

Date

27 August 2013

SCRs Incorporated and other changes

Common Profiles part number changed to Part 3, per ISO request change re SMI-S 1.5

Comments

Editorial notes and DRAFT material are displayed.

Revision 3

Date

4 December 2013

SCRs Incorporated and other changes

References
- Added the reference to the DMTF Diagnostics Job Control profile in the Normative References per SMIS-160-Addenda-Draft-SCR00004
- Updated DMTF references to be current.

Indications
- Rolled forward AlertIndications updates per SMIS-160-Errata-SCR00006.

Comments

Editorial notes are displayed.

DRAFT material is hidden.

Revision 4

Date

25 February 2014

SCRs Incorporated and other changes

None

Comments

Editorial notes and DRAFT material are hidden.

Revision 5

Date

11 August 2014

SCRs Incorporated and other changes

Annex: SMI-S Information Model
- CIM version updated to V2.41 per TSG ballot -- Correct CIM Schema Version in SMI-S.

Comments

Editorial notes and DRAFT material are hidden.

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/

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:

1) Any text, diagram, chart, table or definition reproduced shall 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 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.

All code fragments, scripts, data tables, and sample code in this SNIA document are made available under the following license:

BSD 3-Clause Software License

Copyright (c) 2014, The Storage Networking Industry Association.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of The Storage Networking Industry Association (SNIA) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

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 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.

INTENDED AUDIENCE

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

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

Maturity Level

In addition to informative and normative content, this specification includes guidance about the maturity of emerging material that has completed a rigorous design review but has limited implementation in commercial products. This material is clearly delineated as described in the following sections. The typographical convention is intended to provide a sense of the maturity of the affected material, without altering its normative content. By recognizing the relative maturity of different sections of the standard, an implementer should be able to make more informed decisions about the adoption and deployment of different portions of the standard in a commercial product.

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

Contents

List of Figures

List of Tables

Foreword

The Storage Management Technical Specification is published in several parts. *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* defines profiles that are used by profiles in other parts of this standard. In general, the common profiles do not fully define storage elements, but define non-storage management aspects that are common to storage domains. For example, the Access Points Profile defines a technique that the arrays, switches, or libraries may use to inform clients of non-CIM network interfaces that are available.

Some of the common profiles are based on DMTF profiles. For these profiles, the DMTF profile may be "specialized" to assure SNIA requirements are met.

Parts of this Standard

This standard is subdivided in the following parts:

- *Storage Management Technical Specification, Part 1 Overview, 1.6.1 Rev 5*
- *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5*
- *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5*
- *Storage Management Technical Specification, Part 4 Block Devices, 1.6.1 Rev 5*
- *Storage Management Technical Specification, Part 5 Filesystems, 1.6.1 Rev 5*
- *Storage Management Technical Specification, Part 6 Fabric, 1.6.1 Rev 5*
- *Storage Management Technical Specification, Part 7 Host Elements, 1.6.1 Rev 5*
- *Storage Management Technical Specification, Part 8 Media Libraries, 1.6.1 Rev 5*

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, 4360 ArrowsWest Drive, Colorado Springs, Colorado 80907, U.S.A.

1 Scope

Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5 defines profiles that are supported by profiles defined in the other parts of this standard. The first few clauses provide background material that helps explain the purpose and profiles and recipes (a subset of a profile). Common port profiles are grouped together since they serve as transport-specific variations of a common model. The port profiles are followed by other common profiles The last clause presents recipes that span multiple profiles.

2 Normative References

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

2.1 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)

2.2 DMTF References (Final)

DMTF Final documents are accepted as standards.

DMTF DSP0004, CIM Infrastructure Specification 2.3
http://www.dmtf.org/standards/published_documents/DSP0004V2.3_final.pdf

DMTF DSP0200, CIM Operations over HTTP 1.3
http://www.dmtf.org/standards/published_documents/DSP0200_1.3.0.pdf

DMTF DSP1001, Management Profile Specification Usage Guide
http://www.dmtf.org/standards/published_documents/DSP1001.pdf

DMTF DSP1013:2006, Fan Profile 1.0.1
http://www.dmtf.org/standards/published_documents/DSP1013_1.0.1.pdf

DMTF DSP1015:2006, Power Supply Profile 1.0.1
http://www.dmtf.org/standards/published_documents/DSP1015_1.0.1.pdf

DMTF DSP1011:2006, Physical Asset Profile 1.0.2
http://www.dmtf.org/standards/published_documents/DSP1011_1.0.2.pdf

DMTF DSP1009:2009, Sensors Profile 1.0.2
http://dmtf.org/sites/default/files/standards/documents/DSP1009_1.0.2.pdf

DMTF DSP1025:2009, Software Update Profile 1.0.0
http://www.dmtf.org/standards/published_documents/DSP1025_1.0.0.pdf

DMTF DSP1103, Job Control Profile v1.0.0
http://www.dmtf.org/sites/default/files/standards/documents/DSP1103_1.0.0_0.pdf

2.3 IETF References (Standards or Draft Standards)

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

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

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

IETF RFC 2445 Internet Calendaring and Scheduling Core Object Specification (iCalendar)
http://www.ietf.org/rfc/rfc2445.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 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
http://www.ietf.org/rfc/rfc3280.txt

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

2.4 References under development

DMTF DSP0202 CIM Query Language Specification 1.0
http://www.dmtf.org/standards/published_documents/DSP0202.pdf

DMTF DSP0207 WBEM URI Mapping 1.0
http://www.dmtf.org/standards/published_documents/DSP0207.pdf

DMTF DSP1102:2009, Launch in Context Profile 1.0.0b
http://www.dmtf.org/standards/published_documents/DSP1102_1.0.0b.pdf

DMTF DSP1119, Diagnostics Job Control Profile v1.0.0b
http://www.dmtf.org/sites/default/files/standards/documents/DSP1119_1.0.0b.pdf

Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5

2.5 Other References

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

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

3 Definitions, Symbols, Abbreviations, and Conventions

3.1 General

For the purposes of this document, the definitions, symbols, abbreviations, and conventions given in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* and the following apply.

3.2 Terms

3.2.1
FC-SB-X

Fibre Channel Single-Byte command set used in FICON™[1] devices

3.2.2
SAS

Serial Attached SCSI

3.2.3
SATA

Serial ATA

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.

4 Profile Introduction

4.1 Profile Overview

A profile is a specification that defines the CIM model and associated behavior for an autonomous and self-contained management domain. The CIM model includes the CIM Classes, Associations, Indications, Methods and Properties. The management domain is a set of related management tasks. A profile is uniquely identified by the name, organization and version.

In SMI-S, a profile describes the management interfaces for a class of storage subsystem, typically realized as a hardware of software product. For example, SMI-S includes profiles for arrays, FC-Switches, and logical volume manager software. The boundaries chosen for SMI-S profiles are often those of storage products, but some vendors may package things differently. For example, one vendor may choose to package an Array and an FC Switch into a single product; this can be handled in SMI-S by implementing the Array and FC Switch Profiles for this product.

A profile may add restrictions to usage and behavior, but cannot change CIM defined characteristics. For example, if a property is required in the CIM model, then it is required in a profile. On the other hand, a profile may specify that a property is required even if it is not required by the general CIM model.

In SMI-S, profiles serve several purposes:

- Specification organization - the SMI-S object model (see *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 6 Object Model General Information) is presented as a set of profiles, each describing a type of storage element or behavior,
- Certification - SMI-S profiles form the basis for CTP certification,
- Discovery- profiles are registered with the CIM Server and advertised to clients as part of the CIM model and using SLP (see *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 9 Service Discovery. An SMI-S client uses SLP to determine which CIM Servers host profiles it wishes to manage, then uses the CIM model to discover the actual configurations and capabilities.

A subprofile is a profile that specifies a subset of a management domain. A subprofiles's CIM elements are scoped within a containing profile. Multiple profiles may use the same subprofile. A subprofile is uniquely identified by the name, organization and version.

A profile specification may include a list of the subprofiles it uses. The included subprofiles may be optional or mandatory by the scoping profile. The behavior of a profile is specified in this profile and its included subprofiles.

For example, target devices such as RAID arrays and tape libraries may support Fibre Channel or parallel SCSI connectivity. SMI-S includes an FC Target Port Subprofile and a Parallel SCSI Target Port Subprofile that may optionally be supported by profiles representing target devices. The elements defined in the port subprofiles are scoped to the ComputerSystem in the profile. For example, each LogicalPort subclass has a SystemDevice association to the profile's ComputerSystem.

In addition to sharing the purposes of profiles (as described in this section), subprofiles have these purposes:

- Optional behavior - a profile may allow, but not require, an implementation to support a subprofile. Although a subprofile does not describe a full product, a subprofile should describe an aspect of a product that is recognizable to an knowledgeable end-user such as a storage administrator,
- Reuse of functionality - some storage management behavior is common across different types of storage elements. For example, block virtualization is managed similarly in RAID arrays and logical volume managers. These common sets of functionality are specified as profiles that are shared by several other profiles.

- Decomposition - certain functionality may not be reused multiple places, but is complicated enough to document as a separate profile. For example, Disk Partition management is only used in the Host Discovered Resources profiles, but is complicated enough that it has been documented as a separate profile.

4.1.1 Terminology

A profile collects included subprofiles and provides the filler needed to define the management interfaces of a particular type of subsystem. Profiles are separated into two groups. *Storage profiles* define the management interfaces for storage subsystems such as arrays or FC switches. *Generic profiles* define management interfaces for generic systems that are related to storage management. Storage and generic profiles are specified the same way in SMI-S, but generic profiles are not certified as free-standing entities, only as a dependency of a storage profile.

A *Package* is a profile that whose implementation is mandatory to comply with the requirements of all of its containing top-level profiles. Since a package is always mandatory, it is not registered with the CIM Server. Packages provide decomposition in the specification.

Profiles may be related by *specialization* - where several profiles (or subprofiles) share many common elements, but are specialized for specific implementations. The SMI-S Security profiles are an example; the specializations (Authorization Profile, Security Resource Ownership Profile,...) share some classes and behavior. Profile specialization is only an artifact of the specification. It saves the reader from reading common aspects in multiple places and help the specification stay consistent across the specialized profiles. There is no information in the CIM model about the relationship between generic and specialized profiles.

4.2 Format for Profile Specifications

For each profile there is a set of information that is provided to specify the characteristics and requirements of the profile. Subprofiles are also defined using this format, but they are clearly identified as subprofiles.

Each profile or subprofile is defined in subsections that are described in Table 1.

NOTE CIM schema diagrams are logically part of a profile description. However, they can be rather involved and cannot be easily depicted in a single diagram. As a result, the reader is advised to refer to DMTF characterizations of CIM schema diagrams.

Table 1 - Profile Components (Sheet 1 of 2)

Profile Element	Goal
Description	This section provides a description of the profile and model including an overview of the objectives and functionality. *Functionality* is described in a bullet-form in this section that includes functionality provided by the subprofiles referenced by the profile. If a function is provided by a subprofile, this is indicated, including whether the subprofile is optional or required. Functionality listed in the profile is organized by Levels, and within each Level by FCAPS category, as defined in the SMI-S functionality matrix section <link>. *Instance Diagrams:* One or more instance diagrams to highlight common implementations that employ this section of the Object Model. Instance diagrams also contain classes and associations but represent a particular configuration; multiple instances of an object may be depicted in an instance diagram. Finally, this section may include supporting text for recipes, properties, and methods as needed.
Health & Fault Management Considerations	If a profile provides optional Health & Fault Management capabilities, then this section describes the specifics of these capabilities, including: • A table of the classes that report health information • Tables of possible states of the OperationalStatus and HealthState attributes and descriptions for those elements that report state. • Cause and Effect associations. • Standard Errors produced (including Alert Indications, Errors, CIM Errors, and Health Related Live Cycle Events).
Cascading Considerations	A Profile may be a cascading profile. A **cascading profile** is any Profile that supports the Cascading Subprofile as either a mandatory or recommended subprofile. If the profile is a cascading profile, this section documents cascading considerations in each of the following areas: • Cascaded Resources – Defines the type of resources in the Cascading Profile that are associated to what type of resources in the Leaf Profile and the association. • Ownership Privileges – Identifies the Resource Control Privileges (on leaf resources) that are established by the Cascading Profile. • Limitations on Cascading Subprofile – Identifies any limitations on the Cascading Subprofile that are imposed by the Cascading in effect
Supported Profiles, Subprofiles, and Packages	A list of the names and versions of subprofiles and packages supported by a profile.
Methods of the Profile	This section documents the methods used in this profile. All methods used in recipes shall be documented; optional methods (those not used in recipes) may also be included.
Client Considerations and Recipes	This section documents a set of "recipes" that describe the CIM operations and other steps required to accomplish particular tasks. These recipes do not define the upper bound of what a CIM Server may support, however, they define a lower bound. That is, a CIM Provider implementation shall support these recipes as prescribed to be SMI-S compliant. NOTE A recipe that is defined as part of a subprofile is only required if the subprofile is implemented. All optional behavior in a profile shall be described in a recipe and shall have a capabilities property a client can test to determine whether the optional behavior is supported. The actual capabilities properties are documented in “Classes Used in the Profile” in this table.

Table 1 - Profile Components (Sheet 2 of 2)

Profile Element	Goal
CIM Elements	A table listing the classes, associations, subprofile, packages, and indication filters that this profile (or subprofile) supports, and a brief description of each. Everything listed in this section is mandatory for the profile or subprofile. This section shall not list optional elements. Prior to SMI-S 1.1.0, CIM did not have standard language for indication filters; SMI-S 1.0.x used the proposed WQL query language. This version of SMI-S uses the CQL standard query language. WQL is also supported for backwards compatibility. The Description column for an indication filter specifies whether the filter string is compliant to CQL or WQL. If neither is stated, then the string complies to both CQL and WQL.
Classes Used in the Profile	This section provides one table per class and lists each required and recommended property. For each required or recommended property a brief description on what information is to be encoded is identified. The class tables include a “Flags” column. This can contain “C” (the property is a correlatable name or a format for a name), “D” (the property is a durable name), “M” (the property is modifiable), or “N” (null is a valid value).

5 Recipe Overview

5.1 Recipe Concepts

Recipe: A set of instructions for making something from mixing various ingredients in a particular sequence. The set of ingredients used by a particular recipe is scoped by the particular profile, subprofile or some other well-defined context in which that recipe is defined.

A recipe shall specify an interoperable means for accomplishing a particular task across all conformant implementations. However, a recipe does not necessarily specify the only set of instructions for accomplishing that task. Nor are all tasks that may be accomplished necessarily specified by the set of recipes defined for a particular profile or subprofile.

In order to compress the document, some recipes are implied or assumed. This would include, for instance, that the set of available, interoperable properties are those explicitly defined by a particular profile or subprofile.

For a profile, the set of all defined and implied recipes defines the range of behavior across for which interoperability is mandatory for all conformant implementations. Unless specifically defined in a recipe, other sequences of actions (even simple Create/Delete instance requests) are not guaranteed to have the same results across multiple implementations.

Each recipe defines an interoperable series of interactions (between a SMI-S Client and a SMI-S Server) required to manage storage devices or applications. Another goal is to list the operations required for the CIM Client realize functionality. It is not a goal to comprehensively express the programming logic required to implement the recipe in any particular language. In fact, recipes are limited to the expression of CIM or SLP operations, and may simply reference or describe any of the implementation that may be required beyond that.

5.2 Recipe Pseudo Code Conventions

5.2.1 Overview

A recipe's instructions are written using the pseudo code language defined in this section.

All recipes are prefixed with a summary narrative of the functionality being implemented. This summary may be included explicitly as part of the recipe or reference to the appropriate narrative that can be found elsewhere in the specification.

NOTE The use of optional features (profiles or subprofiles) in recipes shall be clearly identified.

Generic Operations should be used, but for backward compatibility CIM Operations (from CIM Operations over HTTP) may also be used. Arrays grow in size automatically.

5.2.2 General Syntax

<condition> logical statement that evaluates to true (Boolean)

!<condition> tests for false (Boolean)

<action> unspecified list of programming logic that is not important to the understanding of the reader for a particular recipe.

<EXIT: *success message*>Exits the recipe with a success status code. The condition that resulted in the call to exit the recipe was allowable. The implementation subjected to the recipe behaves in accordance to this specification.

<ERROR! *error condition*> Exits the recipe with a failure status code. The condition that resulted in the call to the exit the recipe was not allowable. The implementation subjected to the recipe does not behave in accordance with this specification.

@{recipe} logic flow is contained within the specification of the recipe elsewhere in the specification

<variable> some variable

5.2.3 CIM related variable and methods

5.2.3.1 CIM Instances and Object Names

$name represents a single instance (CIMInstance) with a given variable name

$name.property represents a property in a single instance (CIMInstance)

$name.getObjectPath()
method returns a object name, REF, to the CIM Instance

$name.getNameSpace()
method returns the namespace name for the CIM Instance or Object Name

{value1, value2 ...}
an anonymous array, comprised of selected values of a given type; an anonymous array is an array that is not referable by a variable

EXAMPLE:

{"Joe", "Fred", "Bob", "Celma"}

$name[] represents an array of instances (CIMInstances) with a given variable name; array are initialized by constructing an anonymous array.

EXAMPLE:

Names = {"Joe", "Fred", "Bob", "Celma"}

$name-> represents an object path name (CIMObjectPath)

$name->[] represents an array of object names of a given name

$name->property
represents a property of object $name

$name[].size() returns the number of CIM instances in the array

$name->[].length returns the number of CIM object names in the array

#name[].length returns the number of variable elements in the array

%name[].length returns the number of method arguments elements in the array

5.2.3.2 Extrinsic method arguments

%name represents a CIM Argument that can contain any CIM or other variable.

%name[] represents an array of CIM Arguments

5.2.3.3 Other Variables

#name neither CIM Instance nor Object Name variable. The type may be a string, number or some other special type. Types are defined in the CIM Specification 2.2.

#name[] a non-CIM variable array

"literal" some string literal

5.2.4 Data Structure

Variables can be collected by an array. The array can be indexed by other variable (see 5.2.3.3).

Arguments are always indexed by strings. In other words, the arguments are retrieved from the array by name.

5.2.5 Operations

= assigns right value to left value

== test for equivalency

!= test for not equivalency

< true if the left argument is numerically less than the right argument.

> true if the left argument is numerically greater than the right argument.

<= true if the left argument is numerically less than or equal to the right argument.

>= true if the left argument is numerically greater than or equal to the right argument.

&& condition A AND condition B

|| condition A OR condition B

+, -, *, / addition, subtraction, multiplication and division, respectively

++, -- increment and decrement a variable, respectively; placement of the operator relative to the variable determines whether the operation is completed before or after evaluation

EXAMPLE:

```
#i = 1
#names[] = {"A", "B, "C"}
"B" == #names[++#i]  is true
2 == #i is true
```

EXAMPLE:

```
#i = 2
#names[] = {"A", "B, "C"}
"B" == #names[#i++]  is true
3 == #i is true
```

// comments

nameof returns an Object Name given a CIM Instance. This unitary operator does nothing in other usages.

ISA tests for the name of the CIM Instance or object name

EXAMPLE: if ($SomeName-> ISA CIM_StorageVolume) {
<The Object Name is a reference to a CIM_StorageVolume >
}

5.2.6 Control Operations

The pseudocode used in this specification relies on control operators common to most high-level languages. For example:

- **for**

  EXAMPLE:

  ```
  for #x in <variable array> {
  <actions>
  }
  ```

- **if**

  EXAMPLE:

  ```
  if (<condition>) {
  <actions>
  } ;
  if (<condition>) {
  <actions>
  } else {
  <alternate actions>
  }
  ```

- **do/while**

  EXAMPLE:

  ```
  do {
  <actions>
  } while (<condition>)
  ```

- continue
  Within a **for** loop: initialize loop variable to next available value and restart loop body. Terminate loop if no more loop variable values available. Within a **do/while** loop: transfer control immediately to **while** test.

  EXAMPLE:

  ```
  for #i in <array> {
       if (<some condition>)
          continue;    // process next loop variable
      <alternative>
  }
  ```

- **break**: interrupts the sequence of statement execution within a loop block and exits the loop block altogether. The looping condition is not re-evaluated Statement execution starts at the next statement outside of the loop block.

- **exit**
  Terminate recipe instantly, including termination of any callers.

  EXAMPLE:

  ```
  if (<unexpected condition>)
     exit
  ```

5.2.7 Functions

5.2.7.1 Function Declaration

A function definition is of the form *sub functionName(),* followed by the body of the function enclosed in braces. If parameters are to be passed to a function, then are expressed as a comma-separated list of

arguments within the parentheses following the function name. Each argument is comprised of a data type and an accompanying argument name.

Functions are declared at the beginning of a recipe.

```
sub functionName(integer nArg1, Class &cArg2) {
<actions>
}
```

5.2.7.2 Function Invocation

A function invocation is of the form *&functionName()*. If parameters are to be passed to a function, then are expressed as a comma-separated list within the parentheses following the function name.

```
&functionName(5, pClass)
```

5.2.8 Exception Handling

All operations may produce exceptions or errors. The following construct is used to test for particular errors. Once a particular error is caught, then special exception handling logic is processed. Only CIM Errors can be caught.

```
try {
     <actions>
}
     catch (CIM Exception $Exception) {
 <recovery actions>
 }
 The error received may also be thrown
 throw $Exception
```

The error response returned from the SMI-S implementation is treated as a exception, a "CIM Exception". The catch condition is expressed in terms of the CIM status code returned (e.g., CIM_ERR_NOT_FOUND) as defined in the CIM Operations specification.

The $Exception variable contains a Error instance. The $Exception CIM Instance may be examined like any other CIM Instance. In this language, the $Exception is never null even if the SMI-S implementation does provide one. In this case, the $Exception CIM Instance is empty with the exception of the CIMStatusCode and CIMStatusCodeDescription properties. This properties are populated with the Status and Description returned in the error response from the SMI-S implementation.

5.2.9 Built-in Functions

a) boolean = compare(<variable>, <variable>)

   1) Used to determine if two variables of the same type are equivalent

   2) The variables shall not be CIM instances or object names nor other complex data types or structures

   3) The variables shall be of the same type

b) $instance = newInstance("CIM Classname")

   1) Creates a CIM instance, which does not exist in the CIMOM (yet), that can be later filled in with properties and passed to CreateInstance. The namespace is assumed to be the same that the CIM client connected to.

c) $instance - newInstance("CIM Namespace", "CIM Classname”)

1) Variable of the above method that has the namespace name as an argument

d) boolean = contains(<test value>, <variable array>)

1) Used to test if the variable array contains a value equivalent to the test variable

2) The array shall be of variables of the same types as the test variable.

3) If the equivalency is found with at least one value then the function returns true, else false is returned.

4) If the array is not a simple, or non-CIM, data type, then the test value shall be a CIM property, $SomeInstance.SomeProperty or $SomeObjectname->SomeProperty

e) %Argument = newArgument("Argument Name", <variable>)

1) Creates a CIM Argument of a given name containing a value, CIM or non-CIM

f) $objectPath-> = newObjectPath("Class name", "NameSpace name")

1) Returns a new ObjectPath, built from the supplied arguments;

2) Required to perform the `EnumerateInstances and EnumerateInstanceNames operations`

g) #stringArray[] = #stringVariable.split(#stringParam or “string literal”)

1) Returns an array of strings, built by splitting the string variable around matches of the supplied string parameter

2) Divides the string into substrings, using the string parameter as a delimiter, returning the substrings in an array in the order in which they occurred in the string variable. If there are no occurrences of the string parameter, then the array returned contains only one string element equal to the original string variable.

h) #intVariable = Integer(#stringVariable)

1) Returns the integer that the supplied string represents. If the supplied string does not represent an integer, then an error is thrown.

2) The function will parse and return signed or unsigned integers up to 64-bits in size, and will accept the hyphen ‘-‘ character in the 8-bit ASCII-range of UTF-8 as the first character in the string to indicate a negative number.

i) #datetimeVariable = Datetime(#stringVariable)

1) Returns a variable of Datetime type, as defined by section 2.2.1 the CIM Infrastructure Specification v1.3, that the supplied string represents. If the supplied string does not represent a DateTime object, then an error is thrown.

2) This function will accept strings of the format described in the CIM Infrastructure Specification, including both timestamps and intervals, zero-padded to 25-characters, and will recognize Datetime strings containing asterisk (“*”) characters for fields that are not significant.

5.2.10 Extrinsic method calls

```
<variable> = InvokeMethod ($someobjectname->, "Method Name",
 %InArguments[], %OutArguments[])
```

EXPERIMENTAL

6 Generic Target Ports Profile

6.1 Synopsis

Profile Name: Generic Target Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.9.0 (specialized profiles may need later versions)

Related Profiles for Generic Target Ports: Not defined in this standard.

Central Class: CIM_LogicalPort

Scoping Class: a CIM_System in a separate autonomous profile

The Generic Target Port Profile models the generic behavior of target ports in storage systems such as disk arrays and tape libraries.

This abstract profile specification shall not be directly implemented; implementations shall be based on a profile specification that specializes the requirements of this profile.

6.2 Description

The Generic Target Port Profile models the generic behavior of target ports in storage systems such as disk arrays and tape libraries. Separate profiles specialize the Generic Target Port Profile for Fibre Channel, iSCSI, and other transports. The primary classes of the Generic Target Port Profile are LogicalPort and ProtocolEndpoint, as shown in Figure 5. Instances of subclasses of a LogicalPort (e.g., FCPort, EthernetPort) represent the logical aspects of ports, independent from command protocols (such as SCSI). Instances of subclasses of ProtocolEndpoint (e.g., SCSIProtocolEndpoint or ATA ProtocolEndpoint) represent command protocols in use on the port.

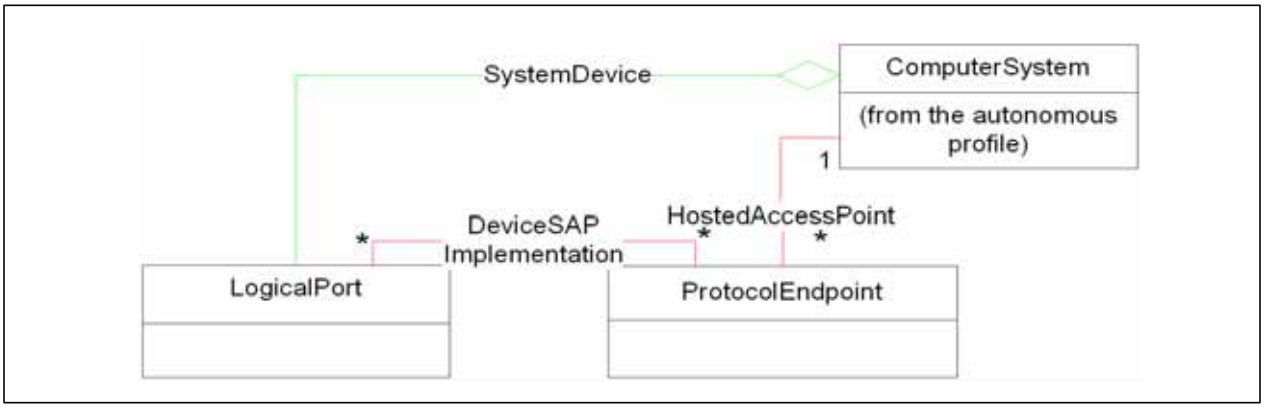

Figure 5 - Generic Target Port Classes

6.3 Implementation

Subclasses of ProtocolEndpoint represent command protocols supported by the port. SCSIProtocolEndpoint represents SCSI as a protocol, independent of specific transports or device types – i.e., the behavior described in the SCSI Primary Commands (SPC) and SCSI Architecture Model (SAM)

specifications from T10. SCSIProtocolEndpoint.Role indicates whether this protocol endpoint instance represents a SCSI Target or target. For target port profiles, Role shall be Target" or "Both Initiator and Target". iSCSIProtocolEndpoint specializes SCSIProtocolEndpoint with additional iSCSI-specific properties.

ATAProtocolEndpoint represents the ATA command protocol. SBPProtocolEndpoint represents Single Byte protocol used with mainframes. ProtocolEndpoint is associated to a System instance with Hosted Access Point.

LogicalPort subclasses specify the type of transport. If the port is subclassed directly from LogicalPort it indicates it is connected to a bus. If the port is further subclassed from NetworkPort it indicates the port is capable of being used in a network. Specializations of this profile shall specify the appropriate subclass of LogicalPort. Figure 6 shows the subclasses of LogicalPort.

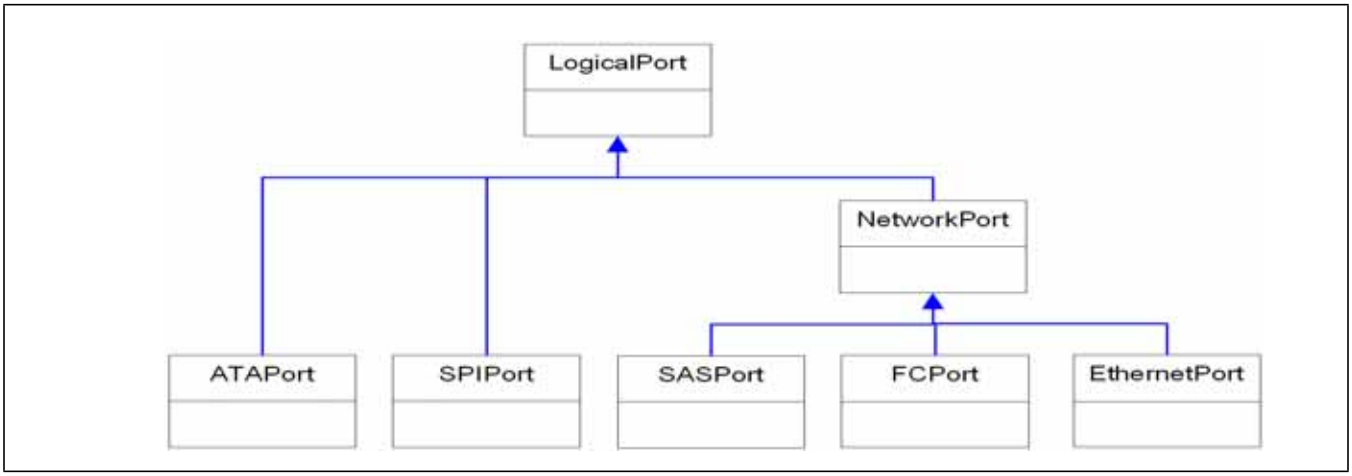

Figure 6 - LogicalPort Class Hierarchy

A property on LogicalPort called UsageRestriction is indicates whether the port is restricted to use as a "front end" (target) or a "back end" (Target) interface or both. Note that port may not have a restriction and the actual point-in-time role is modeled in SCISProtocolEndpoint.Role. SystemDevice associates LogicalPort to a System.

ProtocolEndpoint and LogicalPort are associated with DeviceSAPImplementation. For most transports, the command protocol is implemented in the port hardware and there is 1-1 cardinality between the LogicalPort and ProtocolEndpoint instances. iSCSI is an exception, many-to-many relationships are possible between EthernetPort and iSCSIProtocolEndpoint instances

ProtocolController (in the Mapping and Masking Profile) represents the SCSI (or SB) 'view' of ports and logical devices seen by target systems (e.g., arrays). In a system supporting Mapping and Masking, zero or more views exist; defined by the customer to expose subsets of logical units to certain Targets. SAPAvailableForElement connects ProtocolEndpoint from a target ports profile to SCSIProtocolController instances from the Mapping/Masking Profile. iSCSI and SB have protocol-specific, secondary uses of ProtocolController.

Figure 7 depicts a generic storage device with elements from a target ports profile, the Mapping/Masking Profile, and a target device profile The LogicalDevice object represents logical units that are visible to

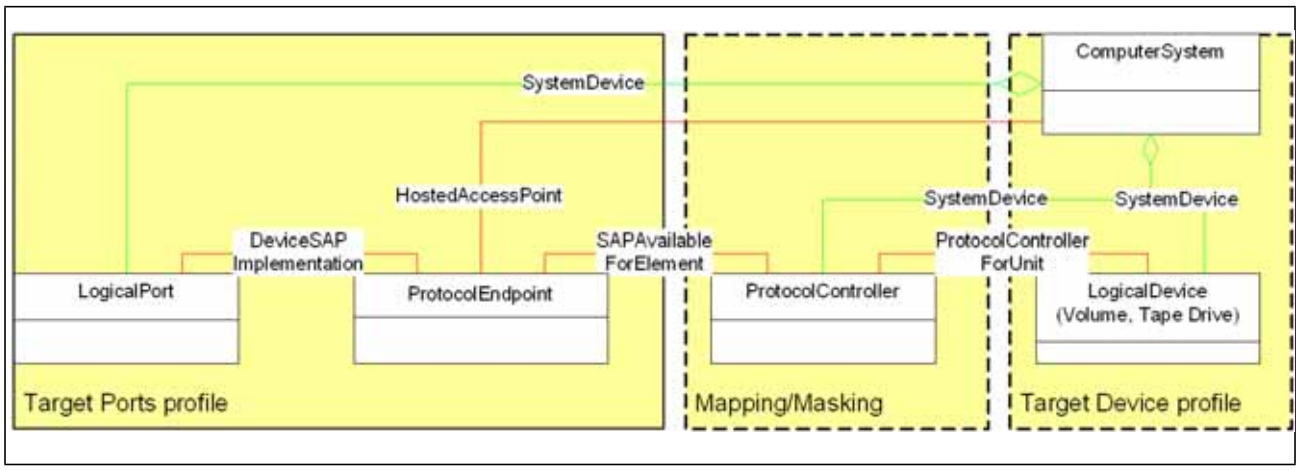

Figure 7 - Generic Target with LUN Masking

external systems. It is subclassed to StorageVolume, TapeDrive, etc. to identify the device type.

6.3.1 Modeling SCSI/SB Logical Units

The SCSI standard inquiry response includes a Device Type property with integers representing types of devices. Most of these devices types have a CIM analog. Devices that are used primarily for management are modeled as SCSIArbitraryLogicalUnit. SCSIArbitraryLogicalUnit.DeviceType maps to SCSI device types. Table 2 describes how common storage devices are modeled in CIM.

Table 2 - Modeling of Common Storage Devices in CIM

SCSI Device Type	Inquiry Device Type	LogicalDevice subclass
DirectAccessDevice	0	DiskDrive or StorageVolume
SequentialAccessDevice	1	TapeDrive
WriteOnceDevice	4	WormDrive
CD-ROM	5	CDROMDrive
MediaChanger	8	MediaTranferDevice
ArrayController	0xc	SCSIArbitraryLogicalUnit DeviceType="SCSI SCC Device"
SES	0xd	SCSIArbitraryLogicalUnit DeviceType="SCSI SES"
Other		SCSIArbitraryLogicalUnit DeviceType="Other"
Unknown		SCSIArbitraryLogicalUnit DeviceType="Uknown"
DirectAccessDevice	0	DiskDrive or StorageVolume

All devices (logical units) visible to external systems shall be modeled.

6.4 Methods of the Profile

6.4.1 Extrinsic Methods

None

6.4.2 Intrinsic Methods

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

6.5 Use Cases

6.6 CIM Elements

Table 3 describes the CIM elements for Generic Target Ports.

Table 3 - CIM Elements for Generic Target Ports

Element Name	Requirement	Description
6.6.1 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.
6.6.2 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
6.6.3 CIM_LogicalPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
6.6.4 CIM_ProtocolEndpoint	Mandatory	ProtocolEndpoint representing support for SCSI, ATA, or SB command set.
6.6.5 CIM_SystemDevice (Port)	Mandatory	Associates ComputerSystem to LogicalPort.

6.6.1 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 4 describes class CIM_DeviceSAPImplementation.

Table 4 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to SCSI, ATA, or SB ProtocolEndpoint.
Antecedent		Mandatory	Reference to Port.

6.6.2 CIM_HostedAccessPoint

Associates ComputerSystem to ProtocolEndpoint. Limit to targets.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 5 describes class CIM_HostedAccessPoint.

Table 5 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in the referencing profile.
Dependent		Mandatory	Reference to SCSIProtocolEndpoint, ATAProtocolEndpoint or SBProtocolEndpoint.

6.6.3 CIM_LogicalPort

Represents the logical aspects of the physical port and may have multiple associated protocols.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 6 describes class CIM_LogicalPort.

Table 6 - SMI Referenced Properties/Methods for CIM_LogicalPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction		Mandatory	Shall be 2 for ports restricted to Front-end only or 4 if the port is unrestricted.
PortType		Mandatory	VALUE and DESC should be set appropriately for each specialized target port profile.

6.6.4 CIM_ProtocolEndpoint

ProtocolEndpoint representing support for SCSI, ATA, or SB command set. Shall be subclassed in specialized profiles.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 7 describes class CIM_ProtocolEndpoint.

Table 7 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription		Mandatory	Shall be the string 'SCSI','ATA', or 'SB'.

6.6.5 CIM_SystemDevice (Port)

Associates ComputerSystem to LogicalPort.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 8 describes class CIM_SystemDevice (Port).

Table 8 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem in the referencing profile.
PartComponent		Mandatory	Reference to Port.

EXPERIMENTAL

EXPERIMENTAL

7 Parallel SCSI (SPI) Target Ports Profile

7.1 Synopsis

Profile Name: SPI Target Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.9.0

Table 9 describes the related profiles for SPI Target Ports.

Table 9 - Related Profiles for SPI Target Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Central Class: CIM_SPIPortt

Scoping Class: a CIM_System in a separate autonomous profile

Models a parallel SCSI port,

7.2 Description

This port represents a SCSI Parallel Interface (SPI).

7.3 Implementation

Because of addressing limits, the port may use multiple SCSI IDs to extend the addressing. The LUN Mapping/Masking common subprofile is not used with this port type. Figure 8 shows an SPI Target Port Instance.

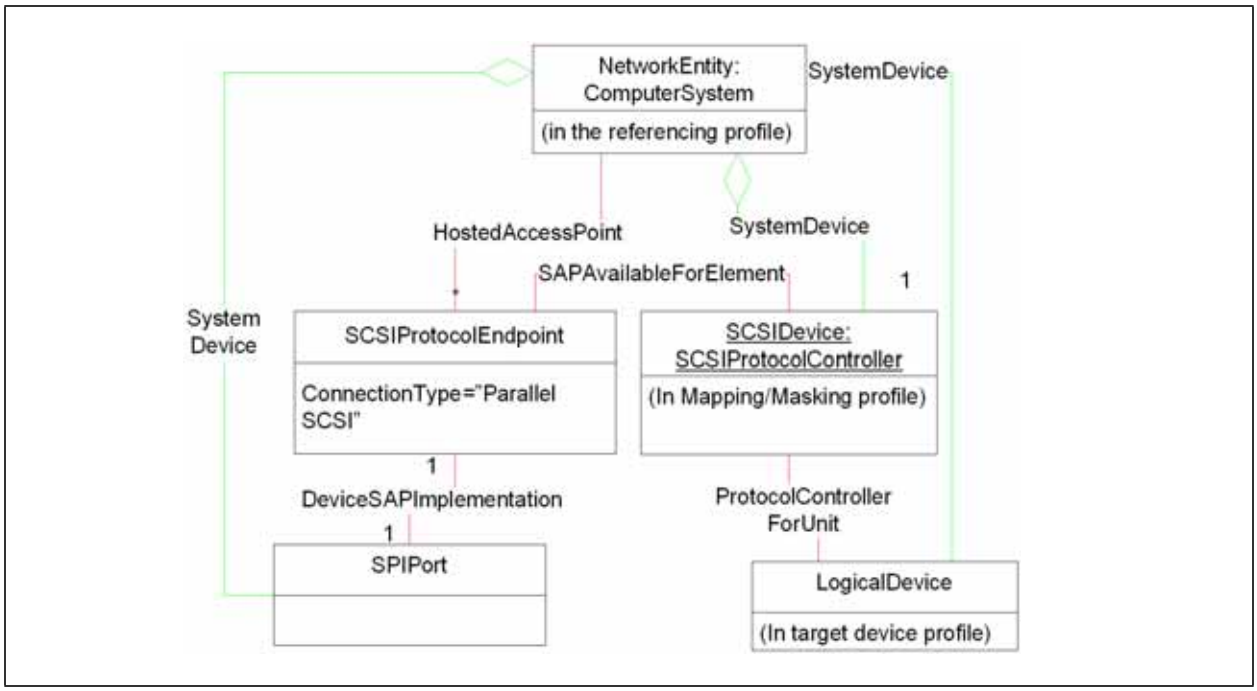

Figure 8 - SPI Target Port Instance Diagram

The SCSIProtocolEndpoint.ConnectionType shall be set to “Parallel SCSI”. The SCSIProtocolEndpoint class is connected to a SPIPort. Attributes of SPIPort define the bus width and speed. The port class inherits the UsageRestriction attribute from LogicalPort. This attribute shall be set to “Front-end only”

7.4 Health and Fault Management

Table 10 shows SPIPort OperationalStatus.

Table 10 - SPIPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

7.5 Methods

7.5.1 Extrinsic Methods of this Subprofile

None

7.6 CIM Elements

Table 11 describes the CIM elements for SPI Target Ports.

Table 11 - CIM Elements for SPI Target Ports

Element Name	Requirement	Description
7.6.1 CIM_DeviceSAPImplementation	Mandatory	Associates front-end SPIPort and target SCSIProtocolEndpoint.
7.6.2 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to SCSIProtocolEndpoint.
7.6.3 CIM_SCSIProtocolEndpoint	Mandatory	Represents management characteristics related to the SCSI command set.
7.6.4 CIM_SPIPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
7.6.5 CIM_SystemDevice (Port)	Mandatory	Associates ComputerSystem to LogicalPort.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SPIPort	Mandatory	CQL -Create SPIPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SPIPort AND SourceInstance.CIM_SPIPort::OperationalStatus <> PreviousInstance.SAS_Port::OperationalStatus	Mandatory	CQL -Modify SPIPort.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SPIPort	Mandatory	CQL -Delete SPIPort.

7.6.1 CIM_DeviceSAPImplementation

Associates front-end SPIPort and target SCSIProtocolEndpoint. Limit to target ProtocolEndpoints and front-end ports. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 12 describes class CIM_DeviceSAPImplementation.

Table 12 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to SPIPort.

7.6.2 CIM_HostedAccessPoint

Associates ComputerSystem to SCSIProtocolEndpoint. Limit to targets (Role = 3). The class definition specializes the CIM_HostedAccessPoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 13 describes class CIM_HostedAccessPoint.

Table 13 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in the referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.

7.6.3 CIM_SCSIProtocolEndpoint

Represents management characteristics related to the SCSI command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 14 describes class CIM_SCSIProtocolEndpoint.

Table 14 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
ConnectionType (added)		Mandatory	Shall be 3 (Parallel SCSI).
Role (added)		Mandatory	Shall be 3 (Target).

7.6.4 CIM_SPIPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 15 describes class CIM_SPIPort.

Table 15 - SMI Referenced Properties/Methods for CIM_SPIPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction (overridden)		Mandatory	Shall be 2 (Front-end Only).
PortType (overridden)		Mandatory	Shall be 140 (SCSI Parallel).

7.6.5 CIM_SystemDevice (Port)

Associates ComputerSystem to LogicalPort.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 16 describes class CIM_SystemDevice (Port).

Table 16 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem in the referencing profile.
PartComponent		Mandatory	Reference to Port.

EXPERIMENTAL

STABLE

8 FC Target Ports Profile

8.1 Synopsis

Profile Name: FC Target Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.9.0

Table 17 describes the related profiles for FC Target Ports.

Table 17 - Related Profiles for FC Target Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Central Class: CIM_FCPort

Scoping Class: a CIM_ComputerSystem in a referencing autonomous profile

8.2 Description

The FC Target Port Subprofile models the Fibre Channel specific aspects of a target storage system.

8.3 Implementation

For Fibre Channel ports, the concrete subclass of LogicalPort is FCPort. FCPort is always associated 1-1 with a SCSIProtocolEndpoint instance.

8.3.1 SMI-S 1.0 backwards compatibility

SCSIProtocolEndpoint was introduced in SMI-S 1.1.0 to enable support for non-FC transports and for non-SCSI protocols. In SMI-S 1.0, FCPort was associated directly to SCSIProtocolController. SCSIProtocolEndpoint, DeviceSAPImplementation, and SAPAvailableForElement are required and are used consistently across all target port subprofiles. To maintain backwards compatibility, ProtocolControllerForPort is still required in this version of SMI-S. But this association will be removed in a future versions and clients should start using the newer model. Figure 9 illustrates a Target Port instance.

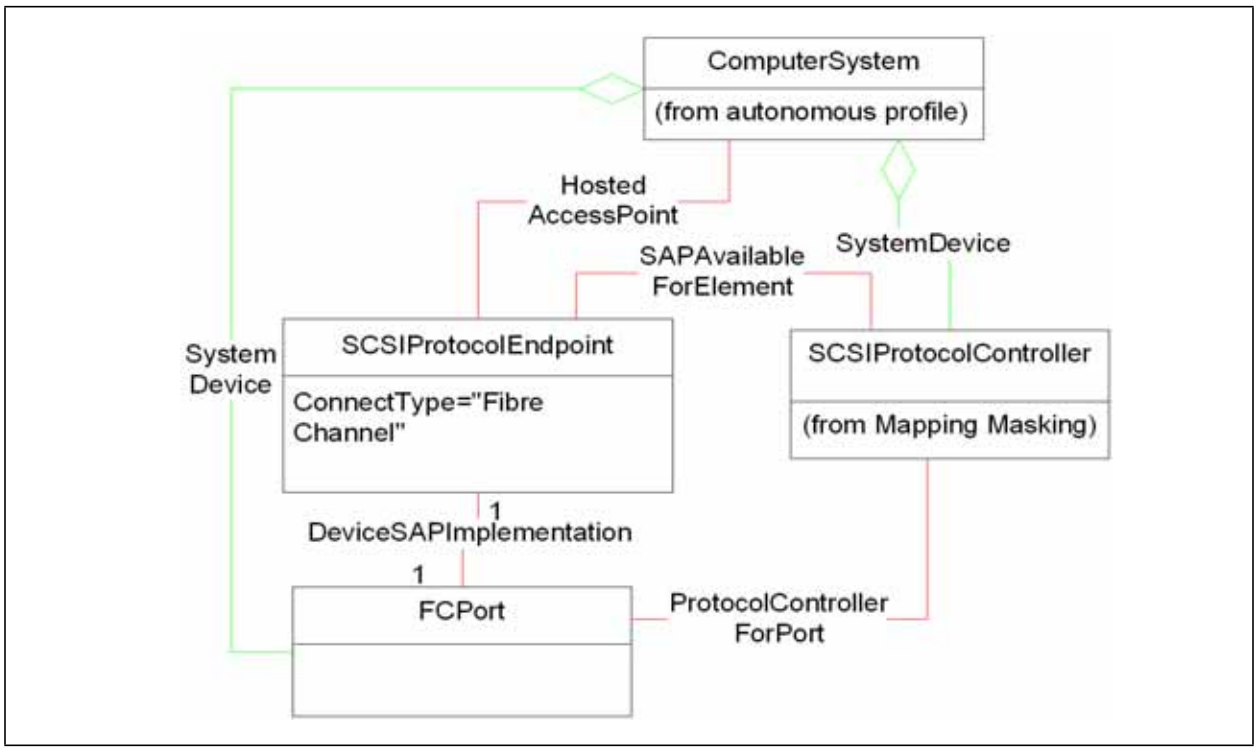

Figure 9 - FC Target Port Instance Diagram

8.4 Durable Names and Correlatable IDs of the Subprofile

FCPort.PermanantAddress shall contain the port's Port WWN.

8.5 Health and Fault Management

Table 18 describes FCPort OperationalStatus.

Table 18 - FCPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

8.6 Supported Profiles and Packages

None

8.7 Extrinsic Methods of this Subprofile

None

8.8 Client Considerations and Recipes

None

8.9 CIM Elements

Table 19 describes the CIM elements for FC Target Ports.

Table 19 - CIM Elements for FC Target Ports

Element Name	Requirement	Description
8.9.1 CIM_DeviceSAPImplementation	Mandatory	Associates FCPort and SCSIProtocolEndpoint.
8.9.2 CIM_FCPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
8.9.3 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to SCSIProtocolEndpoint.
8.9.4 CIM_ProtocolControllerForPort	Conditional	Conditional requirement: Support for the Masking and Mapping profile. Only required if the instrumentation claims compatibility with 1.0.
8.9.5 CIM_SCSIProtocolEndpoint	Mandatory	Represents management characteristics related to the SCSI command set.
8.9.6 CIM_SystemDevice (Port)	Mandatory	Associates controller ComputerSystem to FCPort.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort	Mandatory	Create FCPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Deprecated WQL -Change to FCPort OperationalStatus.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::OperationalStatus <> PreviousInstance.CIM_FCPort::OperationalStatus	Mandatory	CQL -Change to FCPort OperationalStatus.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.Speed <> PreviousInstance.Speed	Optional	Deprecated WQL -Change to FCPort Speed.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::Speed <> PreviousInstance.CIM_FCPort::Speed	Optional	CQL -Change to FCPort Speed.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.NetworkAddresses <> PreviousInstance.NetworkAddresses	Optional	Deprecated WQL -Change to FCPort NetworkAddresses.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::NetworkAddresses <> PreviousInstance.CIM_FCPort::NetworkAddresses	Optional	CQL -Change to FCPort NetworkAddresses.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort	Mandatory	Delete FCPort.

8.9.1 CIM_DeviceSAPImplementation

Associates FCPort and SCSIProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 20 describes class CIM_DeviceSAPImplementation.

Table 20 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to FCPort.

8.9.2 CIM_FCPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 21 describes class CIM_FCPort.

Table 21 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction		Mandatory	Shall be 2 for ports restricted to Front-end only or 4 if the port is unrestricted.
PortType (overridden)		Mandatory	Shall be 0|1|10|11|12|13|14|15|16|17|18 (Unknown or Other or N or NL or F/NL or Nx or E or F or FL or B or G).
PermanentAddress (added)	CD	Mandatory	Port WWN. Shall be 16 unseparated uppercase hex digits.
SupportedCOS (added)		Optional	
ActiveCOS (added)		Optional	
SupportedFC4Types (added)		Optional	
ActiveFC4Types (added)		Optional	
Speed (added)		Optional	Speed in bits per second. Shall be 0, 1062500000 (1GFC), 2125000000 (2GFC), 4250000000 (4GFC), 8500000000 (8GFC), 10518750000 (10GFC), 14025000000 (16GFC), 21037500000 (20GFC) or 28500000000 (32GFC).

Table 21 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
MaxSpeed (added)		Optional	Maximum Port Speed.
NetworkAddresses (added)		Optional	For Fibre Channel end device ports, the Fibre Channel ID. Shall be 16 un-separated upper case hex digits.

8.9.3 CIM_HostedAccessPoint

Associates ComputerSystem to SCSIProtocolEndpoint. Limit to targets. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 22 describes class CIM_HostedAccessPoint.

Table 22 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent (overridden)		Mandatory	
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.

8.9.4 CIM_ProtocolControllerForPort

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for the Masking and Mapping profile.

Table 23 describes class CIM_ProtocolControllerForPort.

Table 23 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForPort

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to SCSIProtocolController.
Dependent		Mandatory	Reference to FCPort.

8.9.5 CIM_SCSIProtocolEndpoint

Represents management characteristics related to the SCSI command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 24 describes class CIM_SCSIProtocolEndpoint.

Table 24 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
ConnectionType (added)		Mandatory	Shall be 2 (Fibre Channel).
Role (added)		Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target).

8.9.6 CIM_SystemDevice (Port)

Associates controller ComputerSystem to FCPort. The class definition specializes the CIM_SystemDevice definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 25 describes class CIM_SystemDevice (Port).

Table 25 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent (overridden)		Mandatory	Reference to ComputerSystem in the referencing profile.
PartComponent (overridden)		Mandatory	Reference to FCPort.

STABLE

STABLE

9 iSCSI Target Ports Subprofile

9.1 Synopsis

Profile Name: iSCSI Target Ports (Component Profile)

Version: 1.6.0

Organization: SNIA

CIM Schema Version: 2.27.0

Related Profiles for iSCSI Target Ports: Not defined in this standard.

Central Class: CIM_EthernetPort

Scoping Class: a CIM_System in a separate autonomous profile

Models an iSCSI target port

9.2 Description

The iSCSI Target Ports Subprofile describes the iSCSI specific aspects of a target device.

9.3 Implementation

iSCSI terminology is different than that used in other parts of SMI-S. Table 26 provides a map of terminology from iSCSI standards and CIM class names used in this standard. iSCSI does have a specific naming requirement for SCSIProtocolController that is described in Table 26.

Table 26 - iSCSI Terminology and SMI-S Class Names

iSCSI Term	CIM Class Name	Notes
Network Entity	ComputerSystem	The Network Entity represents a device or gateway that is accessible from the IP network. A Network Entity shall have one or more Network Portals, each of which can be used to gain access to the IP network by some iSCSI Nodes contained in that Network Entity.
Session	iSCSISession	The group of TCP connections that link a Target with a target form a session (loosely equivalent to a SCSI I-T nexus). TCP connections can be added and removed from a session. Across all connections within a session, a Target sees one and the same target.
Connection	NetworkPipe	A connection is a TCP connection. Communication between the Target and target occurs over one or more TCP connections. The TCP connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units (iSCSI PDUs).
SCSI Port	iSCSIProtocolEndpoint	A SCSI Port using an iSCSI service delivery subsystem. A collection of Network Portals that together act as a SCSI Target or target.

Table 26 - iSCSI Terminology and SMI-S Class Names

Portal Group	SystemSpecificCollection	iSCSI supports multiple connections within the same session; some implementations will have the ability to combine connections in a session across multiple Network Portals. A Portal Group defines a set of Network Portals within an iSCSI Network Entity that collectively supports the capability of coordinating a session with connections spanning these portals. Not all Network Portals within a Portal Group need participate in every session connected through that Portal Group. One or more Portal Groups may provide access to an iSCSI Node. Each Network Portal, as utilized by a given iSCSI Node, belongs to exactly one portal group within that node.
Network Portal	TCPProtocolEndpoint, IPProtocolEndpoint, EthernetPort	The Network Portal is a component of a Network Entity that has a TCP/IP network address and that may be used by an iSCSI Node within that Network Entity for the connection(s) within one of its iSCSI sessions. A Network Portal in a Target is identified by its IP address. A Network Portal in a target is identified by its IP address and its listening TCP port.
Node	SCSIProtocolController	The iSCSI Node represents a single iSCSI Target or iSCSI target. There are one or more iSCSI Nodes within a Network Entity. The iSCSI Node is accessible via one or more Network Portals. An iSCSI Node is identified by its iSCSI Name. The separation of the iSCSI Name from the addresses used by and for the iSCSI Node allows multiple iSCSI nodes to use the same address, and the same iSCSI node to use multiple addresses.

Figure 10 is a class diagram for iSCSI Target Ports and uses the UML instance naming notation (InstanceName:ClassName) with the iSCSI-style names before the CIM names. Figure 26 explains the use of all these objects.

Note that ComputerSystem, SCSIProtocolController and StorageVolume are not actually part of this subprofile; they would be the parts of the Array Profile that associate with the iSCSI-specific classes.

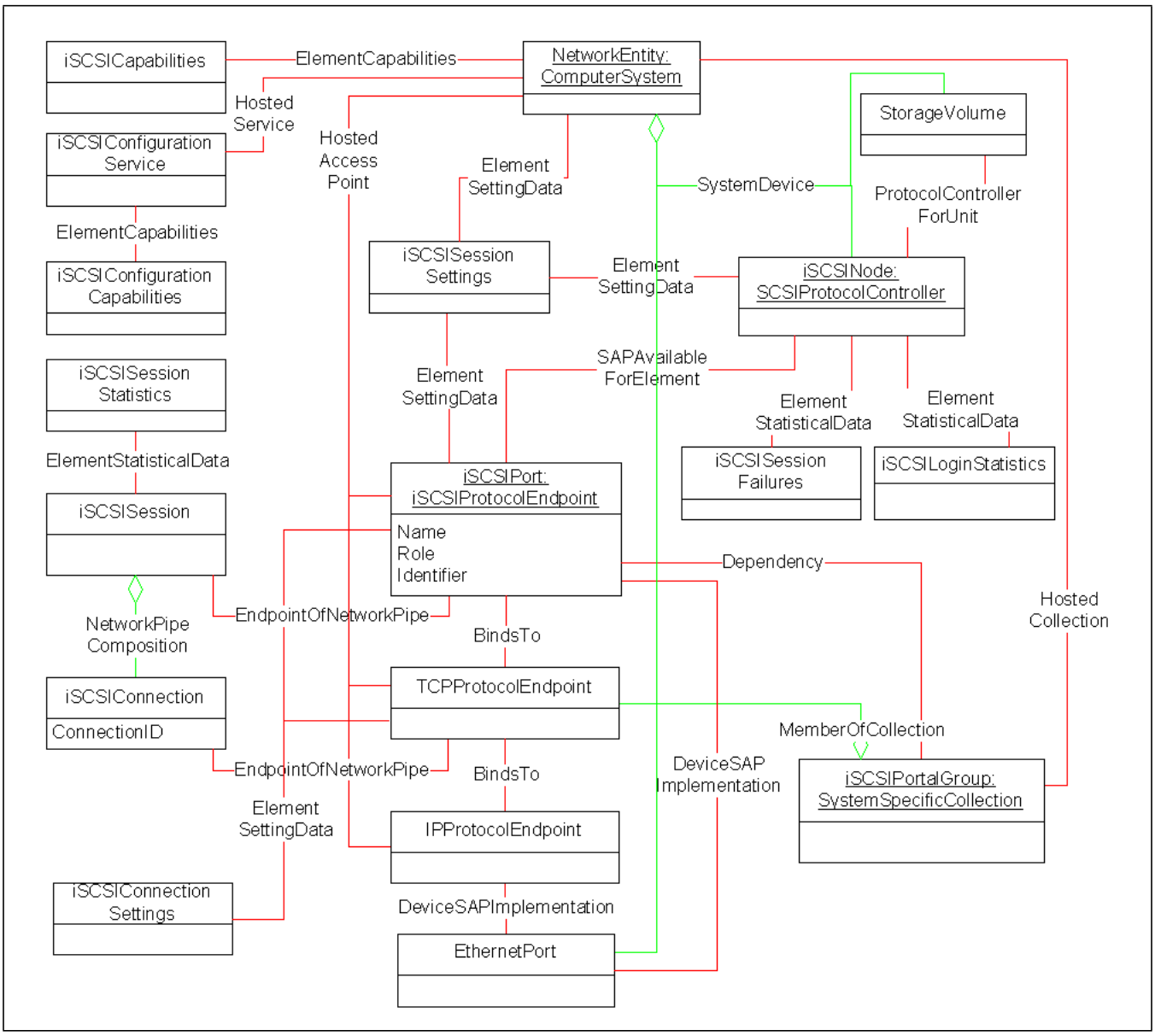

Figure 10 - iSCSI Target Ports Subprofile Instance Diagram

9.3.1 Mapping and Masking Considerations

The class SCSIProtocolController is used in the Mapping and Masking Subprofile to model a “view”, which is a set of logical devices exposed to a Target. It is in a sense a virtual SCSI device, but carries no SCSI device name when used with the other Target Ports subprofiles such as the FC Target Port Subprofile. In fact the class is even not part of these subprofiles.

The iSCSI Target Ports Subprofile however uses SCSIProtocolController to model the iSCSI Node which is the SCSI Device as defined in the SAM specification. It has a SCSI device name which is the iSCSI Node Name. Thus the presence of instances of SCSIProtocolController with this subprofile has multiple meanings. Whereas there may be no instances of SCSIProtocolController with other Target Port subprofiles until created as views by the Mapping and Masking method ExposePaths, instances of

SCSIProtocolControllers as iSCSINodes can be brought into existence by the iSCSI method CreateiSCSINode. The instances can then be used as inputs to ExposePaths to grant access by Targets to logical devices through the Node. This initial SCSIProtocolController that was created as a Node will be the first view. Additional “view” ProtocolControllers created by ExposePaths would carry the same iSCSI Node name to convey that they represent the same underlying Node.

9.3.2 Settings

An iSCSI Session is established between a Target Port and a Target Port through the establishment of an initial iSCSI Connection, which happens during the “Leading” Login. At this time the operational properties for the Session are negotiated and also the operational properties for the initial Connection. Additional Connections for the Session are established through subsequent logins. For many operational properties both the Target and Target have settings that specify the starting position for the negotiation process. The settings for negotiating Session-wide operational properties (found in iSCSISession) are in iSCSISessionSettings. Likewise the settings for negotiating Connection level operational properties (found in iSCSI Connection) are in iSCSIConnectionSettings. For example, iSCSISessionSettings contains the property MaxConnectionsPerSession, which is the value that the local system (which in this subprofile is the Target) would like to use for Session. When the leading login is complete the actual value agreed upon with the Target is in the property MaxConnectionsPerSession in iSCSI Session.

Different implementations may scope the settings classes differently.

iSCSISessionSettings can be associated to any one of the following classes:

- iSCSIProtocolEndpoint: The Settings apply to Sessions created on the iSCSI Port represented by the iSCSIProtocolEndpoint.
- SCSIProtocolController: The Settings apply to Sessions created on all iSCSIProtocolEndpoint belonging to the iSCSI Node represented by the SCSIProtocolController.
- ComputerSystem: The Settings apply to Sessions created on all iSCSIProtocolEndpoints belonging to all SCSIProtocolControllers belonging to the ComputerSystem.

iSCSIConnectionSettings can be associated to any one of the following classes:

- TCPProtocolEndpoint: The Settings apply to each Connection created using the Network Portal represented by the TCPProtocolEndpoint, regardless of which iSCSIProtocolEndpoint owns the Session that the Connection belongs to.
- iSCSIProtocolEndpoint: The Settings apply to Connections using NetworkPortals to which the iSCSIProtocolEndpoint is bound and belonging to Sessions on that same iSCSIProtocolEndpoint.

EXPERIMENTAL

NOTE The support on iSCSI Session is conditional, which means it is only supported when ‘iSCSI Session’ is included in iSCSICapabilities.SupportedFeatures.

EXPERIMENTAL

9.3.3 Durable Names and Correlatable IDs of the Subprofile

The Name property for the iSCSI node (SCSIProtocolController) shall be a compliant iSCSI name as described in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5,* 7.8 "iSCSI Names". NameFormat shall be set to “iSCSI Name”.

The Name property for iSCSIProtocolEndpoint shall be a compliant iSCSI name as described in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5,* 7.8 "iSCSI Names"ConnectionType shall be set to “iSCSI”.

9.4 Health and Fault Management

Table 27 defines the SMI-S-defined meanings of the OperationalStatus property for EthernetPort used in the SB Target Port Profile.

Table 27 - EthernetPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

9.5 Supported Subprofiles and Packages

None

9.6 Methods of this Subprofile

The iSCSIConfigurationService provides the following methods that allow a client to manipulate iSCSIProtocolEndpoints in an iSCSI Target Node. The class iSCSIProtocolController models the iSCSI Target Port. The instance of the service is scoped by an instance of ComputerSystem that represents that Network Entity. The capabilities of this service are defined in the companion class iSCSIConfigurationCapabilities.

9.6.1 CreateiSCSINode

This method creates an iSCSI Node in the form of an instance of SCSIProtocolController. As part of the creation process a SystemDevice association is created between the new SCSIProtocolController and the scoping Network Entity (ComputerSystem) hosting this service.

CreateiSCSINode

IN, string **Alias**,

The iSCSI Alias for the new Node.

OUT, SCSIProtocolController REF **iSCSINode,**

A reference to the new SCSIProtocolController that is created.

9.6.1.1 Return Values

Success

Not Supported

Unspecified Error

Timeout

Failed

Node Creation Not Supported

Alias in use by Other Node

9.6.1.2 Created Instances

SCSIProtocolController

SystemDevice

9.6.1.3 Deleted Instances

None

9.6.1.4 Modified Instances

None

9.6.2 DeleteiSCSINode

The method deletes an instance of SCSIProtocolController representing an iSCSI Node and all associations in which this SCSIProtocolController is referenced. If Sessions are active on iSCSIProtocolEndpoints belonging to this Node an error will be returned. If no Sessions are active the scoped iSCSIProtocolEndpoints will be deleted.

DeleteiSCSINode

IN, SCSIProtocolController REF **iSCSINode**

The SCSIProtocolController to be deleted.

9.6.2.1 Return Values

Success

Not Supported

Unspecified Error

Timeout

Failed

Invalid Parameter

SCSIProtocolController Non-existent

Sessions Active on Node Ports

9.6.2.2 Created Instances

None

9.6.2.3 Deleted Instances

SCSIProtocolController

SystemDevice

iSCSIProtocolEndpoint

HostedAccessPoint

SAPAvailableForElement

BindsTo

9.6.2.4 Modified Instances

None

9.6.3 CreateiSCSIProtocolEndpoint

This method creates an iSCSI Port in the form of an instance of iSCSIProtocolEndpoint. As part of the creation process the iSCSIProtocolEndpoint is 'bound to' the underlying TCPProtocolEndpoints which are specified as inputs by creating instances of the BindsTo association between the new instance and those instances. In addition, an instance of SAPAvailableForElement is created between the specified SCSIProtocolController and the new instance of iSCSIProtocolEndpoint.

CreateiSCSIProtocolEndpoint

IN, SCSIProtocolController REF **iSCSINode**,

The SCSIProtocolController instance representing the iSCSI Node that will contain the iSCSI Port.

IN, uint16 **Role**,

For iSCSI, each iSCSIProtocolEndpoint acts as either a target or a Target endpoint. This property indicates which role this iSCSIProtocolEndpoint implements.

IN, string **Identifier**,

The Identifier shall contain the Target Portal Group Tag (TGPT). Each iSCSIProtocolEndpoint (iSCSI port) associated to a common SCSIProtocolController (iSCSI node) has a unique Identifier. This field is a string that contains 12 hexadecimal digits. If the property IdentifierSelectionSupported in class iSCSIConfigurationCapabilities is false, this parameter shall be set to NULL.

IN, ProtocolEndpoint REF **NetworkPortals[],**

An Array of References to TCPProtocolEndpoints representing Target Network Portals. The TCPProtocolEndpoints specified each shall be associated to an instance of IPProtocolEndpoint via a BindsTo association in order to provide the Target Network Portal functionality. The selected Portal endpoints shall be from the same SystemSpecificCollection, which represents a Portal Group.

OUT, iSCSIProtocolEndpoint REF **iSCSIPort,**

A reference to the new iSCSIProtocolEndpoint that is created.

9.6.3.1 Return Values

Success

Not Supported

Unspecified Error

Timeout

Failed

SCSIProtocolController Non-existent

Role Not Supported By Specified SCSIProtocolController

Identifier In Use, Not Unique

Identifier Selection Not Supported

ProtocolEndpoint Non-Existent

TCPProtocolEndpoint Not Bound To Underlying IPProtocolEndpoint

TCPProtocolEndpoint In Use By Other iSCSIProtocolEndpoint In Same Target SCSIProtocolController.

ProtocolEndpoints Not From Same Endpoint Collection

9.6.3.2 Created Instances

iSCSIProtocolEndpoint

HostedAccessPoint

SAPAvailableForElement

BindsTo

9.6.3.3 Deleted Instances

None

9.6.3.4 Modified Instances

None

9.6.4 DeleteiSCSIProtocolEndpoint

The method deletes an instance of iSCSIProtocolEndpoint and all associations in which this iSCSIProtocolEndpoint is referenced.

DeleteiSCSIProtocolEndpoint

IN, iSCSIProtocolEndpoint REF **iSCSIPort**

The iSCSIProtocolEndpoint to be deleted.

9.6.4.1 Return Values

Success

Not Supported

Unspecified Error

Timeout

Failed

Invalid Parameter

Endpoint Non-existent

9.6.4.2 Created Instances

None

9.6.4.3 Deleted Instances

iSCSIProtocolEndpoint

HostedAccessPoint

SAPAvailableForElement

BindsTo

9.6.4.4 Modified Instances

None

9.6.5 BindiSCSIProtocolEndpoint

This method provides for modification of an existing iSCSI Port by associating a TCPProtocolEndpoint representing a Target Network Portal to the iSCSIProtocolEndpoint. The association is persisted as an instance of BindsTo. The selected Portal endpoint shall be from the same SystemSpecificCollection, which represents a Portal Group, as those endpoints currently bound to the iSCSIProtocolEndpoint.

This action is intended to be reversed by the use of the intrinsic method **'DeleteInstance'**.

BindiSCSIProtocolEndPoint

IN, iSCSIProtocolEndpoint REF **iSCSIPort**,

A reference to the iSCSIProtocolEndpoint

IN, ProtocolEndpoint REF **NetworkPortal**

An instance of TCPProtocolEndpoint representing the Network Portal to be added

9.6.5.1 Return Values

Success

Not Supported

Unspecified Error

Timeout

Failed

Invalid Parameter

ProtocolEndpoint Non-Existent

TCPProtocolEndpoint Not Bound To Underlying IPProtocolEndpoint

ProtocolEndpoint In Use By Other iSCSIProtocolEndpoint In Same Target SCSIProtocolController

ProtocolEndpoint Not From Same Endpoint Collection

9.6.5.2 Created Instances

BindsTo

9.6.5.3 Deleted Instances

None

9.6.5.4 Modified Instances

None

9.7 Client Considerations and Recipes

9.7.1 Discover the iSCSI Target Port capabilities.

```
// DESCRIPTION
```

```
// Discover the iSCSI Target Port capabilities.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The ComputerSystem representing the target system of interest has been
// previously identified and defined in the $NetworkEntity-> variable.

// MAIN
// Step 1. Locate the instance of CIM_iSCSICapabilities associated to the
// target ComputerSystem.
$iSCSICapabilities[] = Associators($NetworkEntity->,
     “CIM_ElementCapabilities”,
     “CIM_iSCSICapabilities”,
     “ManagedElement”,
     “Capabilities”,
     {“MinimumSpecificationVersionSupported”,
     “MaximumSpecificationVersionSupported”,
     “AuthenticationMethodsSupported”,
     “SupportedFeatures”})

if ($iSCSICapabilities[] == null || $iSCSICapabilities[].length != 1) {
    <ERROR! The iSCSI capabilities could not be found>
}
$Capabilities = $iSCSICapabilities[0]
```

9.7.2 Identify the iSCSI Nodes in a target system.

```
// DESCRIPTION
//
// Identify the iSCSI Nodes in a target system.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The ComputerSystem representing the Network Entity of interest has been
// previously identified and defined in the $NetworkEntity-> variable.

// MAIN
// Step 1. Locate the instances of CIM_SCSIProtocolController with a NameFormat
// property value of “iSCSI Name”.
$ProtocolControllers[] = Associators($NetworkEntity->,
     “CIM_SystemDevice”,
     “CIM_SCSIProtocolController”,
     “GroupComponent”,
     “PartComponent”,
     false,
     false,
     {“Name”, “NameFormat”})

// Step 2. Locate the SCSIProtocolControllers that represent the iSCSI Nodes.
$iSCSINodes[]
```

```
#index = 0
for (#i in $ProtocolControllers[]) {
    if ($ProtocolControllers[#i].NameFormat == “iSCSI Name”) {
     // Filter out SCSIProtocolControllers previously encountered.
     if (!contains($ProtocolControllers[#i].Name, #NodeNames[])) {
         #NodeNames[#index] = $ProtocolControllers[#i].Name
         $iSCSINodes[#index++] = $ProtocolControllers[#i]
     }
    }
}
<EXIT: $Nodes[] contains the results>
```

9.7.3 Identify the iSCSI Ports on an given iSCSI node.

```
// DESCRIPTION
// Identify the iSCSI Ports on an given iSCSI node.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The SCSIProtocolController representing an iSCSI Node of interest has
// been previously identified and defined in the $iSCSINode-> variable.

// This function returns the instance(s) of iSCSI ports on the specified
// iSCSI node, or null if none are found.
sub $iSCSIPorts[] getiSCSIPortsOnNode($Node->) {

    // Step 1. Locate the iSCSI Ports, which are represented by instances of
    // iSCSIProtocolEndpoint, on the iSCSI Node of interest.
    $iSCSIPorts[] = Associators($iSCSINode->,
         “CIM_SAPAvailableForElement”,
         “CIM_iSCSIProtocolEndpoint”,
         “ManagedElement”,
         “AvailableSAP”,
         false,
         false,
         {“Name”, “Identifier”, “Role”})

    if ($iSCSIPorts[].length == 0) {
     return (null)
    }
    return ($iSCSIPorts[])
}

// MAIN
$iSCSIPorts[] = &getiSCSIPortsOnNode($iSCSINode->)
```

9.7.4 Identify the iSCSI sessions existing on an iSCSI node.

```
// DESCRIPTION
```

```
// Identify the iSCSI sessions existing on an iSCSI node.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.’iSCSI Session’ is included in iSCSICapabilities.SupportedFeatures.
// 2.The SCSIProtocolController representing the iSCSI Node of interest has
// been previously identified and defined in the $iSCSINode-> variable

// Step 1. Retrieve the CIM_iSCSIProtocolEndpoints for an
// CIM_SCSIProtocolController representing a node.
$iSCSIPorts[] = @getiSCSIPortsOnNode($iSCSINode->)
if ($iSCSIPorts[] == null) {
    <ERROR! No iSCSI ports located on the specified iSCSI node>
}

// Step 2. Retrieve the iSCSI session associated with each iSCSI port.
$iSCSISessions[]
#index = 0
#PropList[] = {“Directionality”, “SessionType”, “TSIH”, “EndPointName”,
     “CurrentConnections”, “InitialR2T”, “ImmediateData”,
     “MaxOutstandingR2T”, “MaxUnsolicitedFirstDataBurstLength”,
     “MaxDataBurstLength”, “AuthenticationMethodUsed”,
     “DataSequenceInOrder”, “DataPDUInOrder”, “ErrorRecoveryLevel”}
for (#i in $iSCSIPorts[]) {
    $Sessions[] = Associators($iSCSIPorts[#i].getObjectPath(),
     “CIM_EndpointOfNetworkPipe”,
     “CIM_iSCSISession”,
     “Antecedent”,
     “Dependent”,
     #PropList[])
    if ($Sessions[] != null && $Sessions[].length == 1) {
     $iSCSISessions[#index++] = $Sessions[0]
    }
}
<EXIT: $iSCSISessions[] contains the iSCSI Sessions>
```

9.7.5 Create an iSCSI Target Node on an iSCSI Network Entity

```
// DESCRIPTION
// Create an iSCSI Target Node on an iSCSI Network Entity
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The ComputerSystem representing the Network Entity of interest has been
// previously identified and defined in the $NetworkEntity-> variable.

// MAIN
// Step 1. Locate the CIM_iSCSIConfigurationService hosted by the System.
$iSCSIConfigurationService->[] = AssociatorNames($NetworkEntity->,
     “CIM_HostedService”,
```

```
     “CIM_iSCSIConfigurationService”,
     “Antecedent”,
     “Dependent”)
if ($iSCSIConfigurationService->[] == null ||
     $iSCSIConfigurationService->[].length == 0) {
    <ERROR! Required iSCSI Configuration Service not available>
}

// Step 2. Examine the capabilities to determine if Node creation is supported.
$ConfigurationCapabilities[] = Associators($iSCSIConfigurationService->[0],
     “CIM_ElementCapabilities”,
     “CIM_iSCSIConfigurationCapabilities”,
     “ManagedElement”,
     “Capabilities”,
     false,
     false,
     {“iSCSINodeCreationSupported  “})
if ($ConfigurationCapabilities[] == null ||
     $ConfigurationCapabilities[].length == 0) {
    <ERROR! Required iSCSI Configuration Service capabilities not available>
}

// Step 3. Create the iSCSI Target Node if supported by the device.
if ($ConfigurationCapabilities[0].iSCSINodeCreationSupported  == true) {
    %InArguments[“Alias”] = “Some Target Alias”
    #ReturnValue = invokeMethod($iSCSIConfigurationService->[0],
         “CreateiSCSINode”,
         %InArguments[],
         %OutArguments[])

    if (#ReturnValue == 0) {
     $NewNode-> = $OutArguments[“iSCSINode”]
     <EXIT: The node was created>
    } else {
     <EXIT: The method returned an error; the Node was not created>
    }
} else {
    <EXIT: Node Creation is not supported>
}
```

9.7.6 Create an iSCSI Target Port on an iSCSI target node.

```
// DESCRIPTION
// This recipe describes how to create an iSCSI Target Port on an iSCSI target
// node.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The object name for the ComputerSystem representing the Network Entity of
```

```
// interest has been previously identified and defined in the $NetworkEntity->
// variable.
// 2. The object name for the SCSIProtocolController representing the iSCSI Node
// within which to create the iSCSI Port has been identified and defined in the //
                          $Node-> variable.
// 3. The object names for one or more TCPProtocolEndpoints representing Target
// Network Portals have been previously identified and defined in the
// Portals->[] array variable.

// MAIN
// Step 1. Find a CIM_iSCSIConfigurationService associated to ComputerSystem
// by HostedService.
$iSCSIConfigurationService->[] = AssociatorNames($NetworkEntity->,
     “CIM_HostedService”,
     “CIM_iSCSIConfigurationService”,
     “Antecedent”,
     “Dependent”)

// Step 2. Examine the associated CIM_iSCSIConfigurationCapabilities to
// determine if Target Port manipulation is supported.
$ConfigurationCapabilities[] = Associators($iSCSIConfigurationService->[0],
     “CIM_ElementCapabilities”,
     “CIM_iSCSIConfigurationCapabilities”,
     “ManagedElement”,
     “Capabilities”,
     false,
     false,
     {“iSCSIProtocolEndpointCreationSupported”})

// Step 3. Given an instance of CIM_SCSIProtocolController representing a
// Node($Node->), and one or more TCPProtocolEndpoints representing Target
// Network Portals(Portals->[]), invoke the method CreateiSCSIProtocolEndpoint
// to create the iSCSIProtocolEndpoint.
if ($ConfigurationCapabilities[0].iSCSIProtocolEndpointCreationSupported == true)
                          {

    %InArguments[“iSCSINode”] = $Node->
    %InArguments[“Role”] = 3// “Target”
    %InArguments[“NetworkPortals”] = Portals->[]

    #ReturnValue = InvokeMethod($iSCSIConfigurationService->[0],
         “CreateiSCSIProtocolEndpoint”,
         %InArguments[],
         %OutArguments[])

    if (#ReturnValue == 0) {
     $NewiSCSIProtocolEndpoint-> = $OutArguments[“iSCSIPort”]
     <EXIT: The ProtocolEndpoint was created>
```

```
    } else {
     <EXIT: The method returned an error; the ProtocolEndpoint was not created>
    }
} else {
    <EXIT: iSCSIProtocolEndpoint creation is not supported>
}
```

9.7.7 Add a Network Portal to a Target Port.

```
// DESCRIPTION
// This recipe describes how to add a Network Portal to a Target Port.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The object name for the ComputerSystem representing the Network Entity of
// interest has been previously identified and defined in the $NetworkEntity->
// variable.
// 2. The object name for the instance of iSCSIProtocolEndpoint representing a
// Port has been previously identified and defined in the $iSCSIPort-> variable.
// 3. The object name for the instance of TCPProtocolEndpoint representing a
// Target Network Portal has been previously identified and defined in the
// $Portal-> variable.

// MAIN
// Step 1. Find a CIM_iSCSIConfigurationService associated to ComputerSystem by //
                            HostedService.
$iSCSIConfigurationService->[] = AssociatorNames($NetworkEntity->,
     “CIM_HostedService”,
     “CIM_iSCSIConfigurationService”,
     “Antecedent”,
     “Dependent”)

// Step 2. Examine the associated CIM_iSCSIConfigurationCapabilities to
// determine if Target Port manipulation is supported.
$ConfigurationCapabilities[] = Associators($iSCSIConfigurationService->[0],
     “CIM_ElementCapabilities”,
     “CIM_iSCSIConfigurationCapabilities”,
     “ManagedElement”,
     “Capabilities”,
     false,
     false,
     {“iSCSIProtocolEndpointCreationSupported”})

// Step 3. Given an instance of CIM_iSCSIProtocolEndpoint representing a
// Port (iSCSIPort->), and an instance of TCPProtocolEndpoint representing a
// Target Network Portal($Portal->), invoke BindiSCSIProtocolEndpoint().
if ($ConfigurationCapabilities[0].iSCSIProtocolEndpointCreationSupported == true)
                            {
```

```
    %InArguments[“iSCSIPort”] = $iSCSIPort->
    %InArguments[“NetworkPortal”] = $Portal->

    #ReturnValue = invokeMethod($iSCSIConfigurationService->[0],
         “BindiSCSIProtocolEndpoint”,
         %InArguments[],
         %OutArguments[])

    if (#ReturnValue == 0) {
     <EXIT: The ProtocolEndpoint was modified>
    } else {
     <EXIT: The method returned an error; the ProtocolEndpoint was not modified>
    }
} else {
    <EXIT: iSCSIProtocolEndpoint modification is not supported>
}
```

9.7.8 Determine the health of Nodes in a target system.

```
//
// DESCRIPTION
// Recipe ISCSI_TRGT08:
// Determine the health of Nodes in a target system.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The object name for the SCSIProtocolController representing
// the iSCSI Node of interest has been previously identified and
// defined in the $iSCSINode-> variable

//
// Step 1.
// Given an instance of CIM_SCSIProtocolController($iSCSINode->) ,
// get the instances of CIM_iSCSISessionFailures and
// CIM_iSCSILoginStatistics associated by ElementStatisticalData.
//
$SessionFailures[] = Associators(
     $iSCSINode->,
     “CIM_ElementStatisticalData”,
     “CIM_iSCSISessionFailures”,
     “ManagedElement”,
     “Stats” );

$LoginStatistics[] = Associators(
     $iSCSINode->,
     “CIM_ElementStatisticalData”,
     “CIM_iSCSILoginStatistics”,
     “ManagedElement”,
     “Stats” );
```

```
<EXIT: The statistics are in $SessionFailures[0] and $LoginStatistics[0] >
```

9.7.9 Determine the health of a Session on a target system.

```
//
// DESCRIPTION
// Recipe ISCSI_TRGT09:
// Determine the health of a Session on a target system.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.'iSCSI Session' is included in iSCSICapabilities.SupportedFeatures.
// 2.The object name for the iSCSISession of interest has been
// previously identified and defined in the $iSCSISession-> variable.

// Step 1.
// Given an instance of CIM_iSCSISession,
// get the instance of CIM_iSCSISessionStatistics
// associated by ElementStatisticalData.
//
$SessionStatistics[] = Associators(
     $iSCSISession->,
     "CIM_ElementStatisticalData",
     "CIM_iSCSISessionStatistics",
     "ManagedElement",
     "Stats" );


<EXIT: The statistics are in $SessionStatistics[0]>
```

9.7.10 Configure the default settings for Sessions created in a target computer system.

```
//
// DESCRIPTION
// Recipe ISCSI_TRGT10:
// Configure the default settings for Sessions created in a target
// computer system.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.'iSCSI Session' is included in iSCSICapabilities.SupportedFeatures.
// 2.The object name for the SCSIProtocolController representing the
// iSCSI Node of interest has been previously identified and defined
// in the $iSCSINode-> variable.

//
// Step 1.
// Find and modify an instance of CIM_iSCSISessionSettings associated
// to a ComputerSystem, CIM_SCSIProtocolController, or
// CIM_iSCSIProtocolEndpoint.
//
```

```
$SessionSettings[] = Associators(
     $iSCSIProtocolEndpoint->,
     “CIM_ElementSettingData”,
     “CIM_iSCSISessionSettings”,
     “ManagedElement”,
     “SettingData” );


#MaxConnectionsPerSession = 4;

$SessionSettings[0].MaxConnectionsPerSession = #MaxConnectionsPerSession;

$ModifyInstance(
     $SessionSettings[0],
     false,
     { “MaxConnectionsPerSession” } );

<EXIT: Success>
```

9.7.11 Configure default settings for Connections on Network Portals used by an iSCSIProtocolEndpoint.

```
//
// DESCRIPTION
// Recipe ISCSI_TRGT11:
// Configure the default settings for iSCSI Connections created on
// Network Portals used by an iSCSIProtocolEndpoint.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The object name for the iSCSI Session of interest has been
// previously identified and defined in the $iSCSISession->
// variable

//
// Step 1.
// Find and modify an instance of CIM_iSCSIConnectionSettings
// associated to a iSCSIProtocolEndpoint($iSCSIProtocolEndpoint->).
//
$ConnectionSettings[] = Associators(
     $iSCSIProtocolEndpoint->,
     “CIM_ElementSettingData”,
     “CIM_iSCSIConnectionSettings”,
     “ManagedElement”,
     “SettingData” );


#MaxRecvDataSegLength = 4096;
```

```
$ConnectionSettings[0].MaxReceiveDataSegmentLength = #MaxRecvDataSegLength;

$ModifyInstance(
     $ConnectionSettings[0],
     false,
     { “MaxReceiveDataSegmentLength” } );`

<EXIT: Success>
```

9.7.12 Get the statistics for a Session on a target system

The statistics are properties in the same class as the health information; see 9.7.9.

9.7.13 Configure Enable/disable header and data digest

See 9.7.11.

9.8 CIM Elements

Table 28 describes the CIM elements for iSCSI Target Ports.

Table 28 - CIM Elements for iSCSI Target Ports

Element Name	Requirement	Description
9.8.1 CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)	Mandatory	
9.8.2 CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)	Mandatory	
9.8.3 CIM_ConcreteDependency	Mandatory	
9.8.4 CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)	Optional	
9.8.5 CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)	Optional	
9.8.6 CIM_ElementCapabilities (iSCSIConfigurationCapabilities to System)	Mandatory	
9.8.7 CIM_ElementCapabilities (iSCSIConfigurationCapabilities to iSCSIConfigurationService)	Conditional	Conditional requirement: Active configuration is supported.
9.8.8 CIM_ElementSettingData (iSCSIConnectionSettings to TCPProtocolEndpoint)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.9 CIM_ElementSettingData (iSCSIConnectionSettings to iSCSIProtocolEndpoint)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.10 CIM_ElementSettingData (iSCSISessionSettings to SCSIProtocolController)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.11 CIM_ElementSettingData (iSCSISessionSettings to System)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 28 - CIM Elements for iSCSI Target Ports

Element Name	Requirement	Description
9.8.12 CIM_ElementSettingData (iSCSISessionSettings to iSCSIProtocolEndpoint)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.13 CIM_ElementStatisticalData (iSCSILoginStatistics to SCSIProtocolController)	Mandatory	
9.8.14 CIM_ElementStatisticalData (iSCSISessionFailures to SCSIProtocolController)	Optional	
9.8.15 CIM_ElementStatisticalData (iSCSISessionStatistics to iSCSISession)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.16 CIM_EndpointOfNetworkPipe (iSCSIConnection to TCPProtocolEndpoint)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.17 CIM_EndpointOfNetworkPipe (iSCSISession to iSCSIProtocolEndpoint)	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.18 CIM_EthernetPort	Optional	
9.8.19 CIM_HostedAccessPoint (System to IPProtocolEndpoint)	Mandatory	
9.8.20 CIM_HostedAccessPoint (System to TCPProtocolEndpoint)	Mandatory	
9.8.21 CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)	Mandatory	
9.8.22 CIM_HostedCollection	Mandatory	
9.8.23 CIM_HostedService	Optional	
9.8.24 CIM_IPProtocolEndpoint	Mandatory	
9.8.25 CIM_MemberOfCollection	Optional	
9.8.26 CIM_NetworkPipeComposition	Optional	
9.8.27 CIM_SAPAvailableForElement	Mandatory	
9.8.28 CIM_SCSIProtocolController	Mandatory	
9.8.29 CIM_SystemDevice (System to EthernetPort)	Mandatory	This association links all EthernetPorts to the scoping system.
9.8.30 CIM_SystemDevice (System to SCSIProtocolController)	Mandatory	This association links SCSIProtocolControllers to the scoping system.
9.8.31 CIM_SystemSpecificCollection	Optional	
9.8.32 CIM_TCPProtocolEndpoint	Mandatory	
9.8.33 CIM_iSCSICapabilities	Mandatory	
9.8.34 CIM_iSCSIConfigurationCapabilities	Conditional	Conditional requirement: Active configuration is supported.
9.8.35 CIM_iSCSIConfigurationService	Optional	
9.8.36 CIM_iSCSIConnection	Optional	
9.8.37 CIM_iSCSIConnectionSettings	Optional	

Table 28 - CIM Elements for iSCSI Target Ports

Element Name	Requirement	Description
9.8.38 CIM_iSCSILoginStatistics	Optional	
9.8.39 CIM_iSCSIProtocolEndpoint	Mandatory	
9.8.40 CIM_iSCSISession	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.41 CIM_iSCSISessionFailures	Optional	
9.8.42 CIM_iSCSISessionSettings	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').
9.8.43 CIM_iSCSISessionStatistics	Optional	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_EthernetPort	Optional	Create EthernetPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_EthernetPort AND SourceInstance.CIM_EthernetPort::OperationalStatus <> PreviousInstance.CIM_EthernetPort::OperationalStatus	Optional	CQL -Modify EthernetPort.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_EthernetPort	Optional	Delete EthernetPort.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_iSCSIProtocolEndpoint	Mandatory	Create iSCSIProtocolEndpoint.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_iSCSIProtocolEndpoint	Mandatory	Delete SCSIProtocolEndpoint.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SCSIProtocolController	Mandatory	Create SCSIProtocolController.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SCSIProtocolController	Mandatory	Delete iSCSIProtocolController.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_iSCSISession	Optional	Create iSCSISession.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_iSCSISession AND SourceInstance.CIM_iSCSISession::CurrentConnections <> PreviousInstance.CIM_iSCSISession::CurrentConnections	Optional	CQL -Modify iSCSISession.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_iSCSISession AND SourceInstance.CurrentConnections <> PreviousInstance.CurrentConnections	Optional	Deprecated WQL -Modify iSCSISession.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_iSCSISession	Optional	Delete iSCSISession.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_iSCSIConnection	Optional	Create iSCSIConnection.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_iSCSIConnection	Optional	Delete iSCSIConnection.

Table 28 - CIM Elements for iSCSI Target Ports

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_iSCSISessionSettings	Conditional	Conditional requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session'). Modify iSCSISessionSettings.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_iSCSIConnectionSettings	Optional	Modify iSCSIConnectionSettings.

9.8.1 CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 29 describes class CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint).

Table 29 - SMI Referenced Properties/Methods for CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.8.2 CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 30 describes class CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint).

Table 30 - SMI Referenced Properties/Methods for CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.8.3 CIM_ConcreteDependency

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: Mandatory

Table 31 describes class CIM_ConcreteDependency.

Table 31 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.8.4 CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 32 describes class CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint).

Table 32 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.5 CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 33 describes class CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint).

Table 33 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.6 CIM_ElementCapabilities (iSCSIConfigurationCapabilities to System)

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 34 describes class CIM_ElementCapabilities (iSCSIConfigurationCapabilities to System).

Table 34 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (iSCSIConfigurationCapabilities to System)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

9.8.7 CIM_ElementCapabilities (iSCSIConfigurationCapabilities to iSCSIConfigurationService)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Active configuration is supported.

Table 35 describes class CIM_ElementCapabilities (iSCSIConfigurationCapabilities to iSCSIConfigurationService).

Table 35 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (iSCSIConfigurationCapabilities to iSCSIConfigurationService)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

9.8.8 CIM_ElementSettingData (iSCSIConnectionSettings to TCPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 36 describes class CIM_ElementSettingData (iSCSIConnectionSettings to TCPProtocolEndpoint).

Table 36 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSIConnectionSettings to TCPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
SettingData		Mandatory	
ManagedElement		Mandatory	

9.8.9 CIM_ElementSettingData (iSCSIConnectionSettings to iSCSIProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 37 describes class CIM_ElementSettingData (iSCSIConnectionSettings to iSCSIProtocolEndpoint).

Table 37 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSIConnectionSettings to iSCSIProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
SettingData		Mandatory	
ManagedElement		Mandatory	

9.8.10 CIM_ElementSettingData (iSCSISessionSettings to SCSIProtocolController)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 38 describes class CIM_ElementSettingData (iSCSISessionSettings to SCSIProtocolController).

Table 38 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSISessionSettings to SCSIProtocolController)

Properties	Flags	Requirement	Description & Notes
SettingData		Mandatory	
ManagedElement		Mandatory	

9.8.11 CIM_ElementSettingData (iSCSISessionSettings to System)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 39 describes class CIM_ElementSettingData (iSCSISessionSettings to System).

Table 39 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSISessionSettings to System)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

9.8.12 CIM_ElementSettingData (iSCSISessionSettings to iSCSIProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 40 describes class CIM_ElementSettingData (iSCSISessionSettings to iSCSIProtocolEndpoint).

Table 40 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSISessionSettings to iSCSIProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
SettingData		Mandatory	
ManagedElement		Mandatory	

9.8.13 CIM_ElementStatisticalData (iSCSILoginStatistics to SCSIProtocolController)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 41 describes class CIM_ElementStatisticalData (iSCSILoginStatistics to SCSIProtocolController).

Table 41 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (iSCSILoginStatistics to SCSIProtocolController)

Properties	Flags	Requirement	Description & Notes
Stats		Mandatory	
ManagedElement		Mandatory	

9.8.14 CIM_ElementStatisticalData (iSCSISessionFailures to SCSIProtocolController)

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: Optional

Table 42 describes class CIM_ElementStatisticalData (iSCSISessionFailures to SCSIProtocolController).

Table 42 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (iSCSISessionFailures to SCSIProtocolController)

Properties	Flags	Requirement	Description & Notes
Stats		Mandatory	
ManagedElement		Mandatory	

9.8.15 CIM_ElementStatisticalData (iSCSISessionStatistics to iSCSISession)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 43 describes class CIM_ElementStatisticalData (iSCSISessionStatistics to iSCSISession).

Table 43 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (iSCSISessionStatistics to iSCSISession)

Properties	Flags	Requirement	Description & Notes
Stats		Mandatory	
ManagedElement		Mandatory	

9.8.16 CIM_EndpointOfNetworkPipe (iSCSIConnection to TCPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 44 describes class CIM_EndpointOfNetworkPipe (iSCSIConnection to TCPProtocolEndpoint).

Table 44 - SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (iSCSIConnection to TCP-ProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.8.17 CIM_EndpointOfNetworkPipe (iSCSISession to iSCSIProtocolEndpoint)

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 45 describes class CIM_EndpointOfNetworkPipe (iSCSISession to iSCSIProtocolEndpoint).

Table 45 - SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (iSCSISession to iSCSI-ProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.8.18 CIM_EthernetPort

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 46 describes class CIM_EthernetPort.

Table 46 - SMI Referenced Properties/Methods for CIM_EthernetPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
PermanentAddress	CD	Mandatory	

9.8.19 CIM_HostedAccessPoint (System to IPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 47 describes class CIM_HostedAccessPoint (System to IPProtocolEndpoint).

Table 47 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to IPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.20 CIM_HostedAccessPoint (System to TCPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 48 describes class CIM_HostedAccessPoint (System to TCPProtocolEndpoint).

Table 48 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to TCPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.21 CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 49 describes class CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint).

Table 49 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.22 CIM_HostedCollection

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: Mandatory

Table 50 describes class CIM_HostedCollection.

Table 50 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.23 CIM_HostedService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 51 describes class CIM_HostedService.

Table 51 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.24 CIM_IPProtocolEndpoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 52 describes class CIM_IPProtocolEndpoint.

Table 52 - SMI Referenced Properties/Methods for CIM_IPProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
IPv4Address	CD	Optional	
IPv6Address	CD	Optional	
ProtocolIFType		Mandatory	

9.8.25 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 53 describes class CIM_MemberOfCollection.

Table 53 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

9.8.26 CIM_NetworkPipeComposition

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 54 describes class CIM_NetworkPipeComposition.

Table 54 - SMI Referenced Properties/Methods for CIM_NetworkPipeComposition

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

9.8.27 CIM_SAPAvailableForElement

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 55 describes class CIM_SAPAvailableForElement.

Table 55 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
AvailableSAP		Mandatory	

9.8.28 CIM_SCSIProtocolController

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 56 describes class CIM_SCSIProtocolController.

Table 56 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ElementName		Mandatory	iSCSI Alias.
Name	CD	Mandatory	
NameFormat		Mandatory	

9.8.29 CIM_SystemDevice (System to EthernetPort)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 57 describes class CIM_SystemDevice (System to EthernetPort).

Table 57 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to EthernetPort)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

9.8.30 CIM_SystemDevice (System to SCSIProtocolController)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 58 describes class CIM_SystemDevice (System to SCSIProtocolController).

Table 58 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIProtocolController)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

9.8.31 CIM_SystemSpecificCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 59 describes class CIM_SystemSpecificCollection.

Table 59 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	

9.8.32 CIM_TCPProtocolEndpoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 60 describes class CIM_TCPProtocolEndpoint.

Table 60 - SMI Referenced Properties/Methods for CIM_TCPProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
PortNumber		Mandatory	
ProtocolIFType		Mandatory	

9.8.33 CIM_iSCSICapabilities

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 61 describes class CIM_iSCSICapabilities.

Table 61 - SMI Referenced Properties/Methods for CIM_iSCSICapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
MinimumSpecificationVersionSupported		Mandatory	
MaximumSpecificationVersionSupported		Mandatory	
AuthenticationMethodsSupported		Mandatory	
SupportedFeatures		Mandatory	Experimental.

9.8.34 CIM_iSCSIConfigurationCapabilities

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Active configuration is supported.

Table 62 describes class CIM_iSCSIConfigurationCapabilities.

Table 62 - SMI Referenced Properties/Methods for CIM_iSCSIConfigurationCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
iSCSINodeCreationSupported		Mandatory	
iSCSIProtocolEndpointCreationSupported		Mandatory	
IdentifierSelectionSupported		Mandatory	

9.8.35 CIM_iSCSIConfigurationService

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Optional

Table 63 describes class CIM_iSCSIConfigurationService.

Table 63 - SMI Referenced Properties/Methods for CIM_iSCSIConfigurationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	

9.8.36 CIM_iSCSIConnection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 64 describes class CIM_iSCSIConnection.

Table 64 - SMI Referenced Properties/Methods for CIM_iSCSIConnection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ConnectionID		Mandatory	
MaxReceiveDataSegmentLength		Mandatory	
MaxTransmitDataSegmentLength		Mandatory	
HeaderDigestMethod		Mandatory	
OtherHeaderDigestMethod		Optional	
DataDigestMethod		Mandatory	
OtherDataDigestMethod		Optional	
ReceivingMarkers		Mandatory	
SendingMarkers		Mandatory	
ActiveiSCSIVersion		Mandatory	
AuthenticationMethodUsed		Mandatory	
MutualAuthentication		Mandatory	

9.8.37 CIM_iSCSIConnectionSettings

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 65 describes class CIM_iSCSIConnectionSettings.

Table 65 - SMI Referenced Properties/Methods for CIM_iSCSIConnectionSettings

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
MaxReceiveDataSegment Length		Mandatory	
PrimaryHeaderDigestMeth od		Mandatory	
OtherPrimaryHeaderDiges tMethod		Optional	
PrimaryDataDigestMethod		Mandatory	
OtherPrimaryDataDigestM ethod		Optional	
SecondaryHeaderDigestM ethod		Mandatory	
OtherSecondaryHeaderDi gestMethod		Optional	
SecondaryDataDigestMeth od		Mandatory	
OtherSecondaryDataDiges tMethod		Optional	
RequestingMarkersOnRec eive		Mandatory	
PrimaryAuthenticationMeth od		Mandatory	
SecondaryAuthenticationM ethod		Mandatory	

9.8.38 CIM_iSCSILoginStatistics

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 66 describes class CIM_iSCSILoginStatistics.

Table 66 - SMI Referenced Properties/Methods for CIM_iSCSILoginStatistics

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
LoginFailures		Optional	
LastLoginFailureTime		Optional	
LastLoginFailureType		Optional	
OtherLastLoginFailureType		Optional	
LastLoginFailureRemoteNodeName		Optional	
LastLoginFailureRemoteAddressType		Optional	
LastLoginFailureRemoteAddress		Optional	
SuccessfulLogins		Optional	
NegotiationLoginFailures		Optional	
AuthenticationLoginFailures		Optional	
AuthorizationLoginFailures		Optional	
LoginRedirects		Optional	
OtherLoginFailures		Optional	
NormalLogouts		Optional	
OtherLogouts		Optional	

9.8.39 CIM_iSCSIProtocolEndpoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 67 describes class CIM_iSCSIProtocolEndpoint.

Table 67 - SMI Referenced Properties/Methods for CIM_iSCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	

Table 67 - SMI Referenced Properties/Methods for CIM_iSCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
Name	CD	Mandatory	
ConnectionType		Mandatory	iSCSI.
Identifier		Mandatory	ISID or TPGT.
ProtocolIFType		Mandatory	Other.
OtherTypeDescription		Mandatory	
Role		Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target).

9.8.40 CIM_iSCSISession

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 68 describes class CIM_iSCSISession.

Table 68 - SMI Referenced Properties/Methods for CIM_iSCSISession

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
Directionality		Mandatory	
SessionType		Mandatory	
TSIH		Mandatory	
EndPointName		Mandatory	
CurrentConnections		Mandatory	
InitialR2T		Mandatory	
ImmediateData		Mandatory	
MaxOutstandingR2T		Mandatory	
MaxUnsolicitedFirstDataBurstLength		Mandatory	
MaxDataBurstLength		Mandatory	
DataSequenceInOrder		Mandatory	
DataPDUInOrder		Mandatory	
ErrorRecoveryLevel		Mandatory	
MaxConnectionsPerSession		Mandatory	
DefaultTimeToWait		Mandatory	
DefaultTimeToRetain		Mandatory	

9.8.41 CIM_iSCSISessionFailures

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 69 describes class CIM_iSCSISessionFailures.

Table 69 - SMI Referenced Properties/Methods for CIM_iSCSISessionFailures

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SessionFailures		Optional	
LastSessionFailureType		Optional	
OtherLastSessionFailureType		Optional	
LastSessionFailureRemoteNodeName		Optional	
SessionDigestFailures		Optional	
SessionConnectionTimeoutFailures		Optional	
SessionFormatErrors		Optional	

9.8.42 CIM_iSCSISessionSettings

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_iSCSICapabilities.SupportedFeatures = '3' ('iSCSI Session').

Table 70 describes class CIM_iSCSISessionSettings.

Table 70 - SMI Referenced Properties/Methods for CIM_iSCSISessionSettings

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
MaxConnectionsPerSession		Mandatory	
InitialR2TPreference		Mandatory	
ImmediateDataPreference		Mandatory	
MaxOutstandingR2T		Mandatory	

Table 70 - SMI Referenced Properties/Methods for CIM_iSCSISessionSettings

Properties	Flags	Requirement	Description & Notes
MaxUnsolicitedFirstDataBurstLength		Mandatory	
MaxDataBurstLength		Mandatory	
DataSequenceInOrderPreference		Mandatory	
DataPDUInOrderPreference		Mandatory	
DefaultTimeToWaitPreference		Mandatory	
DefaultTimeToRetainPreference		Mandatory	
ErrorRecoveryLevelPreference		Mandatory	

9.8.43 CIM_iSCSISessionStatistics

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 71 describes class CIM_iSCSISessionStatistics.

Table 71 - SMI Referenced Properties/Methods for CIM_iSCSISessionStatistics

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
CommandPDUsTransferred		Optional	
ResponsePDUsTransferred		Optional	
BytesTransmitted		Optional	
BytesReceived		Optional	
DigestErrors		Optional	
ConnectionTimeoutErrors		Optional	

EXPERIMENTAL

10 Serial Attached SCSI (SAS) Target Port Subprofile

10.1 Synopsis

Profile Name: SAS Target Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.13.1

Table 72 describes the related profiles for SAS Target Ports.

Table 72 - Related Profiles for SAS Target Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Central Class: CIM_SASPort

Scoping Class: a CIM_System in a separate autonomous profile

10.2 Description

Figure 11 illustrates the Serial Attached SCSI (SAS) Target Port. Serial Attached SCSI is a lower cost network interface for SCSI communication.

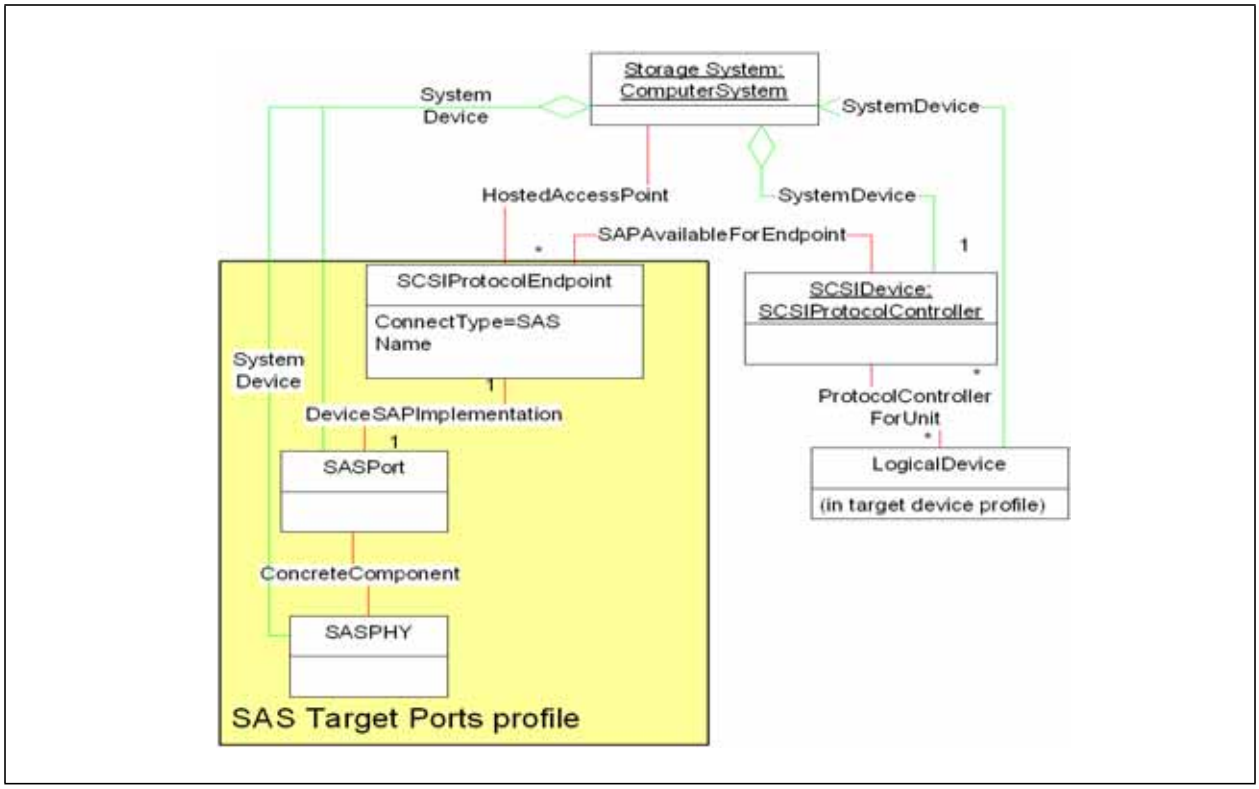

Figure 11 - Serial Attached SCSI (SAS) Target Port Instance Diagram

SCSIProtocolEndpoint.ConnectionType shall be set to “SAS”. SASPort represents the port and is connected to SCSIProtocolEndpoint by DeviceSAPImplemetation. The SASPort contains information about the speed for the bus.

The SASPHY class represents a SAS PHY. A SAS Port may have multiple associated PHYs; generally, all the PHYs are connected to the same initiator or expander and provide additional bandwidth.

10.2.1 Health and Fault Management

Table 73 describes SASPort OperationalStatus.

Table 73 - SASPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

10.3 Methods

10.3.1 Extrinsic Methods of this Subprofile

10.3.2 Intrinsic Methods of this Subprofile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

10.4 Client Considerations and Recipes

None

10.5 CIM Elements

Table 74 describes the CIM elements for SAS Target Ports.

Table 74 - CIM Elements for SAS Target Ports

Element Name	Requirement	Description
10.5.1 CIM_ConcreteComponent	Mandatory	Associates SASPort and SASPHY.
10.5.2 CIM_DeviceSAPImplementation	Mandatory	Associates SASPort and SCSIProtocolEndpoint.
10.5.3 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
10.5.4 CIM_SASPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
10.5.5 CIM_SCSIProtocolEndpoint	Mandatory	Represents management characteristics related to the SCSI command set.
10.5.6 CIM_SystemDevice (Port)	Mandatory	Associates ComputerSystem to LogicalPort.
10.5.7 CIM_SystemDevice (SAS PHY)	Mandatory	Associates ComputerSystem to SASPHY.
10.5.8 SNIA_SASPHY	Mandatory	Several SASPHYs may together be aggregated into a SAS Logical Port.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SASPort	Mandatory	CQL -Create SASPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SASPort AND SourceInstance.CIM_SASPort::OperationalStatus <> PreviousInstance.SAS_Port::OperationalStatus	Mandatory	CQL -Modify SASPort.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SASPort	Mandatory	CQL -Delete SASPort.

10.5.1 CIM_ConcreteComponent

Associates SASPort and SASPHY.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 75 describes class CIM_ConcreteComponent.

Table 75 - SMI Referenced Properties/Methods for CIM_ConcreteComponent

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	Reference to SASPHY.
GroupComponent		Mandatory	Reference to SASPort.

10.5.2 CIM_DeviceSAPImplementation

Associates SASPort and SCSIProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 76 describes class CIM_DeviceSAPImplementation.

Table 76 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to SASPort.

10.5.3 CIM_HostedAccessPoint

Associates ComputerSystem to ProtocolEndpoint. Limit to targets. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 77 describes class CIM_HostedAccessPoint.

Table 77 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in the referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.

10.5.4 CIM_SASPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 78 describes class CIM_SASPort.

Table 78 - SMI Referenced Properties/Methods for CIM_SASPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction (overridden)		Mandatory	Shall be 2 (Front-end Only).
PortType (overridden)		Mandatory	Shall be 94 (SAS).
PermanentAddress (added)		Mandatory	SAS Address. Shall be 16 un-separated upper case hex digits.

10.5.5 CIM_SCSIProtocolEndpoint

Represents management characteristics related to the SCSI command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 79 describes class CIM_SCSIProtocolEndpoint.

Table 79 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).

Table 79 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
ConnectionType (added)		Mandatory	Shall be 8 (SAS).
Role (added)		Mandatory	Shall be 3 (Target).

10.5.6 CIM_SystemDevice (Port)

Associates ComputerSystem to LogicalPort. The class definition specializes the CIM_SystemDevice definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 80 describes class CIM_SystemDevice (Port).

Table 80 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem in the referencing profile.
PartComponent (overridden)		Mandatory	Reference to SASPort.

10.5.7 CIM_SystemDevice (SAS PHY)

Associates ComputerSystem to SASPHY.

Requirement: Mandatory

Table 81 describes class CIM_SystemDevice (SAS PHY).

Table 81 - SMI Referenced Properties/Methods for CIM_SystemDevice (SAS PHY)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent		Mandatory	Reference to SASPHY.

10.5.8 SNIA_SASPHY

Several SASPHYs may together be aggregated into a SAS Logical Port.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 82 describes class SNIA_SASPHY.

Table 82 - SMI Referenced Properties/Methods for SNIA_SASPHY

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
HardwareMinimumPhysicalLinkRate		Mandatory	
HardwareMaximumPhysicalLinkRate		Mandatory	
ProgrammedMinimumPhysicalLinkRate		Mandatory	
ProgrammedMaximumPhysicalLinkRate		Mandatory	
NegotiatedPhysicalLinkRate		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

11 Serial ATA (SATA) Target Ports Profile

11.1 Synopsis

Profile Name: SATA Target Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.13.1

Table 83 describes the related profiles for SATA Target Ports.

Table 83 - Related Profiles for SATA Target Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Central Class: CIM_SASPortt

Scoping Class: a CIM_System in a separate autonomous profile

Model Serial ATA (SATA) target ports.

11.2 Description

Figure 12 illustrates the Serial ATA Target Port Profile. Serial ATA has a simple bus structure. The SATAPort class will include attributes that specifies bus speed and other hardware options.

This model will not be used with LMM common subprofile. All nodes on the bus will have access to each other.

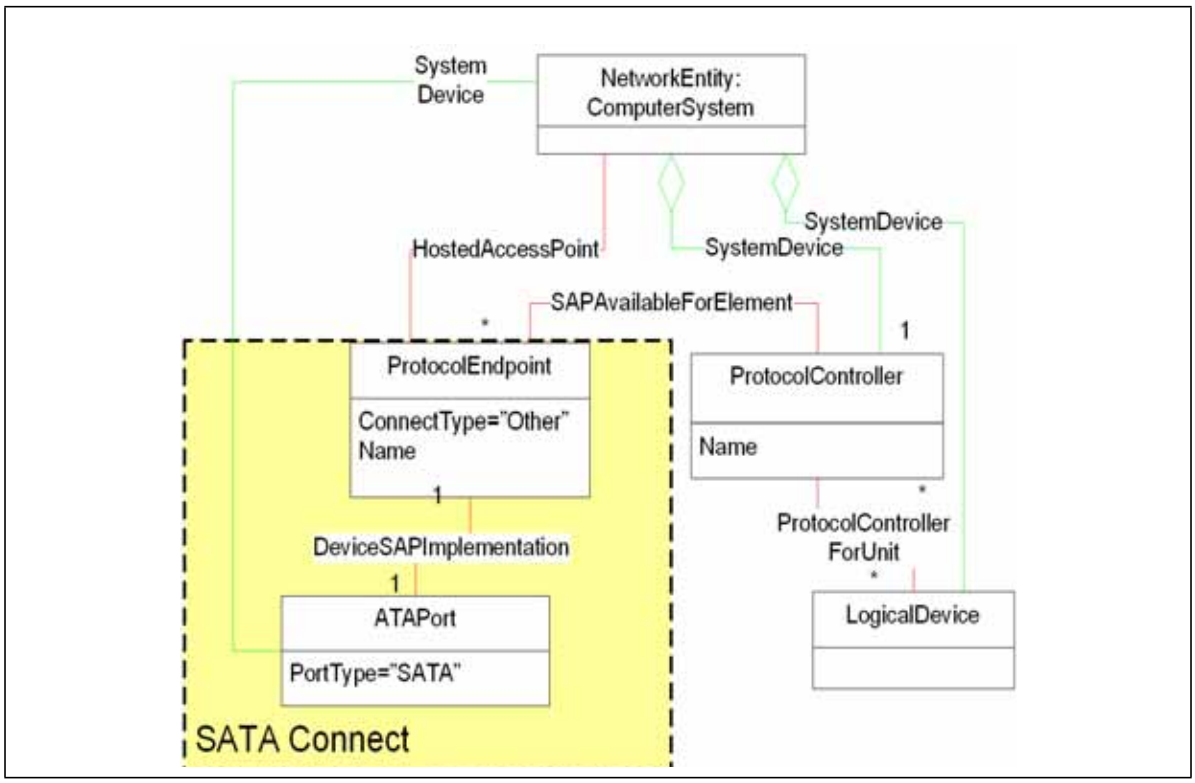

Figure 12 - SATA Target Port Instance Diagram

ProtocolEndpoint.ConnectionType shall be set to “other”. The ProtocolEndPoint class is associated to the ATAPort class with DevImplementation. The ATAPort class contains all the bus operational settings.

11.2.1 Health and Fault Management

Table 84 describes ATAPort OperationalStatus.

Table 84 - ATAPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

11.3 Methods of this Subprofile

None

11.4 Client Considerations and Recipes

None

11.5 CIM Elements

Table 85 describes the CIM elements for SATA Target Ports.

Table 85 - CIM Elements for SATA Target Ports

Element Name	Requirement	Description
11.5.1 CIM_ATAPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
11.5.2 CIM_ATAProtocolEndpoint	Mandatory	Represents management characteristics related to the ATA command set.
11.5.3 CIM_DeviceSAPImplementation	Mandatory	Associates front-end ATAPort and target ATAProtocolEndpoint.
11.5.4 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to ATAProtocolEndpoint.
11.5.5 CIM_SystemDevice (Port)	Mandatory	Associates ComputerSystem to ATAPort.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ATAPort	Mandatory	Create ATAPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ATAPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Modify ATAPort.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ATAPort	Mandatory	Delete ATAPort.

11.5.1 CIM_ATAPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 86 describes class CIM_ATAPort.

Table 86 - SMI Referenced Properties/Methods for CIM_ATAPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	

Table 86 - SMI Referenced Properties/Methods for CIM_ATAPort

Properties	Flags	Requirement	Description & Notes
UsageRestriction (overridden)		Mandatory	Shall be 2 (Front-end Only).
PortType (overridden)		Mandatory	Shall be 92|93 (SATA or SATA2).

11.5.2 CIM_ATAProtocolEndpoint

Represents management characteristics related to the ATA command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 87 describes class CIM_ATAProtocolEndpoint.

Table 87 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'ATA'.
ConnectionType (added)		Mandatory	Shall be 3 (SATA).
Role (added)		Mandatory	Shall be 3 (Target).

11.5.3 CIM_DeviceSAPImplementation

Associates front-end ATAPort and target ATAProtocolEndpoint. Limit to target ProtocolEndpoints and front-end ports. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 88 describes class CIM_DeviceSAPImplementation.

Table 88 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to ATAProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to ATAPort.

11.5.4 CIM_HostedAccessPoint

Associates ComputerSystem to ATAProtocolEndpoint. Limit to targets. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 89 describes class CIM_HostedAccessPoint.

Table 89 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent (overridden)		Mandatory	
Dependent (overridden)		Mandatory	Reference to ATAProtocolEndpoint.

11.5.5 CIM_SystemDevice (Port)

Associates ComputerSystem to ATAPort. The class definition specializes the CIM_SystemDevice definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 90 describes class CIM_SystemDevice (Port).

Table 90 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent (overridden)		Mandatory	Reference to ComputerSystem in referencing profile.
PartComponent (overridden)		Mandatory	Reference to ATAPort.

EXPERIMENTAL

EXPERIMENTAL

12 SB Target Ports Profile

12.1 Synopsis

Profile Name: SB Target Ports (Component Profile)

Version: 1.2.0

Organization: SNIA

CIM Schema Version: 2.13

Table 91 describes the related profiles for SB Target Ports.

Table 91 - Related Profiles for SB Target Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Central Class: CIM_FCPort

Scoping Class: CIM_System

12.2 Description

The SB Target Port Profile models the SB (Single Byte) Fibre Channel specific aspects of a target storage system. The Single Byte protocols are FC4 protocols that support mainframe IO (as opposed to SCSI, which supports IO from non-mainframe systems such as Unix or Windows systems).

The SB Target Port Profile provides a way for storage profiles to model target ports that are dedicated to serving SB hosts attachment. With this support a client will be able to distinguish FC ports that are provided for SCSI access from FC Ports that are provided for mainframe attachment. This is an important distinction for management, since fabric connectivity collections for SB would typically be separate for fabric connectivity collections for SCSI. Similarly, management functions for masking and mapping are somewhat different for SB than SCSI. So, it is important for management applications to be aware of the distinctions.

The SB Target Port Profile specializes the Generic Target Port Profile.

For SB enabled Fibre Channel ports, the concrete subclass of LogicalPort is FCPort. FCPort is always associated 1-1 with a SNIA_SBProtocolEndpoint instance.

12.3 Implementation

Figure 13 illustrates the SB Target Port Profile.

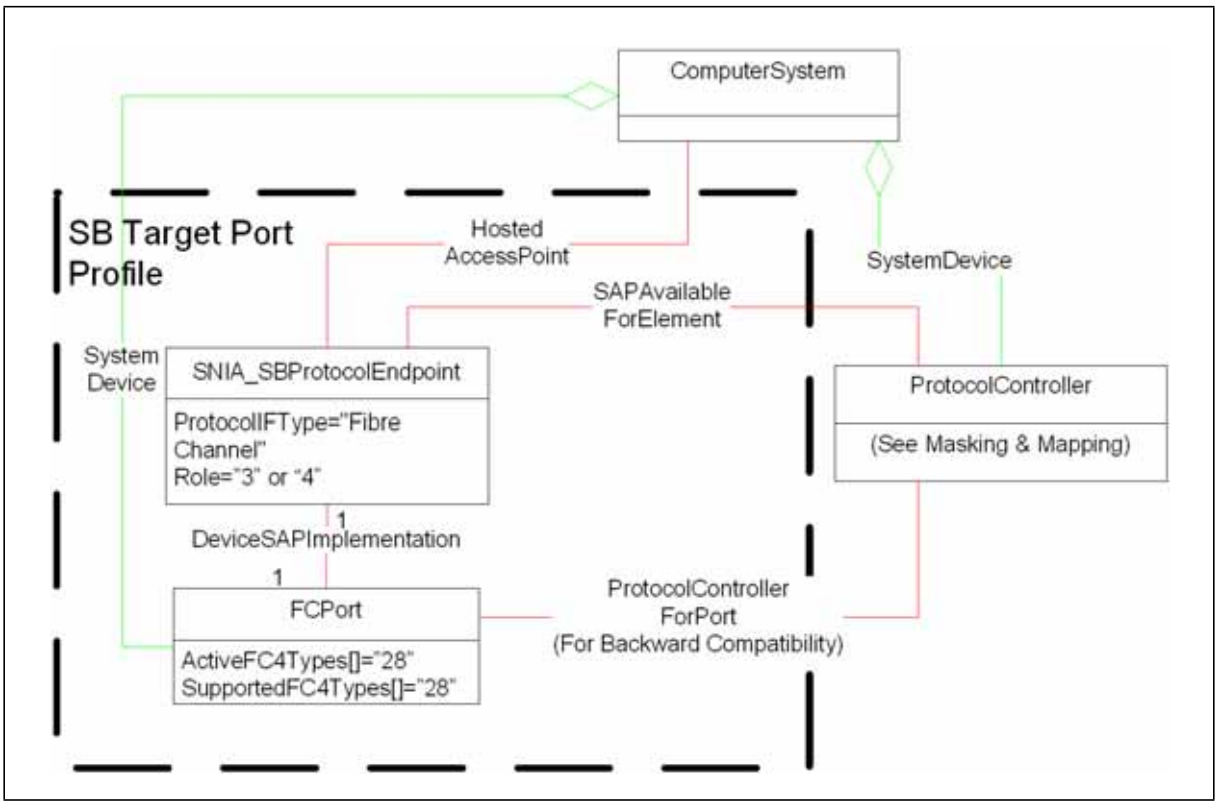

Figure 13 - SB Target Port Instance Diagram

SB Ports are Fibre Channel Ports with the SupportedFC4Types[] and ActiveFC4Types[] arrays holding the value “28” (for “FC-SB-2 Control Unit”). The SupportedFC4Types[] property shall contain the value “28”. The ActiveFC4Types[] property shall contain the value “28” for FCPorts that are actively supporting SB protocols.

The FCPort shall also support an SBProtocolEndpoint with a role property of either “3” (“Target”) or “4” (“Both initiator and target”).

For the SB Target Port Profile, the FCPort is the central class of the Profile.

12.4 Health and Fault Management Consideration

Table 92 defines the SMI-S defined meanings of the OperationalStatus property for FCPorts used in the SB Target Port Profile.

Table 92 - FCPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled

Table 92 - FCPort OperationalStatus

OperationalStatus	Description
InService	Port is in self test
Unknown	

12.5 Cascading Considerations

None

12.6 Methods of the Profile

12.6.1 Extrinsic Methods of the Profile

None

12.6.2 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

12.7 Client Considerations and Recipes

None

12.8 CIM Elements

Table 93 describes the CIM elements for SB Target Ports.

Table 93 - CIM Elements for SB Target Ports

Element Name	Requirement	Description
12.8.1 CIM_DeviceSAPImplementation	Mandatory	Associates front-end FCPort and target SBProtocolEndpoint.
12.8.2 CIM_FCPort	Mandatory	The FCPort provides transport for SB protocols.
12.8.3 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to SBProtocolEndpoint.
12.8.4 CIM_SystemDevice (Port)	Mandatory	Associates controller ComputerSystem to FCPort.
12.8.5 SNIA_SBProtocolEndpoint	Mandatory	This defines the protocol being used on the FC Port as SB (Single Byte).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort	Mandatory	Create FCPort.

Table 93 - CIM Elements for SB Target Ports

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Deprecated WQL -Change to FCPort OperationalStatus.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::OperationalStatus <> PreviousInstance.CIM_FCPort::OperationalStatus	Mandatory	CQL -Change to FCPort OperationalStatus.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort	Mandatory	Delete FCPort.

12.8.1 CIM_DeviceSAPImplementation

Associates front-end FCPort and target SBProtocolEndpoint. Limit to target ProtocolEndpoints and front-end ports. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 94 describes class CIM_DeviceSAPImplementation.

Table 94 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	
Antecedent (overridden)		Mandatory	

12.8.2 CIM_FCPort

The CIM_FCPort class for SB Target Ports is the same as the FC Target Port class, except that the SupportedFC4Types and ActiveFC4Types properties are mandatory, and the SupportedFC4Types shall contain "28" and the ActiveFC4Types should contain "28" to indicate support for FC-SB-2 Control Unit functions. The class definition specializes the CIM_LogicalPort definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 95 describes class CIM_FCPort.

Table 95 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	

Table 95 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction (overridden)		Mandatory	Shall be 2 (Front-end Only).
PortType (overridden)		Optional	
PermanentAddress (added)	CD	Mandatory	Port WWN. Shall be 16 unseparated uppercase hex digits.
SupportedCOS (added)		Optional	
ActiveCOS (added)		Optional	
SupportedFC4Types (added)		Mandatory	For SB Target Ports this array shall contain 28 (FC-SB-2 Control Unit).
ActiveFC4Types (added)		Mandatory	For SB Target Ports this array should contain 28 (FC-SB-2 Control Unit).

12.8.3 CIM_HostedAccessPoint

Associates ComputerSystem to SBProtocolEndpoint. Limit to targets (Role = 3). The class definition specializes the CIM_HostedAccessPoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 96 describes class CIM_HostedAccessPoint.

Table 96 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in the referencing profile.
Dependent (overridden)		Mandatory	Reference to SBProtocolEndpoint.

12.8.4 CIM_SystemDevice (Port)

Associates controller ComputerSystem to FCPort. The class definition specializes the CIM_SystemDevice definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 97 describes class CIM_SystemDevice (Port).

Table 97 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent (overridden)		Mandatory	Reference to ComputerSystem in the referencing profile.
PartComponent (overridden)		Mandatory	Reference to FCPort.

12.8.5 SNIA_SBProtocolEndpoint

The SNIA_SBProtocolEndpoint specializes the Generic Target Port CIM_ProtocolEndpoint. The main difference is that the OtherTypeDescription is set to "SB". The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 98 describes class SNIA_SBProtocolEndpoint.

Table 98 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SB'.
Role (added)		Mandatory	Shall be 3 (Target).
ConnectionType (added)		Mandatory	Shall be 2 (Fibre Channel).

EXPERIMENTAL

EXPERIMENTAL

13 Direct Attach (DA) Ports Profile

13.1 Description

The DAPort (Direct Attach) port models storage systems that attach directly to buses in a host system (e.g., PCI, PCI-E, and chip interfaces on a motherboard). The DAPort can be viewed as both the initiator and Target ports.

This port can not be used with the LUN Mapping and Masking Profile. All volumes served by this port are fully accessible by the host system.

Figure 14 illustrates the Direct Attach (DA) Ports Profile. Volumes served by this port shall be discovered and presented by the Host Discovered Resources Profile.

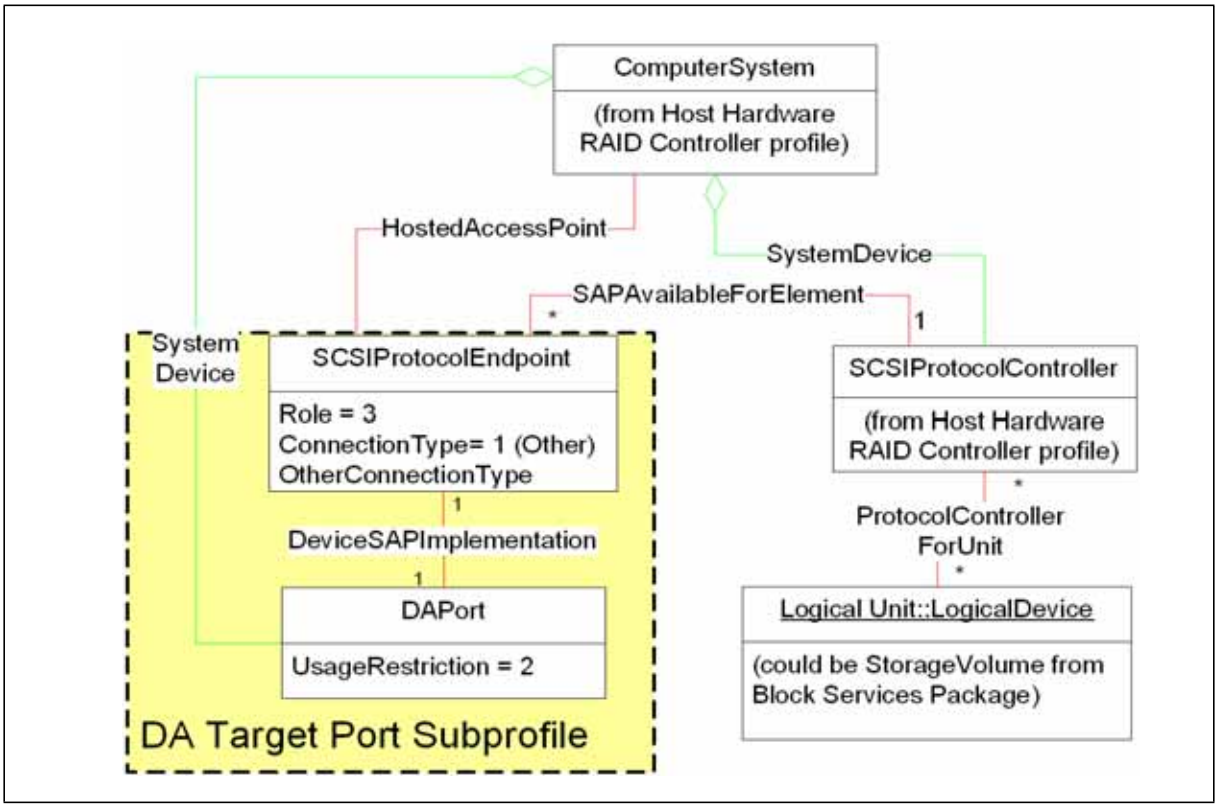

Figure 14 - DA Port Instance Diagram

The DAPort class is connected to the ProtocolEndpoint using the DeviceSAPImplementation association and to the controller ComputerSystem (in the Host Hardware RAID Controller Profile) using SystemDevice. SCSIProtocolController is associated to the controller ComputerSystem from the same ComputerSystem using HostedAccessPoint.

13.2 Health and Fault Management

Not defined in this standard.

13.3 Supported Profiles and Packages

Related Profiles for DA Target Ports: Not defined in this standard.

13.4 Extrinsic Methods

Not defined in this standard.

13.5 Client Considerations and Recipes

Not defined in this standard.

13.6 Registered Name and Version

DA Target Ports version 1.4.0 (Component Profile)

CIM Schema Version: 2.11.0

Specializes SNIA Generic Target Ports version 1.4.0

13.7 CIM Elements

Table 99 describes the CIM elements for DA Target Ports.

Table 99 - CIM Elements for DA Target Ports

Element Name	Requirement	Description
13.7.1 CIM_DAPort	Mandatory	Models the "port" emulated by a host RAID controller.
13.7.2 CIM_DeviceSAPImplementation	Mandatory	Associates DAPort and SCSIProtocolEndpoint.
13.7.3 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to SCSIProtocolEndpoint.
13.7.4 CIM_SCSIProtocolEndpoint	Mandatory	ProtocolEndpoint representing support for SCSI.
13.7.5 CIM_SystemDevice (Port)	Mandatory	Associates controller ComputerSystem to DAPort.

13.7.1 CIM_DAPort

Models the "port" emulated by a host RAID controller. The class definition specializes the CIM_LogicalPort definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 100 describes class CIM_DAPort.

Table 100 - SMI Referenced Properties/Methods for CIM_DAPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	

Table 100 - SMI Referenced Properties/Methods for CIM_DAPort

Properties	Flags	Requirement	Description & Notes
UsageRestriction (overridden)		Mandatory	Shall be 2 (Front-end Only).
PortType (overridden)		Optional	Not defined in this version of the standard.

13.7.2 CIM_DeviceSAPImplementation

Associates DAPort and SCSIProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 101 describes class CIM_DeviceSAPImplementation.

Table 101 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to DAPort.

13.7.3 CIM_HostedAccessPoint

Associates ComputerSystem to SCSIProtocolEndpoint. Limit to targets. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 102 describes class CIM_HostedAccessPoint.

Table 102 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent (overridden)		Mandatory	Reference to controller ComputerSystem.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.

13.7.4 CIM_SCSIProtocolEndpoint

ProtocolEndpoint representing support for SCSI. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 103 describes class CIM_SCSIProtocolEndpoint.

Table 103 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
Role (added)		Mandatory	Shall be 3 (Target).
ConnectionType (added)		Mandatory	Shall be 1 (Other).
OtherConnectionType (added)		Mandatory	

13.7.5 CIM_SystemDevice (Port)

Associates controller ComputerSystem to DAPort. The class definition specializes the CIM_SystemDevice definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 104 describes class CIM_SystemDevice (Port).

Table 104 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent (overridden)		Mandatory	Reference to controller ComputerSystem.
PartComponent (overridden)		Mandatory	Reference to DAPort.

EXPERIMENTAL

EXPERIMENTAL

14 Generic Initiator Ports Profile

14.1 Synopsis

Profile name: Generic Initiator Ports

Version: 1.4.0

Organization: SNIA

CIM schema version: 2.9.0 (later schema versions may be required for specializations)

Central Class: CIM_LogicalPort

Scoping Class: a CIM_System in a separate autonomous profile

The Generic Initiator Port Profile models the generic management interfaces of initiator ports in host adaptors or storage systems.

This abstract profile specification shall not be directly implemented; implementations shall be based on a profile specification that specializes the requirements of this profile.

14.2 Description

The Generic Initiator Port Profile models the generic behavior of initiator ports in host adaptors. It uses the same primary classes as the Generic Target Port Profile (see 6 Generic Target Ports Profile)

14.3 Implementation

The initiator port is modeled as a ProtocolEndpoint connected to a LogicalPort.

The LogicalDevice instances may represent local storage (embedded in the system containing the initiator ports) or remote storage. When it represents remote storage the Name and NameFormat properties are used as correlatable ids to reference the remote device. When the LogicalDevice represents local disk storage, it may be represented as an instance of StorageVolume (subclass of LogicalDevice) or part of an instance of the Disk Drive Lite Profile. A property on LogicalPort called UsageRestriction is available to indicate whether the controller is capable of providing a “front end” (target), a “back end” (initiator), or both interfaces.

Figure 15 depicts the generic model

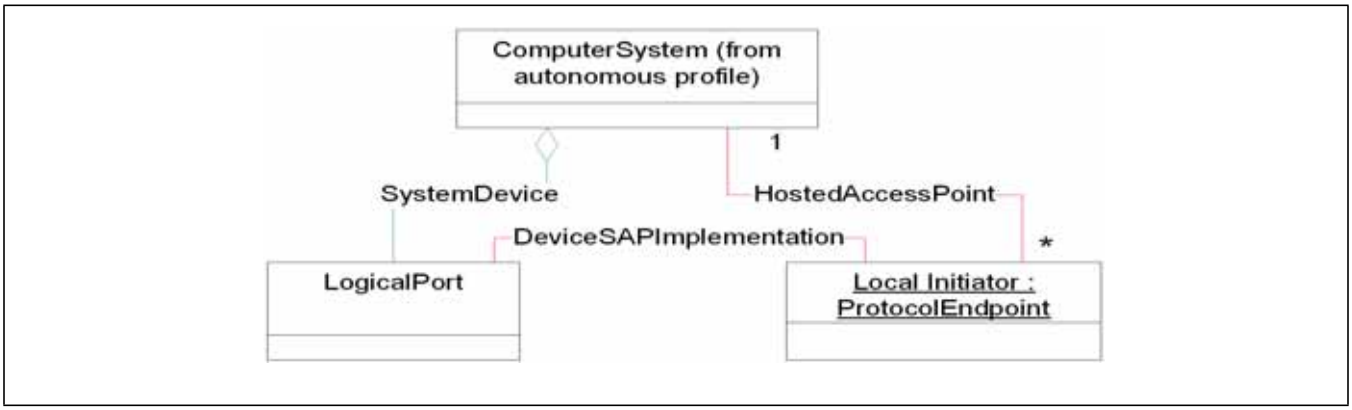

Figure 15 - Generic Initiator Port Model

14.3.1 Remote Device Models

The implementation may optionally include discovered remote elements. There are two optional approaches to modeling remote elements, depending on the capabilities of the underlying host drivers

The first approach is to model a collection of ports representing the local and remote ports that are know to be connected. This approach is appropriate for ATA device and when the underlying drivers or software is limited to information about remote ports and does not include details of the logical devices connected to remote ports. Figure 16 depicts the optional connectivity collection model.

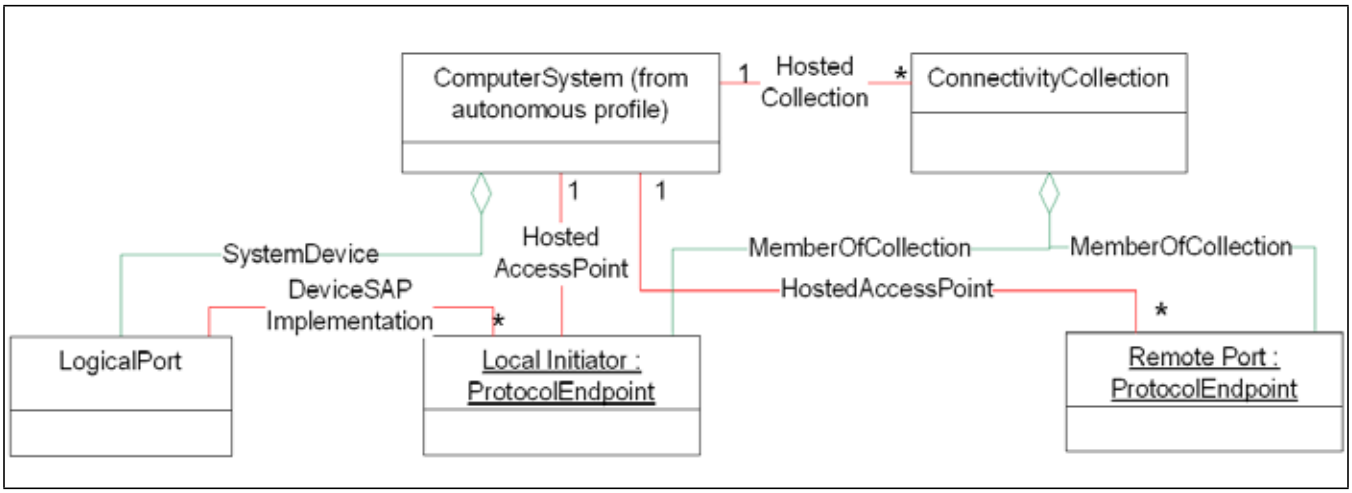

Figure 16 - Optional Connectivity Collection Model

The nature of membership in the collection varies with transports and configuration options. For example, in a parallel SCSI environment, the ConnectivityCollection includes all initiators/targets attached to the bus. In an FC fabric environment, the ConnectivityCollection contains ports that share a zone. In many cases, the ConnectivityCollection could include remote initiators as well as remote devices.

The second approach to modeling remote devices is to include the full initiator/target/logical-unit path model that describes multipath connectivity. This approach has the advantage of including the logical units and including the full path connectivity. The disadvantage is that some OSes handle multipath support in different components from HBA support, making it more efficient to provide the multipath model as part of the Host Discovered Resources Profile. Figure 17 depicts the optional full-path model.

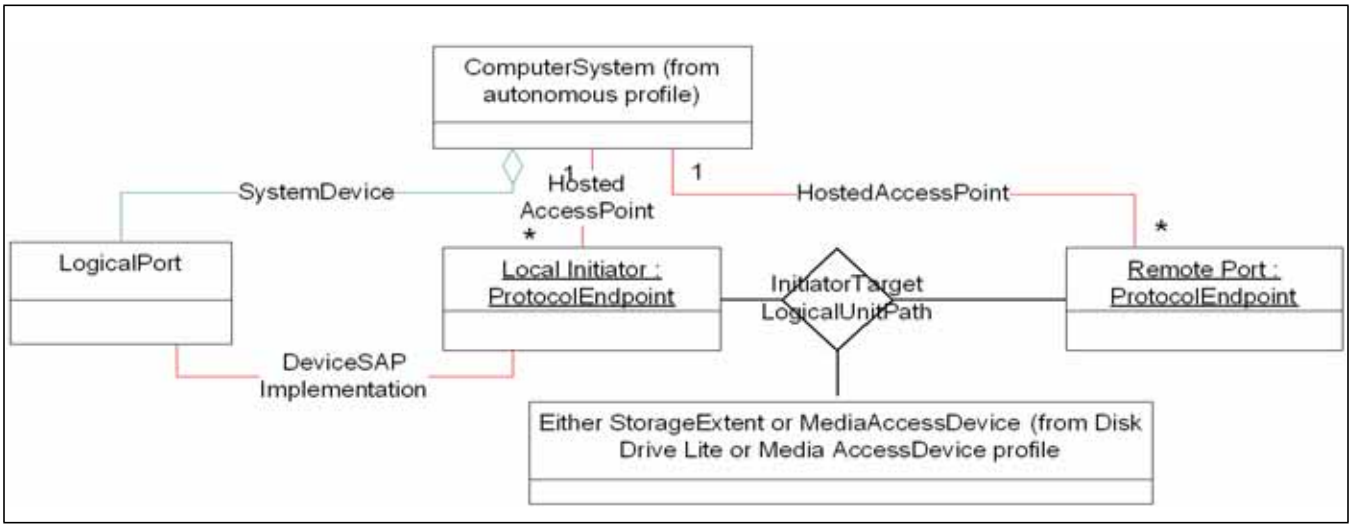

Figure 17 - Optional Full-Path Model

The instrumentation may support the full-path and connectivity collection options by making appropriate ProtocolEndpoints members of ConnectivityCollections.

14.3.1.1 Optional Model for Attached Disks

Disks are modeled using the full-path model and the Disk Drive Lite Profile. The appropriate subclass of InitiatorTargetLogicalUnitPath shall be dependent on whether the disks use the SCSI, ATA or SB commend set. This association references StorageExtent and initiator and target ProtocolEndpoints. The association also provides the disk's logical unit number. The target ProtocolEndpoint referenced from InitiatorTargetLogicalUnitPath shall be the ProtocolEndpoint from the Disk Drive Lite Profile associated indirectly to StorageExtent via DiskDrive. This is the same ProtocolEndpoint described as the optional remote ProtocolEndpoint in initiator ports profiles.

The ProtocolEndpoints may be associated to a ConnectivityCollection representing a collection of logically connected devices, as illustrated in Figure 18.

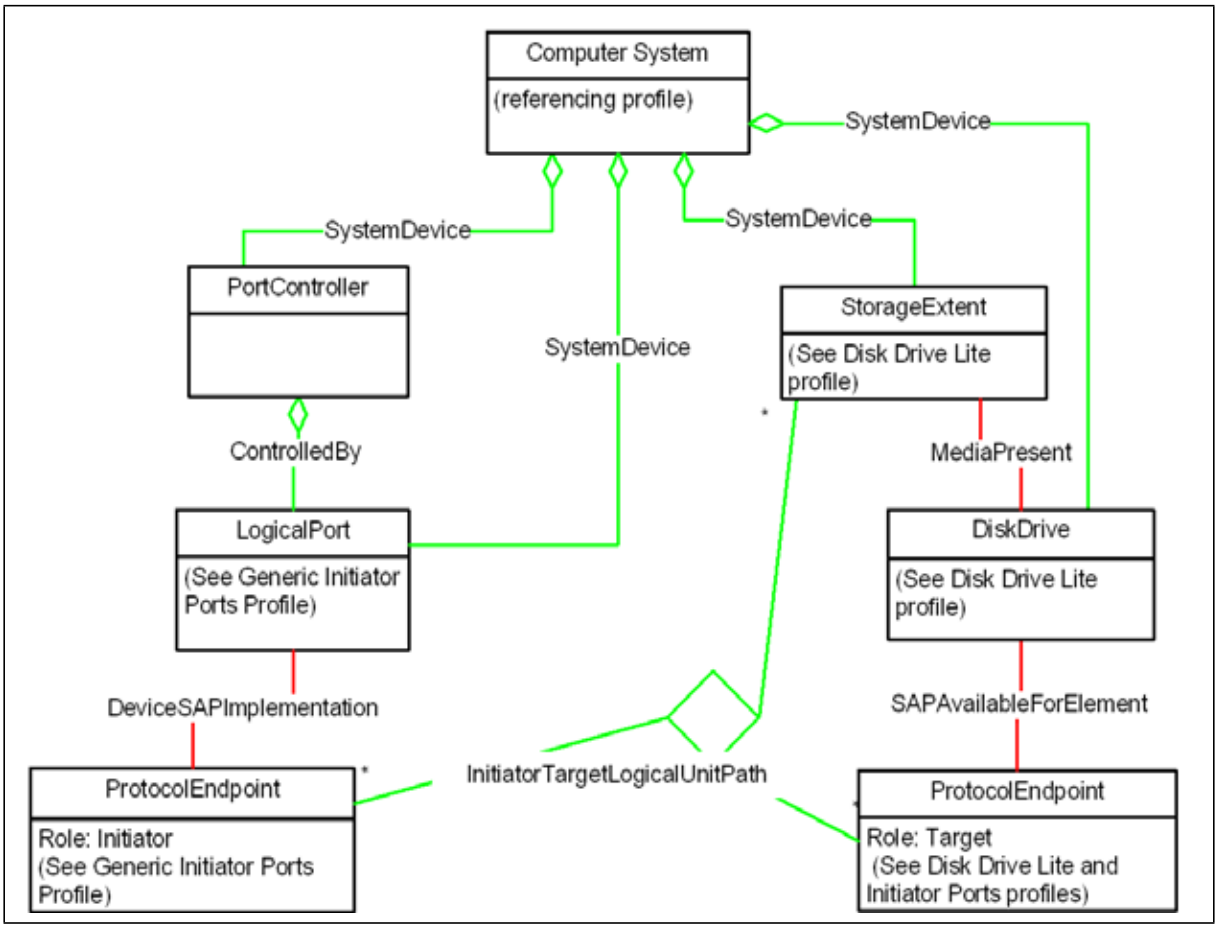

Figure 18 - HBA and Disk Model

14.3.1.2 Optional Model for attached Tape/CD/DVD Drives

The model, illustrated in Figure 19, and requirements are similar to those for disks (see 14.3.1.1), but use the Media Access Device Profile rather than Disk Drive Lite and the appropriate subclass of MediaAccessDevice rather than DiskDrive.

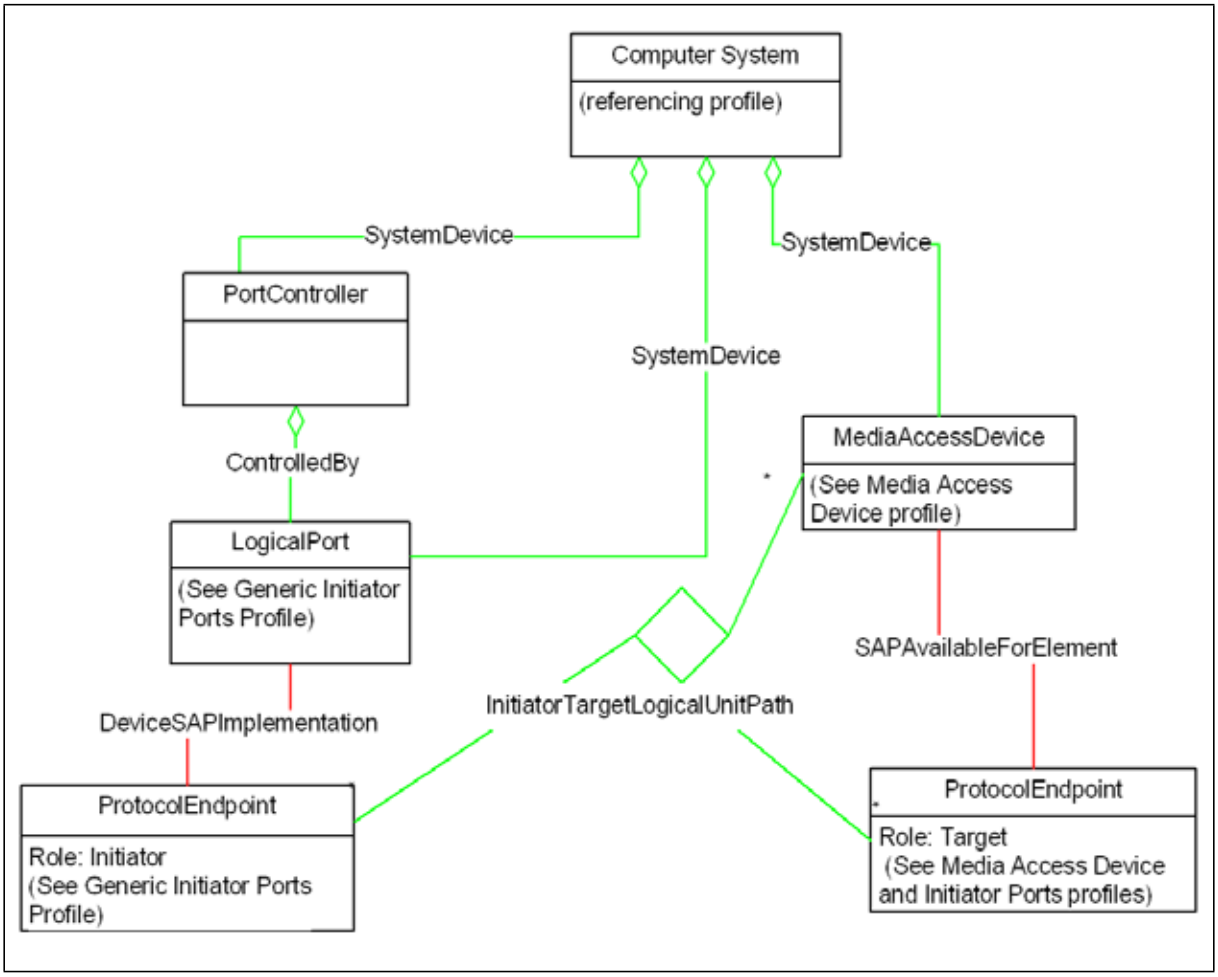

Figure 19 - HBA and Tape or Optical Devices

14.3.1.3 Optional Port Statistics

An implementation of an initiator port profile may optionally support port statistics. Figure 20 depicts the model for port statistics.

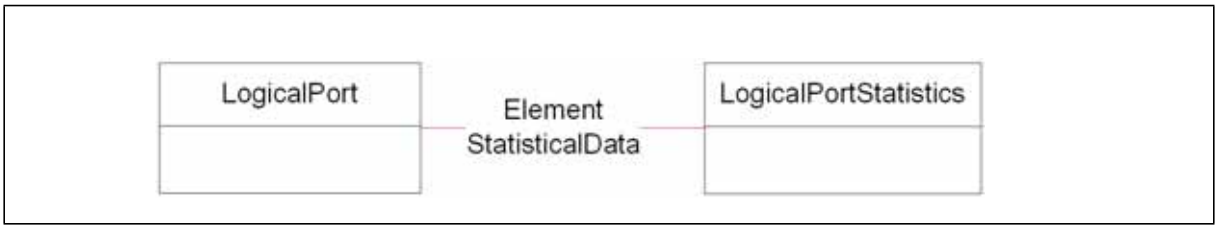

Figure 20 - Port Statistics

If a specialization of the profile specifies a subclass of LogicalPortStatistics (e.g., FCPortStatistics), the implementation should associate that subclass to the appropriate subclass of LogicalPort (e.g., FCPort), as shown in Figure 21. Otherwise, an implementation should use LogicalPortStatistics directly.

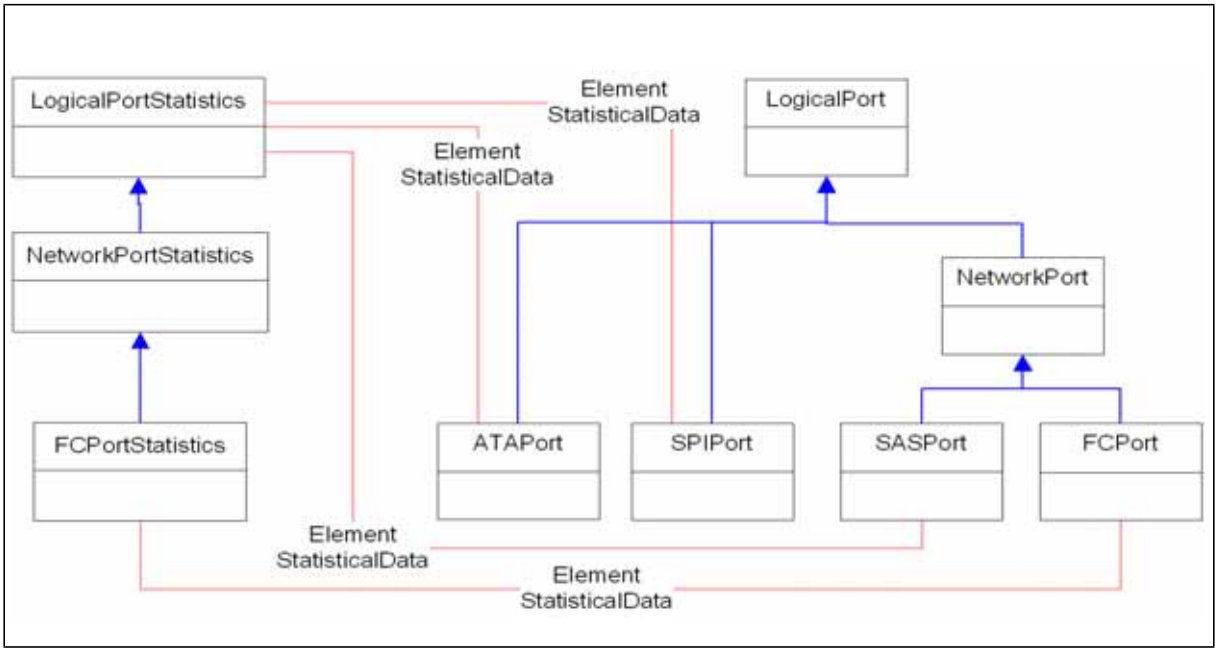

Figure 21 - Port Statistics Hierarchy

14.3.2 Health and Fault Management Considerations

Not defined in this standard.

14.3.3 Cascading Considerations

Not defined in this standard.

14.4 Methods

14.4.1 Extrinsic Methods of this Profile

None

14.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances

- EnumerateInstanceNames

14.5 Use Cases

Not defined in this standard.

14.6 CIM Elements

Table 105 describes the CIM elements for Generic Initiator Ports.

Table 105 - CIM Elements for Generic Initiator Ports

Element Name	Requirement	Description
14.6.1 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
14.6.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator LogicalPort and ProtocolEndpoint.
14.6.3 CIM_ElementStatisticalData (Port Statistics)	Optional	Connects LogicalPort and LogicalPortStatistics.
14.6.4 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
14.6.5 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
14.6.6 CIM_HostedCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates the ConnectivityCollection to the hosting System.
14.6.7 CIM_LogicalPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
14.6.8 CIM_MemberOfCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates ProtocolEndpoints to the ConnectivityCollection.
14.6.9 CIM_ProtocolEndpoint (Initiator)	Mandatory	Represents a protocol (command set) supported by the port. The appropriate subclass (SCSIProtocolEndpoint, ATAProtocolEndpoint, SBProtocolEndpoint) should be used in initiator port specialized profiles.
14.6.10 CIM_ProtocolEndpoint (Target)	Optional	Models protocols of remote ports - target devices and possibly other initiators.
14.6.11 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
14.6.12 SNIA_LogicalPortStatistics	Optional	Statistics for a port.

14.6.1 CIM_ConnectivityCollection

Represents a collection of connected ProtocolEndpoints.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 106 describes class CIM_ConnectivityCollection.

Table 106 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	

14.6.2 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 107 describes class CIM_DeviceSAPImplementation.

Table 107 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ProtocolEndpoint(Initiator).
Antecedent		Mandatory	Reference to LogicalPort.

14.6.3 CIM_ElementStatisticalData (Port Statistics)

Connects LogicalPort and LogicalPortStatistics.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 108 describes class CIM_ElementStatisticalData (Port Statistics).

Table 108 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Port Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Reference to LogicalPort.
Stats		Mandatory	Reference to LogicalPortStatistics.

14.6.4 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 109 describes class CIM_HostedAccessPoint (Initiator).

Table 109 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent		Mandatory	Reference to ProtocolEndpoint(Initiator).

14.6.5 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 110 describes class CIM_HostedAccessPoint (Target).

Table 110 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent		Mandatory	Reference to ProtocolEndpoint(Target).

14.6.6 CIM_HostedCollection (Connectivity Collection)

Associates the ConnectivityCollection to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 111 describes class CIM_HostedCollection (Connectivity Collection).

Table 111 - SMI Referenced Properties/Methods for CIM_HostedCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ConnectivityCollection.
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.

14.6.7 CIM_LogicalPort

Represents the logical aspects of the physical port and may have multiple associated protocols.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 112 describes class CIM_LogicalPort.

Table 112 - SMI Referenced Properties/Methods for CIM_LogicalPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	

Table 112 - SMI Referenced Properties/Methods for CIM_LogicalPort

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction		Mandatory	Shall be 3 for ports restricted to back-end (initiator) only or 4 if the port is unrestricted.
PortType		Mandatory	Initiator port specialized profiles specify the appropriate subset of values.

14.6.8 CIM_MemberOfCollection (Connectivity Collection)

Associates ProtocolEndpoints to the ConnectivityCollection.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 113 describes class CIM_MemberOfCollection (Connectivity Collection).

Table 113 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	Reference to ProtocolEndpoint.
Collection		Mandatory	Reference to ConnectivityCollection.

14.6.9 CIM_ProtocolEndpoint (Initiator)

Represents a protocol (command set) supported by the port.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 114 describes class CIM_ProtocolEndpoint (Initiator).

Table 114 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.

Table 114 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription		Mandatory	Shall be the string 'SCSI', 'ATA', or 'SB'. Initiator port specialized profiles specify the appropriate subset.

14.6.10CIM_ProtocolEndpoint (Target)

Models protocols of remote ports - target devices and possibly other initiators. The appropriate subclass (SCSIProtocolEndpoint, ATAProtocolEndpoint, SBProtocolEndpoint) should be used in initiator port specialized profiles.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 115 describes class CIM_ProtocolEndpoint (Target).

Table 115 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	The values in MOFs map to IETF values and exclude storage. Shall be 1 (Other) and set OtherTypeDescription appropriately.
OtherTypeDescription		Mandatory	Shall be the string 'SCSI', 'ATA', or 'SB'. Initiator port specialized profiles specify the appropriate subset.

14.6.11CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 116 describes class CIM_SystemDevice (Initiator Ports).

Table 116 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent		Mandatory	Reference to LogicalPort.

14.6.12SNIA_LogicalPortStatistics

Statistics for a port.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 117 describes class SNIA_LogicalPortStatistics.

Table 117 - SMI Referenced Properties/Methods for SNIA_LogicalPortStatistics

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
BytesTransmitted		Mandatory	
BytesReceived		Mandatory	
PacketsTransmitted		Mandatory	
PacketsReceived		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

15 Parallel SCSI (SPI) Initiator Ports Profile

15.1 Synopsis

Profile Name: SPI Initiator Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.11.0

Related Profiles for SPI Initiator Ports: Not defined in this standard.

Specializes: Generic Initiator Port Profile

Central Class: CIM_SPIPort

Scoping Class: a CIM_System in a separate autonomous profile

The SPI Initiator Ports Profile models the behavior of a parallel SCSI (SPI) initiator port.

15.2 Description

The SPI Initiator Port Profile defines the model to parallel SCSI ports.

15.3 Implementation

A typical instance diagram is provided in Figure 22.

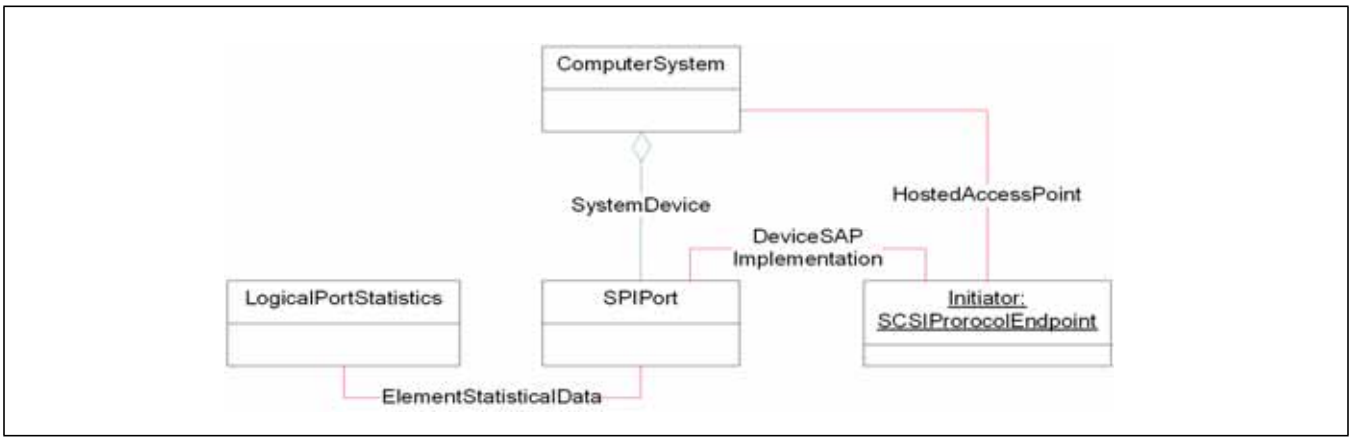

Figure 22 - SPI Initiator Port Instance Diagram

15.3.1 Health and Fault Management Considerations

Table 118 summarizes the Health and Fault Management issues that are unique to this profile.

Table 118 - SPIPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

15.3.2 Cascading Considerations

Not defined in this standard.

15.4 Methods

15.4.1 Extrinsic Methods of this Profile

None

15.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

15.5 Detailed Use Cases and Recipes

None

15.6 CIM Elements

Table 119 describes the CIM elements for SPI Initiator Ports.

Table 119 - CIM Elements for SPI Initiator Ports

Element Name	Requirement	Description
15.6.1 CIM_ConnectivityCollection	Mandatory	Represents a collection of connected SCSIProtocolEndpoints.
15.6.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator SPILogicalPort and SCSIProtocolEndpoint.
15.6.3 CIM_ElementStatisticalData (Port Statistics)	Mandatory	Connects SPIPort and LogicalPortStatistics.
15.6.4 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
15.6.5 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
15.6.6 CIM_HostedCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates the ConnectivityCollection to the hosting System.
15.6.7 CIM_MemberOfCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Represents a collection of connected SCSIProtocolEndpoints.
15.6.8 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
15.6.9 CIM_SCSIProtocolEndpoint (Initiator)	Mandatory	Represents support for the SCSI command set.
15.6.10 CIM_SCSIProtocolEndpoint (Target)	Mandatory	Models protocols of remote ports - target devices and possibly other initiators.
15.6.11 CIM_SPIPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
15.6.12 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
15.6.13 SNIA_LogicalPortStatistics	Optional	Statistics for a port.

15.6.1 CIM_ConnectivityCollection

Represents a collection of connected SCSIProtocolEndpoints. The class definition specializes the CIM_ConnectivityCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 120 describes class CIM_ConnectivityCollection.

Table 120 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	

15.6.2 CIM_DeviceSAPImplementation

Connects Initiator SPILogicalPort and SCSIProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 121 describes class CIM_DeviceSAPImplementation.

Table 121 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Initiator).
Antecedent (overridden)		Mandatory	Reference to SPIPort.

15.6.3 CIM_ElementStatisticalData (Port Statistics)

Connects SPIPort and LogicalPortStatistics. The class definition specializes the CIM_ElementStatisticalData definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 122 describes class CIM_ElementStatisticalData (Port Statistics).

Table 122 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Port Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement (overridden)		Mandatory	Reference to SPIPort.
Stats		Mandatory	Reference to LogicalPortStatistics.

15.6.4 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 123 describes class CIM_HostedAccessPoint (Initiator).

Table 123 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Inititor).

15.6.5 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 124 describes class CIM_HostedAccessPoint (Target).

Table 124 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Target).

15.6.6 CIM_HostedCollection (Connectivity Collection)

Associates the ConnectivityCollection to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 125 describes class CIM_HostedCollection (Connectivity Collection).

Table 125 - SMI Referenced Properties/Methods for CIM_HostedCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ConnectivityCollection.
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.

15.6.7 CIM_MemberOfCollection (Connectivity Collection)

Represents a collection of connected SCSIProtocolEndpoints. The class definition specializes the CIM_MemberOfCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 126 describes class CIM_MemberOfCollection (Connectivity Collection).

Table 126 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Member (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Initiator or Target).
Collection		Mandatory	Reference to ConnectivityCollection.

15.6.8 CIM_SCSIInitiatorTargetLogicalUnitPath

Represents a path between a SCSI initiator, target, and logical unit.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 127 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 127 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
LogicalUnit		Mandatory	Reference to StorageExtent in Disk Drive Lite Profile or MediaAccessDevice in Media Access Device Profile.
Target		Mandatory	Reference to SCSIProtocolEndpoint(Target).
Initiator		Mandatory	Reference to SCSIProtocolEndpoint(Initiatotor).

15.6.9 CIM_SCSIProtocolEndpoint (Initiator)

Represents support for the SCSI command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 128 describes class CIM_SCSIProtocolEndpoint (Initiator).

Table 128 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
ProtocolIFType		Mandatory	Shall be 1 (Other).

Table 128 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
OtherTypeDescription		Mandatory	Shall be the string 'SCSI', 'ATA', or 'SB'. Initiator port specialized profiles specify the appropriate subset.
ConnectionType (added)		Mandatory	Shall be 3 (Parallel SCSI).
Role (added)		Mandatory	Shall be 2 (Initiator).

15.6.10CIM_SCSIProtocolEndpoint (Target)

Models protocols of remote ports - target devices and possibly other initiators. The appropriate subclass (SCSIProtocolEndpoint, ATAProtocolEndpoint, SBProtocolEndpoint) should be used in initiator port specialized profiles. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 129 describes class CIM_SCSIProtocolEndpoint (Target).

Table 129 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	The values in MOFs map to IETF values and exclude storage. Shall be 1 (Other) and set OtherTypeDescription appropriately.
OtherTypeDescription		Mandatory	Shall be the string 'SCSI', 'ATA', or 'SB'. Initiator port specialized profiles specify the appropriate subset.
ConnectionType (added)		Mandatory	Shall be 3 (Parallel SCSI).

15.6.11CIM_SPIPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 130 describes class CIM_SPIPort.

Table 130 - SMI Referenced Properties/Methods for CIM_SPIPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus (overridden)		Mandatory	Shall be 0 (Unknown), 2 (OK), 6 (Error), 10 (Stopped), or 11 (In Service).
UsageRestriction (overridden)		Mandatory	Shall be 3 (Back-end Only).
PortType (overridden)		Mandatory	Shall be 140 (SCSI Parallel).

15.6.12CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports. The class definition specializes the CIM_SystemDevice definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 131 describes class CIM_SystemDevice (Initiator Ports).

Table 131 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent (overridden)		Mandatory	Reference to SPIPort(Initiator).

15.6.13SNIA_LogicalPortStatistics

Statistics for a port.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 132 describes class SNIA_LogicalPortStatistics.

Table 132 - SMI Referenced Properties/Methods for SNIA_LogicalPortStatistics

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
BytesTransmitted		Mandatory	
BytesReceived		Mandatory	
PacketsTransmitted		Mandatory	
PacketsReceived		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

16 iSCSI Initiator Port Profile

16.1 Synopsis

Profile Name: iSCSI Initiator Ports (Component Profile)

Version: 1.2.0

Organization: SNIA

CIM Schema Version: 2.13.1

Table 133 describes the related profiles for iSCSI Initiator Ports.

Table 133 - Related Profiles for iSCSI Initiator Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Specializes: Generic Initiator Port Profile

Central Class: CIM_EthernetPort

Scoping Class: a CIM_System in a separate autonomous profile

Models an adapter (NIC, HBA, TOE) for iSCSI.

16.2 Description

Models an adapter (NIC, HBA, TOE) for iSCSI.

16.3 Implementation

Other port profiles have a single physical port (LogicalPort subclass) associated with each SCSI initiator (SCSIProtocolEndpoint). iSCSI allows multiple connections (each with a single Ethernet port) in a session that acts as a SCSI initiator. This profile includes the subset of classes that model the SCSI initiator and its relationship to logical classes that model physical elements (Ethernet ports).

Figure 23 depicts a configuration with an initiator with two Ethernet ports that are part of a single session that acts as a SCSI initiator. The Ethernet ports (referred to in iSCSI literature as Network Portals) are modeled as instances of EthernetPort, IPProtocolEndpoint, and TCPProtocolEndpoint with 1-1 cardinality. These ports are in the initiator side, the target ports are not required in this profile. Note that all ProtocolEndpoint instances need a HostAccessPoint association to the ComputerSystem, some are omitted to keep the diagram less cluttered.

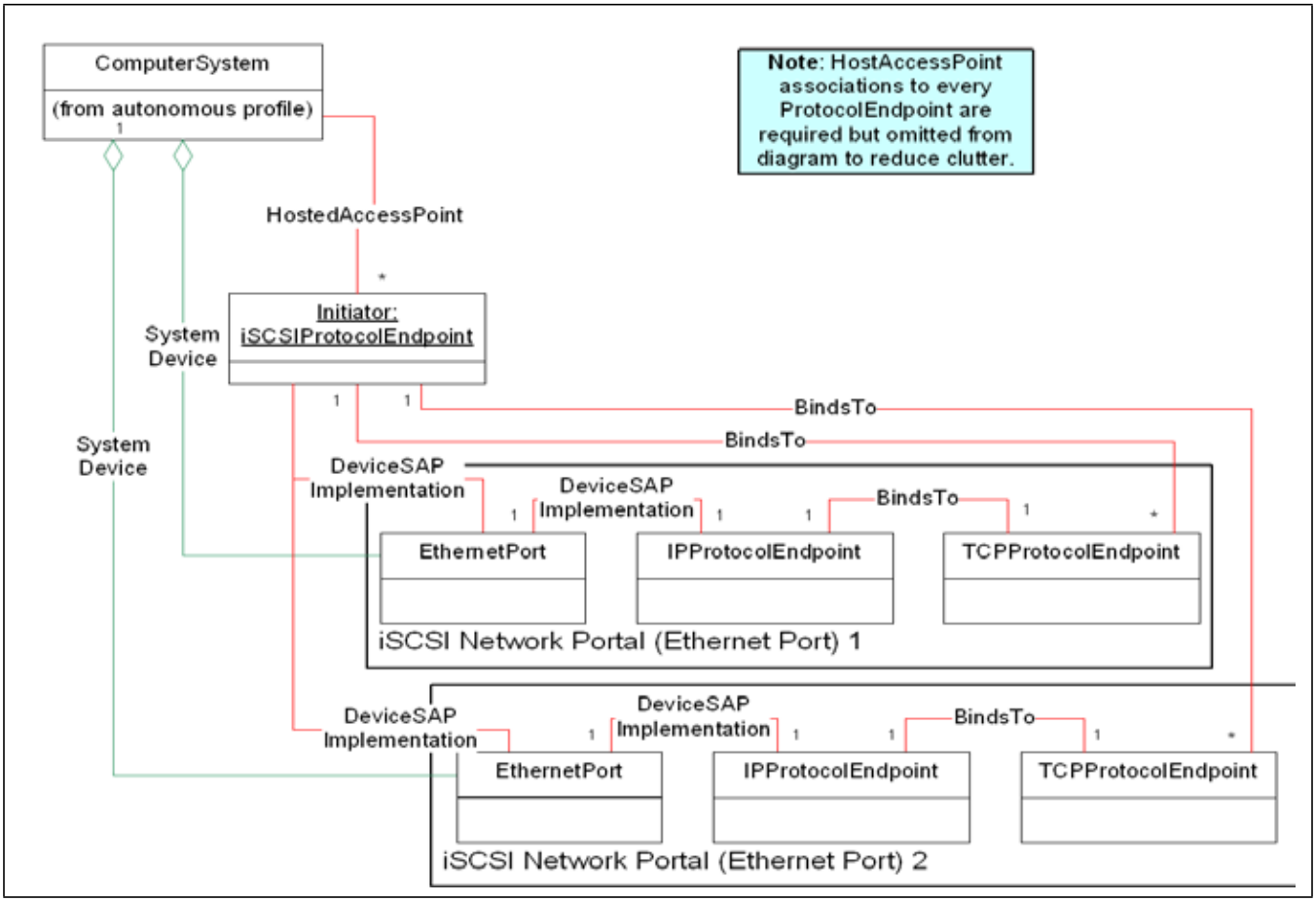

Figure 23 - iSCSI Initiator Port Instance Diagram

16.3.1 Health and Fault Management Considerations

Table 134 describes EthernetPort OperationalStatus.

Table 134 - EthernetPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

16.3.2 Cascading Considerations

Not defined in this standard.

16.4 Methods

16.4.1 Extrinsic Methods of this Profile

None

16.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

16.5 Detailed Use Cases and Recipes

None

16.6 CIM Elements

Table 135 describes the CIM elements for iSCSI Initiator Ports.

Table 135 - CIM Elements for iSCSI Initiator Ports

Element Name	Requirement	Description
16.6.1 CIM_BindsTo (Host Hardware RAID Controller)	Mandatory	
16.6.2 CIM_DeviceSAPImplementation (IPProtocolEndpoint to EthernetPort)	Mandatory	
16.6.3 CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthernetPort)	Mandatory	
16.6.4 CIM_EthernetPort (Host Hardware RAID Controller)	Mandatory	
16.6.5 CIM_HostedAccessPoint (System to IPProtocolEndpoint)	Mandatory	
16.6.6 CIM_HostedAccessPoint (System to TCPProtocolEndpoint)	Mandatory	
16.6.7 CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)	Mandatory	
16.6.8 CIM_IPProtocolEndpoint (Host Hardware RAID Controller)	Mandatory	
16.6.9 CIM_LogicalDevice (Host Hardware RAID Controller)	Optional	

Table 135 - CIM Elements for iSCSI Initiator Ports

Element Name	Requirement	Description
16.6.10 CIM_SystemDevice (System to EthernetPort)	Mandatory	
16.6.11 CIM_SystemDevice (System to LogicalDevice)	Mandatory	
16.6.12 CIM_TCPProtocolEndpoint (Host Hardware RAID Controller)	Mandatory	
16.6.13 CIM_iSCSIProtocolEndpoint (Host Hardware RAID Controller)	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_EthernetPort	Mandatory	Port Creation.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_EthernetPort AND SourceInstance.CIM_EthernetPort::OperationalStatus <> PreviousInstance.CIM_EthernetPort::OperationalStatus	Mandatory	CQL -Port Status Change.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_EthernetPort	Mandatory	Port Removal.

16.6.1 CIM_BindsTo (Host Hardware RAID Controller)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 136 describes class CIM_BindsTo (Host Hardware RAID Controller).

Table 136 - SMI Referenced Properties/Methods for CIM_BindsTo (Host Hardware RAID Controller)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

16.6.2 CIM_DeviceSAPImplementation (IPProtocolEndpoint to EthernetPort)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 137 describes class CIM_DeviceSAPImplementation (IPProtocolEndpoint to EthernetPort).

Table 137 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (IPProtocolEndpoint to EthernetPort)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

16.6.3 CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthernetPort)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 138 describes class CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthernetPort).

Table 138 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthernetPort)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

16.6.4 CIM_EthernetPort (Host Hardware RAID Controller)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 139 describes class CIM_EthernetPort (Host Hardware RAID Controller).

Table 139 - SMI Referenced Properties/Methods for CIM_EthernetPort (Host Hardware RAID Controller)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
PortType		Mandatory	
OperationalStatus		Mandatory	
PermanentAddress	CD	Mandatory	

16.6.5 CIM_HostedAccessPoint (System to IPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 140 describes class CIM_HostedAccessPoint (System to IPProtocolEndpoint).

Table 140 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to IPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

16.6.6 CIM_HostedAccessPoint (System to TCPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 141 describes class CIM_HostedAccessPoint (System to TCPProtocolEndpoint).

Table 141 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to TCPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

16.6.7 CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 142 describes class CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint).

Table 142 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

16.6.8 CIM_IPProtocolEndpoint (Host Hardware RAID Controller)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 143 describes class CIM_IPProtocolEndpoint (Host Hardware RAID Controller).

Table 143 - SMI Referenced Properties/Methods for CIM_IPProtocolEndpoint (Host Hardware RAID Controller)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
IPv4Address	CD	Optional	Maps to IMA_NETWORK_PORTAL_PROPERTIES, ipAddress.
IPv6Address	CD	Optional	Maps to IMA_NETWORK_PORTAL_PROPERTIES, ipAddress.
ProtocolIFType		Mandatory	

16.6.9 CIM_LogicalDevice (Host Hardware RAID Controller)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 144 describes class CIM_LogicalDevice (Host Hardware RAID Controller).

Table 144 - SMI Referenced Properties/Methods for CIM_LogicalDevice (Host Hardware RAID Controller)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	
OperationalStatus		Mandatory	

16.6.10CIM_SystemDevice (System to EthernetPort)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 145 describes class CIM_SystemDevice (System to EthernetPort).

Table 145 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to EthernetPort)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

16.6.11CIM_SystemDevice (System to LogicalDevice)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 146 describes class CIM_SystemDevice (System to LogicalDevice).

Table 146 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to LogicalDevice)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

16.6.12CIM_TCPProtocolEndpoint (Host Hardware RAID Controller)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 147 describes class CIM_TCPProtocolEndpoint (Host Hardware RAID Controller).

Table 147 - SMI Referenced Properties/Methods for CIM_TCPProtocolEndpoint (Host Hardware RAID Controller)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
PortNumber	CD	Mandatory	
ProtocolIFType		Mandatory	

16.6.13CIM_iSCSIProtocolEndpoint (Host Hardware RAID Controller)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 148 describes class CIM_iSCSIProtocolEndpoint (Host Hardware RAID Controller).

Table 148 - SMI Referenced Properties/Methods for CIM_iSCSIProtocolEndpoint (Host Hardware RAID Controller)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	CD	Mandatory	
ProtocolIFType		Mandatory	Other.
OtherTypeDescription		Mandatory	
ConnectionType		Mandatory	iSCSI.
Role		Mandatory	Shall be 2 (Initiator).
Identifier		Mandatory	ISID.

STABLE

17 FC Initiator Ports Profile

17.1 Synopsis

Profile Name: FC Initiator Ports (Component Profile)

Version: 1.6.0

Organization: SNIA

CIM Schema Version: 2.27.0

Table 149 describes the related profiles for FC Initiator Ports.

Table 149 - Related Profiles for FC Initiator Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Specializes: Generic Initiator Ports Profile

Central Class: CIM_FCPort

Scoping Class: a CIM_System in a referencing autonomous profile

The FC Initiator Ports Profile models the behavior of a Fibre Channel port supporting FCP (SCSI command protocol).

17.2 Description

The FC Initiator Ports Profile models the behavior of a Fibre Channel port supporting FCP (SCSI command protocol).

17.3 Implementation

Figure 24 is an example of a single port and drive connected to a single system using Fibre Channel. This instance diagram shows a disk (LogicalDevice in the diagram would be subclassed as something like StorageExtent) in an array, connected by a Fibre Channel port. The full model for the disk is shown in *Storage Management Technical Specification, Part 4 Block Devices, 1.6.1 Rev 5* 11 Disk Drive Lite Subprofile.

SCSIProtocolController is not generally used in initiator contexts. It is included here to be compatible with SMI-S 1.0 clients.

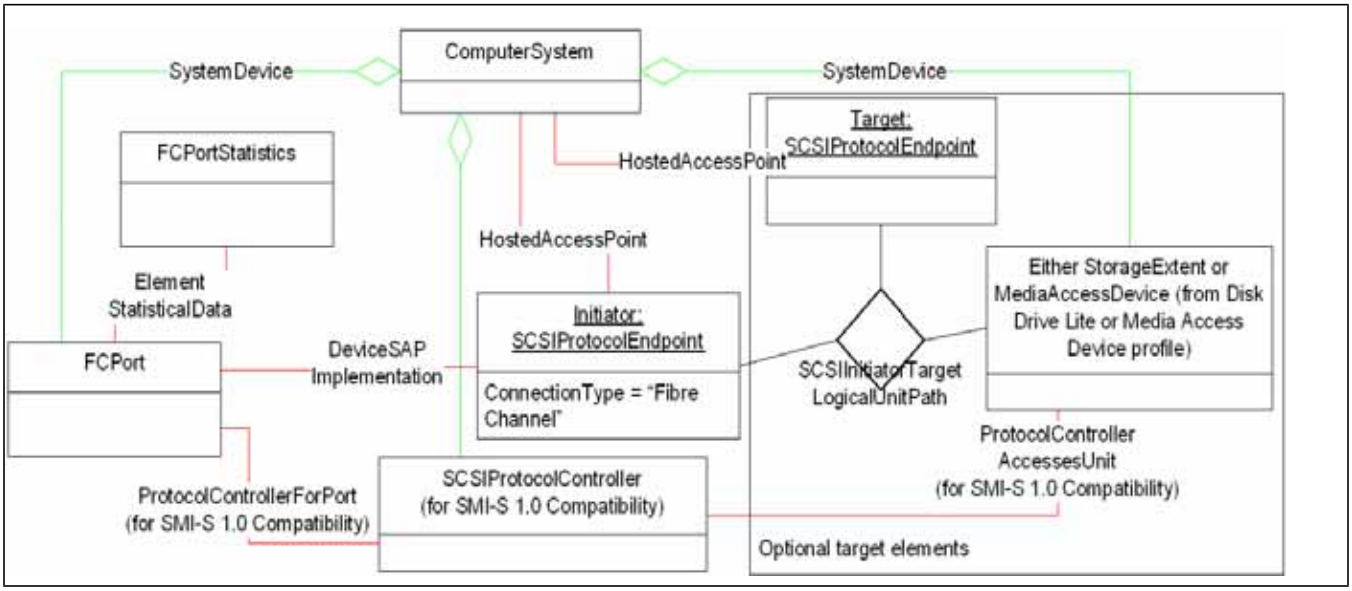

Figure 24 - Fibre Channel Initiator Instance Diagram

17.3.1 Port Statistics

The FCPortStatistics subclass of NetworkPortStatistics is optional. If supported, FCPortStatistics shall be associated to FcPort using ElementStatisticalData.

17.3.2 Logical Port Group (FC Node)

LogicalPortGroup may optionally be used to model the collection of ports that shared a Node WWN (in this case, both ports on a card, but other implementations are in use). If LogicalPortGroup is instantiated, it shall be associated to the ComputerSystem in the referencing profile using HostedCollection and also associated to FCPorts using MemberOfCollection. Figure 25, "FC Node Model" shows to model for FC Nodes.

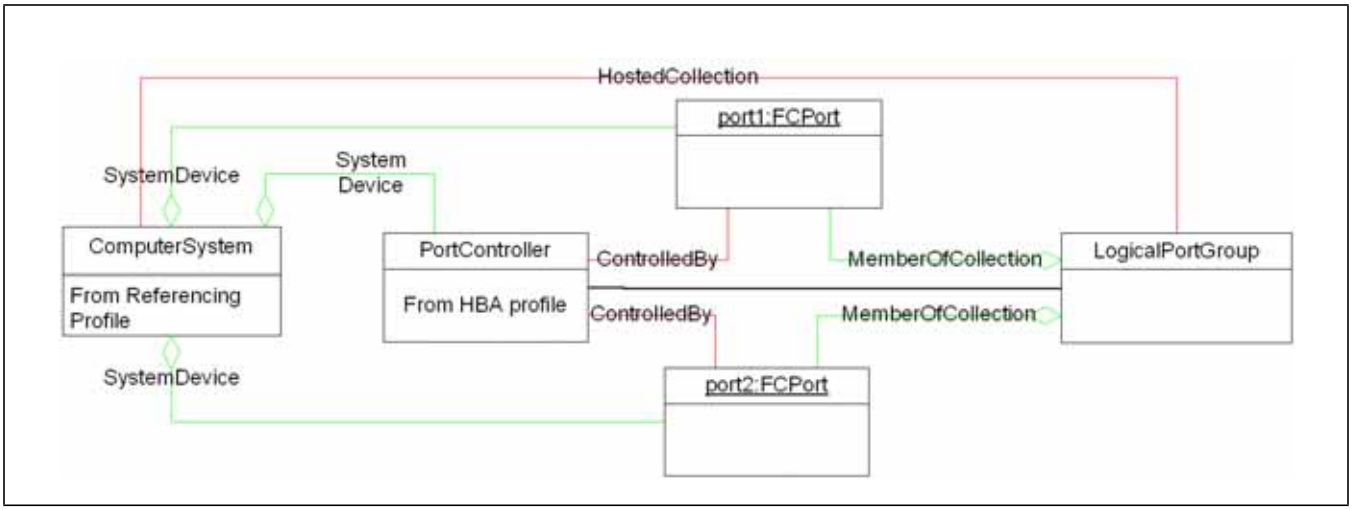

Figure 25 - FC Node Model

17.3.3 Health and Fault Management Considerations

Table 150 summarized the Health and Fault Management considerations specific to this profile.

Table 150 - FCPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

17.3.4 Cascading Considerations

Not defined in this standard.

17.4 Methods

17.4.1 Extrinsic Methods of this Profile

None

17.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

17.5 Use Cases

17.5.1 Get the statistics for each FC port

This recipe is optional and assumes an FCPortStatistics instance is defined for each FCPort instance.

```
//
// DESCRIPTION
//
// Find the FCPortStatistics associated with FC ports
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. A reference to the top-level ComputerSystem in the FC HBA Profile,
```

```
//    which represents the system hosting the HBA, is known as $Host->
//
// Get a list of all the ports
$Ports->[] = AssociatorNames($Host->,// ObjectName
     “CIM_SystemDevice”,// AssocClass
     “CIM_FCPort”,// ResultClass
     “GroupComponent”,   // Role
     “PartComponent”)    // ResultRole

if ($Ports->[] == null || $Ports->[].length == 0) {
     <ERROR! No FC Ports on the host system!>
}

for (#i in $Ports->[] ) {
    // Get a list of FCPortStatistics associated with each port
    // Should only be exactly one FCPortStatistics instance
    $Stats->[] = AssociatorNames($Ports->[#i],// ObjectName
     “CIM_ElementStatisticalData”,// AssocClass
     “CIM_FCPortStatistics”,// ResultClass
     “ManagedElement”,// Role
     “Stats”)     // ResultRole
    if ($Stats->[] == null || $Ports->[].length == 0) {
     <ERROR! Each FCPort shall have an associated FCPortStatistics>
    } else {
        if ( $Stats->[].length > 1) {
         <ERROR: More than 1 FCPortStatistics associated with a port>
     }
    }
    // $Stats[0]-> holds that stats
}
```

17.6 CIM Elements

Table 151 describes the CIM elements for FC Initiator Ports.

Table 151 - CIM Elements for FC Initiator Ports

Element Name	Requirement	Description
17.6.1 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
17.6.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator FCPort and SCSIProtocolEndpoint.
17.6.3 CIM_ElementStatisticalData (Port Statistics)	Conditional	Conditional requirement: support for the FC HBA profile. Connects FCPort and FCPortStatistics.
17.6.4 CIM_FCPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
17.6.5 CIM_FCPortStatistics	Conditional	Conditional requirement: support for the FC HBA profile. Statistics for a port.
17.6.6 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
17.6.7 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.

Table 151 - CIM Elements for FC Initiator Ports

Element Name	Requirement	Description
17.6.8 CIM_HostedCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates the ConnectivityCollection to the hosting System.
17.6.9 CIM_MemberOfCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates SCSIProtocolEndpoints to the ConnectivityCollection.
17.6.10 CIM_ProtocolControllerForPort	Optional	Deprecated. Associates SCSIProtocolController to FCPort.
17.6.11 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
17.6.12 CIM_SCSIProtocolController	Optional	Deprecated. Represents a SCSI logical unit inventory.
17.6.13 CIM_SCSIProtocolEndpoint (Initiator)	Mandatory	Represents support for the SCSI command set.
17.6.14 CIM_SCSIProtocolEndpoint (Target)	Optional	Models remote ports - target devices and possibly other initiators.
17.6.15 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort	Optional	Create FCPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <>PreviousInstance.OperationalStatus	Optional	Deprecated WQL -Modify FCPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::OperationalStatus <>PreviousInstance.CIM_FCPort::OperationalStatus	Optional	CQL -Modify FCPort.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort	Optional	Delete FCPort.

17.6.1 CIM_ConnectivityCollection

Represents a collection of connected ProtocolEndpoints.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 152 describes class CIM_ConnectivityCollection.

Table 152 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	

17.6.2 CIM_DeviceSAPImplementation

Connects Initiator FCPort and SCSIProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 153 describes class CIM_DeviceSAPImplementation.

Table 153 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to Initiator SCSIProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to FCPort.

17.6.3 CIM_ElementStatisticalData (Port Statistics)

Connects FCPort and FCPortStatistics. The class definition specializes the CIM_ElementStatisticalData definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: support for the FC HBA profile.

Table 154 describes class CIM_ElementStatisticalData (Port Statistics).

Table 154 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Port Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement (overridden)		Mandatory	Reference to FCPort.
Stats (overridden)		Mandatory	Reference to FCPortStatistics.

17.6.4 CIM_FCPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 155 describes class CIM_FCPort.

Table 155 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus (overridden)		Mandatory	Shall be 0 (Unknown), 2 (OK), 6 (Error), 10 (Stopped), or 11 (In Service).
UsageRestriction		Mandatory	Shall be 3 for ports restricted to back-end (initiator) only or 4 if the port is unrestricted.
PortType (overridden)		Mandatory	Shall be 0|1|10|11|12|13|14|15|16|17|18 (Unknown or Other or N or NL or F/NL or Nx or E or F or FL or B or G).
ElementName (added)		Mandatory	Port Symbolic Name.
Speed (added)		Mandatory	Speed in bits per second. Shall be 0, 1062500000 (1GFC), 2125000000 (2GFC), 4250000000 (4GFC), 8500000000 (8GFC), 10518750000 (10GFC), 14025000000 (16GFC), 21037500000 (20GFC) or 28500000000 (32GFC).
MaxSpeed (added)		Mandatory	Maximum Port Speed.
PortNumber (added)		Optional	
PermanentAddress (added)	CD	Optional	Port WWN. PermanentAddress is optional when used as a back-end port in a device. This may be overridden in profiles that use this profile. Shall be 16 un-separated upper case hex digits. See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
NetworkAddresses (added)		Optional	For Fibre Channel end device ports, the Fibre Channel ID. Shall be 16 un-separated upper case hex digits.
SupportedCOS (added)		Optional	Shall be 0 (unknown), 1 (Class 1), 2 (Class 2), 3, (Class 3), 4 (Class 4), 6 (Class 6), or 7 (Class 7).
ActiveCOS (added)		Optional	Shall be 0 (unknown), 1 (Class 1), 2 (Class 2), 3, (Class 3), 4 (Class 4), 6 (Class 6), or 7 (Class 7).
SupportedFC4Types (added)		Optional	
ActiveFC4Types (added)		Optional	
LinkTechnology (added)		Mandatory	Shall be 4 (FC).
SupportedMaximumTransmissionUnit (added)		Mandatory	
ActiveMaximumTransmissionUnit (added)		Optional	
PortDiscriminator (added)		Conditional	Experimental. Conditional requirement: support for the Storage HBA profile.Shall include 11 (FC Native) and 12 (HBA).

17.6.5 CIM_FCPortStatistics

Statistics for a port. The class definition specializes the SNIA_LogicalPortStatistics definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: support for the FC HBA profile.

Table 156 describes class CIM_FCPortStatistics.

Table 156 - SMI Referenced Properties/Methods for CIM_FCPortStatistics

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
BytesTransmitted (overridden)		Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.TxWords. Multiply word count by 4.
BytesReceived (overridden)		Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.RxWords. Multiply word count by 4.
PacketsTransmitted (overridden)		Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.TxFrames.
PacketsReceived (overridden)		Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.RxFrames.
CRCErrors (added)		Mandatory	Maps to HBA_PortStatistics.InvalidCRCCount.
LinkFailures (added)		Mandatory	Maps to HBA_PortStatistics.LinkFailureCount.
PrimitiveSeqProtocolErrCount (added)		Mandatory	
LossOfSignalCounter (added)		Mandatory	Maps to HBA_PortStatistics.LossOfSignalCount.
InvalidTransmissionWords (added)		Mandatory	Maps to HBA_PortStatistics.InvalidTxWordCount.
StatisticTime (added)		Optional	Time last measurement was taken.
LIPCount (added)		Mandatory	
NOSCount (added)		Mandatory	
ErrorFrames (added)		Mandatory	
DumpedFrames (added)		Mandatory	
LossOfSyncCounter (added)		Mandatory	Maps to HBA_PortStatistics.LossOfSynchCount.

17.6.6 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 157 describes class CIM_HostedAccessPoint (Initiator).

Table 157 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Initiator).

17.6.7 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 158 describes class CIM_HostedAccessPoint (Target).

Table 158 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Target).

17.6.8 CIM_HostedCollection (Connectivity Collection)

Associates the ConnectivityCollection to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 159 describes class CIM_HostedCollection (Connectivity Collection).

Table 159 - SMI Referenced Properties/Methods for CIM_HostedCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ConnectivityCollection.
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.

17.6.9 CIM_MemberOfCollection (Connectivity Collection)

Associates SCSIProtocolEndpoints to the ConnectivityCollection. The class definition specializes the CIM_MemberOfCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 160 describes class CIM_MemberOfCollection (Connectivity Collection).

Table 160 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Member (overridden)		Mandatory	Reference to target or initiator SCSIProtocolEndpoint.
Collection		Mandatory	Reference to ConnectivityCollection.

17.6.10CIM_ProtocolControllerForPort

Deprecated. Associates SCSIProtocolController to FCPort.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 161 describes class CIM_ProtocolControllerForPort.

Table 161 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForPort

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to SCSIProtocolController.
Dependent		Mandatory	Reference to initiator FCPort.

17.6.11CIM_SCSIInitiatorTargetLogicalUnitPath

Represents a path between a SCSI initiator, target, and logical unit.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 162 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 162 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
LogicalUnit		Mandatory	Reference to StorageExtent in Disk Drive Lite Profile or MediaAccessDevice in Media Access Device Profile.
Initiator		Mandatory	Reference to SCSIProtocolEndpoint(Initiator).
Target		Mandatory	Reference to SCSIProtocolEndpoint(Target).

17.6.12CIM_SCSIProtocolController

Deprecated. Represents a SCSI logical unit inventory.

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 163 describes class CIM_SCSIProtocolController.

Table 163 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	
OperationalStatus		Optional	
MaxUnitsControlled		Optional	

17.6.13CIM_SCSIProtocolEndpoint (Initiator)

Represents support for the SCSI command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 164 describes class CIM_SCSIProtocolEndpoint (Initiator).

Table 164 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
ConnectionType (added)		Mandatory	Shall be 2 (Fibre Channel).
Role (added)		Mandatory	Shall be 2 (Initiator) or 4 (Both Initiator and Target).

17.6.14CIM_SCSIProtocolEndpoint (Target)

Models remote ports - target devices and possibly other initiators. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 165 describes class CIM_SCSIProtocolEndpoint (Target).

Table 165 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	The values in MOFs map to IETF values and exclude storage. Shall be 1 (Other) and set OtherTypeDescription appropriately.
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
Role (added)		Mandatory	Should be set appropriately by the instrumentation. If not known, use 0 (Unknown).
ConnectionType (added)		Mandatory	Shall be 8 (FC).

17.6.15CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports. The class definition specializes the CIM_SystemDevice definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 166 describes class CIM_SystemDevice (Initiator Ports).

Table 166 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent (overridden)		Mandatory	Reference to FCPort.

STABLE

EXPERIMENTAL

18 SAS Initiator Ports Profile

18.1 Synopsis

Profile Name: SAS Initiator Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.11.0

Related Profiles for SAS Initiator Ports: Not defined in this standard.

Specializes: Generic Initiator Port Profile

Central Class: CIM_SASPort

Scoping Class: a CIM_System in a separate autonomous profile

The SAS Initiator Port Profile models the management of a Serial Attached SCSI port that initiates commands to devices.

18.2 Description

The SAS Initiator Port Profile defines the model Serial Attached SCSI (SAS) ports. A typical instance diagram is provided in Figure 26.

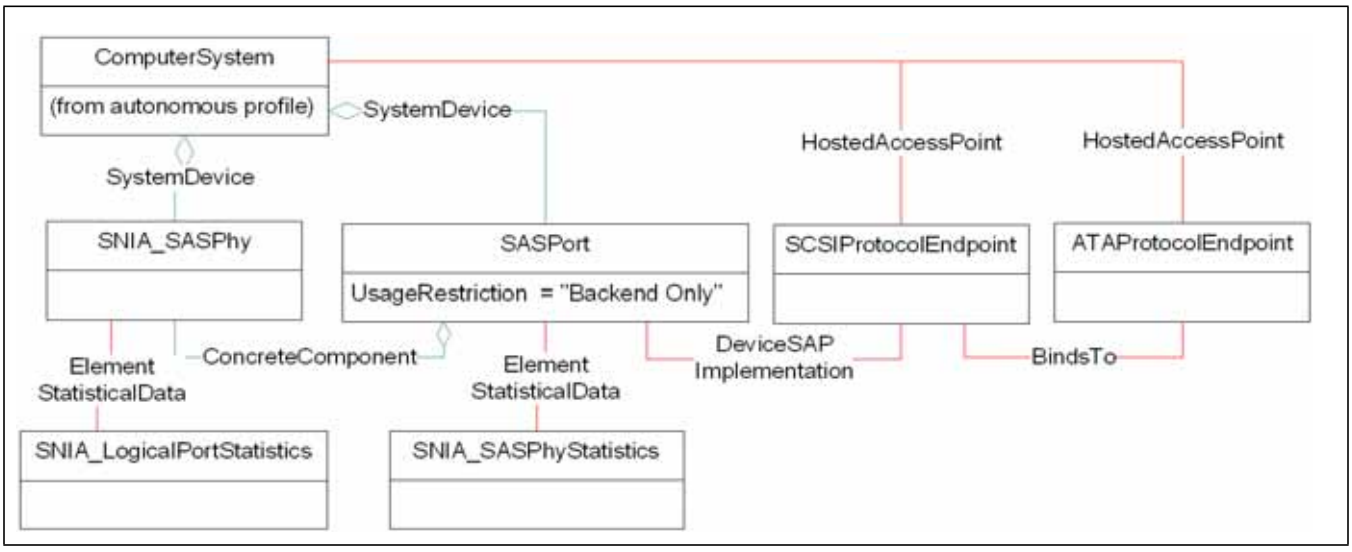

Figure 26 - SAS Initiator Port Model

The SASPhy class represents a SAS PHY. A SAS Port may have multiple associated PHYs; generally, all the PHYs are connected to the same target or expander and provide additional bandwidth.

SASPort represents a SAS initiator port which is an aggregation of SASPHY instances.

SNIA_SASPhyStatistics is optional and may be associated to SASPhy to hold PHY error statistics.

SNIA_LogicalPortStatistics is optional and may be associated to SASPort to hold port I/O statistics.

ATAProtocolEndpoint associated to SCSIProtocolEndpoint using BindsTo shall be used to represent support for ATA (SATA) tunneled over SCSI.

18.2.1 Health and Fault Management Considerations

Table 167 summarizes the Health and Fault Management issues that are unique to this profile.

Table 167 - SASPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

18.3 Methods of the profile

Not defined in this standard.

18.4 Client Considerations and Recipes

Not defined in this standard.

18.5 CIM Elements

Table 168 describes the CIM elements for SAS Initiator Ports.

Table 168 - CIM Elements for SAS Initiator Ports

Element Name	Requirement	Description
18.5.1 CIM_ATAProtocolEndpoint (Initiator)	Optional	Initiator ATA endpoints.
18.5.2 CIM_BindsTo	Optional	Associates SCSIProtocolEndpoint and ATAProtocolEndpoint.
18.5.3 CIM_ConcreteComponent	Mandatory	Associates SASPort and SASPHY.
18.5.4 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
18.5.5 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator SASLogicalPort and SCSIProtocolEndpoint.
18.5.6 CIM_ElementStatisticalData (PHY Statistics)	Optional	Associates SASPort and SASPhyStatistics.
18.5.7 CIM_ElementStatisticalData (Port Statistics)	Optional	Connects LogicalPort and LogicalPortStatistics.
18.5.8 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
18.5.9 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
18.5.10 CIM_HostedCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates the ConnectivityCollection to the hosting System.

Table 168 - CIM Elements for SAS Initiator Ports

Element Name	Requirement	Description
18.5.11 CIM_MemberOfCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates ProtocolEndpoints to the ConnectivityCollection.
18.5.12 CIM_SASPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
18.5.13 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
18.5.14 CIM_SCSIProtocolEndpoint (Initiator)	Mandatory	Represents support for the SCSI command set.
18.5.15 CIM_SCSIProtocolEndpoint (Target)	Optional	Models remote ports - target devices and possibly other initiators.
18.5.16 CIM_SystemDevice (Initiator PHY)	Mandatory	Associates system to initiator SAS PHYs.
18.5.17 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
18.5.18 SNIA_LogicalPortStatistics	Optional	Statistics for a port.
18.5.19 SNIA_SASPHY	Mandatory	A PHY on a SAS HBA, Expander, or device.
18.5.20 SNIA_SASPhyStatistics	Optional	Statistics for a SAS PHY.

18.5.1 CIM_ATAProtocolEndpoint (Initiator)

Initiator ATA endpoints.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 169 describes class CIM_ATAProtocolEndpoint (Initiator).

Table 169 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription		Mandatory	Shall be 'ATA'.
ConnectionType		Mandatory	Shall be 3 (SATA).
Role		Mandatory	Shall be 3 (Target).

18.5.2 CIM_BindsTo

Associates SCSIProtocolEndpoint and ATAProtocolEndpoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 170 describes class CIM_BindsTo.

Table 170 - SMI Referenced Properties/Methods for CIM_BindsTo

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ATAProtocolEndpoint.
Antecedent		Mandatory	Reference to SCSIProtocolEndpoint.

18.5.3 CIM_ConcreteComponent

Associates SASPort and SASPHY.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 171 describes class CIM_ConcreteComponent.

Table 171 - SMI Referenced Properties/Methods for CIM_ConcreteComponent

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	Reference to SASPHY.
GroupComponent		Mandatory	Reference to SASPort.

18.5.4 CIM_ConnectivityCollection

Represents a collection of connected ProtocolEndpoints.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 172 describes class CIM_ConnectivityCollection.

Table 172 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	

18.5.5 CIM_DeviceSAPImplementation

Connects Initiator SASLogicalPort and SCSIProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 173 describes class CIM_DeviceSAPImplementation.

Table 173 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Initiator).
Antecedent (overridden)		Mandatory	Reference to SASPort.

18.5.6 CIM_ElementStatisticalData (PHY Statistics)

Associates SASPort and SASPhyStatistics.

Requirement: Optional

Table 174 describes class CIM_ElementStatisticalData (PHY Statistics).

Table 174 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (PHY Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Referenceto SASPort.
Stats		Mandatory	Reference to SASPhyStatistics.

18.5.7 CIM_ElementStatisticalData (Port Statistics)

Connects LogicalPort and LogicalPortStatistics.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 175 describes class CIM_ElementStatisticalData (Port Statistics).

Table 175 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Port Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Reference to LogicalPort.
Stats		Mandatory	Reference to LogicalPortStatistics.

18.5.8 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 176 describes class CIM_HostedAccessPoint (Initiator).

Table 176 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Initiator).

18.5.9 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 177 describes class CIM_HostedAccessPoint (Target).

Table 177 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint(Target).

18.5.10CIM_HostedCollection (Connectivity Collection)

Associates the ConnectivityCollection to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 178 describes class CIM_HostedCollection (Connectivity Collection).

Table 178 - SMI Referenced Properties/Methods for CIM_HostedCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ConnectivityCollection.
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.

18.5.11CIM_MemberOfCollection (Connectivity Collection)

Associates ProtocolEndpoints to the ConnectivityCollection. The class definition specializes the CIM_MemberOfCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 179 describes class CIM_MemberOfCollection (Connectivity Collection).

Table 179 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Member (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.
Collection		Mandatory	Reference to ConnectivityCollection.

18.5.12CIM_SASPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 180 describes class CIM_SASPort.

Table 180 - SMI Referenced Properties/Methods for CIM_SASPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus (overridden)		Mandatory	Shall be 0 (Unknown), 2 (OK), 6 (Error), 10 (Stopped), or 11 (In Service).
UsageRestriction (overridden)		Mandatory	Shall be 3 (Back-end Only).
PortType (overridden)		Mandatory	Shall be 94 (SAS).
PermanentAddress (added)		Mandatory	SAS Address. Shall be 16 un-separated upper case hex digits. See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.

18.5.13CIM_SCSIInitiatorTargetLogicalUnitPath

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 181 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 181 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
LogicalUnit		Mandatory	Reference to StorageExtent in Disk Drive Lite Profile or MediaAccessDevice in Media Access Device Profile.
Target		Mandatory	Reference to SCSIProtocolEndpoint(Target).
Initiator		Mandatory	Reference to SCSIProtocolEndpoint(Initiatotor).

18.5.14CIM_SCSIProtocolEndpoint (Initiator)

Represents support for the SCSI command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 182 describes class CIM_SCSIProtocolEndpoint (Initiator).

Table 182 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
ConnectionType (added)		Mandatory	Shall be 8 (SAS).
Role (added)		Mandatory	Shall be 2 (Initiator).

18.5.15CIM_SCSIProtocolEndpoint (Target)

Models remote ports - target devices and possibly other initiators. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Optional

Table 183 describes class CIM_SCSIProtocolEndpoint (Target).

Table 183 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	The values in MOFs map to IETF values and exclude storage. Shall be 1 (Other) and set OtherTypeDescription appropriately.
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
Role (added)		Mandatory	SCSI target or initiator role. Should be set appropriately by the instrumentation. If not know, use 0 (Unknown).
ConnectionType (added)		Mandatory	Shall be 8 (SAS).

18.5.16CIM_SystemDevice (Initiator PHY)

Associates system to initiator SAS PHYs.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 184 describes class CIM_SystemDevice (Initiator PHY).

Table 184 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator PHY)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent		Mandatory	Reference to SASPHY.

18.5.17CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports. The class definition specializes the CIM_SystemDevice definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 185 describes class CIM_SystemDevice (Initiator Ports).

Table 185 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent (overridden)		Mandatory	Reference to back-end (initiator) SASPorts.

18.5.18SNIA_LogicalPortStatistics

Statistics for a port.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 186 describes class SNIA_LogicalPortStatistics.

Table 186 - SMI Referenced Properties/Methods for SNIA_LogicalPortStatistics

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
BytesTransmitted		Mandatory	
BytesReceived		Mandatory	
PacketsTransmitted		Mandatory	
PacketsReceived		Mandatory	

18.5.19SNIA_SASPHY

A PHY on a SAS HBA, Expander, or device.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 187 describes class SNIA_SASPHY.

Table 187 - SMI Referenced Properties/Methods for SNIA_SASPHY

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	

Table 187 - SMI Referenced Properties/Methods for SNIA_SASPHY

Properties	Flags	Requirement	Description & Notes
HardwareMinimumPhysicalLinkRate		Mandatory	
HardwareMaximumPhysicalLinkRate		Mandatory	
ProgrammedMinimumPhysicalLinkRate		Mandatory	
ProgrammedMaximumPhysicalLinkRate		Mandatory	
NegotiatedPhysicalLinkRate		Mandatory	

18.5.20SNIA_SASPhyStatistics

Statistics for a SAS PHY.

Requirement: Optional

Table 188 describes class SNIA_SASPhyStatistics.

Table 188 - SMI Referenced Properties/Methods for SNIA_SASPhyStatistics

Properties	Flags	Requirement	Description & Notes
InvalidDwordCount		Mandatory	
RunningDisparityErrorCount		Mandatory	
LossOfDwordSyncCount		Mandatory	
ResetProblemCount		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

19 ATA Initiator Ports Profile

19.1 Synopsis

Profile Name: ATA Initiator Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.13.1

Related Profiles for ATA Initiator Ports: Not defined in this standard.

Specializes: Generic Initiator Port Profile

Central Class: CIM_ATAPort

Scoping Class: a CIM_System in a separate autonomous profile

The ATA Initiator Ports Profile models the management of a PATA or SATA port that initiates commands to devices.

19.2 Description

The ATA Initiator Port Profile describes the model for Parallel or Serial ATA Ports with optional attached drives.

19.3 Implementation

The port is modeled as ATAPort (with PortType set to ATA for PATA ports or SATA) and ATAProtocolEndpoint associated by DeviceSAPImplementation. Attached drives are optionally modeled as subclasses of LogicalDevice (e.g., StorageVolume, TapeDrive) which are associated via SAPAvailableToElement to ATAProtocolEndpoint.

Figure 27 shows a class diagram for this profile.

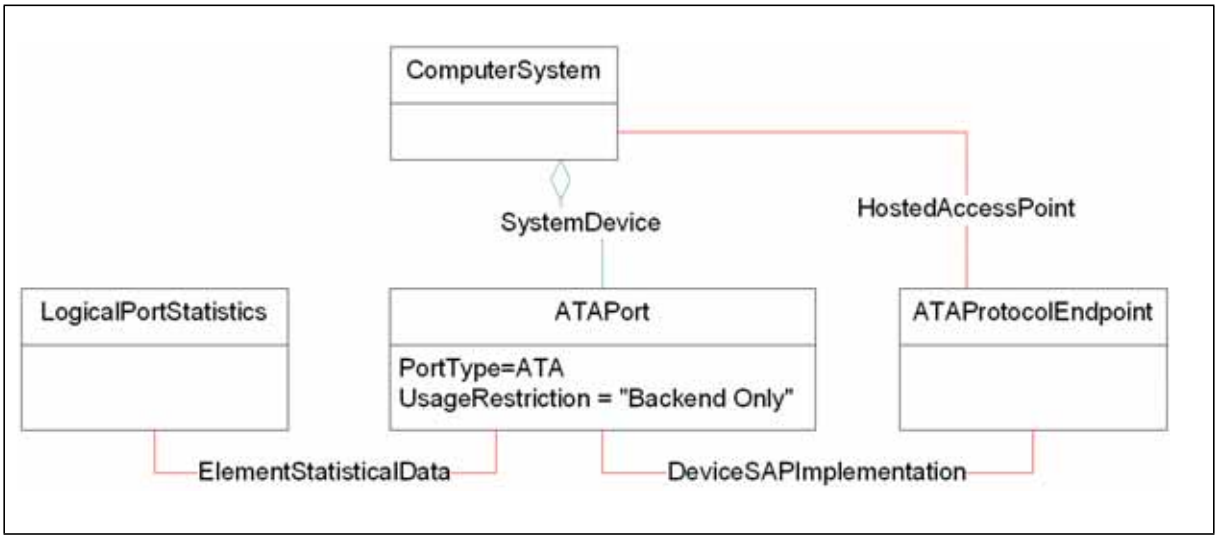

Figure 27 - ATA Initiator Port Class Diagram

19.3.1 Health and Fault Management Consideration

Table 189 summarizes the Health and Fault Management considerations that are specific to this profile.

Table 189 - ATAPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

19.3.2 Cascading Considerations

Not defined in this standard.

19.4 Methods of the Profile

19.4.1 Extrinsic Methods of the Profile

None.

19.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

19.5 Client Considerations and Recipes

None

19.6 CIM Elements

Table 190 describes the CIM elements for ATA Initiator Ports.

Table 190 - CIM Elements for ATA Initiator Ports

Element Name	Requirement	Description
19.6.1 CIM_ATAInitiatorTargetLogicalUnitPath	Optional	Represents a path between an ATA initiator, target, and logical unit.
19.6.2 CIM_ATAPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
19.6.3 CIM_ATAProtocolEndpoint (Initiator)	Mandatory	ProtocolEndpoints associated to initiator ports.
19.6.4 CIM_ATAProtocolEndpoint (Target)	Mandatory	Models remote ports - target devices and possibly other initiators.
19.6.5 CIM_ConnectivityCollection	Mandatory	Represents a collection of connected ATAProtocolEndpoints.
19.6.6 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator ATALogicalPort and ATAProtocolEndpoint.
19.6.7 CIM_ElementStatisticalData (Port Statistics)	Mandatory	Connects ATAPort and LogicalPortStatistics.
19.6.8 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
19.6.9 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
19.6.10 CIM_HostedCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates the ConnectivityCollection to the hosting System.
19.6.11 CIM_MemberOfCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Represents a collection of connected ATAProtocolEndpoints.
19.6.12 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
19.6.13 SNIA_LogicalPortStatistics	Optional	Statistics for a port.

19.6.1 CIM_ATAInitiatorTargetLogicalUnitPath

Represents a path between an ATA initiator, target, and logical unit.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 191 describes class CIM_ATAInitiatorTargetLogicalUnitPath.

Table 191 - SMI Referenced Properties/Methods for CIM_ATAInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
LogicalUnit		Mandatory	Reference to StorageExtent in Disk Drive Lite Profile or MediaAccessDevice in Media Access Device Profile.
Target		Mandatory	Reference to ATAProtocolEndpoint(Target).
Initiator		Mandatory	Reference to ATAProtocolEndpoint(Initiatotor).

19.6.2 CIM_ATAPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 192 describes class CIM_ATAPort.

Table 192 - SMI Referenced Properties/Methods for CIM_ATAPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction		Mandatory	Shall be 3 for ports restricted to back-end (initiator) only or 4 if the port is unrestricted.
PortType (overridden)		Mandatory	Shall be 91(ATA), 92(SATA) or 93(SATA2).
Name (added)	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.

19.6.3 CIM_ATAProtocolEndpoint (Initiator)

ProtocolEndpoints associated to initiator ports. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 193 describes class CIM_ATAProtocolEndpoint (Initiator).

Table 193 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name (overridden)	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
ProtocolIFType		Mandatory	Shall be 1 (Other).

Table 193 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
OtherTypeDescription (overridden)		Mandatory	Shall be 'ATA'.
Role (added)		Mandatory	Shall be 2 (Initiator).
ConnectionType (added)		Mandatory	Shall be 2 (ATA for PATA ports) or 3 (SATA).

19.6.4 CIM_ATAProtocolEndpoint (Target)

Models remote ports - target devices and possibly other initiators. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 194 describes class CIM_ATAProtocolEndpoint (Target).

Table 194 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	The values in MOFs map to IETF values and exclude storage. Shall be 1 (Other) and set OtherTypeDescription appropriately.
OtherTypeDescription (overridden)		Mandatory	Shall be 'ATA'.
Role (added)		Mandatory	Should be set appropriately by the instrumentation. If unknown, use 0 (Unknown).
ConnectionType (added)		Mandatory	Shall be 2 (ATA for PATA ports) or 3 (SATA).

19.6.5 CIM_ConnectivityCollection

Represents a collection of connected ATAProtocolEndpoints. The class definition specializes the CIM_ConnectivityCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 195 describes class CIM_ConnectivityCollection.

Table 195 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	

19.6.6 CIM_DeviceSAPImplementation

Connects Initiator ATALogicalPort and ATAProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 196 describes class CIM_DeviceSAPImplementation.

Table 196 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to ATAProtocolEndpoint(Initiator).
Antecedent (overridden)		Mandatory	Reference to ATAPort.

19.6.7 CIM_ElementStatisticalData (Port Statistics)

Connects ATAPort and LogicalPortStatistics. The class definition specializes the CIM_ElementStatisticalData definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 197 describes class CIM_ElementStatisticalData (Port Statistics).

Table 197 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Port Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement (overridden)		Mandatory	Reference to ATAPort.
Stats		Mandatory	Reference to LogicalPortStatistics.

19.6.8 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 198 describes class CIM_HostedAccessPoint (Initiator).

Table 198 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to ATAProtocolEndpoint(Inititor).

19.6.9 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 199 describes class CIM_HostedAccessPoint (Target).

Table 199 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent (overridden)		Mandatory	Reference to ATAProtocolEndpoint(Target).

19.6.10CIM_HostedCollection (Connectivity Collection)

Associates the ConnectivityCollection to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 200 describes class CIM_HostedCollection (Connectivity Collection).

Table 200 - SMI Referenced Properties/Methods for CIM_HostedCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ConnectivityCollection.
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.

19.6.11CIM_MemberOfCollection (Connectivity Collection)

Represents a collection of connected ATAProtocolEndpoints. The class definition specializes the CIM_MemberOfCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 201 describes class CIM_MemberOfCollection (Connectivity Collection).

Table 201 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Member (overridden)		Mandatory	Reference to ATAProtocolEndpoint(Initiator or Target).
Collection		Mandatory	Reference to ConnectivityCollection.

19.6.12CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports. The class definition specializes the CIM_SystemDevice definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 202 describes class CIM_SystemDevice (Initiator Ports).

Table 202 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent (overridden)		Mandatory	Reference to ATAPort(Initiator).

19.6.13SNIA_LogicalPortStatistics

Statistics for a port.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 203 describes class SNIA_LogicalPortStatistics.

Table 203 - SMI Referenced Properties/Methods for SNIA_LogicalPortStatistics

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
BytesTransmitted		Mandatory	
BytesReceived		Mandatory	

Table 203 - SMI Referenced Properties/Methods for SNIA_LogicalPortStatistics

Properties	Flags	Requirement	Description & Notes
PacketsTransmitted		Mandatory	
PacketsReceived		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

20 FC-SB-x Initiator Ports Profile

20.1 Synopsis

Profile Name: SB Initiator Ports (Component Profile)

Version: 1.4.0

Organization: SNIA

CIM Schema Version: 2.13.0

Table 204 describes the related profiles for SB Initiator Ports.

Table 204 - Related Profiles for SB Initiator Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

The FC-SB-x Initiator Ports Profile models initiator ports that support the FC-SB-x protocol.

20.2 Description

The FC-SB-x Initiator Ports Profile models initiator ports that support the FC-SB-x protocol.

20.3 Implementation

Figure 28 is an example of a single initiator port. The instance diagram shows a disk (LogicalDevice in the diagram would be subclassed as something like StorageExtent) in an array, connected by a Fibre Channel port. The full model for the disk is shown in the Disk Drive Lite Profile. SBProtocolController is not generally used in initiator contexts. It is included here to be compatible with SMI-S 1.0 clients.

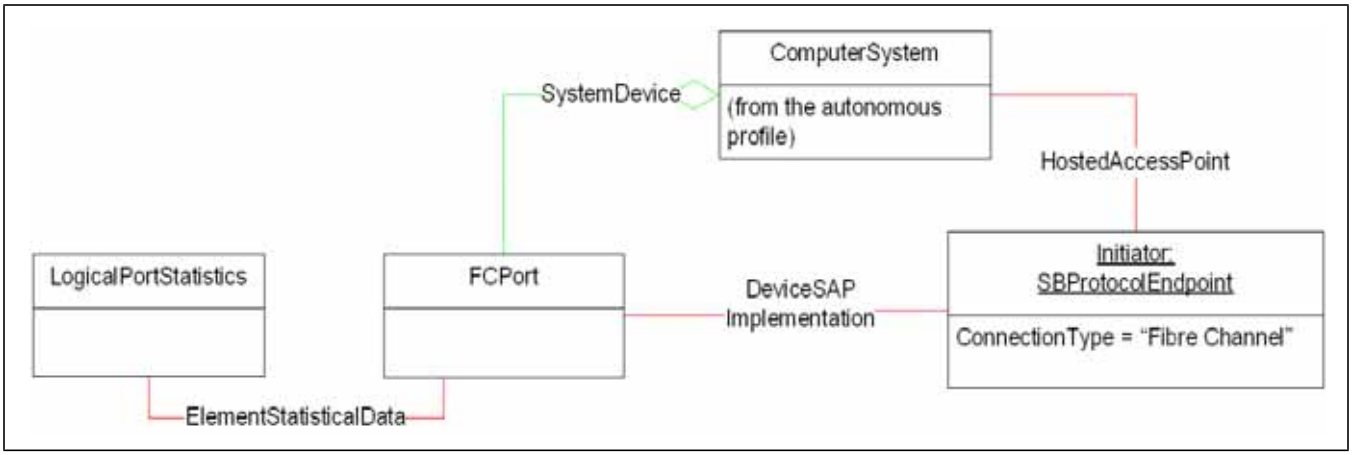

Figure 28 - Fibre Channel Initiator Instance Diagram

20.3.1 Health and Fault Management Considerations

Table 205 summarizes the Health and Fault Management considerations specific to this profile.

Table 205 - FCPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

20.3.2 Cascading Considerations

Not defined in this standard.

20.4 Methods

20.4.1 Extrinsic Methods of the Profile

None.

20.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

20.5 Client Considerations and Recipes

None

20.6 CIM Elements

Table 206 describes the CIM elements for SB Initiator Ports.

Table 206 - CIM Elements for SB Initiator Ports

Element Name	Requirement	Description
20.6.1 CIM_ConnectivityCollection	Mandatory	Represents a collection of connected SBrotocolEndpoints.
20.6.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator SBLogicalPort and SBProtocolEndpoint.
20.6.3 CIM_ElementStatisticalData (Port Statistics)	Mandatory	Connects SBPort and LogicalPortStatistics.
20.6.4 CIM_FCPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
20.6.5 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
20.6.6 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
20.6.7 CIM_HostedCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates the ConnectivityCollection to the hosting System.
20.6.8 CIM_MemberOfCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Represents a collection of connected SBProtocolEndpoints.
20.6.9 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
20.6.10 SNIA_LogicalPortStatistics	Optional	Statistics for a port.
20.6.11 SNIA_SBInitiatorTargetLogicalUnitPath	Optional	
20.6.12 SNIA_SBProtocolEndpoint (Initiator)	Mandatory	Represents a protocol (command set) supported by the port. The appropriate subclass (SCSIProtocolEndpoint, ATAProtocolEndpoint, SBProtocolEndpoint) should be used in initiator port specialized profiles.
20.6.13 SNIA_SBProtocolEndpoint (Target)	Mandatory	Target or non-local ProtocolEndpoint.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort	Optional	Create FCPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::OperationalStatus <> PreviousInstance.CIM_FCPort::OperationalStatus	Optional	CQL -Modify FCPort.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort	Optional	Delete FCPort.

20.6.1 CIM_ConnectivityCollection

Represents a collection of connected SBProtocolEndpoints. The class definition specializes the CIM_ConnectivityCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 207 describes class CIM_ConnectivityCollection.

Table 207 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	

20.6.2 CIM_DeviceSAPImplementation

Connects Initiator SBLogicalPort and SBProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 208 describes class CIM_DeviceSAPImplementation.

Table 208 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SBProtocolEndpoint(Initiator).
Antecedent (overridden)		Mandatory	Reference to FCPort.

20.6.3 CIM_ElementStatisticalData (Port Statistics)

Connects SBPort and LogicalPortStatistics. The class definition specializes the CIM_ElementStatisticalData definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 209 describes class CIM_ElementStatisticalData (Port Statistics).

Table 209 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Port Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement (overridden)		Mandatory	Reference to FCPort.
Stats		Mandatory	Reference to LogicalPortStatistics.

20.6.4 CIM_FCPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 210 describes class CIM_FCPort.

Table 210 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction (overridden)		Mandatory	Shall be 3 for ports restricted to Back-end only or 4 if the port is unrestricted.
PortType (overridden)		Mandatory	Shall be 0|1|10|11|12|13|14|15|16|17|18 (Unknown or Other or N or NL or F/NL or Nx or E or F or FL or B or G).
ElementName (added)		Mandatory	Port Symbolic Name.
Speed (added)		Mandatory	
MaxSpeed (added)		Mandatory	Port Supported Speed from HBA API.
PortNumber (added)		Optional	
PermanentAddress (added)	CD	Optional	Port WWN. PermanentAddress is optional when used as a backend port in a device. This may be overridden in profiles that use this profile. See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
NetworkAddresses (added)		Optional	For Fibre Channel end device ports, the Fibre Channel ID.
SupportedCOS (added)		Optional	
ActiveCOS (added)		Optional	
SupportedFC4Types (added)		Optional	
ActiveFC4Types (added)		Optional	
LinkTechnology (added)		Mandatory	
SupportedMaximumTransmissionUnit (added)		Mandatory	
ActiveMaximumTransmissionUnit (added)		Optional	

20.6.5 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 211 describes class CIM_HostedAccessPoint (Initiator).

Table 211 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent		Mandatory	Reference to ProtocolEndpoint(Initiator).

20.6.6 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 212 describes class CIM_HostedAccessPoint (Target).

Table 212 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent		Mandatory	Reference to ProtocolEndpoint(Target).

20.6.7 CIM_HostedCollection (Connectivity Collection)

Associates the ConnectivityCollection to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 213 describes class CIM_HostedCollection (Connectivity Collection).

Table 213 - SMI Referenced Properties/Methods for CIM_HostedCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ConnectivityCollection.
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.

20.6.8 CIM_MemberOfCollection (Connectivity Collection)

Represents a collection of connected SBProtocolEndpoints. The class definition specializes the CIM_MemberOfCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 214 describes class CIM_MemberOfCollection (Connectivity Collection).

Table 214 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Member (overridden)		Mandatory	Reference to SBProtocolEndpoint(Initiator or Target).
Collection		Mandatory	Reference to ConnectivityCollection.

20.6.9 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 215 describes class CIM_SystemDevice (Initiator Ports).

Table 215 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent		Mandatory	Reference to LogicalPort.

20.6.10SNIA_LogicalPortStatistics

Statistics for a port.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 216 describes class SNIA_LogicalPortStatistics.

Table 216 - SMI Referenced Properties/Methods for SNIA_LogicalPortStatistics

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
BytesTransmitted		Mandatory	
BytesReceived		Mandatory	
PacketsTransmitted		Mandatory	
PacketsReceived		Mandatory	

20.6.11SNIA_SBInitiatorTargetLogicalUnitPath

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 217 describes class SNIA_SBInitiatorTargetLogicalUnitPath.

Table 217 - SMI Referenced Properties/Methods for SNIA_SBInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
UsePreferredPath		Optional	Boolean indicating whether preferred path processing is required.
PreferredPath		Optional	Boolean indicating whether this is a preferred path.
PathGroupState		Optional	One of 0(Unknown), 2(Path grouping not supported), 3(Reset), 4(Grouped), or 5(Ungrouped).
PathGroupMode		Optional	One of 0(Unknown), 2(None), 3(Single path), or 4(Multipath). Single path and multipath only valid if PathGroupState is grouped.
PathGroupID		Optional	String containing the ID from the OS, only valid if PathGroupState is Grouped.
LogicalUnit		Mandatory	Reference to StorageExtent in Disk Drive Lite Profile or MediaAccessDevice in Media Access Device Profile.
Target		Mandatory	Reference to SCSIProtocolEndpoint(Target).
Initiator		Mandatory	Reference to SCSIProtocolEndpoint(Initiator).

20.6.12SNIA_SBProtocolEndpoint (Initiator)

Represents a protocol (command set) supported by the port. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 218 describes class SNIA_SBProtocolEndpoint (Initiator).

Table 218 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
ProtocolIFType (overridden)		Mandatory	Shall be 1 (Other).

Table 218 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
OtherTypeDescription (overridden)		Mandatory	Shall be 'SB'.
ConnectionType (added)		Mandatory	Shall be 2 (Fibre Channel).
Role (added)		Mandatory	Shall be 2 (Initiator) or 4 (Both Initiator and Target).

20.6.13SNIA_SBProtocolEndpoint (Target)

Target or non-local ProtocolEndpoint. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 219 describes class SNIA_SBProtocolEndpoint (Target).

Table 219 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType (overridden)		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be 'SB'.
Role (added)		Mandatory	Should be set appropriately by the instrumentation. If not know, use 0 (Unknown).
ConnectionType (added)		Mandatory	Shall be 2 (Fibre Channel).

EXPERIMENTAL

DEPRECATED

21 Backend Ports Subprofile

The functionality of the Backend Ports Subprofile has been subsumed by 17 FC Initiator Ports Profile.

The Backend Ports Subprofile is defined in section 7.3.3.13 of SMI-S 1.0.2. Any instrumentation that complies to the Fibre Channel Initiator Port Profile defined in this specification may also claim compliance to that version of the Backend Ports Subprofile and may register as both a 1.2.0 Fibre Channel Initiator Port Subprofile and 1.0.2 Backend Ports Subprofile.

DEPRECATED

EXPERIMENTAL

22 FCoE Initiator Ports Profile

22.1 Synopsis

Profile Name: FCoE Initiator Ports (Component Profile)

Version: 1.6.0

Organization: SNIA

CIM Schema Version: 2.27.0

Related Profiles for FCoE Initiator Ports: Not defined in this standard.

Specializes: Generic Initiator Ports Profile

Central Class: CIM_FCPort

Scoping Class: CIM_ComputerSystem in the Base Server Profile (or some other autonomous profile)

The FCoE Initiator Ports Profile models the behavior of the Fibre Channel over Ethernet (FCoE) functionality of a Converged Network Adaptor (CNA).

22.2 Description

The FCoE Initiator Ports Profile is a component profile that models the behavior of the Fibre Channel over Ethernet (FCoE) functionality of a Converged Network Adaptor (CNA).

A CNA may support functionality beyond FCoE, including generic TCP/IP support. Functionality of CNAs other than FCoE is outside the scope of this profile.

22.3 Implementation

Figure 29 shows the model for classes in the FCoE Initiator Ports Profile

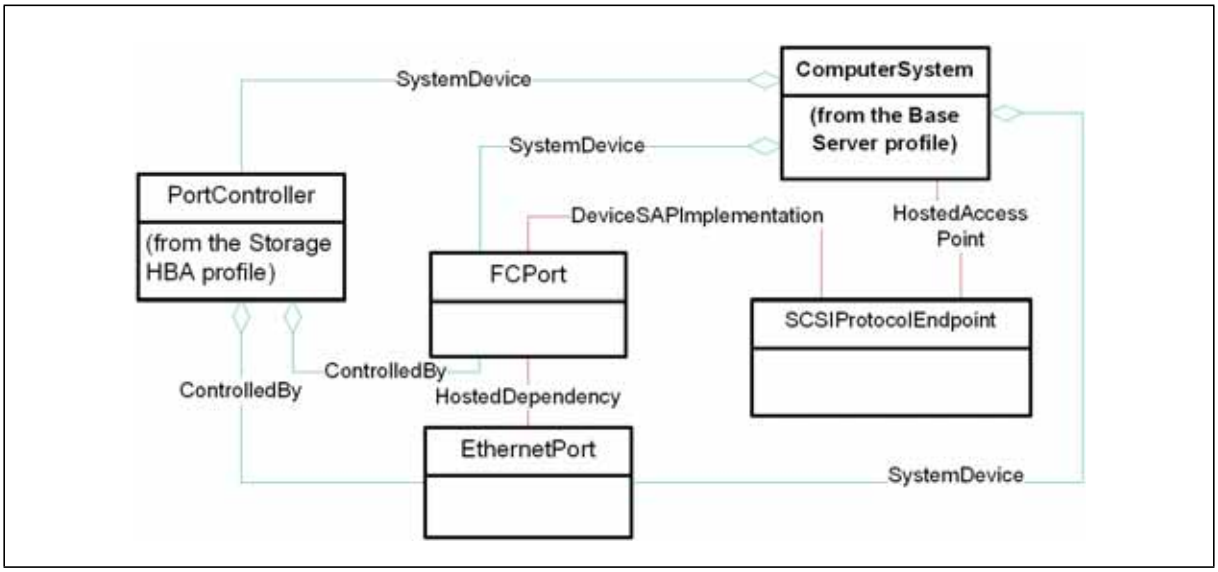

Figure 29 - FCoE Initiator Instance Diagram

Each FCoE port shall be modeled with single instance each of FCPort, EthernetPort, and SCSIProtocolEndpoint. FCPort and EthernetPort shall be associated with HostedDependency. FCPort and EthernetPort shall each be associated to the ComputerSystem in the Base Server Profile with SystemDevice. FCPort and SCSIProtocolEndpoint shall be associated with DeviceSAPImplementation. SCSIProtocolEndpoint shall be associated to the ComputerSystem (defined in the Base Server Profile) with HostedAccessPoint.

22.3.1 Relationship to Storage HBA Profile

The FCoE Initiator ports profile is used in conjunction with the Storage HBA Profile (see *Storage Management Technical Specification, Part 7 Host Elements, 1.6.1 Rev 5* Clause 6: Storage HBA Profile). The Storage HBA Profile models the management of HBA cards independent of connectivity, and the FCoE Ethernet Ports Profile models the management of FCoE ports on an HBA.

The ControlledBy association defined in *Storage Management Technical Specification, Part 7 Host Elements, 1.6.1 Rev 5* Clause 6: Storage HBA Profile shall reference instance of FCPort and shall not reference instances of EthernetPort.

22.3.2 Optional target model

Figure 30 shows an example of a single port and drive connected to a single system using Fibre Channel. This instance diagram shows a disk (LogicalDevice in the diagram would be subclassed as something like StorageExtent) in an array, connected by a Fibre Channel port. The full model for the disk is shown in 11 Disk Drive Lite Subprofile.

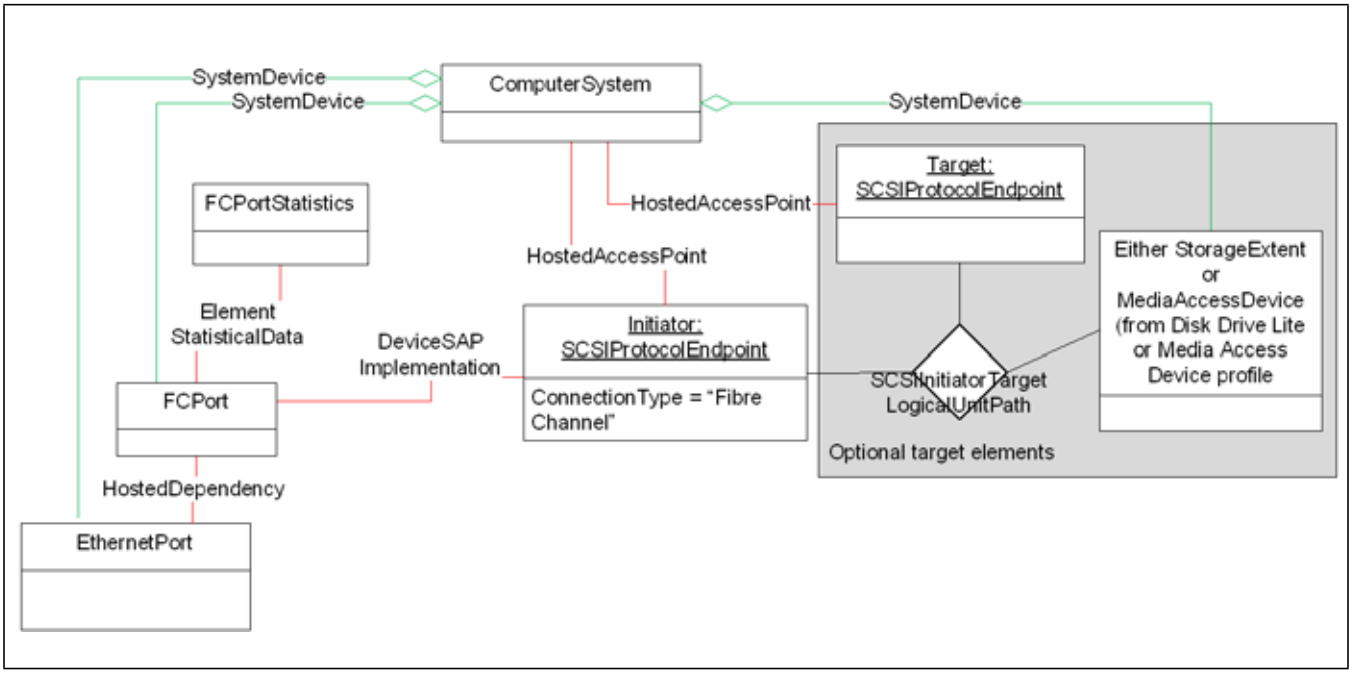

Figure 30 - Optional Target Element Model

22.3.3 Port Statistics

The FCPortStatistics subclass of NetworkPortStatistics is optional. If supported, FCPortStatistics shall be associated to FcPort using ElementStatisticalData.

22.3.4 Logical Port Group (FC Node)

LogicalPortGroup may optionally be used to model the collection of ports that shared a Node WWN (in this case, both ports on a card, but other implementations are in use). If LogicalPortGroup is instantiated, it shall be associated to the ComputerSystem in the referencing profile using HostedCollection and also associated to FCPorts using MemberOfCollection.

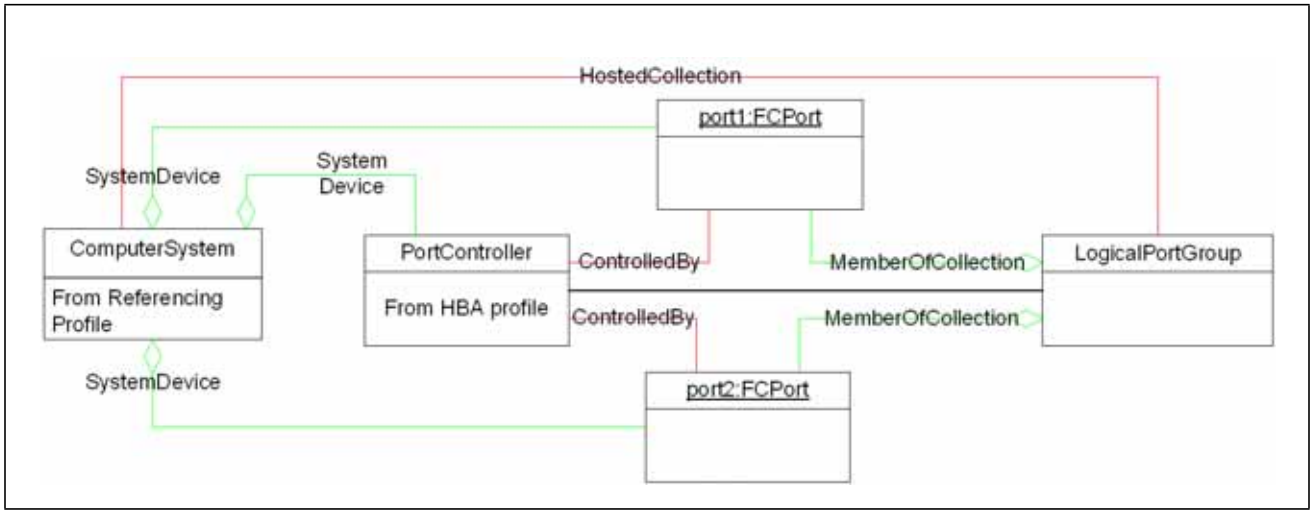

Figure 31 - Logical Port Group Model

22.3.5 Health and Fault Management Considerations

Table 220 summarized the Health and Fault Management considerations specific to this profile.

Table 220 - FCPort OperationalStatus

OperationalStatus	Description
(2) OK	Port is online
(6) Error	Port has a failure
(10) Stopped	Port is disabled
(11) InService	Port is in Self Test
(0) Unknown	

22.3.6 Cascading Considerations

Not defined in this standard.

22.4 Methods

22.4.1 Extrinsic Methods of this Profile

None

22.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported

are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

22.5 Detailed Use Cases and Recipes

None

22.6 CIM Elements

Table 221 describes the CIM elements for FCoE Initiator Ports.

Table 221 - CIM Elements for FCoE Initiator Ports

Element Name	Requirement	Description
22.6.1 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
22.6.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator LogicalPort and ProtocolEndpoint.
22.6.3 CIM_ElementStatisticalData (Port Statistics)	Optional	Connects LogicalPort and LogicalPortStatistics.
22.6.4 CIM_EthernetPort	Mandatory	
22.6.5 CIM_FCPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
22.6.6 CIM_FCPortStatistics	Mandatory	Statistics for a port.
22.6.7 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
22.6.8 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
22.6.9 CIM_HostedCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates the ConnectivityCollection to the hosting System.
22.6.10 CIM_HostedCollection (FC Node)	Optional	Associates the LogicalPortGroup (Fibre Channel Node) to the hosting System.
22.6.11 CIM_HostedDependency (NetworkPort to FCPort)	Mandatory	Association between EthernetPort and FCPort.
22.6.12 CIM_LogicalPortGroup	Optional	Collection of Fibre Channel ports that share a Node WWN.
22.6.13 CIM_MemberOfCollection (Connectivity Collection)	Conditional	Conditional requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections. Associates ProtocolEndpoints to the ConnectivityCollection.
22.6.14 CIM_MemberOfCollection (FC Node)	Optional	Associates FCPort to the LogicalPortGroup.
22.6.15 CIM_ProtocolEndpoint (Initiator)	Mandatory	Represents a protocol (command set) supported by the port. The appropriate subclass (SCSIProtocolEndpoint, ATAProtocolEndpoint, SBProtocolEndpoint) should be used in initiator port specialized profiles.

Table 221 - CIM Elements for FCoE Initiator Ports

Element Name	Requirement	Description
22.6.16 CIM_ProtocolEndpoint (Target)	Optional	Models protocols of remote ports - target devices and possibly other initiators.
22.6.17 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
22.6.18 CIM_SCSIProtocolEndpoint (Initiator)	Mandatory	
22.6.19 CIM_SCSIProtocolEndpoint (Target)	Optional	Models remote ports - target devices and possibly other initiators.
22.6.20 CIM_SystemDevice (Ethernet Port)	Mandatory	Associates system to ethernet ports.
22.6.21 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to FCPorts.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort	Optional	CQL -Creation of an FCoE Port.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::OperationalStatus <> PreviousInstance.CIM_FCPort::OperationalStatus	Optional	CQL -Modify FCPort.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort	Optional	CQL -Deletion of an FCoE Port.

22.6.1 CIM_ConnectivityCollection

Represents a collection of connected ProtocolEndpoints.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 222 describes class CIM_ConnectivityCollection.

Table 222 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	

22.6.2 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 223 describes class CIM_DeviceSAPImplementation.

Table 223 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to FCPort.

22.6.3 CIM_ElementStatisticalData (Port Statistics)

Connects LogicalPort and LogicalPortStatistics. The class definition specializes the CIM_ElementStatisticalData definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 224 describes class CIM_ElementStatisticalData (Port Statistics).

Table 224 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Port Statistics)

Properties	Flags	Requirement	Description & Notes
ManagedElement (overridden)		Mandatory	Reference to FC subclass.
Stats (overridden)		Mandatory	Reference to FCPortStatistics.

22.6.4 CIM_EthernetPort

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 225 describes class CIM_EthernetPort.

Table 225 - SMI Referenced Properties/Methods for CIM_EthernetPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
LinkTechnology		Mandatory	Shall be 2 (Ethernet).
OperationalStatus		Mandatory	Shall be 0 (Unknown), 2 (OK), 6 (Error), 10 (Stopped), or 11 (In Service).
PermanentAddress	CD	Mandatory	The MAC Address. Shall be formatted as 12 un-separated upper case hex digits.

22.6.5 CIM_FCPort

Represents the logical aspects of the physical port and may have multiple associated protocols. The class definition specializes the CIM_LogicalPort definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 226 describes class CIM_FCPort.

Table 226 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction		Mandatory	Shall be 3 for ports restricted to back-end (initiator) only or 4 if the port is unrestricted.
PortType (overridden)		Mandatory	Shall be 0 (Unknown), 1 (Other), N (10), or Nx (13).
ElementName (added)		Mandatory	Port Symbolic Name.
Speed (added)		Mandatory	Speed in bits per second. Shall be 0, 1062500000 (1GFC), 2125000000 (2GFC), 4250000000 (4GFC), 8500000000 (8GFC), 10312500000 (10GE/10GFCoE), 14025000000 (16GFC), or 28500000000 (32GFC).
MaxSpeed (added)		Mandatory	Maximum Port Speed.
PortNumber (added)		Optional	
PermanentAddress (added)	CD	Optional	Port WWN. PermanentAddress is optional when used as a back-end port in a device. This may be overridden in profiles that use this profile. Shall be 16 un-separated upper case hex digits.
NetworkAddresses (added)		Optional	For Fibre Channel end device ports, the Fibre Channel ID. Shall be 16 un-separated upper case hex digits.
SupportedCOS (added)		Optional	List of supported classes of services. Shall include 0 (unknown), 1 (Class 1), 2 (Class 2), 3, (Class 3), 4 (Class 4), 6 (Class 6), or 7 (Class 7).
ActiveCOS (added)		Optional	List of active classes of services. Shall include 0 (unknown), 1 (Class 1), 2 (Class 2), 3, (Class 3), 4 (Class 4), 6 (Class 6), or 7 (Class 7).
SupportedFC4Types (added)		Optional	
ActiveFC4Types (added)		Optional	
LinkTechnology (added)		Mandatory	Shall be 4 (FC).
SupportedMaximumTransmissionUnit (added)		Mandatory	

Table 226 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Flags	Requirement	Description & Notes
ActiveMaximumTransmissionUnit (added)		Optional	
PortDiscriminator (added)		Mandatory	Experimental. Shall include 10 (FCoE) and 12 (HBA).

22.6.6 CIM_FCPortStatistics

. The class definition specializes the SNIA_LogicalPortStatistics definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 227 describes class CIM_FCPortStatistics.

Table 227 - SMI Referenced Properties/Methods for CIM_FCPortStatistics

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
BytesTransmitted		Mandatory	
BytesReceived		Mandatory	
PacketsTransmitted		Mandatory	
PacketsReceived		Mandatory	
CRCErrors (added)		Mandatory	Maps to HBA API HBA_PortStatistics.InvalidCRCCount.
LinkFailures (added)		Mandatory	Maps to HBA API HBA_PortStatistics.LinkFailureCount.
PrimitiveSeqProtocolErrCount (added)		Mandatory	
LossOfSignalCounter (added)		Mandatory	Maps to HBA API HBA_PortStatistics.LossOfSignalCount.
InvalidTransmissionWords (added)		Mandatory	Maps to HBA API HBA_PortStatistics.InvalidTxWordCount.
StatisticTime (added)		Optional	Time last measurement was taken.
LIPCount (added)		Mandatory	
NOSCount (added)		Mandatory	
ErrorFrames (added)		Mandatory	
DumpedFrames (added)		Mandatory	
LossOfSyncCounter (added)		Mandatory	Maps to HBA API HBA_PortStatistics.LossOfSynchCount.

22.6.7 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 228 describes class CIM_HostedAccessPoint (Initiator).

Table 228 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent		Mandatory	Reference to ProtocolEndpoint(Initiator).

22.6.8 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 229 describes class CIM_HostedAccessPoint (Target).

Table 229 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.
Dependent		Mandatory	Reference to ProtocolEndpoint(Target).

22.6.9 CIM_HostedCollection (Connectivity Collection)

Associates the ConnectivityCollection to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 230 describes class CIM_HostedCollection (Connectivity Collection).

Table 230 - SMI Referenced Properties/Methods for CIM_HostedCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ConnectivityCollection.
Antecedent		Mandatory	Reference to ComputerSystem in referencing profile.

22.6.10CIM_HostedCollection (FC Node)

Associates the LogicalPortGroup (Fibre Channel Node) to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 231 describes class CIM_HostedCollection (FC Node).

Table 231 - SMI Referenced Properties/Methods for CIM_HostedCollection (FC Node)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem.
Dependent		Mandatory	Reference to LogicalPortGroup.

22.6.11CIM_HostedDependency (NetworkPort to FCPort)

Association between EthernetPort and FCPort.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 232 describes class CIM_HostedDependency (NetworkPort to FCPort).

Table 232 - SMI Referenced Properties/Methods for CIM_HostedDependency (NetworkPort to FCPort)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to EthernetPort.
Dependent		Mandatory	Reference to FCPort.

22.6.12CIM_LogicalPortGroup

Represents the Fibre Channel Node.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 233 describes class CIM_LogicalPortGroup.

Table 233 - SMI Referenced Properties/Methods for CIM_LogicalPortGroup

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque.
Name	D	Mandatory	Fibre Channel Node WWN.
NameFormat		Mandatory	Shall be 'WWN'.
ElementName		Mandatory	Node Symbolic Name.

22.6.13CIM_MemberOfCollection (Connectivity Collection)

Associates ProtocolEndpoints to the ConnectivityCollection. The class definition specializes the CIM_MemberOfCollection definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections that are not RemoteReplicationCollections.

Table 234 describes class CIM_MemberOfCollection (Connectivity Collection).

Table 234 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Connectivity Collection)

Properties	Flags	Requirement	Description & Notes
Member (overridden)		Mandatory	Reference to ProtocolEndpoint.
Collection		Mandatory	Reference to ConnectivityCollection.

22.6.14CIM_MemberOfCollection (FC Node)

Associates FCPort to the LogicalPortGroup.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 235 describes class CIM_MemberOfCollection (FC Node).

Table 235 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (FC Node)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to LogicalPortGroup.
Member		Mandatory	Reference to FCPort.

22.6.15CIM_ProtocolEndpoint (Initiator)

Represents a protocol (command set) supported by the port.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 236 describes class CIM_ProtocolEndpoint (Initiator).

Table 236 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	

Table 236 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name	C	Mandatory	See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7.6.3 Standard Formats for Port Names.
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription		Mandatory	Shall be the string 'SCSI', 'ATA', or 'SB'. Initiator port specialized profiles specify the appropriate subset.

22.6.16CIM_ProtocolEndpoint (Target)

Models protocols of remote ports - target devices and possibly other initiators. The appropriate subclass (SCSIProtocolEndpoint, ATAProtocolEndpoint, SBProtocolEndpoint) should be used in initiator port specialized profiles.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 237 describes class CIM_ProtocolEndpoint (Target).

Table 237 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	The values in MOFs map to IETF values and exclude storage. Shall be 1 (Other) and set OtherTypeDescription appropriately.
OtherTypeDescription		Mandatory	Shall be the string 'SCSI', 'ATA', or 'SB'. Initiator port specialized profiles specify the appropriate subset.

22.6.17CIM_SCSIInitiatorTargetLogicalUnitPath

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 238 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 238 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
LogicalUnit		Mandatory	
Initiator		Mandatory	
Target		Mandatory	

22.6.18CIM_SCSIProtocolEndpoint (Initiator)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 239 describes class CIM_SCSIProtocolEndpoint (Initiator).

Table 239 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription		Mandatory	Shall be the string 'SCSI'.
ConnectionType		Mandatory	Shall be 2 (Fibre Channel).
Role		Mandatory	Shall be 2 (Initiator).

22.6.19CIM_SCSIProtocolEndpoint (Target)

Models remote ports - target devices and possibly other initiators.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 240 describes class CIM_SCSIProtocolEndpoint (Target).

Table 240 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
Role		Mandatory	Should be set appropriately by the instrumentation. If not know, use 0 (Unknown).
ProtocolIFType		Mandatory	The values in MOFs map to IETF values and exclude storage. Shall be 1 (Other) and set OtherTypeDescription to 'SCSI'.
OtherTypeDescription		Mandatory	Shall be the string 'SCSI'.
ConnectionType		Mandatory	Shall be 8 (FC).

22.6.20CIM_SystemDevice (Ethernet Port)

Associates system to ethernet ports.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 241 describes class CIM_SystemDevice (Ethernet Port).

Table 241 - SMI Referenced Properties/Methods for CIM_SystemDevice (Ethernet Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent		Mandatory	Reference to EthernetPort.

22.6.21CIM_SystemDevice (Initiator Ports)

Associates system to FCPorts. The class definition specializes the CIM_SystemDevice definition in the Generic Initiator Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 242 describes class CIM_SystemDevice (Initiator Ports).

Table 242 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem.
PartComponent (overridden)		Mandatory	Reference to FCPort.

EXPERIMENTAL

STABLE

23 Access Points Subprofile

23.1 Description

The Access Points Subprofile provides addresses of remote access points for management services.

This is modeled using a RemoteServiceAccessPoint linked to the managed system using a HostedAccessPoint association.

A management service is typically associated with all elements in a system, but in some cases, a management service relates to a subset of elements. The scope of a RemoteServiceAccessPoint may be constrained to a subset of elements using SAPAvailableForElement. If the service referenced in RemoteServiceAccessPoint is not referenced by any SAPAvailableForElement associations, then the service described by RemoteServiceAccessPoint shall apply to all the elements of the system referenced via HostedAccessPoints. This type of system-wide service is depicted in Figure 32.

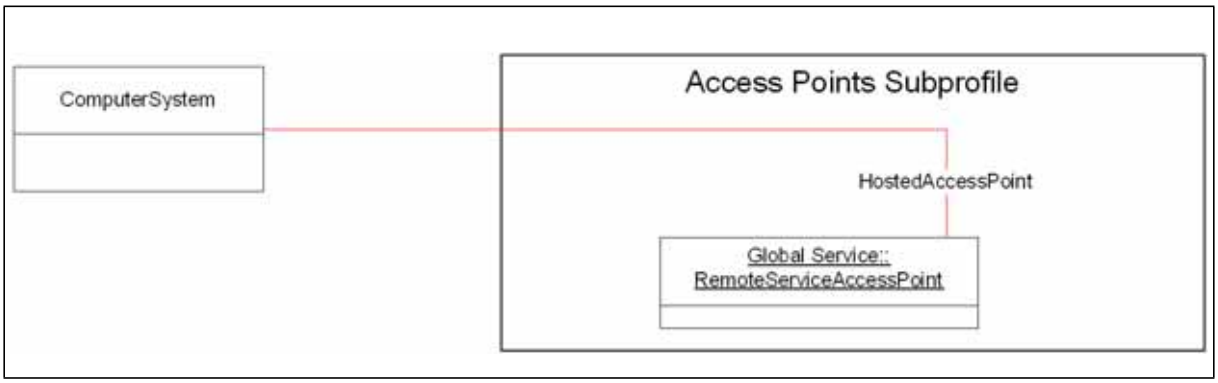

Figure 32 - System-wide Remote Access Point

If the service referenced in RemoteServiceAccessPoint is referenced by any SAPAvailableForElement associations, then the service described by RemoteServiceAccessPoint shall apply to the subset of elements referenced via SAPAvailabelForElement associations. The HostedAccessPoint association between RemoteServiceAccessPoint is still mandatory (so the client can readily associate the service to a specific storage system).

Figure 33 depicts a configuration with two RemoveServiceAccessPoint instances. One represents a system-wide service and the other represents a service that applies just to certain devices.

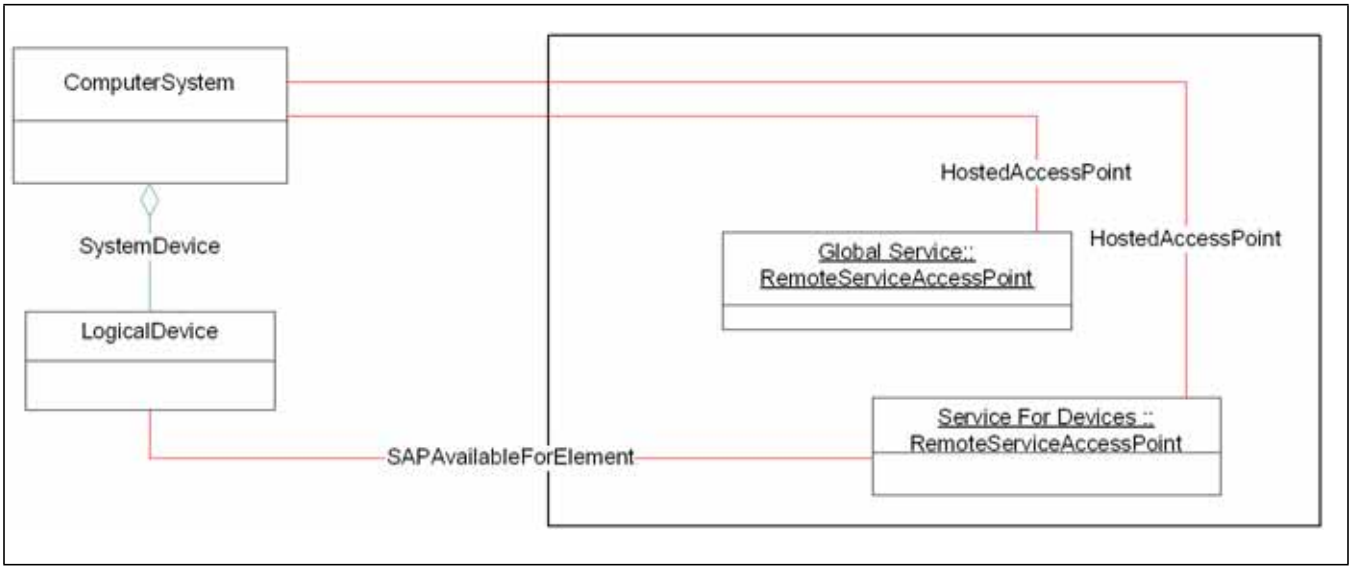

Figure 33 - Access Point Instance Diagram

The exposed management services may represent a web UI that can be launched by a web browser, a telnet interface, or some vendor-specific interface. RemoteServiceAccessPoint InfoFormat property describes the format of the AccessIfo property; valid options include “URL” and FQDN”. In a URL, the text before the “://” is referred to as the “scheme”. A URL with an http or HTTPS scheme is often a web/HTML page, but HTTP can be used for other purposes. Table 243 specifies the requirements for InfoFormat, AccessInfo, and the scheme subset of a URL AccessInfo.

Table 243 - RemoteAccessPoint InfoFormat and AccessInfo Properties

InfoFormat	AccessInfo Scheme	Description
“URL”	“http” or “https”	The references URL shall be a valid web page. It should provide element management for the system or elements referenced by the associated HostedAccessPoint association.
“Other” with OtherInfoFormatDescription = "Non-UI URL"	“http” or” https”	Used for HTTP URLs that do not reference a valid web UI.
“URL”	anything other than “http” and “https”	May be used. No standard behavior is specified.
others from the MOF	n/a	May be used. No standard behavior is specified.

23.2 Health and Fault Management Considerations

Not defined in this standard.

23.3 Cascading Considerations

Not defined in this standard.

23.4 Supported Subprofiles and Packages

Not defined in this standard.

23.5 Methods of this Profile

Not defined in this standard.

23.6 Client Considerations and Recipes

Not defined in this standard.

23.7 Registered Name and Version

Access Points version 1.3.0 (Component Profile)

23.8 CIM Elements

Table 244 describes the CIM elements for Access Points.

Table 244 - CIM Elements for Access Points

Element Name	Requirement	Description
23.8.1 CIM_HostedAccessPoint	Mandatory	Associate the RemoteServiceAccessPoint to the System on which it is hosted.
23.8.2 CIM_RemoteServiceAccessPoint	Mandatory	A ServiceAccessPoint for management tools.
23.8.3 CIM_SAPAvailableForElement	Optional	This association identifies the element that is serviced by the RemoteServiceAccessPoint.

23.8.1 CIM_HostedAccessPoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 245 describes class CIM_HostedAccessPoint.

Table 245 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Hosting System.
Dependent		Mandatory	The access point(s) that are hosted on this System.

23.8.2 CIM_RemoteServiceAccessPoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 246 describes class CIM_RemoteServiceAccessPoint.

Table 246 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	
ElementName		Mandatory	User Friendly name.
AccessInfo		Mandatory	Management Address.
InfoFormat		Mandatory	The format of the Management Address. For interoperability, this shall be 'URL' (200).

23.8.3 CIM_SAPAvailableForElement

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 247 describes class CIM_SAPAvailableForElement.

Table 247 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The managed element.
AvailableSAP		Mandatory	The service access point.

STABLE

DEPRECATED

24 Cascading Subprofile

NOTE The Cascading Subprofile is scheduled for removal for SMI-S 2.0. The functionality of this profile will not be replaced in SMI-S 2.0. In SMI-S 2.0, cascading elements are to be done as profile specific cascading component profile or the cascading elements are to simply be embedded in the profile that is doing the cascading. The SNIA would like to hear from anyone that has implemented the Cascading Subprofile. If your company or organization has implemented this profile and is a member of the SNIA, please contact the SMI-S Core Technical Working Group or indicate your preference to keep this profile in SMI-S 2.0 during member reviews and ballots. If your company or organization has implemented this profile and is not a member of the SNIA, please indicate your preference to keep this profile as part of SMI-S using the SNIA feedback portal: http://www.snia.org/tech_activities/feedback/ .

24.1 Description

The Cascading Subprofile defines the set of classes, methods and behavior used to model cross profile dependencies and references. This includes modeling of cross CIM server references when the referenced profile is managed by another CIM server.

Examples of SMI-S Profiles that should support the Cascading Subprofile include Storage Virtualizer, NAS Heads and Volume Managers. However, other profiles may also support the Cascading Subprofile for cross profile references. For example, an Array Profile may support the Cascading Subprofile to effect cross profile references used in “remote copy.”

For ease of documentation, a profile that supports the Cascading Subprofile is referred to as a **cascading profile**. The profile referenced is referred to as a **leaf profile**. For example, storage virtualization would support the Cascading Subprofile and would be a cascading profile. It would reference storage volumes in one or more Array profiles. In such configurations, the Array profiles would be referred to as Leaf profiles.

The cascading subprofile defines a common approach to “stitching” resources in the cascading profile to resources in the leaf profiles. While the general mechanism used is common, the specifics may vary depending on the resources that are stitched together. For example, a Storage Virtualization Profile would stitch StorageExtents (in the virtualizer) to StorageVolumes (in arrays). But a Volume Manager would stitch LogicalDisks (in the volume manager) to StorageVolumes (in arrays or virtualizers).

The cascading subprofile defines how to model the relationships between CIM Servers when there are CIM Servers of Leaf profiles that are referenced by a CIM Server of the cascading profile, and how a client manages the interaction between CIM Servers in a cascading configuration (including CIM Server credentials).

In addition to the Cascading Subprofile, there are two related subprofiles that may also be supported by the cascading profile or the leaf profiles. They are the Credential Management Subprofile, which defines the classes, methods and behavior for managing the credentials used by a CIM server of the cascading profile when accessing (different) CIM Servers of Leaf profiles. The second is the Security Resource Ownership Subprofile (or a specialization of this subprofile) which defines the classes, methods and behavior of recording ownership in the leaf profiles. The usage of these subprofiles will be referenced in this subprofile, but their definition is contained in separate subprofile specifications.

The Cascading Subprofile provides block-level configuration management in the current version of SMI-S.

The Cascading Subprofile defines cascading of resources at the block level. That is, a Cascading Profile uses Block storage resources of the leaf profiles. These are StorageVolumes or LogicalDisks. In the current version of SMI-S the model will only be tested in the context of cascading for block storage.

24.1.1 Instance Diagrams

There are three aspects of the cascading subprofile that are illustrated separately:

- Logical Topology (usage of leaf resources by cascading profiles)
- Resource Allocation/Deallocation
- CIM Server Topology (usage of CIM Servers by other CIM Servers)

In addition, there are the relationships between the Cascading Subprofile and the Security Resource Ownership Subprofile and the Credential Management Subprofile. This relationship will be illustrated, but the details of those subprofiles are documented in their own sections.

24.1.1.1 Logical Topology

Figure 34 illustrates the basic constructs for modeling the logical topology represented by cascading profiles. The cascading profile is the top box. The modeling for the cascading subprofile is in the dashed box (in the Cascading Profile). The leaf profile is the lower box. Note that for the basic modeling of the logical topology of cascading, there are no modeling requirements on the leaf profile.

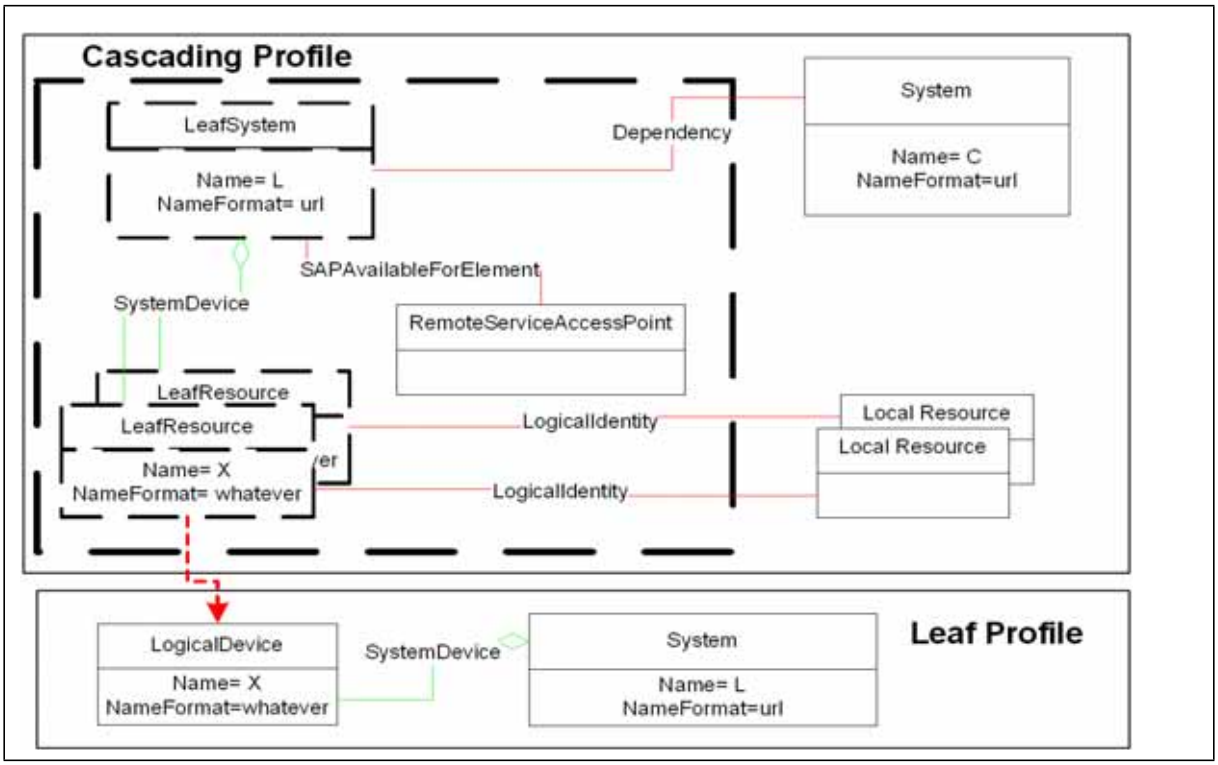

Figure 34 - Instance Diagram for Logical Topology

NOTE The dashed classes in Figure 34 are instances that are cached in the Cascading Profile. They are redundant with the instances maintained by the Leaf profile. The dashed arrows between the Cacsading Profile and the Leaf Profile signifies “stitching” based on durable names or correlatable ids for the resources represented. The dashed arrows **are not** instantiated associations.

If the Cascading Subprofile is supported by the Cascading Profile, then there will be support for instantiating “leaf” “top level object” (e.g., ComputerSystems) and “leaf” LogicalDevices (e.g., StorageVolumes) in those Leaf Profiles that are “visible” to the Cascading Profile (device). The instances of the “leaf” “top level object” can be found by traversing the CascadingDependency association from the “top level object” of the Cascading Profile.

The leaf resources (logical devices) that are visible to the Cascading Profile have an association (e.g., SystemDevice association) to the “leaf” top level object (e.g., ComputerSystem) that has exposed them to the Cascading Profile.

The top level object, Hosted or SystemDevice association and LogicalDevices mirrors information that is in the Leaf Profile. In some Cascading Profile configurations, the Cascading Profile may want to subscribe to life cycle indications on the devices of interest in the Leaf Profile. However, that is a consideration of the Cascading Profile. It is not required as part of the Cascading Subprofile.

From the top level object (e.g., ComputerSystem) of the Leaf, there may be a SAPAvailableForElement association to a RemoteServiceAccessPoint instance. The RemoteServiceAccessPoint identifies information need for access to the management interface to the Leaf system. This management interface may or may not be a CIM interface.

The expectation is that the model represented in Figure 34 will be automatically maintained by the Cascading Profile (and providers). There are no methods for client manipulation of this model. In the case of the RemoteServiceAccessPoint instance, the expectation is that discovery of leaf systems would be an automatic process (e.g., SLP discovery of SMI-S Profiles and Servers) and that the provider would record the access information based on its discovery processes.

In the simplest form of cascading, this is sufficient to model the logical topology of the cascading. However, many implementations will need to go further (see 24.1.1.2).

24.1.1.2 Resource Allocation/Deallocation

In some cascading environments, it is necessary to distinguish between resources that are “visible” to the Cascading Profile from resources that are actually “in use.” For example, a Volume Manager or storage virtualization system may be able to “see” a number of storage volumes (logical units) through its ports. But this does not necessarily mean that is has allocated and is using them. A separate step is required to “prepare” the resources for use. In the case of storage virtualization systems, this step would include assigning the storage to a storage pool in the virtualizer.

To readily discern which storage volumes (logical devices) are “visible” and which volumes are assigned, two collections are defined. The collection of “visible” resources is the “RemoteResources” collection. The collection of assigned resources is the “AllocatedResources” collection. This is illustrated in Figure 35.

The SNIA_AllocationService may or may not exist. The actual function of Allocation may be implemented as a side effect of other methods. For example, allocating a Leaf StorageVolume may occur as a side effect of CreateOrModifyStoragePool, where an extent (e.g., leaf StorageVolume) is added to a StoragePool. The semantics of CreateOrModifyStoragePool constructs all the necessary associations for the StorageExtent (and may also have the semantics of an implied allocation of the StorageVolume).

To determine if allocation or deallocation are explicit (via allocate/deallocate method calls) or implicit (side effect of another method), the client should inspect the “AsynchronousMethodsSupported” and “SynchronousMethodsSupported” properties of the SNIA_CascadingCapabilities instance for the System.

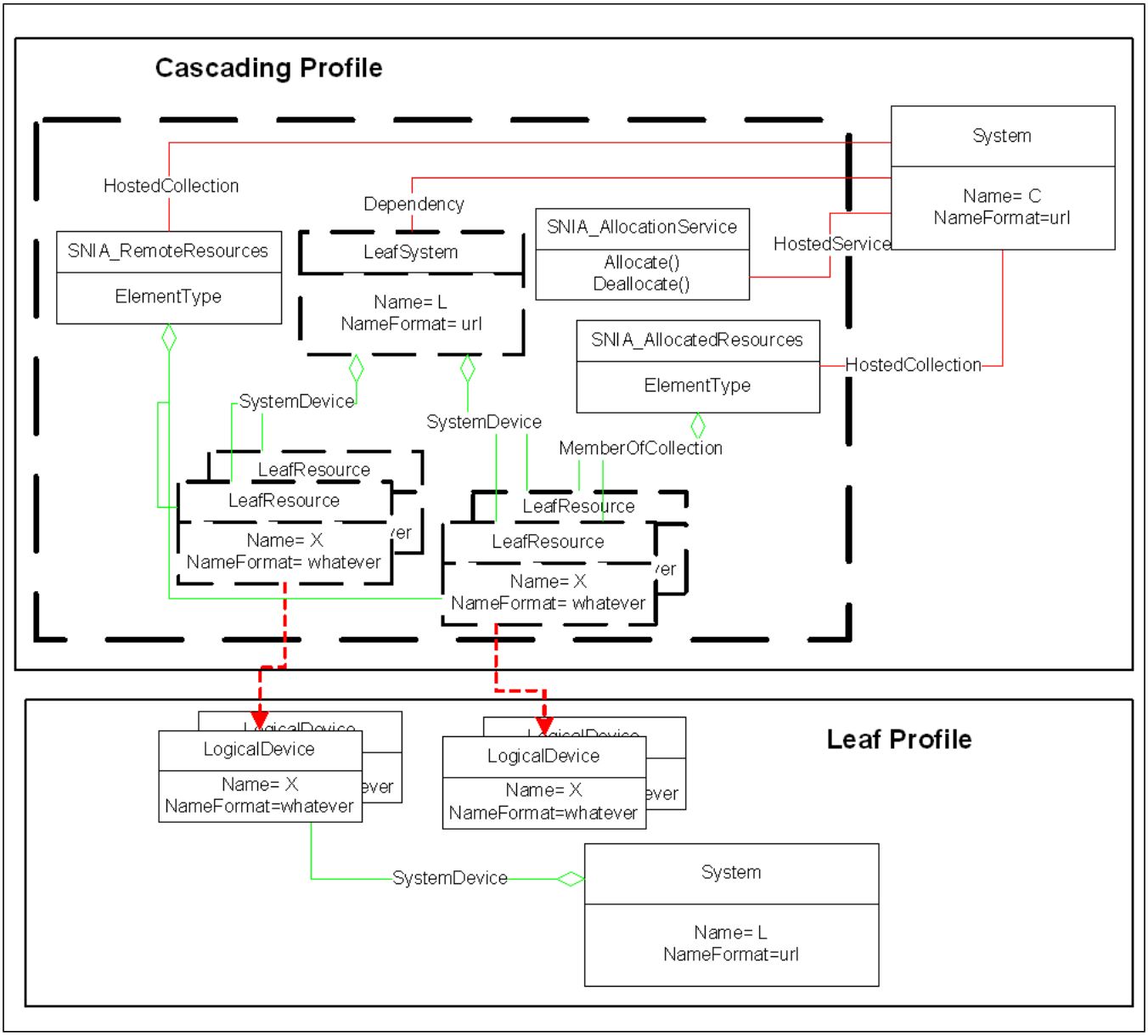

Figure 35 - Resource Allocation/Deallocation Instance Diagram

24.1.1.3 CIM Server Topology

In addition to a cascading system using leaf systems and its resources, a cascading profile may also model the dependencies between the CIM Server of the cascading profile and the CIM Servers of the Leaf Profiles. This is illustrated in Figure 36.

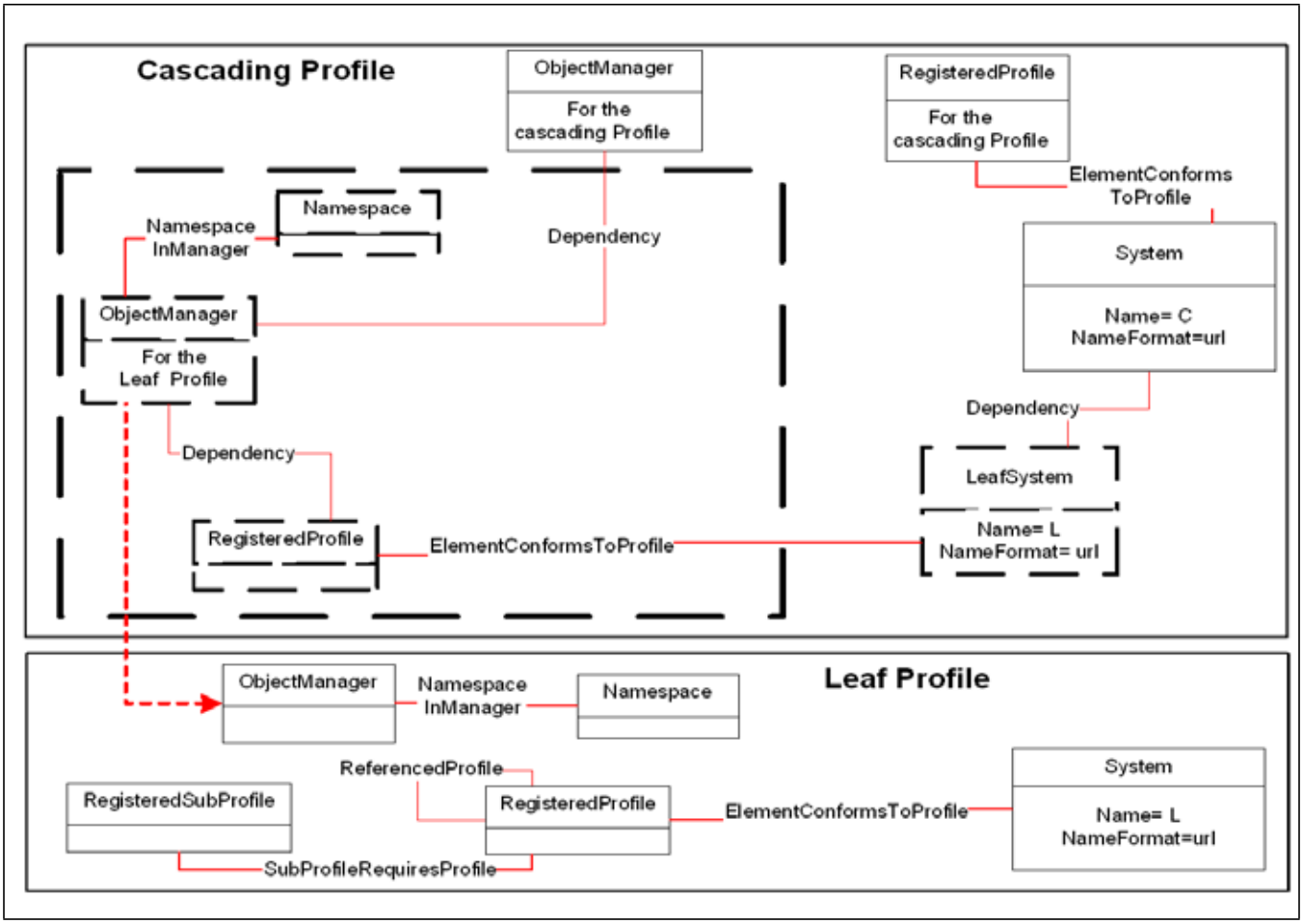

Figure 36 - Cascading Server Topology

As with the logical topology, the server topology is effected by caching Leaf information in the cascading profile. Specifically, the cached instances from the leaf profiles are:

ObjectManager – to allow the dependency between ObjectManagers to be instantiated in the cascading profile.

Namespace – to provide cached information on the namespace of the leaf CIM Server. This would be the Interop Namespace for accessing the Server Profile of the CIM Server.

RegisteredProfile – to identify the Profile of the Leaf Profile (e.g., Array or Virtualizer).

In addition, the necessary associations (HostedProfile, NamespaceInManager and ElementConformsToProfile) would be instantiated to connect the relevant instances.

The actual dependence between the CIM Server (ObjectManager) of the Cascading Profile and the CIM Server (ObjectManager) of the Leaf systems is represented by instances of Dependency.

24.1.1.4 Cascading with the Resource Ownership Subprofile

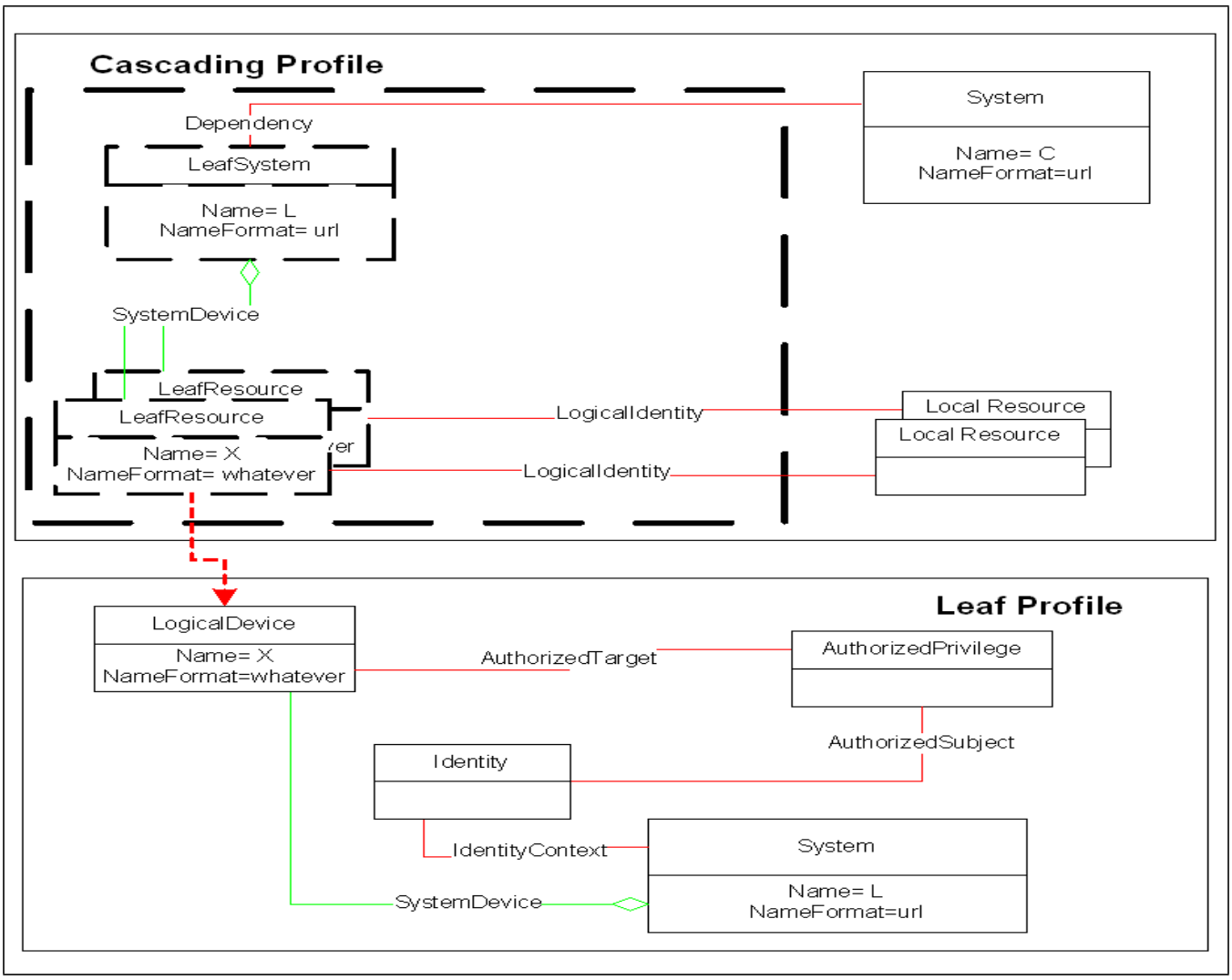

Figure 37 - Instance Diagram for Cascading with Resource Ownership

Figure 37 illustrates cascading when used in conjunction with the Security Resource Ownership Profile. The Security Resource Ownership (or a specialization of it) would be implemented in the Leaf Profile.

24.1.1.5 Cascading with the Credentials Management Subprofile

As an extension of the modeling of CIM Server topology, a cascading profile may implement the Credentials Management Subprofile. When this is done it extends the modeling for the Server topology as illustrated in Figure 38.

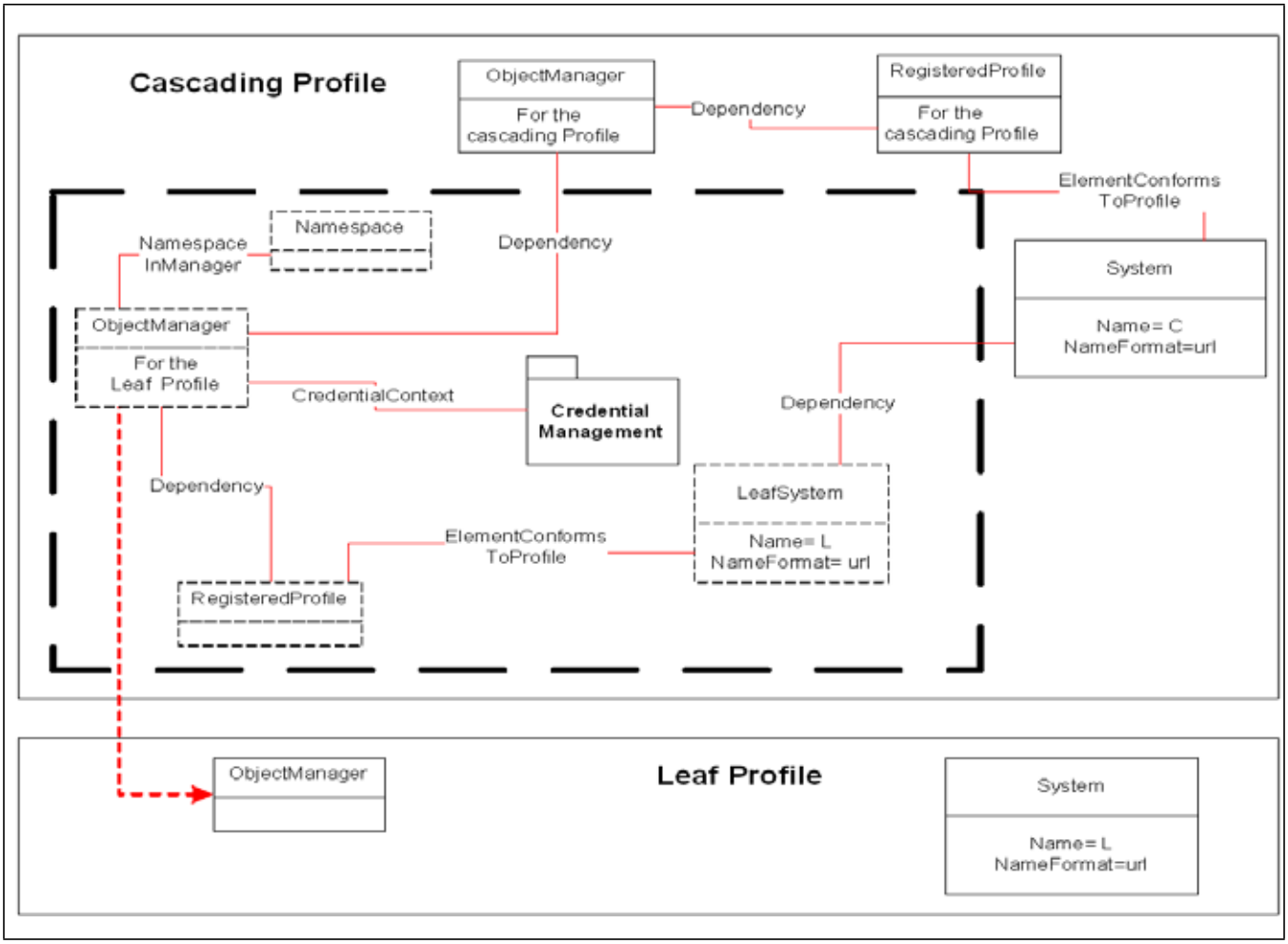

Figure 38 - Instance Diagram for Cascading with Credential Management Subprofile

The Credential Management information would be associated with the CIM Server ObjectManager instance for a Leaf system. The Credential Management Subprofile would identify how the cascading system would authenticate itself with the Leaf system.

24.1.1.6 Modeling for Defining Cascading Capabilities

As indicated in previous discussions, only parts of the Cascading subProfile are mandatory. For a list of what elements are mandatory, see Table 250. In order to make it relatively easy for clients to determine what is supported, implementation of the SNIA_CascadingCapabilities class is mandatory if cascading is supported. The modeling for this class is illustrated in Figure 39.

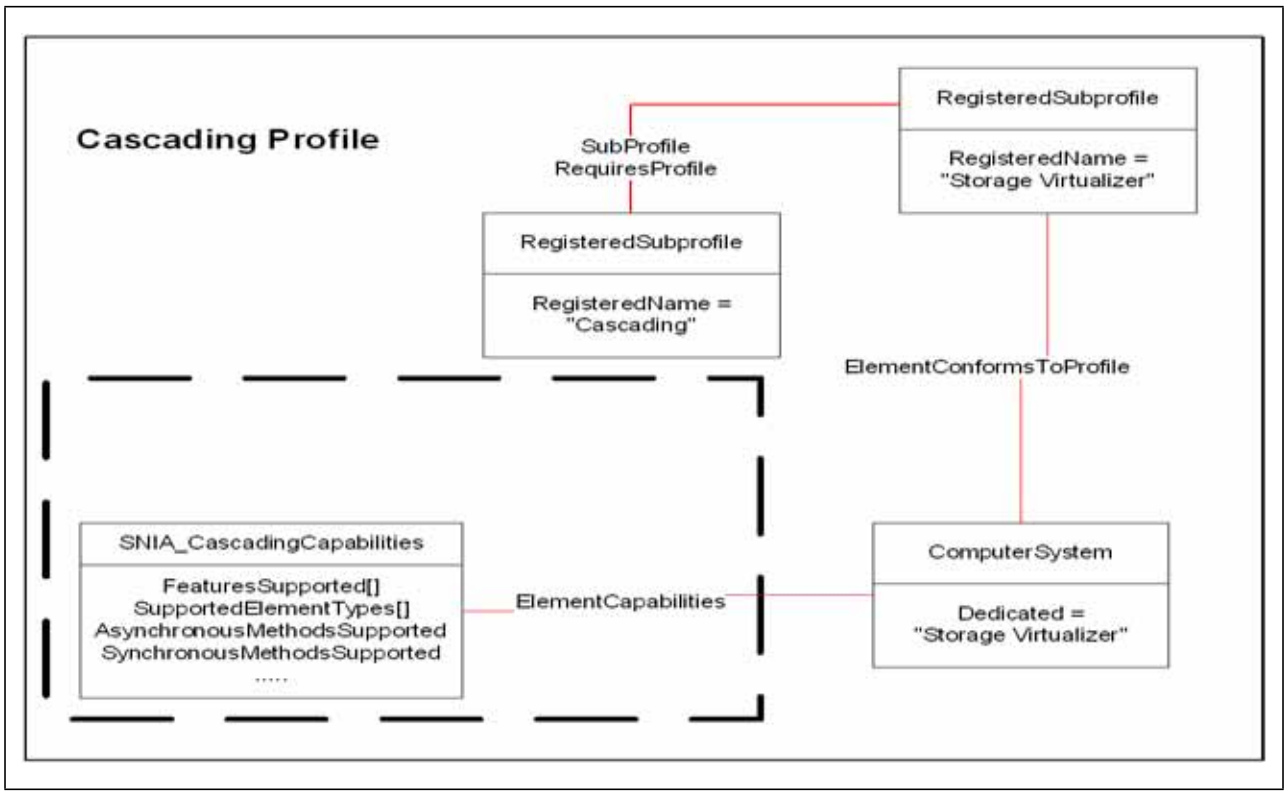

Figure 39 - Modeling of Cascading Capabilities

The SNIA_CascadingCapabilities instance would be found by doing association traversal from the RegisteredSubprofile for cascading following the ElementCapabilites association.

The properties of SNIA_CascadingCapabilities are defined as follows:

- FeaturesSupported - This is an array that defines the cascading features that are supported by the implementation of the Cascading Profile. The values are "Ownership", "Leaf Credentials", "OM Dependencies" and "Allocation Service".
- SupportedElementTypes - This is an array that defines the type of “Remote Resource” ManagedElements that are supported by the implementation. For this version of SMI-S, only StorageVolumes and LogcialDisks are supported.
- AsynchronousMethodsSupported – This is an array that defines any asynchronous methods supported for allocation or deallocation of leaf resources. The values are “Allocation” or “Deallocation”.
- SynchronousMethodsSupported – This is an array that defines any synchronous methods supported for allocation or deallocation of leaf resources. The values are “Allocation” or “Deallocation”.

The Cascading Subprofile uses durable names of leaf resources for stitching together the Leaf Profile and its resources to the corresponding instances in the Cascading Profile.

The CIM Server of the Cascading Profile may use indications (or provider poll on access) to keep its model accurate.

24.2 Health and Fault Management Considerations

24.2.1 Reporting Health of Leaf Systems, Resources and Object Managers

A Cascading Profile should not report health of leaf resources without verifying the health of those resources (via direct reference to the Leaf Profile). The Cascading Profile may keep health properties in its local copy of the instances for leaf resources for its own purposes, but it should always refer to the leaf profile on requests from clients.

A request for a health property (e.g., OperationalStatus) should result in a request to the underlying leaf resource for the information. If the leaf resource is not available (e.g., the connection to the CIM Server is broken) the Cascading Profile may report health from its local copy of the instance.

24.2.2 Cascading Indications of Health

Given a Cascading Profile is dependent upon leaf resources, the CIM Server of the Cascading Profile may chose to subscribe to health (OperationalStatus) indications on the leaf resources it is actively using (allocated resources). Generally speaking, health problems on leaf resources will translate to health problems on one or more resources in the Cascading Profile. For example, if a StorageVolume in the Array (leaf) profile has an OperationalStatus of “Error”, this may cause one or more StorageVolumes in a Virtualizer that is using the array to either be in error or be degraded.

Health indications should cascade. However, how they cascade will depend on where and how the leaf resources are used.

However a cascading profile discovers a problem with leaf resources, then it may be reflected in operational status of the cascader’s resources.

24.3 Cascading Considerations

Not defined in this standard.

24.4 Supported Subprofiles and Packages

Table 248 describes the supported profiles for Cascading.

Table 248 - Supported Profiles for Cascading

Profile Name	Organization	Version	Requirement	Description
Server	SNIA	1.6.0	Mandatory	

24.5 Methods of this Subprofile

Table 249 summarized the extrinsic methods supported by the Cascading Subprofile.

Table 249 - Extrinsic Methods Supported by Cascading Subprofile

Method	Created Instances	Deleted Instances	Modified Instances
Allocate	MemberOfCollection	N/A	N/A
Deallocate	N/A	MemberOfCollection	N/A

24.5.1 Allocate

Starts a job to allocate remote resources (from the RemoteResources collection) to the AllocatedResources collection.

Allocate (

[IN, Description (Enumeration indicating the type of element being allocated. This type value shall match the type of the instances.),

ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8" },

Values { "Unknown", "Reserved", "Any Type", "StorageVolume", "StorageExtent", "StoragePool", "ComputerSystem", "LogicalDisk", "FileShare" }]

Only “3” (StorageVolume) is supported in this version of SMI-S.

uint16 **ElementType**;

[IN (false), OUT, Description (Reference to the job (may be null if job completed).)]

CIM_ConcreteJob REF **Job**,

[IN, Description (The reference to the AllocatedResource collection to which Elements are being added.)]

SNIA_AllocatedResources REF **Collection**,

[IN, Description (Array of strings containing representations of references to CIM_ManagedElement instances, that are being allocated to the AllocatedResources Collection.)]

string **InElements[]**);

Error returns are:

{ "Job Completed with No Error", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter", "In Use", "DMTF Reserved", "Method Parameters Checked - Job Started", "Method Reserved", "Vendor Specific" }]

24.5.2 Deallocate

Starts a job to remove remote resources (from the AllocatedResources collection) and return them to the RemoteResources collection.

Deallocate (

[IN, Description (Enumeration indicating the type of element being deallocated. This type value shall match the type of the instances.),

ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8" },

Values { "Unknown", "Reserved", "Any Type", "StorageVolume", "StorageExtent", "StoragePool", "ComputerSystem", "LogicalDisk", "FileShare" }]

Only “3” (StorageVolume) is supported in this version of SMI-S.

uint16 **ElementType**;

[IN (false), OUT, Description (Reference to the job (may be null if job completed).)]

CIM_ConcreteJob REF **Job**,

[IN, Description (The reference to the AllocatedResource collection from which Elements are being removed.)]

SNIA_AllocatedResources REF **Collection**,

[IN, Description (Array of strings containing representations of references to CIM_ManagedElement instances, that are being deallocated from the AllocatedResources Collection.")]

string **InElements[]**);

Error returns are:

{ "Job Completed with No Error", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter", "In Use", "DMTF Reserved", "Method Parameters Checked - Job Started", "Method Reserved", "Vendor Specific" }

24.6 Client Considerations and Recipes

24.6.1 Recipe MPCP01: Determining Resources used by cascading Profiles

This recipe is not defined in this standard. It will be included in a future revision, based on implementation experience.

24.6.2 Recipe MPCP02: Monitoring the existence of Cascading Profiles

This recipe is not defined in this standard. It will be included in a future revision, based on implementation experience.

24.6.3 OPTIONAL: Recipe MPCP03: Allocation of Leaf Resources

This recipe is not defined in this standard. It will be included in a future revision, based on implementation experience.

24.6.4 OPTIONAL: Recipe MPCP04: Deallocation of Leaf Resources

This recipe is not defined in this standard. It will be included in a future revision, based on implementation experience.

24.6.5 Recipe MPCP05: Monitoring the existence of “Stitching” between Profiles

This recipe is not defined in this standard. It will be included in a future revision, based on implementation experience.

24.6.6 Supported SNIA_CascadingCapabilities Patterns

The SNIA_CascadingCapabilities patterns in Table 250 are formally recognized and supported by this version of SMI-S.

Table 250 - Cascading Capabilities Patterns

FeaturesSupported	SupportedElementTypes	SynchronousMethods Supported	AsynchronouosMethods Supported
none	StorageVolume	none	none
Ownership, Leaf Credentials, OM Dependencies, Allocation Service	StorageVolume	Allocation Deallocation	Allocation Deallocation
Allocation Service	StorageVolume	Allocation Deallocation	none

Table 250 - Cascading Capabilities Patterns (Continued)

FeaturesSupported	SupportedElementTypes	SynchronousMethods Supported	AsynchronouosMethods Supported
Allocation Service	StorageVolume	none	Allocation Deallocation
Ownership, Leaf Credentials, OM Dependencies	StorageVolume	none	none

24.7 Registered Name and Version

Cascading version 1.3.0 (Component Profile)

24.8 CIM Elements

Table 251 describes the CIM elements for Cascading.

Table 251 - CIM Elements for Cascading

Element Name	Requirement	Description
24.8.1 CIM_ComputerSystem (Leaf System)	Mandatory	'Top level' system that represents the leaf system.
24.8.2 CIM_Dependency (Object Managers)	Conditional	Conditional requirement: This is required if Leaf ObjectManagers are modeled. This associates the Object Manager of the Leaf System to the Object Manager of the Cascading System.
24.8.3 CIM_Dependency (Profile to Object Manager)	Conditional	Conditional requirement: This is required if RegisteredProfiles of Leaf systems are modeled. This associates the RegisteredProfile of a leaf system to the Object Manager of the leaf system.
24.8.4 CIM_Dependency (Systems)	Mandatory	This associates the Leaf System to the Cascading System.
24.8.5 CIM_ElementCapabilities	Mandatory	This associates the CascadingCapabilities to the cascading system (e.g., ComputerSystem).
24.8.6 CIM_ElementConformsToProfile (Leaf)	Conditional	Conditional requirement: This is required if RegisteredProfiles of Leaf systems are modeled. This associates the RegisteredProfile of the Leaf Profile to the Leaf system (e.g., ComputerSystem).
24.8.7 CIM_HostedCollection (Allocated Resources)	Mandatory	This would associate the AllocatedResources collection to the top level system for the Cascading Profile (e.g., Storage Virtualizer).
24.8.8 CIM_HostedCollection (Remote Resources)	Conditional	Conditional requirement: This is required if SNIA_RemoteResources is modeled. This would associate the RemoteResources collection to the top level system for the Cascading Profile (e.g., Storage Virtualizer).
24.8.9 CIM_HostedService (Allocation Service)	Conditional	Conditional requirement: This is required if SNIA_AllocationService is modeled. This associates the AllocationService to the system in the cascading profile that hosts the service.

Table 251 - CIM Elements for Cascading

Element Name	Requirement	Description
24.8.10 CIM_HostedService (Object Manager)	Conditional	Conditional requirement: This is required if Leaf ObjectManagers are modeled. This associates the ObjectManager to the system in the cascading profile that hosts the service.
24.8.11 CIM_LogicalDisk	Conditional	Conditional requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '7' ('LogicalDisk'). A remote LogicalDisk that is imported to the referencing profile.
24.8.12 CIM_LogicalIdentity (General)	Mandatory	Associates local resource (e.g., StorageExtent) to a remote (imported) resource (e.g., StorageVolume or LogicalDisk).
24.8.13 CIM_LogicalIdentity (LogicalDisk)	Conditional	Conditional requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '7' ('LogicalDisk'). Associates local StorageExtent to a remote (imported) LogicalDisk.
24.8.14 CIM_LogicalIdentity (StorageVolume)	Conditional	Conditional requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '3' ('StorageVolume'). Associates local StorageExtent to a remote (imported) StorageVolume.
24.8.15 CIM_MemberOfCollection (Allocated Resources)	Mandatory	This supports collecting leaf resources. This is required to support the AllocatedResources collection.
24.8.16 CIM_MemberOfCollection (Remote Resources)	Conditional	Conditional requirement: This is required if SNIA_RemoteResources is modeled. This supports collecting leaf resources. This is optional when used to support the RemoteResources collection (the RemoteResources collection is optional).
24.8.17 CIM_Namespace (Leaf)	Conditional	Conditional requirement: This is required if Leaf ObjectManagers are modeled. There would be one for every namespace supported.
24.8.18 CIM_NamespaceInManager (Leaf)	Conditional	Conditional requirement: This is required if Leaf ObjectManagers are modeled. This associates the namespace to the ObjectManager.
24.8.19 CIM_ObjectManager (Leaf)	Optional	This is the Object Manager service of the CIM Server.
24.8.20 CIM_RegisteredProfile (Leaf)	Optional	A registered profile that is supported by the CIM Server.
24.8.21 CIM_RemoteServiceAccessPoint (Leaf)	Optional	CIM_RemoteServiceAccessPoint represents the management interface to a leaf system.
24.8.22 CIM_SAPAvailableForElement	Conditional	Conditional requirement: This is required if CIM_RemoteServiceAccessPoint is modeled. Represents the association between a RemoteServiceAccessPoint and the leaf System to which it provides access.
24.8.23 CIM_StorageVolume	Conditional	Conditional requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '3' ('StorageVolume'). A remote StorageVolume that is imported to the referencing profile.
24.8.24 CIM_SystemDevice (Leaf Devices)	Conditional	Conditional requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '3'|'4'|'7' (StorageVolume | StorageExtent | LogicalDisk). This association links **LogicalDevice** remote resources to the scoping system. This is used to associate the remote resources with the System that manages them.

Table 251 - CIM Elements for Cascading

Element Name	Requirement	Description
24.8.25 SNIA_AllocatedResources	Mandatory	This is a SystemSpecificCollection for collecting leaf resources that have been deployed for use in the Cascading profile (e.g., StorageVolumes assigned to a virtualizer's StoragePool).
24.8.26 SNIA_AllocationService	Optional	This service provides methods for allocating and deallocating leaf resources.
24.8.27 SNIA_CascadingCapabilities	Mandatory	This defines the cascading capabilities supported by the implementation of the profile.
24.8.28 SNIA_RemoteResources	Optional	This is a SystemSpecificCollection for collecting leaf resources that may be allocated by the system of the Cascading profile (e.g., StorageVolumes assigned to a virtualizer's StoragePool).

24.8.1 CIM_ComputerSystem (Leaf System)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 252 describes class CIM_ComputerSystem (Leaf System).

Table 252 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Leaf System)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	Unique identifier for the leaf system. E.g., IP address.
ElementName		Mandatory	User friendly name.
OtherIdentifyingInfo	C	Mandatory	
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	Overall status of the Leaf system.
NameFormat		Mandatory	Format for Name property.
Dedicated		Mandatory	Indicates that this computer system is dedicated to operation as a leaf system.
PrimaryOwnerContact	M	Optional	Contact a details for owner.
PrimaryOwnerName	M	Optional	Owner of the Leaf system.

24.8.2 CIM_Dependency (Object Managers)

CIM_Dependency is an association between an Object Manager of a Leaf System and the Object Manager of the Cascading System (ComputerSystem). If the Leaf System and the Cascading System are supported by the same Object Manager, then no Dependency would exist.

CIM_Dependency is not subclassed from anything.

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: This is required if Leaf ObjectManagers are modeled.

Table 253 describes class CIM_Dependency (Object Managers).

Table 253 - SMI Referenced Properties/Methods for CIM_Dependency (Object Managers)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Object Manager of the Cascading System.
Dependent		Mandatory	The Object Manager of the Leaf System.

24.8.3 CIM_Dependency (Profile to Object Manager)

CIM_Dependency is an association between RegisteredProfile and the Object Manager that provides the management interface.

CIM_Dependency is not subclassed from anything.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if RegisteredProfiles of Leaf systems are modeled.

Table 254 describes class CIM_Dependency (Profile to Object Manager).

Table 254 - SMI Referenced Properties/Methods for CIM_Dependency (Profile to Object Manager)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Leaf Object Manager.
Dependent		Mandatory	The RegisteredProfile for the Leaf System.

24.8.4 CIM_Dependency (Systems)

CIM_Dependency is an association between a Leaf System and the Cascading System (ComputerSystem). The specific nature of the dependency is determined by associations between resources of the cascading system and resources of the leaf system.

CIM_Dependency is not subclassed from anything.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 255 describes class CIM_Dependency (Systems).

Table 255 - SMI Referenced Properties/Methods for CIM_Dependency (Systems)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Cascading System.
Dependent		Mandatory	The Leaf System.

24.8.5 CIM_ElementCapabilities

CIM_ElementCapabilities represents the association between ManagedElements (i.e.,ComputerSystem) and their capabilities (e.g., SNIA_CascadingCapabilities).

CIM_ElementCapabilities is not subclassed from anything.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 256 describes class CIM_ElementCapabilities.

Table 256 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

24.8.6 CIM_ElementConformsToProfile (Leaf)

CIM_ElementConformsToProfile is the association between the RegisteredProfile of the leaf profile and the system of the leaf (i.e., leaf ComputerSystem).

CIM_ElementConformsToProfile is not subclassed from anything.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if RegisteredProfiles of Leaf systems are modeled.

Table 257 describes class CIM_ElementConformsToProfile (Leaf).

Table 257 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (Leaf)

Properties	Flags	Requirement	Description & Notes
ConformantStandard		Mandatory	The RegisteredProfile of the leaf system.
ManagedElement		Mandatory	Reference to the top-level system of the leaf profile.

24.8.7 CIM_HostedCollection (Allocated Resources)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Cascading Subprofile, it is used to associate the Allocated Resources to the top level Computer System of the Cascading Profile.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 258 describes class CIM_HostedCollection (Allocated Resources).

Table 258 - SMI Referenced Properties/Methods for CIM_HostedCollection (Allocated Resources)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

24.8.8 CIM_HostedCollection (Remote Resources)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Cascading Subprofile, it is used to associate the Remote Resources to the top level Computer System of the Cascading Profile.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if SNIA_RemoteResources is modeled.

Table 259 describes class CIM_HostedCollection (Remote Resources).

Table 259 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

24.8.9 CIM_HostedService (Allocation Service)

CIM_HostedService is an association between a Service (SNIA_AllocationService) and the System (ComputerSystem) on which the functionality resides.

CIM_HostedService is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if SNIA_AllocationService is modeled.

Table 260 describes class CIM_HostedService (Allocation Service).

Table 260 - SMI Referenced Properties/Methods for CIM_HostedService (Allocation Service)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	>The AllocationService hosted on the System.

24.8.10CIM_HostedService (Object Manager)

CIM_HostedService is an association between a Service (SNIA_AllocationService) and the System (ComputerSystem) on which the functionality resides.

CIM_HostedService is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if Leaf ObjectManagers are modeled.

Table 261 describes class CIM_HostedService (Object Manager).

Table 261 - SMI Referenced Properties/Methods for CIM_HostedService (Object Manager)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

24.8.11CIM_LogicalDisk

A remote LogicalDisk that is imported to the referencing profile. If the referencing profile has access to the leaf profile, the data in this class should reflect what the referencing profile obtains from that profile. If the referencing profile does not have access to the leaf profile, then this should be filled out as best can be done.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '7' ('LogicalDisk').

Table 262 describes class CIM_LogicalDisk.

Table 262 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name		Mandatory	OS Device Name.
NameFormat		Mandatory	Format for name.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	Value shall be 2|3|6|8|15 (OK or Degraded or Error or Starting or Dormant).
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks of capacity consumed from the parent StoragePool.
ConsumableBlocks		Mandatory	The number of blocks usable by consumers.
IsBasedOnUnderlyingRedundancy		Mandatory	

Table 262 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
InstallDate	N	Optional	Not Specified in this version of the Profile.
StatusDescriptions	N	Optional	Not Specified in this version of the Profile.
HealthState	N	Optional	Not Specified in this version of the Profile.
EnabledState	N	Optional	Not Specified in this version of the Profile.
OtherEnabledState	N	Optional	Not Specified in this version of the Profile.
RequestedState	N	Optional	Not Specified in this version of the Profile.
EnabledDefault	N	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	N	Optional	Not Specified in this version of the Profile.

24.8.12CIM_LogicalIdentity (General)

Associates local resource (e.g., StorageExtent) to a remote (imported) resource (e.g., StorageVolume or LogicalDisk).

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 263 describes class CIM_LogicalIdentity (General).

Table 263 - SMI Referenced Properties/Methods for CIM_LogicalIdentity (General)

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	This is a reference to the remote (imported) resource.
SameElement		Mandatory	This is a reference to the local resource that maps to the remote (imported) resource.

24.8.13CIM_LogicalIdentity (LogicalDisk)

Associates local StorageExtent to a remote (imported) LogicalDisk.

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '7' ('LogicalDisk').

Table 264 describes class CIM_LogicalIdentity (LogicalDisk).

Table 264 - SMI Referenced Properties/Methods for CIM_LogicalIdentity (LogicalDisk)

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	This is a reference to the remote (imported) LogicalDisk.
SameElement		Mandatory	This is a reference to the local StorageExtent that maps to the remote (imported) LogicalDisk.

24.8.14CIM_LogicalIdentity (StorageVolume)

Associates local StorageExtent to a remote (imported) StorageVolume.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '3' ('StorageVolume').

Table 265 describes class CIM_LogicalIdentity (StorageVolume).

Table 265 - SMI Referenced Properties/Methods for CIM_LogicalIdentity (StorageVolume)

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	This is a reference to the remote (imported) StorageVolume.
SameElement		Mandatory	This is a reference to the local StorageExtent that maps to the remote (imported) StorageVolume.

24.8.15CIM_MemberOfCollection (Allocated Resources)

This use of MemberOfCollection is to collect all resource instances (in the AllocatedResources collection). Each association is created as a result of the Allocate method or as a side effect of a cascading profile specific operation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 266 describes class CIM_MemberOfCollection (Allocated Resources).

Table 266 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

24.8.16CIM_MemberOfCollection (Remote Resources)

This use of MemberOfCollection is to collect all resource instances (in the RemoteResources collection). Each association (and the RemoteResources collection, itself) is created through external means.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if SNIA_RemoteResources is modeled.

Table 267 describes class CIM_MemberOfCollection (Remote Resources).

Table 267 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

24.8.17CIM_Namespace (Leaf)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if Leaf ObjectManagers are modeled.

Table 268 describes class CIM_Namespace (Leaf).

Table 268 - SMI Referenced Properties/Methods for CIM_Namespace (Leaf)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
ObjectManagerCreationClassName		Mandatory	
ObjectManagerName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ClassType		Mandatory	
DescriptionOfClassType		Mandatory	Mandatory if ClassType is set to 'Other'.
ClassInfo		Optional	Deprecated in the MOF, but required for 1.0 compatibility.
DescriptionOfClassInfo		Optional	Deprecated in the MOF, but mandatory for 1.0 compatibility. Mandatory if ClassInfo is set to 'Other'.

24.8.18CIM_NamespaceInManager (Leaf)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if Leaf ObjectManagers are modeled.

Table 269 describes class CIM_NamespaceInManager (Leaf).

Table 269 - SMI Referenced Properties/Methods for CIM_NamespaceInManager (Leaf)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

24.8.19CIM_ObjectManager (Leaf)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 270 describes class CIM_ObjectManager (Leaf).

Table 270 - SMI Referenced Properties/Methods for CIM_ObjectManager (Leaf)

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
ElementName		Mandatory	
Description		Mandatory	
OperationalStatus		Mandatory	
Started		Mandatory	
StopService()		Optional	

24.8.20CIM_RegisteredProfile (Leaf)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 271 describes class CIM_RegisteredProfile (Leaf).

Table 271 - SMI Referenced Properties/Methods for CIM_RegisteredProfile (Leaf)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	This is a unique value for the profile instance.
RegisteredOrganization		Mandatory	This is the official name of the organization that created the Profile. For SMI-S profiles, this would be SNIA.
OtherRegisteredOrganization		Optional	
RegisteredName		Mandatory	This is the name assigned by the organization that created the profile.
RegisteredVersion		Mandatory	This is the version number of the organization that defined the profile.
AdvertiseTypes		Mandatory	Defines the advertisement of this profile. If the property is null then no advertisement is defined. A value of 1 is used to indicate 'other' and a 3 is used to indicate 'SLP'.
AdvertiseTypeDescriptions		Optional	This shall not be NULL if 'Other' is identified in AdvertiseType.

24.8.21CIM_RemoteServiceAccessPoint (Leaf)

CIM_RemoteServiceAccessPoint is an instance that provides access information for accessing the actual leaf profile via a management interface.

CIM_RemoteServiceAccessPoint is not subclassed from CIM_ServiceAccessPoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 272 describes class CIM_RemoteServiceAccessPoint (Leaf).

Table 272 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint (Leaf)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the management interface.
SystemName		Mandatory	The name of the Computer System hosting the management interface.
CreationClassName		Mandatory	The CIM Class name of the management interface.
Name		Mandatory	The unique name of the management interface.

24.8.22CIM_SAPAvailableForElement

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_RemoteServiceAccessPoint is modeled.

Table 273 describes class CIM_SAPAvailableForElement.

Table 273 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Leaf System.
AvailableSAP		Mandatory	

24.8.23CIM_StorageVolume

A remote StorageVolume that is imported to the referencing profile. If the referencing profile has access to the leaf profile, the data in this class should reflect what the referencing profile obtains from that profile. If the referencing profile does not have access to the leaf profile, then this should be filled out as best can be done.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '3' ('StorageVolume').

Table 274 describes class CIM_StorageVolume.

Table 274 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name	CD	Mandatory	The identifier for this volume.
OtherIdentifyingInfo	CD	Optional	Additional correlatable names.
IdentifyingDescriptions		Optional	
NameFormat		Mandatory	The type of identifier in the Name property.
NameNamespace		Mandatory	The namespace that defines uniqueness for the NameFormat.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	Value shall be 2|3|6|8|15 (OK or Degraded or Error or Starting or Dormant).
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks of capacity consumed from the parent StoragePool.
ConsumableBlocks		Mandatory	The number of blocks usable by consumers.
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	

Table 274 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
InstallDate	N	Optional	Not Specified in this version of the Profile.
StatusDescriptions	N	Optional	Not Specified in this version of the Profile.
HealthState	N	Optional	Not Specified in this version of the Profile.
EnabledState	N	Optional	Not Specified in this version of the Profile.
OtherEnabledState	N	Optional	Not Specified in this version of the Profile.
RequestedState	N	Optional	Not Specified in this version of the Profile.
EnabledDefault	N	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	N	Optional	Not Specified in this version of the Profile.

24.8.24CIM_SystemDevice (Leaf Devices)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if SNIA_CascadingCapabilities.SupportedElementTypes = '3'|'4'|'7' (StorageVolume | StorageExtent | LogicalDisk).

Table 275 describes class CIM_SystemDevice (Leaf Devices).

Table 275 - SMI Referenced Properties/Methods for CIM_SystemDevice (Leaf Devices)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Leaf Computer System that contains this device.
PartComponent		Mandatory	The logical device that is managed by a computer system.

24.8.25SNIA_AllocatedResources

An instance of a default SNIA_AllocatedResources defines the set of remote (leaf) resources that are allocated and in use by the Cascading Profile.

SNIA_AllocatedResources is subclassed from CIM_SystemSpecificCollection.

At least one instance of the SNIA_AllocatedResources shall exist for a Profile and shall be hosted by one of the ComputerSystems of that Profile.

Created By: Static

Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 276 describes class SNIA_AllocatedResources.

Table 276 - SMI Referenced Properties/Methods for SNIA_AllocatedResources

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the AllocatedResources collection (e.g., AllocatedVolumes).
ElementType		Mandatory	The type of remote resources collected by the AllocatedResources collection. For this version of SMI-S, the only value supported is '3' (StorageVolume).

24.8.26SNIA_AllocationService

The SNIA_AllocationService class provides methods for allocating and deallocating remote resources for use in the Cascading Profile.

The SNIA_AllocationService class is subclassed from CIM_Service.

There may be an instance of the SNIA_AllocationService if Allocation or Deallocation are supported.

The methods that are supported can be determined from the SynchronousMethodsSupported and AsynchronousMethodsSupported properties of the SNIA_CascadingCapabilities.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 277 describes class SNIA_AllocationService.

Table 277 - SMI Referenced Properties/Methods for SNIA_AllocationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
Allocate()		Optional	Support for this method is optional. This method allocates remote (leaf) resources to the AllocatedResources collection.
Deallocate()		Optional	Support for this method is optional. This method is used to remove remote (leaf) resources from the AllocatedResources collection.

24.8.27SNIA_CascadingCapabilities

An instance of the SNIA_CascadingCapabilities class defines the specific support provided with the implementation of the Cascading Profile.

There would be zero or one instance of this class in a profile. There would be none if the profile did not support the Cascading Subprofile. There would be exactly one instance if the profile did support the Cascading Subprofile.

SNIA_CascadingCapabilities class is subclassed from CIM_Capabilities.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 278 describes class SNIA_CascadingCapabilities.

Table 278 - SMI Referenced Properties/Methods for SNIA_CascadingCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
FeaturesSupported		Mandatory	ValueMap { '2', '3', '4', '5' }, Values {'Ownership', 'Leaf Credentials', 'OM Dependencies', 'Allocation Service'}.
SupportedElementTypes		Mandatory	For this version of SMI-S, only the value '3' (StorageVolume) is supported. ValueMap { '2', '3', '4', '5', '6', '7', '8' }, Values {'Any Type', 'StorageVolume', 'StorageExtent', 'StoragePool', 'ComputerSystem', 'LogicalDisk', 'FileShare'}.
SupportedSynchronousActions		Mandatory	ValueMap { '2', '3' }, Values {'Allocation', 'Deallocation'}.
SupportedAsynchronousActions		Mandatory	ValueMap { '2', '3' }, Values {'Allocation', 'Deallocation'}.

24.8.28SNIA_RemoteResources

An instance of a default SNIA_RemoteResources defines the set of remote (leaf) resources that are available to be used by the Cascading Profile.

SNIA_RemoteResources is subclassed from CIM_SystemSpecificCollection.

One instance of the SNIA_RemoteResources would exist for each Element type for a Profile and shall be hosted by one of the ComputerSystems of that Profile.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 279 describes class SNIA_RemoteResources.

Table 279 - SMI Referenced Properties/Methods for SNIA_RemoteResources

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the RemoteResources collection (e.g., RemoteVolumes).
ElementType		Mandatory	The type of remote resources collected by the RemoteResources collection. For this version of SMI-S, the only value supported is '3' (StorageVolume).

DEPRECATED

STABLE

25 Health Package

25.1 Description

Failures and abnormal occurrences are a common and expected part of monitoring, controlling, and configuring devices and applications. A SMI-S client needs to be prepared at all times to trap unexpected situations and take appropriate action. This package defines the general mechanisms used in the expression of health in SMI-S. This package does not define the particular way a particular profile, subprofile, or package reports health.

This package builds on *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 5 Health and Fault Management. In particular, this package defines the basis of all the sections that currently and will exist in this specification or future versions of same.

25.1.1 Error Reporting Mechanism

Error are reports for many reasons. Not all the reasons are directly related to the operation being imposed on the implementation by the client. It is therefore necessary for the client to be able to distinguish between errors that are associated to problems in the formation and invocation of a method, extrinsic or intrinsic, or are related to other conditions.

The client application may need to reform the method call itself, by fixing parameters for example, or the client may need to stop what its attempting. At a basic level, the client needs to know that this operation will succeed at all, given the prevailing conditions on the managed element. A client may also need to notify the end-user of the situation that is preventing the client from fulfilling its function. A HFM application may need to investigate the failure and develop a prognosis.

The types of errors are categorized in the three following types.

a) Errors associated to the method call

b) Errors caused by adverse prevailing conditions in the managed element

c) Errors causes by adverse prevailing conditions in the WBEM Server or related, infrastructural components

Obviously, the method called may not exist. There may be a spelling mistake for the method name. One or more of the parameters may be incorrectly formed, expressed, or otherwise invalid. The first type of error, type a, is designed to inform the client that the operation attempted is still valid, but that the request was faulty. The intent of such an error is to tell the client what is wrong with the method call and allow the method to be invoked again.

On the other hand, the device or application may be in some failure condition which prevents it from honoring this particular or several method calls. This type of error, type b, tells the client that the it is unlikely that the method being attempted will be honored. Specifically, the method execution is blocked by the prevailing condition being described in the error itself. Given the presence of both type a and type b error situations, the implementationshould report the type b error. In this case, it does not matter how many fixes are made to the method call, the method call will fail anyway.

The WBEM Service is a separate architectural element from the managed element itself. It can fail, even though the methods and the managed element itself are without error. For example, the WBEM Server may allow only a limited number of concurrent connection or request and reject all others. The server may be shutting down or starting up and thus be unable to process any requests at the time. Unlike type b errors, type c errors are usually transient in nature. Since a failure in the WBEM Server or its components constitutes a communications failure, the reporting of type c errors shall take precedence over all other existing error type conditions.

The WBEM Server returns a error response or a results response to the request, which contains the operation previously mentioned. Errors in WBEM may be reported through two ways. The status code itself provides basic failure information. The number of status codes is very limited. Also on conveyed on the error response, is a Error instance. The Error provides vastly most information than the status code and, as such, is a superior mechanism for reporting errors.

The CIM Error provides attributes to express the categorization and severity of the error. More importantly, the CIM Error and AlertIndication, to be discussed later, contain the exact expression of the nature of the error and additional parameters to that error.

25.1.2 Event Reporting Mechanism

It is not sufficient to simply report the adverse conditions of the device or application through the error reporting mechanism. Many of the adverse conditions that would be reported to a client application attempting control or configuration operations are also of interest to client applications monitoring the very same device or application.

The CIM Event model provides a special class for reporting event conditions, AlertIndication. The AlertIndication is used to report a device or application conditions that may also be represented in one or more other instances. When the implementation detects the presence of a supported condition, it generates an AlertIndication to those listening clients.

It is recommended that the type b and type c errors reported in also be reported through AlertIndications.

25.1.3 Standard Events

The expression of Error or an Alertindication is not entirely meaningful to the SMI-S client without the standardization. A client can use these classes to determine the category, severity, and some other characteristics of the event, but the client can not determine the exact nature of the event without this standardization.

Standard events are registered and this registry is maintained by some organization or company, like SNIA.

Primary event identification and characterization properties:

- OwningEntity
  This property defines the registration entity for the event. The entities that are in scope for SMI-S are “DMTF” and “SNIA”. If the OwningEntity is neither of these, then this specification provides no meaning for this event.
- MessageID
  This property defines an event identifier that is unique for the OwningEntity. The combination of the OwningEntity and MessageID defines the entry in the registry.
- Message
  This property contains the message that can be forwarded to the end-user. The message is built from using the static, MessageFormatString, and dynamic, MessageArguments, components. This text may be localized. This text is not intended for programmatic processing
- MessageArguments
  This property defines the variable content for the message. The client would programmatically process the arguments to get further details on the nature of the event. For example, the message argument can tell the client which method parameter has a problem and what the problem is.
- MessageFormatString
  This property defines the static component of the message. This property is not included in the event instance itself and is only present in the event registry.

25.1.4 Reporting Health

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 effect the corrective action being attempted.

When components fail or become degraded, they can cause other components to fail or become degraded. For an HFM application to report or attempt to diagnose the problem, the device or application should express what the cause and effect relationships are that define the extent of the components affected by the failure or degradation. The RelatedElementCausingError class provides just such a mechanism.

The cause and effect relationships identified by the RelatedElementCausingError association may be a chain of cause and effect relationships with many levels. Given that devices or applications are sometimes subject to several levels of decomposition, each level of may have its own set of these associations that represent the ranking of cause and effect relationships and their effect on the parent component on the given level.

25.1.5 Computer System Operational Status

For most profiles, the ComputerSystem class is used to define the top or head of the object hierarchy. A profile may allow for partitioning or clustering by having more than one ComputerSystem, but one ComputerSystem often represents the device or application representation. In this role, it is important the summary of the health of the device or application is declared in the ComputerSystem instance.

Table 280 - OperationalStatus Details

Primary Operational Status	Subsidiary Operational Status	Description
2 “OK”		The system has a good status.
2 “OK”	4 “Stressed”	The system is stressed, for example the temperature is over limit or there is too much IO in progress.
2 “OK”	5 “Predictive Failure”	The system will probably fail sometime soon.
3 “Degraded”		The system is operational but not at 100% redundancy. A component has suffered a failure or something is running slow.
6 “Error”		An error has occurred causing the system to stop. This error may be recoverable with operator intervention.
6 “Error”	7 “Non-recoverable error”	A severe error has occurred. Operator intervention is unlikely to fix it.
6 “Error”	16 “Supporting entity in error”	A modeled element has failed.
12 “No contact”		The provider knows about the array but has not talked to it since last reboot.
13 “Lost communication”		The provider used to be able to communicate with the array, but has now lost contact.
8 “Starting”		The system is starting up.
9 “Stopping”		The system is shutting down.
10 “Stopped”		The data path is OK but shut down, the management channel is still working.

OperationalStatus is an array. The primary and subsidiary statuses are both OperationalStatus property, and are summarized in Table 280. If the subsidiary operational status is present in the array, it is intended to provide additional clarification to the primary operational status. The implementation shall report one of the above combinations of statuses. It may also report additional statuses beyond the ones defined in Table 280.

The operational status combinations listed in Table 280 that include descriptions about “provider” (i.e., the CIM Provider), are only valid in those cases where the implementation of SMI-S employs a proxy provider.

The operational statuses listed in Table 280 shall not be used to report the status of the WBEM Server itself.

EXPERIMENTAL

25.1.6 Event Reporting

The implementation may report Event or AlertIndication instances. The profile, subprofile, or package that includes this package defines whether or not these events are supported and when the events are produced.

If the support Event or AlertIndication is implemented, then the implementation shall also support the common messages through both Errors and AlertIndications. This means that the implementation produce the common event listed in the registry when the condition, also described in the registry, is present.

It is mandatory to report error conditions through both AlertIndication or Lifecycle indication 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 heated, 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.

25.1.7 Fault Region

If the device or application is itself attempting to rectify an adverse condition reported through a standard error, then the implementation shall report what corrective action, if any, it is taking. This is necessary to prevent a HFM application from also trying to rectify the very same condition. An HFM application should avoid a interfering with ongoing corrective action taken by the device or application itself.

The corrective action may be a process, like hardware diagnostics or volume rebuild. In which case, the above requirement is fulfilled by expressing the instances representing the process.

The corrective action may be a state change, like reboot. In which case, the above requirement is fulfilled by expressing the state change in some CIM Instances.

In all cases, the profile, subprofile, or package that includes this package defines the standard events included and the associated, possible corrective actions taken in response to these events.

25.1.8 RelatedElementCausingError

This package provides a mechanism in which the effect of a component failure on other components can be reported. the RelatedElementCausingError association defines what components are causing a particular component to failure or become degraded.

Some effects are more germane to the failure or degradation than others. In other words, there are primary and second effects. This association provides a mechanism for ranking the effect. The implementing shall provide the EffectCorrelation property, but it recommended that the implementation also provide the FailureRelationshipInitiated and Ranking properties

If there are these cause and effect relationships, the RelatedElementCausingError association should be implemented to report the causes of the failure or degradation.

EXPERIMENTAL

25.1.9 HealthState

The HealthState property in LogicalDevice defines the state for a particular component. The OperationalStatus defines operational status. For example, a disk or port may be taken off-line for service. The component's health may still be OK or not OK. The two properties, when used in combination, disambiguate the health of the component. For example, a OperationStatus of 10 "Stopped" and a HealthState of 30 "Major Failure" means that the component is off-line and has failed. While a OperationalStatus of 10 "Stopped" and a HealthState of 5 "OK" for the very same component means that although the component is off-line, the component is still in good working order.

The HealthState of a component should not represent the health of any other component as well by way of a summary or aggregate health state. However, if the component is itself relies on other components for its health, because the component itself is an aggregate of components, then the HealthState may represent a summary HealthState by side-effect.

HealthState is a mandatory for all system device logical devices that are defined by the profile or subprofile that includes this package. It is recommended that HealthState is something other than 0 "Unknown". However, a component may report "Unknown" after it has reported one of the other HealthStates. When HealthState changes from 5 "OK", it is mandatory that a LogicalDevice report some other HealthState (e.g., 30 "Major Failure") before reporting 0 "Unknown". Such a requirement is necessary, so that the client can notice the adverse state change via polling or indication before the component is no longer responding.

25.2 Health and Fault Management Considerations

Not defined in this standard.

25.3 Cascading Considerations

Not defined in this standard.

25.4 Supported Subprofiles and Packages

Not defined in this standard.

25.5 Client Considerations and Recipes

Not defined in this standard.

Not defined in this standard.

25.6 Registered Name and Version

Health version 1.2.0 (Component Profile)

25.7 CIM Elements

Table 281 describes the CIM elements for Health.

Table 281 - CIM Elements for Health

Element Name	Requirement	Description
25.7.1 CIM_ComputerSystem	Mandatory	
25.7.2 CIM_LogicalDevice	Mandatory	

Table 281 - CIM Elements for Health

Element Name	Requirement	Description
25.7.3 CIM_RelatedElementCausingError	Optional	
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::OperationalStatus[*] <> PreviousInstance.CIM_ComputerSystem::OperationalStatus[*]	Mandatory	CQL -Operational Status change of the device and application.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Deprecated WQL -Operational Status change of the device and application.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDevice AND SourceInstance.CIM_LogicalDevice::HealthState <> PreviousInstance.CIM_LogicalDevice::HealthState	Mandatory	CQL -Health State change of the logical component.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDevice AND SourceInstance.HealthState <> PreviousInstance.HealthState	Mandatory	Deprecated WQL -Health State change of the logical component.

25.7.1 CIM_ComputerSystem

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 282 describes class CIM_ComputerSystem.

Table 282 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	
OperationalStatus		Mandatory	Overall status of the Host.

25.7.2 CIM_LogicalDevice

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 283 describes class CIM_LogicalDevice.

Table 283 - SMI Referenced Properties/Methods for CIM_LogicalDevice

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
HealthState		Mandatory	Reports the health of the component beyond the operational status.

25.7.3 CIM_RelatedElementCausingError

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 284 describes class CIM_RelatedElementCausingError.

Table 284 - SMI Referenced Properties/Methods for CIM_RelatedElementCausingError

Properties	Flags	Requirement	Description & Notes
FailureRelationshipInitiated		Optional	Reports the date and time when this cause and effect was created. The population of this property is RECOMMENDED.
EffectCorrelation		Mandatory	Describes the general nature of the cause and effect correlation.
Ranking		Optional	Describes the order of effect from 1, the highest effect, on. If there is only one of these associations between two elements, the ranking shall 1. Once more associations are added, then it RECOMMENDED that the implementation assist the client by stating which of the cause and effect relationship should be reviewed and addressed first. This property assists a client in accomplishing a triage of known problems.
Antecedent		Mandatory	Element causing the failure.
Dependent		Mandatory	

STABLE

STABLE

26 Job Control Subprofile

26.1 Description

In some profiles, some or all of the methods described may take some time to execute (longer than a HTTP time-out). In this case, a mechanism is needed to handle asynchronous execution of the method as a 'Job'.

This subprofile defines the constructs and behavior for job control for SNIA profiles that make use of the subprofile.

NOTE The subprofile describes a specific use of the constructs and properties involved. The actual CIM capability may be more, but this specification clearly states what clients may depend on in SNIA profiles that implement the Job Control Subprofile.

26.1.1 Instance Diagram

A normal instance diagram is provided in Figure 40.

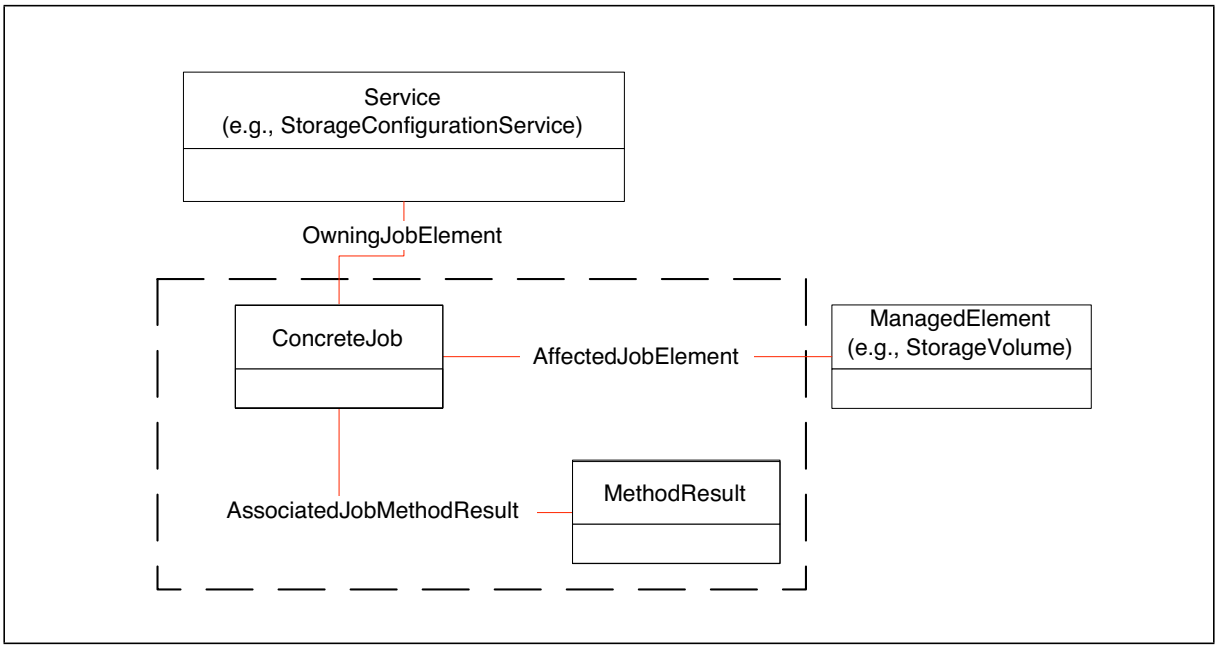

Figure 40 - Job Control Subprofile Model

When the Job Control Subprofile is implemented and a client executes a method that executes asynchronously, a reference to an instance of ConcreteJob is returned and the return value for the method is set to “Method parameters checked - job started”.

The ConcreteJob instance allows the progress of the method to be checked, and instance Indications can be used to subscribe for Job completion.

The associations OwningJobElement and AffectedJobElement are used to indicate the service whose method created the job by side-effect and the element being affected by the job. The job itself may create, modify and/or delete many elements during its execution. The nature of this affect is the creation or deletion of the instances or associations or the modification of instance properties. These elements, albeit regular instances or associations, are said to be *affected* by the job. The elements linked by AffectedJobElement may change through the execution of the job, and in addition, the job may be associated to more than one Input and/or Output elements or other elements affected by side-effect. Input

and Output elements are those referenced by method parameters of the same type, input and output parameters respectively.

EXPERIMENTAL

The following set of rules defines the nature of the AffectedJobElement associations for a given job in terms of the references passed as parameters to the service method that spawned the job. Obviously, the distinction of Input element from Output element in the following rules only makes sense if these parameters are not both Input and Output elements.

- If all Elements created by the method exists immediately upon the return from the method, then AffectedJobElement shall reference the Output Element.
- If the Output Element, one or more, does not exist until the job has completed, the AffectedJobElement shall reference the Input Element until the job completes, at which time AffectedJobElement shall then reference the Output Element instead.
- In the event the job fails and the Output Element created during the job and referenced by AffectedJobElement is no longer available, AffectedJobElement shall revert to referencing the Input Element.
- If the method affects elements without referencing elements as Output parameters, then the AffectedJobElement Association shall reference the Input element, one or more.
- If the method only modifies the elements referenced with method parameters, then the AffectedJobElement association references the modified elements. Elements modified by the job shall be reference by this association.
- If the method affects elements but references no elements as either Input or Output parameters or the only Input elements referenced are those of the elements to be deleted, then AffectJobElement associations shall exist to other elements that are affected by the job.
- Other elements whose references are not used in the method invocation, but that are created or modified by side-effect of the job's execution shall be associated to the job via the AffectJobElement association, but may cease to be associated once the job has finished execution.

The lifetime of a completed job instance, and thus the AffectedJobElement association to the appropriate Element is currently implementation dependent. However, the set of AffectedJobElement associations to Input and Output element present when the job finishes execution shall remain until the job is deleted.

26.1.2 MethodResult

Jobs are produced by side effect of the invocation of an extrinsic method. Reporting the resulting Job is the purpose of this subprofile. The MethodResult class is used to report the extrinsic method called and the parameters passed to the method. In this way, third party observers of a CIMOM can tell what the job is and what it is doing. A MethodResult instance contains the LifeCycle indications that have been or would have been produced as the result of the extrinsic method invocation. That is, the instance contains the indications whether or not there were the appropriate indication subscription at the time the indication were produced.

A client may fetch the method lifecycle indication produced when the method was called from the PreCallIndication attribute. This indication, an instance of InstMethodCall, contains the input parameters provided by the client that called the method.

A client may fetch the method lifecycle indication produced once the method execution was completed from the PostCallIndication. This indication contains the input parameters provided by the client that called the method and output parameters returned by the method implementation. Parameters that are

both input and output parameters will contain the output parameter provided by the method implementation.

EXPERIMENTAL

26.1.3 OperationalStatus for Jobs

The OperationalStatus property is used to communicate that status of the job that is created. As such, it is critical that implementations are consistent in how this property is set. The values that shall be supported consistently are:

- 2 “OK” - combined with 17 “Completed” to indicate that the job completed with no error.
- 6 “Error” - combined with 17 “Completed” to indicate that the job did not complete normally and that an error occurred.
- 10 “Stopped” implies a clean and orderly stop.
- 17 “Completed” indicates the Job has completed its operation. This value should be combined with either 2 “OK” or 6 “Error, so that a client can tell if the complete operation passed (Completed with OK), and failure (Completed with Error).

26.1.4 JobState for Jobs

The JobState property is used to communicate Job specific states and statuses.

- 2 “New” - Job was created but has not yet started
- 3 “Starting” - Job has started
- 4 “Running” - Job is current executing
- 5 “Suspended” - Job has been suspended. The Job may be suspended for many reasons like it has been usurped by a higher priority or a client has suspended it (not described within this subprofile).
- 6 “Shutting Down” - Job is completing its work, has been terminated, or has been killed. The Job may be cleaning up after only having completed some of its work.
- 7 “Completed” - Job has completed normally, its work has been completed successfully.
- 8 “Terminated” - Job has been terminated
- 9 “Killed” - Job has been aborted. The Job may not cleanup after itself.
- 10 “Exception” - Job failed and is in some abnormal state. The client may fetch the error conditions from the job. See 26.5.2.

Table 285 maps the standard mapping between the OperationalStatus and JobState properties on ConcreteJob. The actual values of the properties are listed in Table 285with the associated value from the property’s ValueMap qualifier.

Table 285 - OperationalStatus to Job State Mapping

OperationalStatus	JobState	Job is
2 “OK”, 17 “Completed”	7 “Completed”	Completed normally
6 “Error”, 17 “Completed”	10 “Exception”	Completed abnormally
10 “Stopped”	8 “Terminated”	Terminated

Table 285 - OperationalStatus to Job State Mapping

OperationalStatus	JobState	Job is
6 “Error”	9 “Killed”	Aborted / Killed
2 “OK”	4 “Running”	Executing
15 “Dormant”	2 “New”	Created but not yet executing
2 “OK”, 8 “Starting”	3 “Starting”	Starting up
2 “OK”	5 “Suspended”	Suspended
2 “OK”, 9 “Stopping”	6 “Shutting Down”	Terminated and potentially cleaning up
6 “Error”	6 “Shutting Down”	Killed and is aborting

26.1.5 Determining How Long a Job Remains after Execution

The Job shall report how long it will remain after it has finished executing, fails on its own, is terminated, or is killed. The TimeBeforeRemoval attribute reports a datetime offset.

The TimeBeforeRemoval and DeleteOnCompletion attributes are related. If the DeleteOnCompletion is FALSE, then the Job shall remain until is it explicitly deleted. If the DeleteOnCompletion is TRUE, then the Job shall exist for the length of time specified in the TimeBeforeRemoval attribute. An implementation may not support the setting of the DeleteOnCompletion attribute because it does not support the client modifying the Job instance.

The amount of time specified in the TimeBeforeRemoval should be five or more minutes. This amount of time allows a client to recognize that the Job has failed and retrieve the Error.

26.2 Health and Fault Management

The implementation should report CIM Errors from the ConcreteJob.GetError() method. See 25 Health Package for details.

EXPERIMENTAL

The standards messages specific to this profile are listed in Table 286. See 8 Standard Messages in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* for a description of standard messages and the list all standard messages

Table 286 - Standard Message for Job Control Subprofile

Message ID	Message Name
DRM22	Job failed to start
DRM23	Job was halted

EXPERIMENTAL

26.3 Cascading Considerations

Not defined in this standard.

26.4 Support Subprofiles and Packages

Not defined in this standard.

26.5 Methods of the Profile

26.5.1 Job Modification

A Job instance may be modified. The DeleteOnCompletion and TimeBeforeRemoval properties are writable. If the intrinsic ModifyInstance method is supported, then the setting of both attributes shall be supported.

EXPERIMENTAL

26.5.2 Getting Error Conditions from Jobs

```
uint32 GetError(
                                [Out, EmbeddedObject] string Error);
```

This method is used to fetch the reason for the job failure. The type of failure being report is when a Job stops executing on its own. That is, the Job was not killed or terminated. An Embedded Object, encoded in a string, shall returned if the method is both supported and the job has failed. The Job shall report the 10 “Exception” status when the Job has failed on its own.

The GetError method should be supported.

The Error string contains a Error instance. See 25 Health Package for details on how to process this CIM Instance.

EXPERIMENTAL

26.5.3 Suspending, Killing or Terminating a Job

A Job may be suspended, terminated or killed. Suspending a Job means that the Job will not be executing and be suspended until it is resumed. Terminating a job means to request that the Job stop executing and that the Job clean-up its state prior to completing. Killing a job means to request that the Job abort executing, usually meaning there is little or no clean-up of Job state.

```
uint32 RequestStateChange(
                            [In] RequestedState,
                            [In] TimeoutPeriod);
```

A client may request a state change on the Job.

- RequestedState - The standard states that can be requested are “Start”, “Suspend”, “Terminate”, “Kill”, “Service”. A new Job may be started. A suspended Job may be resumed, using the “Started” requested status. A executing Job may be suspended, terminated, or killed. A new or executing Job may be put into the “Service” state. The “Service” state is vendor specific. An implementation can indicate what state transitions are supported by not returning the 4 098 “Invalid State Transition” return code
- TimeoutPeriod - The client expects the state transition to occur within the specified amount of time. The implementation may support the method but not this parameter.

Return codes:

- 0 “Completed with No Error”
- 1 “Not Supported” - The method is not supported
- 2 “Unknown/UnSpecified Error” - Failure for some vendor specific reason

- 3 “Can not complete within Timeout Period” - The requested amount of time is less than how long the requested state transition takes
- 4 “Failed”
- 5 “Invalid Parameters” - The parameters are incorrect
- 6 “In Use” - Another client has requested a state change that has not completed
- 4 096 “Method Parameters Checked - Transition Started” - The method can return before the state transition completes. This error code tells that calling that this situation has occurred
- 4 097 “Invalid State Transition” - The state change requested is invalid for the current state. 4 098 “Use of Timeout Parameter Not Supported” - This implementation does not support the TimeoutPeriod parameter. A client may pass a NULL for the TimeoutPeriod and try again. There is no mechanism to determine what state changes are supported by a particular implementation. Such a mechanism is planned for a future version of this specification.
- 4 099 “Busy” - A state change is underway in the Job and, as such, the state can not be changed. An implementation may use this return code to indicate the job can not be suspended, killed, or terminated at all or in the current phase of execution

26.6 Client Considerations and Recipes

If the operation will take a while (longer than an HTTP timeout), a handle to a newly minted ConcreteJob is returned. This allows the job to continue in the background. Note a few things:

- The job is associated to the Service via OwningJobElement and is also linked to the object being modified/created via AffectedJobElement. For example, a job to create a StorageVolume may start off pointing to a Pool until the Volume is instantiated at which point the association would change to the StorageVolume.
- These jobs do not have to get instantiated! If the method completes quickly, a null can be returned as a handle, as illustrated in Figure 41.
- It may take some time before the Job starts.
- A Job may be terminated or killed.
- Jobs may be modified.
- Jobs may be restarted.

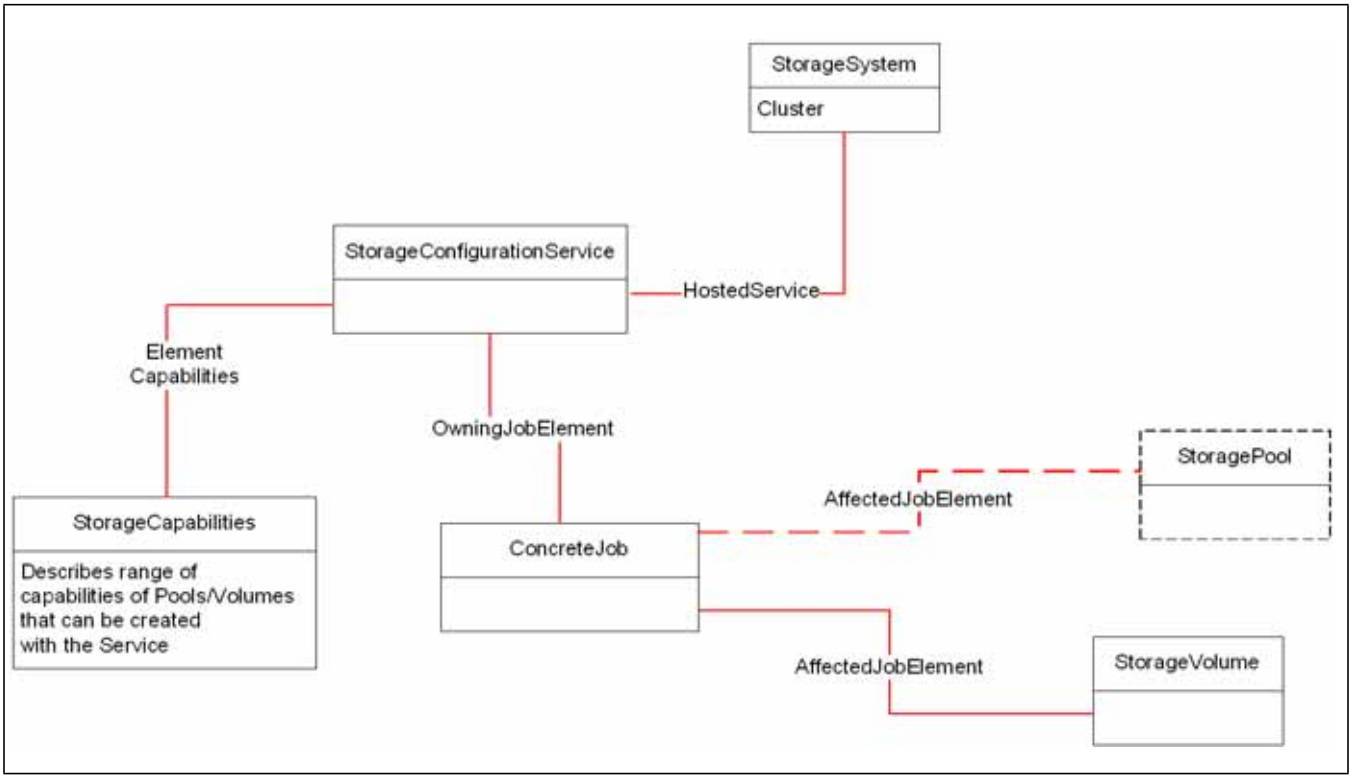

Figure 41 - Storage Configuration

26.7 Registered Name and Version

Job Control version 1.5.0 (Component Profile)

26.8 CIM Elements

Table 287 describes the CIM elements for Job Control.

Table 287 - CIM Elements for Job Control

Element Name	Requirement	Description
26.8.1 CIM_AffectedJobElement	Mandatory	
26.8.2 CIM_AssociatedJobMethodResult	Mandatory	
26.8.3 CIM_ConcreteJob	Mandatory	
26.8.4 CIM_MethodResult	Mandatory	
26.8.5 CIM_OwningJobElement	Mandatory	
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.CIM_ConcreteJob::JobStatus <> PreviousInstance.CIM_ConcreteJob::JobStatus	Optional	CQL -Deprecated. Modification of Job Status for a Concrete Job.

Table 287 - CIM Elements for Job Control

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.JobStatus <> PreviousInstance.JobStatus	Optional	Deprecated WQL -Deprecated: Modification of Job Status for a Concrete Job.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.CIM_ConcreteJob::PercentComplete <> PreviousInstance.CIM_ConcreteJob::PercentComplete	Mandatory	CQL -Modification of Percentage Complete for a Concrete Job.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.PercentComplete <> PreviousInstance.PercentComplete	Mandatory	Deprecated WQL -Modification of Percentage Complete for a Concrete Job.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND ANY SourceInstance.CIM_ConcreteJob::OperationalStatus[*] = 17 AND ANY SourceInstance.CIM_ConcreteJob::OperationalStatus[*] = 2	Mandatory	CQL -Modification of Operational Status for a Concrete Job to 'Complete' and 'OK'.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.OperationalStatus = 17 AND SourceInstance.OperationalStatus = 2	Mandatory	Deprecated WQL -Modification of Operational Status for a Concrete Job to 'Complete' and 'OK'.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND ANY SourceInstance.CIM_ConcreteJob::OperationalStatus[*] = 17 AND ANY SourceInstance.CIM_ConcreteJob::OperationalStatus[*] = 6	Mandatory	CQL -Modification of Operational Status for a Concrete Job to 'Complete' and 'Error'.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.OperationalStatus = 17 AND SourceInstance.OperationalStatus = 6	Mandatory	Deprecated WQL -Modification of Operational Status for a Concrete Job to 'Complete' and 'Error'.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.CIM_ConcreteJob::JobState <> PreviousInstance.CIM_ConcreteJob::JobState	Mandatory	CQL -Modification of Job State for a Concrete Job.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.JobState <> PreviousInstance.JobState	Mandatory	Deprecated WQL -Modification of Job State for a Concrete Job.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ConcreteJob	Mandatory	Creation of a ConcreteJob.

26.8.1 CIM_AffectedJobElement

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 288 describes class CIM_AffectedJobElement.

Table 288 - SMI Referenced Properties/Methods for CIM_AffectedJobElement

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	The ManagedElement affected by the execution of the Job.
AffectingElement		Mandatory	The Job that is affecting the ManagedElement.

26.8.2 CIM_AssociatedJobMethodResult

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 289 describes class CIM_AssociatedJobMethodResult.

Table 289 - SMI Referenced Properties/Methods for CIM_AssociatedJobMethodResult

Properties	Flags	Requirement	Description & Notes
Job		Mandatory	The Job that has parameters.
JobParameters		Mandatory	The parameters for the method which by side-effect created the Job.

26.8.3 CIM_ConcreteJob

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 290 describes class CIM_ConcreteJob.

Table 290 - SMI Referenced Properties/Methods for CIM_ConcreteJob

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
Name		Mandatory	The user-friendly name for this instance of Job. In addition, the user-friendly name can be used as a property for a search or query. (Note: Name does not have to be unique within a namespace.)".
OperationalStatus		Mandatory	Describes whether the Job is running or not.
JobStatus		Optional	Add additional detail beyond OperationalStatus about the runtime status of the Job. This property is free form and vendor specific.
JobState		Mandatory	Add additional detail beyond the OperationalStatus about the runtime state of the Job.
ElapsedTime		Optional	The time interval that the Job has been executing or the total execution time if the Job is complete.

Table 290 - SMI Referenced Properties/Methods for CIM_ConcreteJob

Properties	Flags	Requirement	Description & Notes
PercentComplete		Mandatory	The percentage of the job that has completed at the time that this value is requested. Optimally, the percentage should reflect the amount of work accomplished in relation to the amount of work left to be done. 0 percent complete means that the job has not started and 100 percent complete means the job has finished all its work. However, in the degenerate case, 50 percent complete means that the job is running and may remain that way until the job completes.
DeleteOnCompletion		Mandatory	Indicates whether or not the job should be automatically deleted upon completion. If this property is set to false and the job completes, then the extrinsic method DeleteInstance shall be used to delete the job versus updating this property. Even if the Job is set to delete on completion, the job shall remain for some period of time, see GetError() method.
ErrorCode		Optional	A vendor specific error code. This is set to zero if the job completed without error.
ErrorDescription		Optional	A free form string containing the vendor error description.
TimeBeforeRemoval		Mandatory	The amount of time the job will exist after the execution of the Job if DeleteOnCompletion is set to FALSE. Jobs that complete successfully or fail shall remaining for at least this period of time before being removed from the model (CIMOM).
GetError()		Mandatory	This method is used to retrieve the error that caused the Job to fail. The Job shall remain in the model long enough to allow client to a) notice that the job was stopped executing and b) to retrieve the error using this method. There are not requirements for how long the job must remain; however, it is suggested that the Job remain for at least five minutes. JobStatus=10 (Exception) tell the client that the job failed and this method can be called to retrieve the reason why embedded in the CIM_Error, see GetError() method.
RequestStateChange()		Optional	This method changes the state of the job. The client may suspend, terminate, or shutdown the job. To terminate a job means to request a clean shutdown of the job, have it finish some portion of it's work and terminate or to roll back the changes done by the job to date. The implement can make the choice which behavior. To kill a job means to abort the job, perhaps leaving some element of the work partially done and in an unknown state.

26.8.4 CIM_MethodResult

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 291 describes class CIM_MethodResult.

Table 291 - SMI Referenced Properties/Methods for CIM_MethodResult

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
PreCallIndication		Mandatory	Contains a copy of the CIM_InstMethodCall produced when the configuration or control change method was called. This Embedded Instance shall contain the configuration or control change extrinsic method name (MethodName) and parameters (MethodParameters).
PostCallIndication		Mandatory	Contains a copy of the CIM_InstMethodCall produced when the configuration or control change method has completed execution and control was returned to the client. This Embedded Instance shall contain the configuration or control change extrinsic method name (MethodName) and parameters (MethodParameters).

26.8.5 CIM_OwningJobElement

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 292 describes class CIM_OwningJobElement.

Table 292 - SMI Referenced Properties/Methods for CIM_OwningJobElement

Properties	Flags	Requirement	Description & Notes
OwningElement		Mandatory	The ManagedElement responsible for the creation of the Job. (e.g., StorageConfigurationService).
OwnedElement		Mandatory	The Job created by the ManagedElement.

STABLE

STABLE

27 Location Subprofile

27.1 Description

Associated with product information, a PhysicalPackage may also have a location. This is indicated using an instance of a Location class and the PhysicalElementLocation association.

27.1.1 Instance Diagram

Figure 42 illustrates a typical instance diagram.

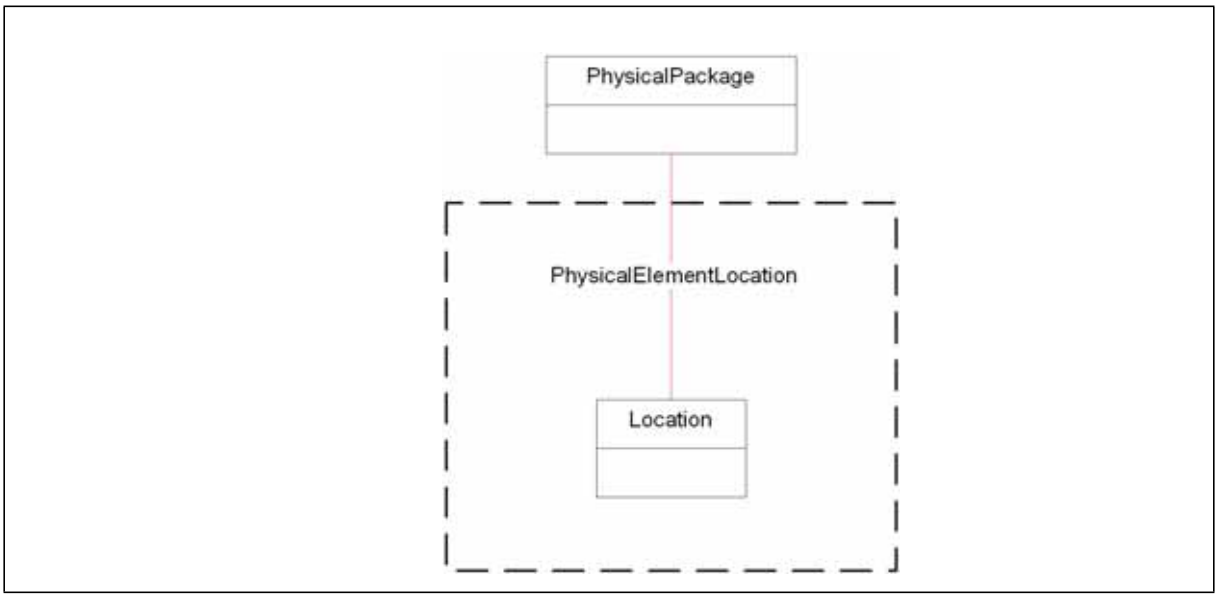

Figure 42 - Location Instance

27.2 Health and Fault Management Considerations

Not defined in this standard.

27.3 Cascading Considerations

Not defined in this standard.

27.4 Supported Subprofiles and Packages

None.

27.5 Methods of the Profile

None.

27.6 Client Considerations and Recipes

None

27.7 Registered Name and Version

Location version 1.4.0 (Component Profile)

27.8 CIM Elements

Table 293 describes the CIM elements for Location.

Table 293 - CIM Elements for Location

Element Name	Requirement	Description
27.8.1 CIM_Location	Mandatory	
27.8.2 CIM_PhysicalElementLocation	Mandatory	Associates the location to package.

27.8.1 CIM_Location

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 294 describes class CIM_Location.

Table 294 - SMI Referenced Properties/Methods for CIM_Location

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	A free-form string defining a label for the Location.
PhysicalPosition		Mandatory	A free-form string indicating the placement of a PhysicalElement.
ElementName		Optional	User-friendly name.
Address		Optional	A free-form string indicating a street, building or other type of address for the PhysicalElement's Location.

27.8.2 CIM_PhysicalElementLocation

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 295 describes class CIM_PhysicalElementLocation.

Table 295 - SMI Referenced Properties/Methods for CIM_PhysicalElementLocation

Properties	Flags	Requirement	Description & Notes
Element		Mandatory	
PhysicalLocation		Mandatory	

STABLE

DEPRECATED

28 Extra Capacity Set Subprofile

The functionality of the Extra Capacity Set Subprofile has been replaced by the 30 Multiple Computer System Subprofile.

The Extra Capacity Set Subprofile is defined in section B.8 of SMI-S 1.0.2.

DEPRECATED

DEPRECATED

29 Cluster Subprofile

The functionality of the Cluster Subprofile has been subsumed by 30 Multiple Computer System Subprofile.

The Cluster Subprofile is defined in section 7.3.3.3 of SMI-S 1.0.2. Any instrumentation that complies to the Multiple Computer System Subprofile defined in this specification may also claim compliance to that version of the Cluster Subprofile and may register as both a 1.1.0 Multiple Computer System Subprofile and 1.0.2 Cluster Subprofile.

DEPRECATED

STABLE

30 Multiple Computer System Subprofile

30.1 Description

The Multiple Computer System Subprofile models multiple systems that cooperate to present a “virtual” computer system with additional capabilities or redundancy. This virtual aggregate system is sometimes referred to as a cluster. and is illustrated in Figure 43.

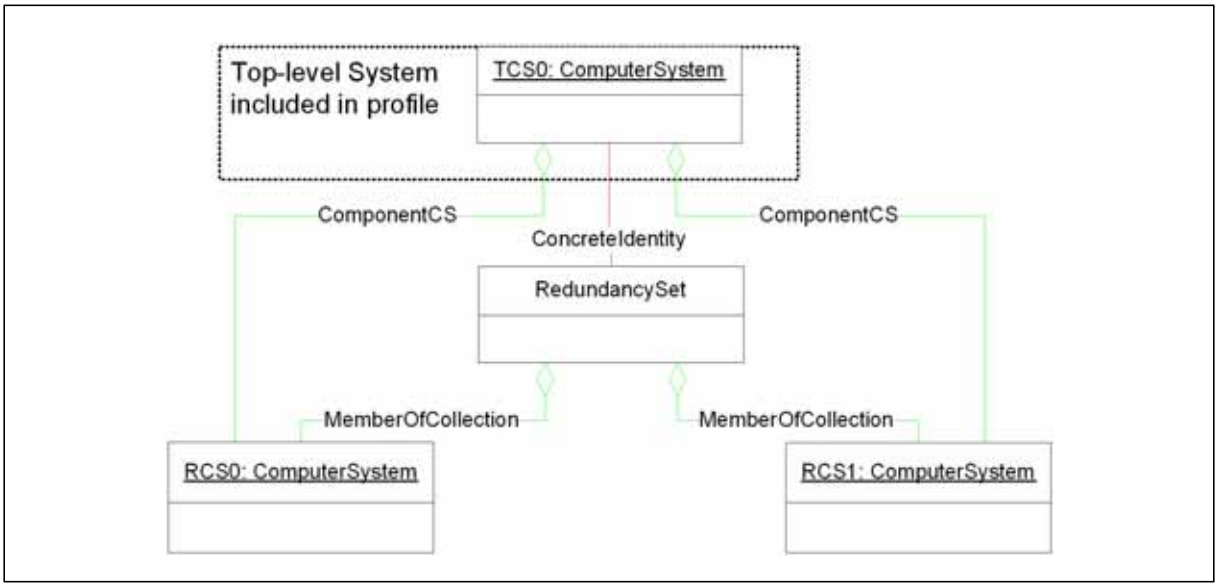

Figure 43 - Two Redundant Systems Instance Diagram

The general pattern for the redundancy aspect of Multiple Systems uses an instance of RedundancySet to aggregate multiple “real” ComputerSystem instances (labeled RCS0 and RCS1 in the diagram). Another ComputerSystem instance (TCS0) is associated to the RedundancySet instance using a ConcreteIdentity association and is associated to the real ComputerSystems using ComponentCS.

30.1.1 Top Level System

The top (“virtual”) system in this diagram (labeled TCS0) is referred to as the Top Level System. Note that for single-system configurations, the top-level system is the only system. Top-level systems have characteristics different from the underlying ComputerSystem instances.

The Top Level System is associated to the registered profile described in 40 Server Profile. Other elements such as LogicalDevices (ports, volumes), ServiceAccessPoints, and Services are associated to the top-level system if these elements are supported by multiple underlying systems (for example, the underlying systems provide failover and/or load balancing). Alternatively, elements can be associated to an underlying system if that system is a single point of failure. For example, a RAID array may associate StorageVolume instances to a top-level system since these are available when one underlying system (RAID controller) fails, all the port elements are associated to one underlying system because the ports become unavailable when this system fails.

The Dedicated property is required for top-level systems. Each profile defines the values that are appropriate for Dedicated.

30.1.2 Non-Top-Level Systems

Each ComputerSystem instance shall have a unique Name property. For non-top-level systems, Name may be vendor-unique; in which case, NameFormat shall be set to “Other”.

ComputerSystem.Dedicated should not be used in non-top-level systems.

Non-top-level systems shall not be associated to registered profiles or subprofiles.

Each non-top-level ComputerSystem shall be associated to the top-level system using ComponentCS. Note that non-top-level systems may not be members of a RedundancySet. For example, a top-level system may be associated to a RedundancySet with two systems as described in Figure 43 and also associated via ComponentCS to another Computer (not a member of a RedundancySet) representing a service processor.

30.1.3 Types of RedundancySets

The TypeOfSet property of RedundancySet is a list describing the types of redundancy. Its values are summarized in Table 296.

Table 296 - Redundancy Type

Redundancy Type	Description
N+1	All ComputerSystems are active, are unaware and function independent of one another. However, there exists at least one extra ComputerSystem to achieve functionality.
Load Balanced	All computer systems are active. However, their functionality is not independent of each other. Their functioning is determined by some sort of load balancing algorithm (implemented in hardware and/or software). 'Sparing' is implied (i.e., each computer system can be a spare for the other(s).
Sparing	All computer systems are active and are aware of each other. However, their functionality is independent until failover. Each computer system can be a spare for the other(s).
Limited Sparing	All members are active, and they may or may not be aware of each and they are not spares for each other. Instead, their redundancy is indicated by the IsSpare relationship.
Other/Unspecified	The relationship between the computer systems is not specified.

30.1.4 Multiple Tiers of Systems

The diagram above describes two tiers of systems; the real systems (labeled RCS0 and RCS1) in the lower tier are aggregated into a top-level system (TCS0) in the upper tier. There may be more than two tiers, as depicted in Figure 44.

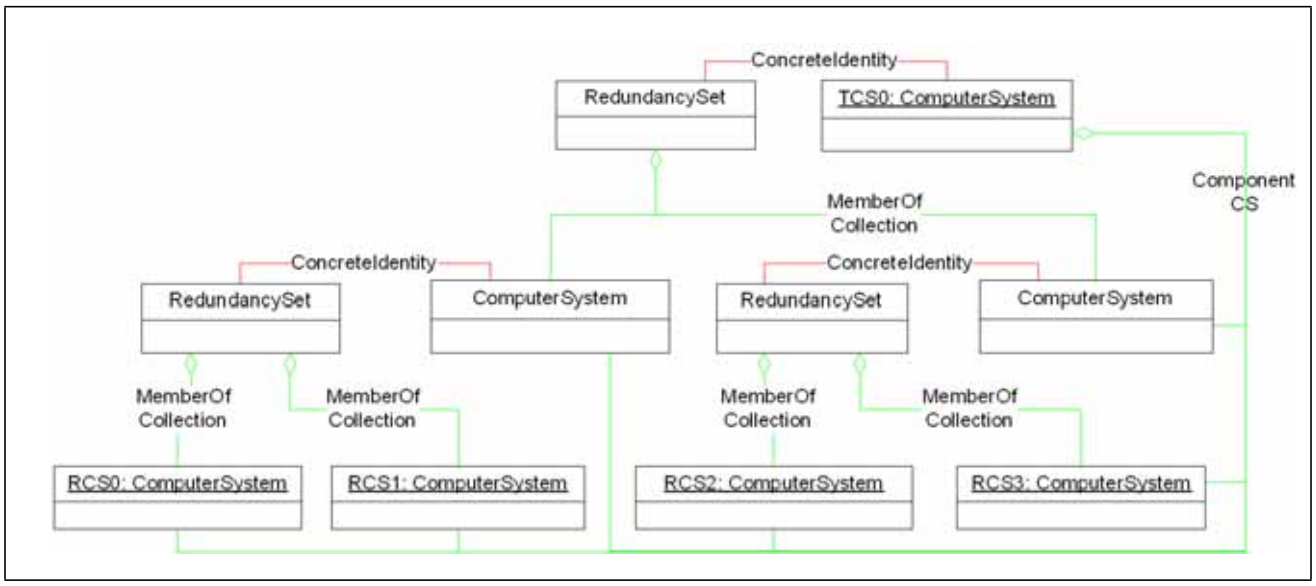

Figure 44 - Multiple Redundancy Tier Instance Diagram

The systems in the bottom tier (RCS0-RCS3) represent "real" systems.

RedundancySet.TypeOfSet can be used as part of multiple tier configurations to describe different types of redundancy at different tiers. For example, a virtualization system has four controllers that operate in pairwise redundancy. This could be modeled using the model in the diagram above and setting TypeOfSet in the top RedundancySet to “N+1” and setting TypeOfSet to “LoadBalancing” in the lower two RedundacySets.

30.1.5 Associations between ComputerSystems and other Logical Elements

SystemDevice associates device (subclasses of LogicalDevice such as LogicalPort or StorageVolume) and ComputerSystem instances. The cardinality of SystemDevice is one-to-many; a LogicalDevice may be associated with one and only one ComputerSystem. If the device availability is equivalent to that of the top-level system, it shall be associated to the top-level system via SystemDevice. If the device may become unavailable while the system as a whole remains available, the device shall be associated to a non-top-level system that has availability equivalent to the device. This system could be a real system or a system in an intermediate tier (representing some redundancy less than full redundancy).

This same approach shall be used for all other logical CIM elements with associations to systems. For example, HostedService and HostedAccessPoint shall associate elements (services, access points, and protocol endpoints) to the ComputerSystem with availability to the element.

Based on the arrangement of systems in figure 31, associations from systems to service and capabilities classes shall not be lower than associations to other classes. For the purpose of formally stating this rule, each ComputerSystem is assigned a level number. The profile's top-level ComputerSystem has level number 0. The ComputerSystem instances that are members of RedundancySets associated via ConcreteIdentity to the top-level system have level number 1. The members of redundancy sets associated to the level number 1 systems via ConcreteIdentity have level number 2. In general, the ComputerSystem members of redundancy sets associated to the level number n systems via ConcreteIdentity have level number n+1. The level of non-system objects is the level of the

ComputerSystem instance associated to the object via associations such as SystemDevice, HostedAccessPoint, HostedService, or ElementCapabilities.

Figure 45 demonstrates these system level numbers using the same configuration from Figure 44. Note that ComponentCS diagrams are omitted from this diagram to avoid clutter.

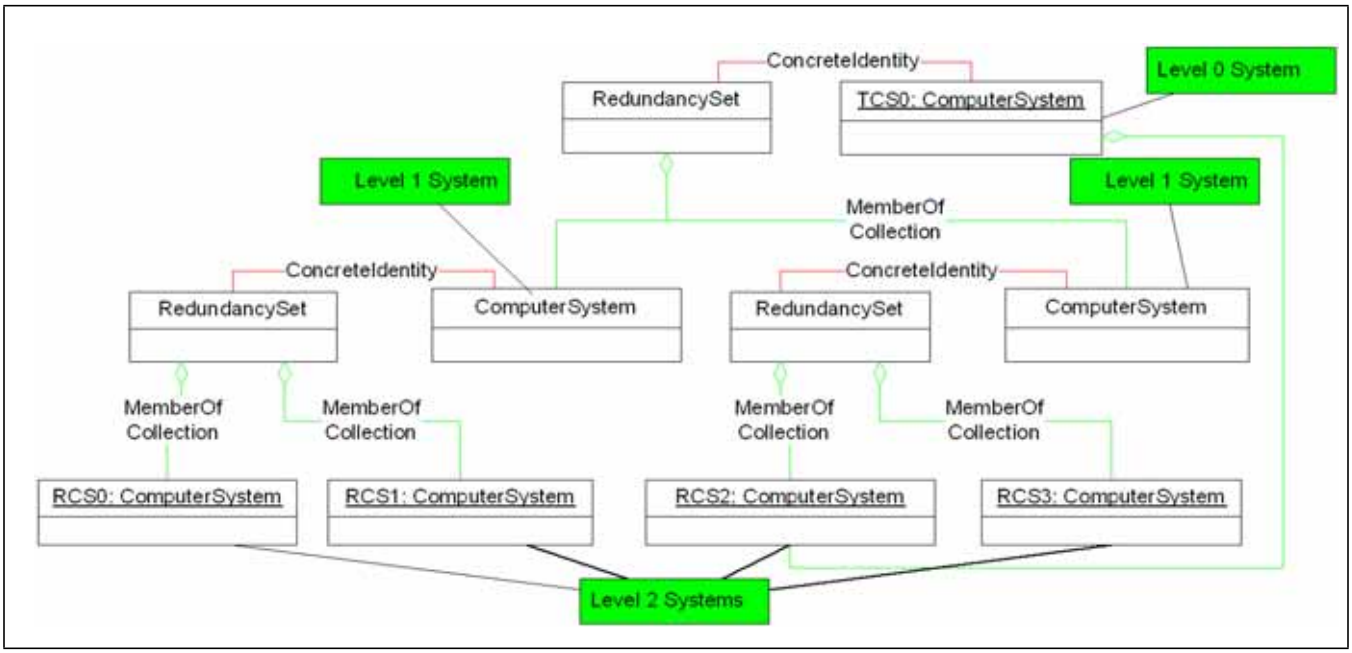

Figure 45 - System Level Numbers

All subclasses of CIM_Service and CIM_Capabilities shall have a level number less than or equal to the level number of storage classes (ports, volumes, etc.) that are influenced by the properties and methods of the Service and Capabilities classes. In some cases, different storage classes are influenced by different Service or Capabilities classes; the “level number less than or equal to” requirement may apply differently to different Service/Capabilities classes. It is always valid to associate Service and Capabilities classes to the top-level ComputerSystem since the level number of the top-level system (0) is always less than or equal to the level number of any other system.

Example 1 - An array with two controllers is modeled as a top-level ComputerSystem with real systems representing the controllers. The system’s storage volumes remain available when one controller fails, but each LogicalPort becomes unavailable when a controller fails. The StorageVolumes should be associated to the top-level ComputerSystem and the LogicalPorts should be associated to one of the real ComputerSystems.

Example 2 - An array with four pair-wise redundant controllers. Each LogicalPort is associated with a pair of controllers - if one controller in a pair fails, the port is still accessible through the alternate controller. This corresponds to Figure 44; the ports should be associated with one of the ComputerSystems in the middle tier.

A provider shall delete and create associations between ComputerSystems and logical elements (e.g., ports, logical devices) during failover or failback to represent changes in availability. This includes SystemDevice, HostedAccessPoint, HostedService, or HostedFileSystem associations (and other associations weak to systems). The effect of the creation and deletion of associations is to switch these elements from one ComputerSystem to another. The profiles that include Multiple Computer System Subprofile shall specify the affected associations and indications for creation and deletion of these associations.

30.1.6 Associations between ComputerSystems and PhysicalPackages and Products

The relationship between ComputerSystems, PhysicalPackages, and Products is defined in the Physical Package Package (see 31 Physical Package Package) which may be required by the profile including the Multiple Computer System Subprofile. Typically, the top-level system is associated to a PhysicalPackage which is associated to a Product. Non-top-level systems may also be associated to PhysicalPackage and indirectly to a Product. If all underlying ComputerSystems share the same physical package, a single PhysicalPackage should be associated to the upper ComputerSystem.

The relationships between ComputerSystems, redundancy sets, and CIM logical elements serve as a redundancy topology - informing the client of the availability of subsets of logical elements. The relationships between PhysicalPackages and logical elements serve as a physical topology. These two topologies need not be equivalent. Consider these examples:

Example 1: a RAID array with a single controller (no redundancy); the controller and all backend disks are housed in a single chassis. This is modeled as a single ComputerSystem, no RedundancySets, no ComponentCS associations, and a single PhysicalPackage with a single associated Product.

Example 2: a RAID array with two redundant controllers; both controllers and all backend disks share a single chassis. In this case, the redundancy topology matches Figure 43. The top-level ComputerSystem is associated to a PhysicalPackage with a single associated Product.

Example 3: two arrays described in example 1 are assembled as part of common rack and sold as a single product. Note that although there are two controllers, there is no redundancy - the two controllers act completely independently. This is modeled as two top-level computer systems attached to separate PhysicalPackages (representing the two internal chassis); These two PhysicalPackages have a Container association to third PhysicalPackage representing the assembly - which has an association to a Product.

Example 4: two arrays described in Example 1 are assembled as part of a common rack and also share a high-speed trunk and a mutual failover capability. This failover capability means the two controllers share a RedundancySet and common top-level system. The result is similar to example 2, but each real ComputerSystem is now associated to separate PhysicalPackages which have Contiainer associations to a common PhysicalPackage.

30.1.7 Storage Systems without Multiple Systems

In configurations where the instrumentation does not model multiple ComputerSystem instances, all the associations described above reference the one and only ComputerSystem.

30.1.8 Durable Names and Correlatable IDs of the Subprofile

This subprofile does not impose any requirements on names. The requirements for ComputerSystem names are defined in the profiles that depend on Multiple Computer System Subprofile and in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 7 Correlatable and Durable Names. Clients should not expect that a network name or IP address is exposed as a ComputerSystem property. The Access Points Subprofile should be used to model a network access point.

30.2 Health and Fault Management Considerations

The requirements for OperationalStatus of a ComputerSystem are discussed in 25 Health Package.

30.3 Cascading Considerations

None

30.4 Supported Subprofiles and Packages

Table 297 describes the supported profiles for Multiple Computer System.

Table 297 - Supported Profiles for Multiple Computer System

Profile Name	Organization	Version	Requirement	Description
Storage Server Asymmetry	SNIA	1.4.0	Optional	

30.5 Methods of the Profile

This subprofile does not include any extrinsic methods. A client may use this subprofile to discover information about the topology of computer systems, but cannot change the topology.

30.6 Client Considerations and Recipes

A client cannot generally, interoperably navigate the redundancy topology using ComponentCS because some Component CS associations may not parallel RedundancySet associations. But a client may use ComponentCS selectively to speed up certain tasks. In particular, a client may locate the top-level system from other ComputerSystems using ComponentCS.

30.6.1 Find Top-level Computer Systems

Top-level systems are the only objects in SMI-S associated to RegisteredProfile via ElementConformsToProfile. (See 41.5.5.)

30.6.2 Find the Top-level Computer System for any LogicalDevice

```
/
// DESCRIPTION:
// Find the Top-level Computer System for any CIM_LogicalDevice
//
// Preconditions:
//  $Device - Reference the LogicalDevice
//
// Find Systems associated to $Device
$Systems->[] = AssociatorNames($Device->,   // ObjectName
        “CIM_SystemDevice”,             // AssocClass
        “CIM_System”,                   // ResultClass
        “PartComponent”,                // Role
        “GroupComponent”)               // ResultRole
if ($Systems == null || $Systems->[].size != 1) {
    <ERROR! must be exactly one ComputerSystem Associated via
         SystemDevice to each LogicalDevice instance>
}

// System->[0] is the associated system; see if it’s the
// top-level system for the scoping profile.  All ComponentCS
// association GroupComponent references must refer to the
// profile’s top-level system.
```

```
$UpperSystems->[] = AssociatorNames($System->[0],
     “CIM_ComponentCS”,// AssocClass
     “CIM_ComputerSystem”,// ResultClass
     “PartComponent”,// Role
     “GroupComponent”)   // ResultRole
if ($UpperSystems != null && $UpperSystems->[].size > 1) {
// The restriction below is a characteristic of this subprofile
// and matches the DMTF Partinion white paper.
    <ERROR! must be no more than one ComputerSystem Associated
         via ComponentCS to each LogicalDevice instance>
}
// If an upper system was found, it must be the top-level
// system; if not, then the system associated to the device
// must be the top-level system
if ($UpperSystems->[].size == 1) {
  $TopLevelSystem =  $UpperSystems->[0]
} else {
  $TopLevelSystem = $System->[0]
}

// The remaining steps are not needed to locate the top-level
// system, but validate the classes and associations.
//
// The system associated to the device may also be part of a RedundancySet.
// If so, follow a chain from that system to the RedundancySet, then
// follow ConcreteIdentity to a system - then check to see if it has
// ConponentCS to the top-level system.  Keep iterating till no more
// RedundancySets - this must be the same system as TopLevelSystem.
do {
    // Get the RedundancySet that $System->[0] is a member of
    $RedundancySets->[] = AssociatorNames($System->[0],
                “CIM_MemberOfCollection”,
                “CIM_RedundancySet”,
                “Member”,
                “Collection”)
    if ($RedundancySets == null || $RedundancySets->[].size ==0) {
        #InARedundancySet = false
    } else {
        #InARedundancySet = true
     // Error is more than one RedundancySet
     if ($RedundancySets->[].size != 1) {
         <ERROR: A system cannot be the member of multiple RedundancySets>
     }
        $Systems->[] = AssociatorNames($RedundancySets->[0],   // ObjectName
            “CIM_LogicalIdentity”,          // AssocClass
            “CIM_System”,                   // ResultClass
            “SameElement”,                  // Role
```

```
            “SystemElement”)                // ResultRole
          if ($Systems == null || $Systemss->[].size != 1) {
            <ERROR: There must be exactly one System associated to each
             RedundancySet>
     }
     // if System->[0] is not the TopLevelSystem, it must have ComponentCS
     if ($System->[0] != $TopLevelSystem) {
            $UpperSystems->[] = AssociatorNames($System->[0],
                “CIM_ComponentCS”,// AssocClass
             “CIM_ComputerSystem”,// ResultClass
             “PartComponent”,// Role
                “GroupComponent”)   // ResultRole
            if ($UpperSystems == null && $UpperSystems->[].size != 1) {
                <ERROR: must be no more than one ComputerSystem Associated
                   via ComponentCS to each LogicalDevice instance>
         }
            if ($UpperSystems->[0] != $TopLevelSystem) {
               <ERROR: The one end of every ComponentCS must be the Top Level
              system>
            }
     }
    }
} while (#InARedundancySet)

// The top-level system must be associated to a RegisteredProfile
$Profiles->[] = AssociatorNames($TopLevelSystem,
        “CIM_ElementConformsToProfile”,
        “CIM_RegisteredProfile”,
        NULL, NULL)
if ($Profiles == null || $Profiles->[].size == 0) {
    <ERROR: Top-Level system not associated to RegisteredProfile>
}
```

30.7 Registered Name and Version

Multiple Computer System version 1.2.0 (Component Profile)

30.8 CIM Elements

Table 298 describes the CIM elements for Multiple Computer System.

Table 298 - CIM Elements for Multiple Computer System

Element Name	Requirement	Description
30.8.1 CIM_ComponentCS	Mandatory	Associates non-top-level systems to the top-level system.
30.8.2 CIM_ComputerSystem (Non-Top-Level System)	Mandatory	Non-Top-level System.
30.8.3 CIM_ConcreteIdentity	Mandatory	Associates aggregate (possibly top-level) ComputerSystem and RedundancySet.

Table 298 - CIM Elements for Multiple Computer System

Element Name	Requirement	Description
30.8.4 CIM_IsSpare	Optional	Associates the ComputerSystem that may be used as a spare to the RedundancySet of ActiveComputerSystem.
30.8.5 CIM_MemberOfCollection	Mandatory	Associates RedundancySet and its member ComputerSystems.
30.8.6 CIM_RedundancySet	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Creation of a ComputerSystem instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Deletion of a ComputerSystem instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Deprecated WQL -Change of Operational Status of a ComputerSystem instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::OperationalStatus <> PreviousInstance.CIM_ComputerSystem::OperationalStatus	Mandatory	CQL -Change of Operational Status of a ComputerSystem instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_RedundancySet AND SourceInstance.RedundancyStatus <> PreviousInstance.RedundancyStatus	Mandatory	Deprecated WQL -Change of redundancy status.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_RedundancySet AND SourceInstance.CIM_RedundancySet::RedundancyStatus <> PreviousInstance.CIM_RedundancySet::RedundancyStatus	Mandatory	CQL -Change of redundancy status.

30.8.1 CIM_ComponentCS

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 299 describes class CIM_ComponentCS.

Table 299 - SMI Referenced Properties/Methods for CIM_ComponentCS

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Top-Level ComputerSystem; must be assocated to a RegisteredProfile.
PartComponent		Mandatory	The contained (Sub)ComputerSystem.

30.8.2 CIM_ComputerSystem (Non-Top-Level System)

Non-Top-level system.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 300 describes class CIM_ComputerSystem (Non-Top-Level System).

Table 300 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Non-Top-Level System)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	
NameFormat		Mandatory	Non-top-level system names are not correlatable, any format is valid.
ElementName		Mandatory	
OperationalStatus		Mandatory	

30.8.3 CIM_ConcreteIdentity

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 301 describes class CIM_ConcreteIdentity.

Table 301 - SMI Referenced Properties/Methods for CIM_ConcreteIdentity

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	
SameElement		Mandatory	

30.8.4 CIM_IsSpare

Associates the ComputerSystem that may be used as a spare to the RedundancySet of ActiveComputerSystem.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 302 describes class CIM_IsSpare.

Table 302 - SMI Referenced Properties/Methods for CIM_IsSpare

Properties	Flags	Requirement	Description & Notes
SpareStatus		Mandatory	
FailoverSupported		Mandatory	
Dependent		Mandatory	The RedundancySet.
Antecedent		Mandatory	The spare system.

30.8.5 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 303 describes class CIM_MemberOfCollection.

Table 303 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

30.8.6 CIM_RedundancySet

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 304 describes class CIM_RedundancySet.

Table 304 - SMI Referenced Properties/Methods for CIM_RedundancySet

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
RedundancyStatus		Mandatory	The redundancy status shall be either 'Unknown' 0, 'Redundant' 2, or 'Redundancy Lost'. The implementation should report 2 or 3 most of the time, although it may report 0 sometimes. It should report 2 when there is at least one spare per the RedundancySet. It should report 3 when there are no more spares (via IsSpare association) per the RedundancySet.
TypeOfSet		Mandatory	

STABLE

STABLE

31 Physical Package Package

31.1 Description

Physical Package Package models information about a storage system's physical package and optionally about internal sub-packages. A System is 'realized' using a SystemPackaging association to a PhysicalPackage (or a subclasses such as Chassis). The physical containment model can then be built up using Container associations and subclasses (such as PackageInChassis).

Physical elements are described as products using the Product class and ProductPhysicalComponent associations, as shown in Figure 46. The Product instances may be built up into a hierarchy using the ProductParentChild association. The Product class holds information such as vendor name, serial number and version.

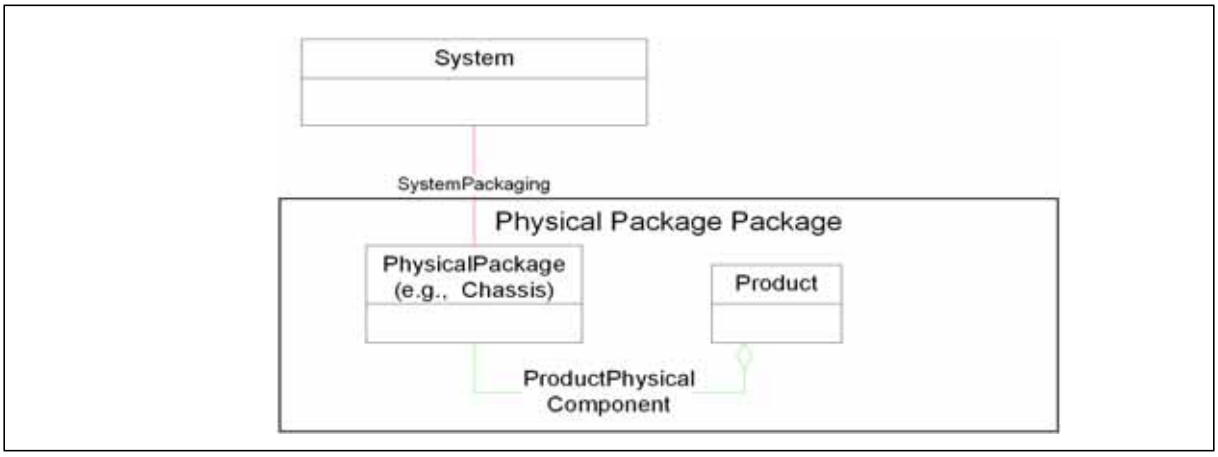

Figure 46 - Physical Package Package Mandatory Classes

31.1.1 Well Defined Subcomponents

In addition to defining physical packages at the "System" level, PhysicalPackage may also be defined at a lower, subcomponent level. For example, PhysicalPackage is used in the Disk Drive Lite Subprofile and for devices supported by storage media libraries (e.g., TapeDrive and ChangerDevice). If the subcomponents are supported by the Profile, they should model their physical packaging. When subcomponents are modeled, there shall be a container relationship between their physical package and the containing package (e.g., the System level physical package). In addition, there shall be a ProductParentChild association between the subcomponent Product and the parent Product.

The Physical Package constructs may also be used to model other aspects of the environment. However, this is not mandatory. Note that each logical device may be realized by a card. The cards are contained in a controller chassis.

When establishing physical packages for logical device subcomponents (e.g., disk drives, changers, etc.) the provider shall populate both Container and Realizes associations. When establishing physical packages for processor subcomponents

(e.g., non-top level systems.) the provider shall populate both Container and SystemPackaging associations. When establishing the Product instances for the subcomponent packages the provider shall populate the ProductParentChild association to the parent product. This is illustrated in Figure 47.

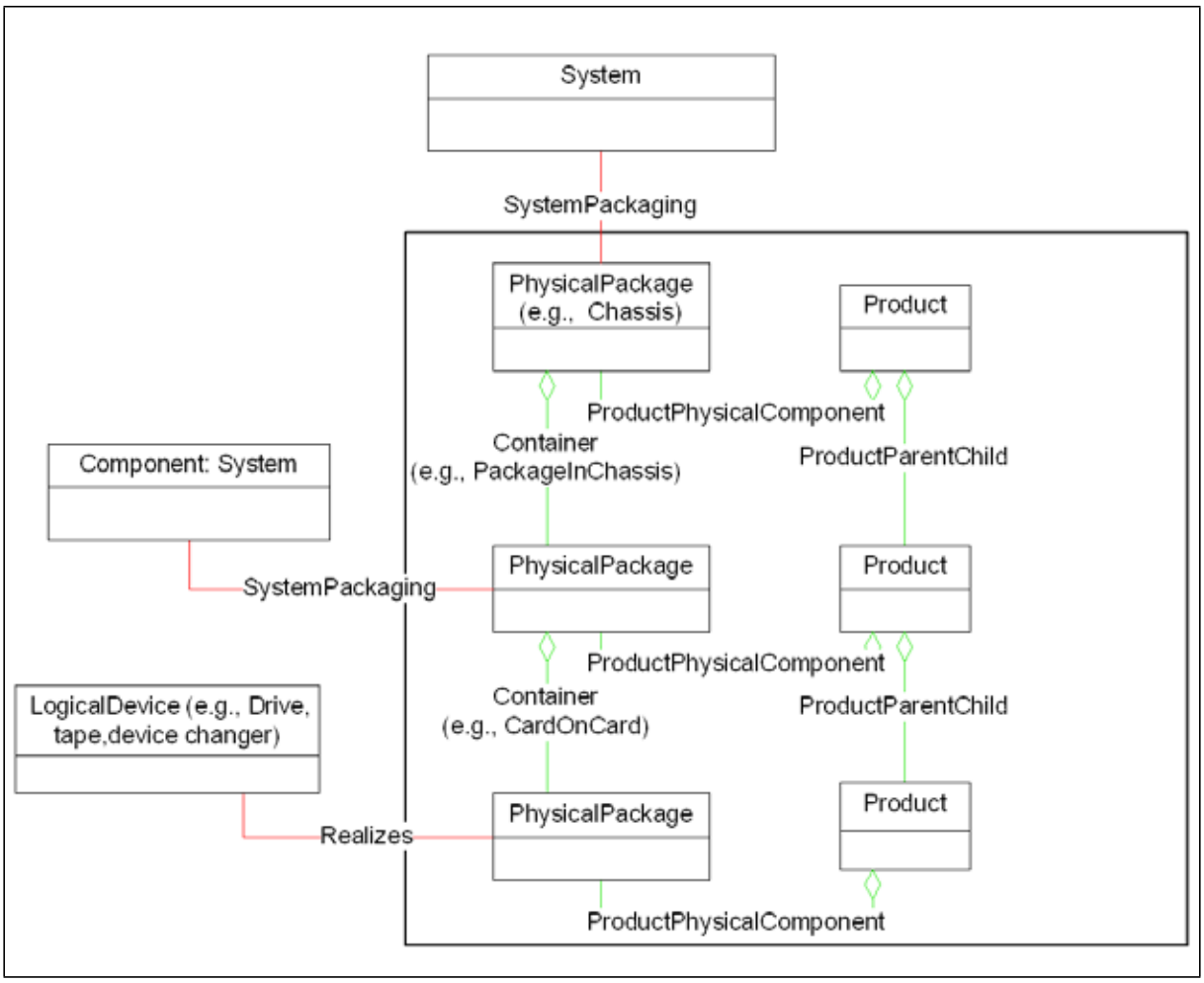

Figure 47 - Modeling for well defined subcomponents

31.1.2 Multiple Product Identities

Instrumentation may optionally describe multiple product identities for a physical package, for example, product information for both an OEM and vendor. This information should be modeled as multiple instances of CIM_Product associated with the LogicalIdentity association. The Product instance that clients should treat as primary is directly associated with PhysicalPackage via ProductPhysicalComponent. Additional product instances are associated with the primary product using the LogicalIdentity association.

Figure 48 shows an example of the use of mandatory and optional physical package classes.

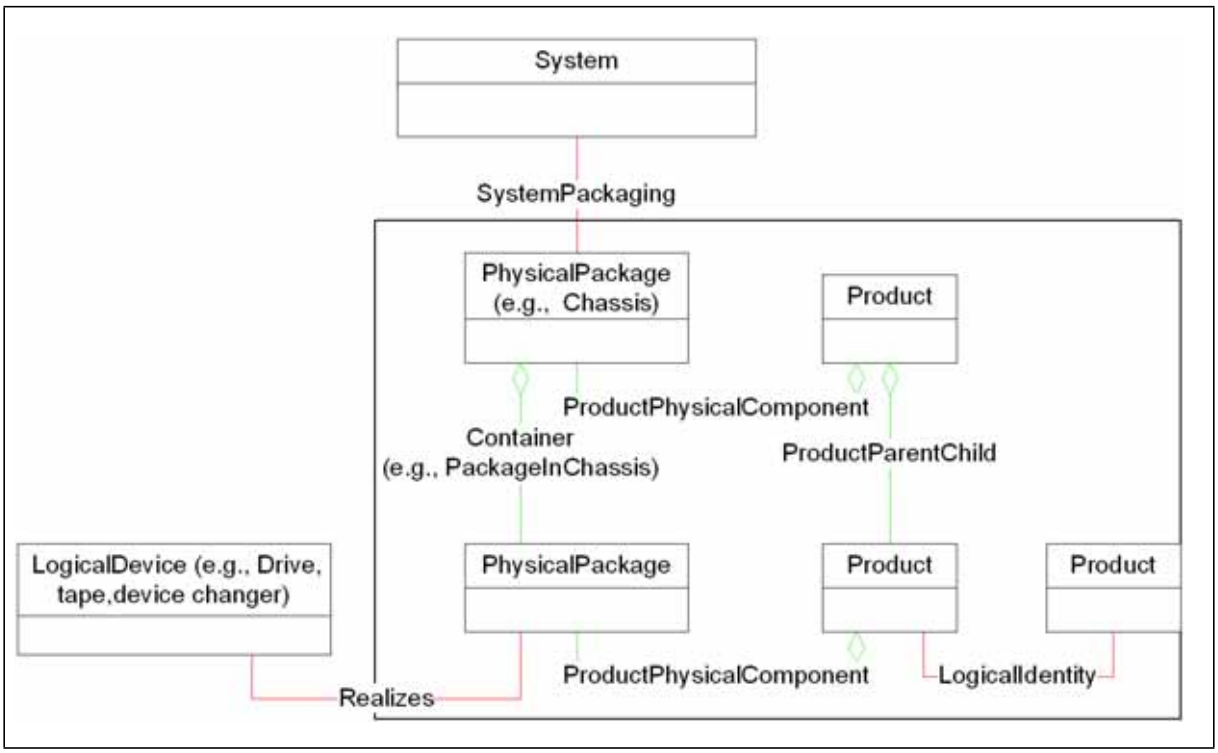

Figure 48 - Physical Package Package with Optional Classes

31.2 Health and Fault Management Considerations

Not defined in this standard.

31.3 Cascading Considerations

Not defined in this standard.

31.4 Supported Subprofiles and Packages

Related Profiles for Physical Package: Not defined in this standard.

31.5 Methods of this Profile

Not defined in this standard.

31.6 Client Considerations and Recipes

31.6.1 Find Asset Information

Information about a system is modeled in PhysicalPackage. PhysicalPackage may be subclassed to Chassis; the more general PhysicalPackage is used here to accommodate device implementations that are deployed in multiple chassis. PhysicalPackage has an associated Product with physical asset information such as Vendor and Version.

31.6.2 Finding Product information

To locate product information (Vendor, Serial number and product versions) information about a device that is conforms to the profile, you would start with the “top-level” computer system and traverse the SystemPackaging to the PhysicalPackage (e.g., a Chassis). From the PhysicalPackage, the client would then traverse the ProductPhysicalComponent association to locate the Product instance. The primary Vendor, Serial Number and version for the device is in the Product instance associated with the PhysicalPackage. Additional product identities may be associated with the primary Product using the LogicalIdentity association.

31.6.3 Finding Asset information

There are certain subcomponents of a device that a client may be interested in locating. For example, disk drives in an array or changer devices in a library. To locate the asset information of these subcomponents, the client would follow the ProductParentChild association from the system Product to lower level Products.

Alternatively, if the client is starting from a LogicalDevice, it can locate the PhysicalPackage by following the Realizes association from the LogicalDevice. From the PhysicalPackage, the client can find the Product information by traversing the ProductPhysicalComponent association.

31.7 Registered Name and Version

Physical Package version 1.5.0 (Component Profile)

CIM Schema Version: 2.23

31.8 CIM Elements

Table 305 describes the CIM elements for Physical Package.

Table 305 - CIM Elements for Physical Package

Element Name	Requirement	Description
31.8.1 CIM_Container	Optional	Associates a PhysicalPackage to its component physical packages (e.g., Drives in a Storage System).
31.8.2 CIM_LogicalIdentity	Optional	Associates the primary product information to secondary product information.
31.8.3 CIM_PhysicalElementLocation	Conditional	Conditional requirement: Support for the Location profile. Associates the physical package of the system to its location.
31.8.4 CIM_PhysicalPackage (Component)	Optional	A physical package for a component of the overall system.
31.8.5 CIM_PhysicalPackage (System)	Mandatory	The physical package for the overall system.
31.8.6 CIM_Product (Component)	Optional	The product information for a physical package that is a component of the system.
31.8.7 CIM_Product (System)	Mandatory	The product information for the physical package of the system.
31.8.8 CIM_ProductParentChild	Optional	If more than one product comprises a system, this association should be used to indicate the 'parent' product.
31.8.9 CIM_ProductPhysicalComponent (Component)	Optional	Associates a component physical package to its product information.

Table 305 - CIM Elements for Physical Package

Element Name	Requirement	Description
31.8.10 CIM_ProductPhysicalComponent (System)	Mandatory	Associates the system physical package to its product information.
31.8.11 CIM_SystemPackaging (Component)	Optional	Associates a component system and its physical components.
31.8.12 CIM_SystemPackaging (System)	Mandatory	Associates the top level system and its physical package.

31.8.1 CIM_Container

Associates a PhysicalPackage to its component physical packages (e.g., Drives in a Storage System). This may be subclassed (e.g., PackageInChassis or CardOnCard), but only the Container properties are required.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 306 describes class CIM_Container.

Table 306 - SMI Referenced Properties/Methods for CIM_Container

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	A reference to the higher level physical package.
PartComponent		Mandatory	A reference to a lower level physical package.

31.8.2 CIM_LogicalIdentity

Associates the primary product information to secondary product information. The secondary product information might be the OEM product information.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 307 describes class CIM_LogicalIdentity.

Table 307 - SMI Referenced Properties/Methods for CIM_LogicalIdentity

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	A reference to the primary product information.
SameElement		Mandatory	A reference to a secondary product information.

31.8.3 CIM_PhysicalElementLocation

Associates the physical package of the system to its location.

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: Support for the Location profile.

Table 308 describes class CIM_PhysicalElementLocation.

Table 308 - SMI Referenced Properties/Methods for CIM_PhysicalElementLocation

Properties	Flags	Requirement	Description & Notes
PhysicalLocation		Mandatory	The reference to the location of the system physical package.
Element		Mandatory	The reference to the system physical package.

31.8.4 CIM_PhysicalPackage (Component)

A physical package for a component of the overall system. There may be multiple instances of a component physical package. For certain component physical packages, this "generic" physical package may be the same as physical packages defined in other component profiles (e.g., this physical package is the same instance as the physical package defined in Disk Drive Lite. It is preferred that component physical packages are modeled in their respective component profiles. However, the component physical package defined in this profile is intended for components (e.g., non-Top Level Systems in the Multiple Computer System Profile) that do not model their physical package.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 309 describes class CIM_PhysicalPackage (Component).

Table 309 - SMI Referenced Properties/Methods for CIM_PhysicalPackage (Component)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Tag		Mandatory	
ElementName		Optional	
Name		Optional	
Manufacturer		Mandatory	
Model		Mandatory	
SerialNumber		Optional	
Version		Optional	
PartNumber		Optional	

31.8.5 CIM_PhysicalPackage (System)

The physical package for the overall system. There shall be only one instance of this class for an autonomous profile (e.g., Array or Fabric).

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 310 describes class CIM_PhysicalPackage (System).

Table 310 - SMI Referenced Properties/Methods for CIM_PhysicalPackage (System)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Tag		Mandatory	
ElementName		Optional	
Name		Optional	
Manufacturer		Mandatory	
Model		Mandatory	
SerialNumber		Optional	
Version		Optional	
PartNumber		Optional	

31.8.6 CIM_Product (Component)

The product information for a physical package that is a component of the system.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 311 describes class CIM_Product (Component).

Table 311 - SMI Referenced Properties/Methods for CIM_Product (Component)

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
IdentifyingNumber		Mandatory	
Vendor		Mandatory	
Version		Mandatory	
ElementName		Mandatory	

31.8.7 CIM_Product (System)

The product information for the physical package of the system.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 312 describes class CIM_Product (System).

Table 312 - SMI Referenced Properties/Methods for CIM_Product (System)

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
IdentifyingNumber		Mandatory	
Vendor		Mandatory	
Version		Mandatory	
ElementName		Mandatory	

31.8.8 CIM_ProductParentChild

If more than one product comprises a system, this association should be used to indicate the 'parent' product.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 313 describes class CIM_ProductParentChild.

Table 313 - SMI Referenced Properties/Methods for CIM_ProductParentChild

Properties	Flags	Requirement	Description & Notes
Parent		Mandatory	A reference to the parent (System or Component) product.
Child		Mandatory	A reference to a component product.

31.8.9 CIM_ProductPhysicalComponent (Component)

Associates a component physical package to its product information.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 314 describes class CIM_ProductPhysicalComponent (Component).

Table 314 - SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent (Component)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	A reference to the product information for a component of the system.
PartComponent		Mandatory	A reference to the component physical package.

31.8.10CIM_ProductPhysicalComponent (System)

Associates the system physical package to its product information.

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 315 describes class CIM_ProductPhysicalComponent (System).

Table 315 - SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent (System)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The reference to the product information for the system.
PartComponent		Mandatory	The reference to the system physical package.

31.8.11CIM_SystemPackaging (Component)

Associates a component system and its physical components. The ComputerSystemPackage subclass should be used if the referenced system is subclassed as ComputerSystem.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 316 describes class CIM_SystemPackaging (Component).

Table 316 - SMI Referenced Properties/Methods for CIM_SystemPackaging (Component)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The component system that has a physical package.
Antecedent		Mandatory	The reference to the PhysicalPackage of the component system.

31.8.12CIM_SystemPackaging (System)

Associates the top level system and its physical package. The ComputerSystemPackage subclass should be used if the referenced system is subclassed as ComputerSystem.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 317 describes class CIM_SystemPackaging (System).

Table 317 - SMI Referenced Properties/Methods for CIM_SystemPackaging (System)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The system that has a physical package.
Antecedent		Mandatory	The reference to the PhysicalPackage of the top level system.

STABLE

EXPERIMENTAL

32 Power Supply Profile

32.1 Synopsis

Profile Name: Power Supply (Component Profile)

Version: 1.0.1

Organization: SNIA

CIM Schema Version: 2.17.0

Table 318 describes the related profiles for Power Supply.

Table 318 - Related Profiles for Power Supply

Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0.0a	Optional	

Specializes: DMTF Power Supply Profile

The SNIA Power Supply Profile specializes DSP1015: the DMTF Power Supply Profile by adding indications.

32.2 Description

The SNIA Power Supply Profile specializes the DMTF Power Supply Profile by adding indications. No other changes are made to the DMTF profile.

32.3 Implementation

See DSP1015: the DMTF Power Supply Profile.

32.3.1 Health and Fault Management Consideration

None

32.3.2 Cascading Considerations

None

32.4 Methods

See DSP1015: the DMTF Power Supply Profile.

32.5 Use Cases

See DSP1015: the DMTF Power Supply Profile.

32.6 CIM Elements

Table 319 describes the CIM elements for Power Supply.

Table 319 - CIM Elements for Power Supply

Element Name	Requirement	Description
32.6.1 CIM_ElementCapabilities	Conditional	Conditional requirement: Support for CIM_EnabledLogicalElementCapabilities.
32.6.2 CIM_EnabledLogicalElementCapabilities	Optional	
32.6.3 CIM_IsSpare	Optional	
32.6.4 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for Power Supply redundancy.
32.6.5 CIM_OwningCollectionElement	Conditional	Conditional requirement: Support for Power Supply redundancy.
32.6.6 CIM_PowerSupply	Mandatory	
32.6.7 CIM_RedundancySet	Optional	
32.6.8 CIM_SuppliesPower	Optional	
32.6.9 CIM_SystemDevice	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_PowerSupply	Mandatory	Creation of a PowerSupply instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_PowerSupply	Mandatory	Deletion of a PowerSupply instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_PowerSupply AND SourceInstance.CIM_PowerSupply::OperationalStatus<> PreviousInstance.CIM_PowerSupply::OperationalStatus	Mandatory	CQL -Change of Operational Status of a PowerSupply instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_PowerSupply AND SourceInstance.CIM_PowerSupply::EnabledState <> PreviousInstance.CIM_PowerSupply::EnabledState	Mandatory	CQL -Change of EnabledState of a PowerSupply instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_RedundancySet AND SourceInstance.CIM_RedundancySet::RedundancyStatus <> PreviousInstance.CIM_RedundancySet::RedundancyStatus	Conditional	Conditional requirement: Support for Power Supply redundancy. CQL -Change of redundancy status.

32.6.1 CIM_ElementCapabilities

CIM_ElementCapabilities is used to associate CIM_PowerSupply with CIM_EnabledLogicalElementCapabilities that describes the capabilities of CIM_PowerSupply.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for CIM_EnabledLogicalElementCapabilities.

Table 320 describes class CIM_ElementCapabilities.

Table 320 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Reference to CIM_PowerSupply.
Capabilities		Mandatory	Reference to CIM_EnabledLogicalElementCapabilities.

32.6.2 CIM_EnabledLogicalElementCapabilities

CIM_EnabledLogicalElementCapabilities represents the capabilities of the power supply.

Requirement: Optional

Table 321 describes class CIM_EnabledLogicalElementCapabilities.

Table 321 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
RequestedStatesSupported		Mandatory	Array that contains the supported requested states for the instance of CIM_PowerSupply. Shall include 2 (Enabled), 3 (Disabled), 6 (Offline), or 11 (Reset).
ElementNameEditSupported		Mandatory	
MaxElementNameLen		Conditional	Conditional requirement: Support for Element Name editing.Conditional on Support for Element Name editing.
ElementName		Mandatory	User-friendly name.

32.6.3 CIM_IsSpare

CIM_IsSpare is used to associate CIM_PowerSupply with CIM_RedundancySet that the CIM_PowerSupply is a member of and where CIM_PowerSupply represents a spare power supply.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 322 describes class CIM_IsSpare.

Table 322 - SMI Referenced Properties/Methods for CIM_IsSpare

Properties	Flags	Requirement	Description & Notes
SpareStatus		Mandatory	Shall be 0 (Unknown), 1 (Cold Standby), or 2 (Hot Standby).
FailoverSupported		Mandatory	Shall be are 2 (Automatic), 3 (Manual) or 4 (Both Manual and Automatic).
Antecedent		Mandatory	The RedundancySet.
Dependent		Mandatory	PowerSupply.

32.6.4 CIM_MemberOfCollection

CIM_MemberOfCollection is used to associate CIM_PowerSupply with CIM_RedundancySet that the CIM_PowerSupply is a member of.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for Power Supply redundancy.

Table 323 describes class CIM_MemberOfCollection.

Table 323 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

32.6.5 CIM_OwningCollectionElement

CIM_OwningCollectionElement is used to associate CIM_RedundancySet with CIM_ComputerSystem that the CIM_RedundancySet is a member of. The instance of CIM_OwningCollectionElement is conditional on having instantiation of the CIM_RedundancySet class.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for Power Supply redundancy.

Table 324 describes class CIM_OwningCollectionElement.

Table 324 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

32.6.6 CIM_PowerSupply

CIM_PowerSupply is used to represent the power supply.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 325 describes class CIM_PowerSupply.

Table 325 - SMI Referenced Properties/Methods for CIM_PowerSupply

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	Key.
SystemName		Mandatory	Key.

Table 325 - SMI Referenced Properties/Methods for CIM_PowerSupply

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key.
DeviceID		Mandatory	Key.
TotalOutputPower		Mandatory	Shall match 0 when the power supply's total output power is unknown.
ElementName		Mandatory	
OperationalStatus		Mandatory	
HealthState		Mandatory	
EnabledState		Mandatory	Shall be 2 (Enabled), 3 (Disabled), 5 (Not Applicable) or 6 (Enabled but Offline).
RequestedState		Mandatory	SHall be 2 (Enabled), 3 (Disabled), 5 (No Change), 6 (Offline), 11 (Reset) or 12 (Not Applicable).
RequestStateChange()		Conditional	Conditional requirement: updating requested states.The implementation shall support this method, but the method may always return 'Not Supported.'.

32.6.7 CIM_RedundancySet

CIM_RedundancySet is used to represent the aggregation of redundant power supplies.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 326 describes class CIM_RedundancySet.

Table 326 - SMI Referenced Properties/Methods for CIM_RedundancySet

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	shall be formatted as a free formed string of variable length (pattern ".+").
RedundancyStatus		Mandatory	
TypeOfSet		Mandatory	Shall be 2 (N+1), 3 (Load Balanced), 4 (Sparing) or 5 (Limited Sparing).
MinNumberNeeded		Mandatory	shall match 0 when the minimum number of power supplies needed for the redundancy is unknown.
Failover()		Optional	

32.6.8 CIM_SuppliesPower

CIM_SuppliesPower is used to associate CIM_PowerSupply with CIM_ManagedSystemElement that the power supply represented by the CIM_PowerSupply instance supplies power to.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 327 describes class CIM_SuppliesPower.

Table 327 - SMI Referenced Properties/Methods for CIM_SuppliesPower

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	Shall reference the instance of the subclass of CIM_ManagedSystemElement representing element receiving the power.

32.6.9 CIM_SystemDevice

CIM_SystemDevice is used to associate CIM_PowerSupply with CIM_ComputerSystem that the CIM_PowerSupply is a member of.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 328 describes class CIM_SystemDevice.

Table 328 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

33 Fan Profile

33.1 Synopsis

Profile Name: Fan (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.19.1

Table 329 describes the related profiles for Fan.

Table 329 - Related Profiles for Fan

Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0.0a	Optional	
Sensors	SNIA	1.5.0	Optional	

Specializes: DMTF Fan Profile

Central Class: CIM_Fan

Scoping Class: CIM_ComputerSystem

The SNIA Fan Profile specializes DSP1013: the DMTF Fan Profile by adding indications.

33.2 Description

The SNIA Fan Profile specializes the DMTF Fan Profile by adding indications. Other changes are made to the DMTF Profile, include making a couple of classes conditional and elaborating on the definition of RedundancyStatus.

33.3 Implementation

See DSP1013: the DMTF Fan Profile.

33.3.1 Health and Fault Management Consideration

None

33.3.2 Cascading Considerations

None

33.4 Methods

See DSP1013: the DMTF Fan Profile.

33.5 Use Cases

See DSP1013: the DMTF Fan Profile.

33.6 CIM Elements

Table 330 describes the CIM elements for Fan.

Table 330 - CIM Elements for Fan

Element Name	Requirement	Description
33.6.1 CIM_AssociatedCooling	Optional	
33.6.2 CIM_AssociatedSensor	Optional	
33.6.3 CIM_ElementCapabilities	Conditional	Conditional requirement: Support for CIM_EnabledLogicalElementCapabilities.
33.6.4 CIM_EnabledLogicalElementCapabilities	Optional	
33.6.5 CIM_Fan	Mandatory	
33.6.6 CIM_HostedCollection	Optional	
33.6.7 CIM_IsSpare	Optional	
33.6.8 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for Fan redundancy.
33.6.9 CIM_NumericSensor	Optional	
33.6.10 CIM_OwningCollectionElement	Conditional	Conditional requirement: Support for Fan redundancy.
33.6.11 CIM_RedundancySet (Fan Redundancy)	Optional	
33.6.12 CIM_Sensor	Optional	
33.6.13 CIM_SystemDevice	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_Fan	Mandatory	Creation of a Fan instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_Fan	Mandatory	CQL -Deletion of a Fan instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_Fan AND SourceInstance.CIM_Fan::OperationalStatus <> PreviousInstance.CIM_Fan::OperationalStatus	Mandatory	CQL -Change of Operational Status of a Fan instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_Fan AND SourceInstance.CIM_Fan::EnabledState <> PreviousInstance.CIM_Fan::EnabledState	Mandatory	CQL -Change of EnabledState of a Fan instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_RedundancySet AND SourceInstance.CIM_RedundancySet::RedundancyStatus <> PreviousInstance.CIM_RedundancySet::RedundancyStatus	Conditional	Conditional requirement: Support for Fan redundancy. CQL -Change of redundancy status.

33.6.1 CIM_AssociatedCooling

CIM_AssociatedCooling associates CIM_Fan with a subclass of CIM_ManagedSystemElement.

Requirement: Optional

Table 331 describes class CIM_AssociatedCooling.

Table 331 - SMI Referenced Properties/Methods for CIM_AssociatedCooling

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to CIM_Fan.
Dependent		Mandatory	Shall reference an instance of a subclass of CIM_ManagedSystemElement for which the fan is providing cooling.

33.6.2 CIM_AssociatedSensor

This is described in the DMTF profile, but is missing from the CIM Elements table.

Requirement: Optional

Table 332 describes class CIM_AssociatedSensor.

Table 332 - SMI Referenced Properties/Methods for CIM_AssociatedSensor

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to CIM_Fan.
Antecedent		Mandatory	Reference to Sensor or NumericSensor.

33.6.3 CIM_ElementCapabilities

SNIA makes this conditional on the existence of CIM_EnabledLogicalElementCapabilities. The class definition specializes the CIM_ElementCapabilities definition in the Fan profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for CIM_EnabledLogicalElementCapabilities.

Table 333 describes class CIM_ElementCapabilities.

Table 333 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Reference to CIM_Fan.
Capabilities		Mandatory	Reference to CIM_EnabledLogicalElementCapabilities.

33.6.4 CIM_EnabledLogicalElementCapabilities

CIM_EnabledLogicalElementCapabilities represents the capabilities of the Fan.

Requirement: Optional

Table 334 describes class CIM_EnabledLogicalElementCapabilities.

Table 334 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
RequestedStatesSupported	N	Mandatory	Array that contains the supported requested states for the instance of CIM_Fan. Shall be an empty array or contain any combination of: 2 (Enabled), 3 (Disabled), or 11 (Reset).
ElementNameEditSupported		Mandatory	property shall have a value of TRUE when the implementation supports client modification of the ElementName property of the associated instance of CIM_Fan.
MaxElementNameLen		Conditional	Conditional requirement: The ElementNameEditSupported property has a value of TRUE.Conditional on Support for Element Name editing.

33.6.5 CIM_Fan

CIM_Fan is used to represent the fan.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 335 describes class CIM_Fan.

Table 335 - SMI Referenced Properties/Methods for CIM_Fan

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
DeviceID		Mandatory	Key.
OperationalStatus		Mandatory	
HealthState		Mandatory	
VariableSpeed		Mandatory	
DesiredSpeed		Conditional	Conditional requirement: Support for the SetSpeed method (VariableSpeed=\TRUE\')andEnabledStateisNOT3(Disabled).'
ActiveCooling		Mandatory	Shall have the value TRUE.
EnabledState		Mandatory	This property shall match the values 0 (Unknown), 2(Enabled), 3(Disabled) or 5(Not Applicable).
RequestedState		Mandatory	This property shall have a value of 2 (Enabled), 3 (Disabled) , 5 (No Change), 11 (Reset) or 12 (Not Applicable).
ElementName		Mandatory	This property shall be modifiable when the ElementNameEditSupported property of the associated CIM_EnabledLogicalElementCapabilities instance has a value of TRUE.

Table 335 - SMI Referenced Properties/Methods for CIM_Fan

Properties	Flags	Requirement	Description & Notes
SetSpeed()		Conditional	Conditional requirement: Support for the SetSpeed method (VariableSpeed=\TRUE\')andEnabledStateisNOT3(Disabled).'
RequestStateChange()		Conditional	Conditional requirement: The RequestedStatesSupported property has a value of 2 (Enabled), 3 (Disabled) or 11 (Reset)in the CIM_EnabledLogicalElementCapabilities for the Fan..

33.6.6 CIM_HostedCollection

HostedCollection defines a SystemSpecificCollection in the context of a scoping System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 336 describes class CIM_HostedCollection.

Table 336 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Key: Shall reference a CIM_ComputerSystem instance of which a CIM_RedundancySet instance is a member.
Dependent		Mandatory	Key: Shall reference a CIM_RedundancySet instance.

33.6.7 CIM_IsSpare

CIM_IsSpare is used to associate CIM_Fan with CIM_RedundancySet that the CIM_Fan is a member of and where CIM_Fan represents a spare Fan.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 337 describes class CIM_IsSpare.

Table 337 - SMI Referenced Properties/Methods for CIM_IsSpare

Properties	Flags	Requirement	Description & Notes
SpareStatus		Mandatory	Shall be 0 (Unknown) or 3 (Cold Standby).
FailoverSupported		Mandatory	Shall be 2 (Automatic), 3 (Manual) or 4 (Both Manual and Automatic).
Antecedent		Mandatory	Key: Shall reference a CIM_RedundancySet instance of which a CIM_Fan instance is a member and where the CIM_Fan instance is a Spare Fan.
Dependent		Mandatory	Key: Shall reference the Spare Fan.

33.6.8 CIM_MemberOfCollection

CIM_MemberOfCollection associates CIM_Fan with the CIM_RedundancySet instance of which CIM_Fan is a member.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for Fan redundancy.

Table 338 describes class CIM_MemberOfCollection.

Table 338 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Key: Shall reference a CIM_RedundancySet instance of which a CIM_Fan instance is a member.
Member		Mandatory	Key: Shall reference a CIM_Fan instance.

33.6.9 CIM_NumericSensor

The CIM_NumericSensor class is defined by the Sensors Profile. The requirements denoted here are in addition to those mandated by the Sensors Profile.

Requirement: Optional

Table 339 describes class CIM_NumericSensor.

Table 339 - SMI Referenced Properties/Methods for CIM_NumericSensor

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
DeviceID		Mandatory	Key.
SensorType		Mandatory	Shall be set to 5 (Tachometer).
BaseUnits		Mandatory	Shalll be 19 (RPM).
RateUnits		Mandatory	Shall be 0 (None).

33.6.10CIM_OwningCollectionElement

SNIA makes this conditional on the existence of CIM_RedundancySet. The class definition specializes the CIM_OwningCollectionElement definition in the Fan profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for Fan redundancy.

Table 340 describes class CIM_OwningCollectionElement.

Table 340 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	Key: Shall reference a CIM_RedundancySet instance.
OwningElement		Mandatory	Key: Shall reference a CIM_ComputerSystem instance of which a CIM_RedundancySet instance is a member.

33.6.11CIM_RedundancySet (Fan Redundancy)

SNIA specializes this to further define the RedundancyStatus property. The class definition specializes the CIM_RedundancySet definition in the Fan profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 341 describes class CIM_RedundancySet (Fan Redundancy).

Table 341 - SMI Referenced Properties/Methods for CIM_RedundancySet (Fan Redundancy)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
ElementName		Mandatory	shall be formatted as a free formed string of variable length (pattern ".*").
RedundancyStatus (overridden)		Mandatory	Shall be 2 (Fully Redundant), 3 (Degraded Redundancy), 4 (Redundancy Lost) or 5 (Overall Failure).
TypeOfSet		Mandatory	Shall be 2 (N+1), 3 (Load Balanced), 4 (Sparing) or 5 (Limited Sparing).
MinNumberNeeded		Mandatory	Shall be 0 when the minimum number of fans needed for the redundancy is unknown.
Failover()		Optional	

33.6.12CIM_Sensor

The CIM_Sensor class is defined by the Sensors Profile. The requirements denoted here are in addition to those mandated by the Sensors Profile.

Requirement: Optional

Table 342 describes class CIM_Sensor.

Table 342 - SMI Referenced Properties/Methods for CIM_Sensor

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.

Table 342 - SMI Referenced Properties/Methods for CIM_Sensor

Properties	Flags	Requirement	Description & Notes
DeviceID		Mandatory	Key.
SensorType		Mandatory	Shall be set to 5 (Tachometer).

33.6.13CIM_SystemDevice

CIM_SystemDevice associates CIM_Fan with the CIM_ComputerSystem instance of which CIM_Fan is a member.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 343 describes class CIM_SystemDevice.

Table 343 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Key: Shall reference a CIM_ComputerSystem instance of which a CIM_Fan instance is a member.
PartComponent		Mandatory	Key: Shall reference a CIM_Fan instance.

EXPERIMENTAL

EXPERIMENTAL

34 Sensors Profile

34.1 Synopsis

Profile Name: Sensors (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.19.1

Related Profiles for Sensors: Not defined in this standard.

Specializes: DMTF Sensors Profile

Central Class: CIM_Sensor

Scoping Class: CIM_ComputerSystem

The SNIA Sensors Profile specializes DSP1009: the DMTF Sensors Profile by adding indications.

34.2 Description

The SNIA Sensors Profile specializes the DMTF Sensors Profile by adding indications. No other changes are made to the DMTF Profile.

34.3 Implementation

See DSP1009: the DMTF Sensors Profile.

34.3.1 Health and Fault Management Consideration

None

34.3.2 Cascading Considerations

None

34.4 Methods

See DSP1009: the DMTF Sensors Profile.

34.5 Use Cases

See DSP1009: the DMTF Sensors Profile.

34.6 CIM Elements

Table 344 describes the CIM elements for Sensors.

Table 344 - CIM Elements for Sensors

Element Name	Requirement	Description
34.6.1 CIM_AssociatedSensor	Optional	This is used to associate the instance of CIM_Sensor (or CIM_NumericSensor) with the instance of a subclass of CIM_ManagedElement.
34.6.2 CIM_ElementCapabilities	Optional	This is used to associate CIM_Sensor (or CIM_NumericSensor) with an instance of CIM_EnabledLogicalElementCapabilities that describes the capabilities of CIM_Sensor.
34.6.3 CIM_EnabledLogicalElementCapabilities	Optional	This is used to represent the capabilities of the sensor as it applies to the properties of CIM_Sensor.
34.6.4 CIM_NumericSensor	Optional	
34.6.5 CIM_Sensor	Mandatory	The implementation shall instantiate an instance of CIM_Sensor, but this could be the subclass CIM_NumericSensor.
34.6.6 CIM_SystemDevice	Mandatory	CIM_SystemDevice is used to associate the instance of CIM_Sensor (or CIM_NumericSensor) with the instance of CIM_ComputerSystem of which the CIM_Sensor instance is a member.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_Sensor	Mandatory	Creation of a Sensor instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_Sensor	Mandatory	Deletion of a Sensor instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_Sensor AND SourceInstance.CIM_Sensor::OperationalStatus <> PreviousInstance.CIM_Sensor::OperationalStatus	Mandatory	CQL -Change of Operational Status of a Sensor instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_Sensor AND SourceInstance.CIM_Sensor::EnabledState <> PreviousInstance.CIM_Sensor::EnabledState	Mandatory	CQL -Change of EnabledState of a Sensor instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_Sensor AND SourceInstance.CIM_Sensor::CurrentState <> PreviousInstance.CIM_Sensor::CurrentState	Mandatory	CQL -Change of Current State of a Sensor instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_NumericSensor AND SourceInstance.CIM_Sensor::CurrentReading <> PreviousInstance.CIM_Sensor::CurrentReading	Mandatory	CQL -Change of Current Reading of a Sensor instance.

34.6.1 CIM_AssociatedSensor

CIM_AssociatedSensor associates CIM_Sensor or CIM_NumericSensor with a subclass of CIM_ManagedSystemElement.

Requirement: Optional

Table 345 describes class CIM_AssociatedSensor.

Table 345 - SMI Referenced Properties/Methods for CIM_AssociatedSensor

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Shall be a reference to a specific instance of CIM_Sensor or CIM_NumericSensor.
Dependent		Mandatory	Shall reference an instance of a subclass of CIM_ManagedElement for which the sensor is monitoring.

34.6.2 CIM_ElementCapabilities

CIM_ElementCapabilities is used to associate CIM_Sensor with the CIM_EnabledLogicalElementCapabilities instance that describes the capabilities of the Sensor.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 346 describes class CIM_ElementCapabilities.

Table 346 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

34.6.3 CIM_EnabledLogicalElementCapabilities

CIM_EnabledLogicalElementCapabilities represents the capabilities of the Sensor.

Requirement: Optional

Table 347 describes class CIM_EnabledLogicalElementCapabilities.

Table 347 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
RequestedStatesSupported		Mandatory	This property is an array that contains the supported requested states for the instance of CIM_Sensor or CIM_NumericSensor. This property shall be the super set of the values to be used as the RequestedState parameter in the RequestStateChange( ) method. The valid values of the property shall be an empty array or contain any combination of the following values: 2 (Enabled), 3 (Disabled), or 11 (Reset).
ElementNameEditSupported		Mandatory	This property shall have a value of TRUE when the implementation supports client modification of the ElementName property of the associated CIM_Sensor or CIM_NumericSensor instance.
MaxElementNameLen		Conditional	Conditional requirement: This property shall be implemented when the CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported property has a value of TRUE.

34.6.4 CIM_NumericSensor

CIM_NumericSensor is used to represent an analog sensor. The CIM_NumericSensor class is mandatory when the CIM_Sensor class is not implemented. The class definition specializes the CIM_NumericSensor definition in the Sensors profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 348 describes class CIM_NumericSensor.

Table 348 - SMI Referenced Properties/Methods for CIM_NumericSensor

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
DeviceID		Mandatory	Key.
BaseUnits		Mandatory	None.
UnitModifier		Mandatory	None.
RateUnits		Mandatory	None.
CurrentReading		Mandatory	None.
LowerThresholdNonCritical		Conditional	Conditional requirement: This property shall be mandatory when the CIM_NumericSensor.SupportedThresholds array contains a value of 0 (LowerThresholdNonCritical).This property shall be settable only if the CIM_NumericSensor.SettableThresholds array contains a value of 0 (LowerThresholdNonCritical).
UpperThresholdNonCritical		Conditional	Conditional requirement: This property shall be mandatory when the CIM_NumericSensor.SupportedThresholds array contains a value of 1 (UpperThresholdNonCritical).This property shall be settable only if the CIM_NumericSensor.SettableThresholds array contains a value of 1 (UpperThresholdNonCritical).
LowerThresholdCritical		Conditional	Conditional requirement: This property shall be mandatory when the CIM_NumericSensor.SupportedThresholds array contains a value of 2 (LowerThresholdCritical).This property shall be settable only if the CIM_NumericSensor.SettableThresholds array contains a value of 2 (LowerThresholdCritical).
UpperThresholdCritical		Conditional	Conditional requirement: This property shall be mandatory when the CIM_NumericSensor.SupportedThresholds array contains a value of 3 (UpperThresholdCritical).This property shall be settable only if the CIM_NumericSensor.SettableThresholds array contains a value of 3 (UpperThresholdCritical).
LowerThresholdFatal		Conditional	Conditional requirement: This property shall be mandatory when the CIM_NumericSensor.SupportedThresholds array contains a value of 4 (LowerThresholdFatal).This property shall be settable only if the CIM_NumericSensor.SettableThresholds array contains a value of 4 (LowerThresholdFatal).

Table 348 - SMI Referenced Properties/Methods for CIM_NumericSensor

Properties	Flags	Requirement	Description & Notes
UpperThresholdFatal		Conditional	Conditional requirement: This property shall be mandatory when the CIM_NumericSensor.SupportedThresholds array contains a value of 5 (UpperThresholdFatal).This property shall be settable only if the CIM_NumericSensor.SettableThresholds array contains a value of 5 (UpperThresholdFatal).
SupportedThresholds	N	Mandatory	This property is an array that contains the list of the implemented thresholds: 0 (LowerThresholdNonCritical), 1 (UpperThresholdNonCritical), 2 (LowerThresholdCritical), 3 (UpperThresholdCritical), 4 (LowerThresholdFatal), and 5 (UpperThresholdFatal). When the implementation does not support any of these threshold properties, the property shall be an empty array.
SettableThresholds	N	Mandatory	This property is an array that contains the list of the settable implemented thresholds: 0 (LowerThresholdNonCritical), 1 (UpperThresholdNonCritical), 2 (LowerThresholdCritical), 3 (UpperThresholdCritical), 4 (LowerThresholdFatal), and 5 (UpperThresholdFatal). The array shall contain the subset of values in the CIM_NumericSensor.SupportedThresholds array. When the implementation does not support any of the settable threshold properties, the property shall be an empty array.
SensorType		Mandatory	None.
PossibleStates		Mandatory	See section 7.3 of the DMTF Sensors Profile version 1.0.2.
CurrentState		Mandatory	The CIM_NumericSensor.CurrentState property shall have a value of one of the elements in the CIM_NumericSensor.PossibleStates array.
ElementName		Mandatory	The CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported property shall have a value of TRUE when the implementation supports client modification of the ElementName property of the associated CIM_NumericSensor instance.
OtherSensorTypeDescription		Conditional	Conditional requirement: The OtherSensorTypeDescription property shall be mandatory when the SensorType property is set to a value of 1 (Other).The OtherSensorTypeDescription property shall be formatted as a free-formed string of variable length (pattern \.*\').'
EnabledState		Mandatory	This property may have the values 2 (Enabled), 3 (Disabled) or 5 (Not Applicable).
RequestedState		Mandatory	The RequestedState property shall have a value of 12 (Not Applicable), a value of 5 (No Change), or a value that is contained in the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated CIM_EnabledLogicalElementCapabilities instance (i.e., 2 (Enabled), 3 (Disabled) or 11(Reset)).
OperationalStatus		Mandatory	None.
HealthState		Mandatory	None.
RequestStateChange()		Conditional	Conditional requirement: When a CIM_EnabledLogicalElementCapabilities instance is associated with the Central Instance and the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property is a non-empty array, sensor state management shall be supported.
RestoreDefaultThresholds( )		Conditional	Conditional requirement: The CIM_NumericSensor.RestoreDefaultThresholds( ) method shall be implemented and shall not return a value of 1 (Unsupported) when the CIM_NumericSensor.SettableThresholds property is a non-empty array.

34.6.5 CIM_Sensor

CIM_Sensor is used to represent a discrete sensor. Either the CIM_Sensor class (or its subclass) CIM_NumericSensor class is mandatory. The class definition specializes the CIM_Sensor definition in the Sensors profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 349 describes class CIM_Sensor.

Table 349 - SMI Referenced Properties/Methods for CIM_Sensor

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
DeviceID		Mandatory	Key.
SensorType		Mandatory	none.
PossibleStates		Mandatory	See section 7.2 of the DMTF Sensors Profile version 1.0.2.
CurrentState		Mandatory	The CIM_Sensor.CurrentState property shall have a value of one of the elements in the CIM_Sensor.PossibleStates array.
ElementName		Mandatory	The CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported property shall have a value of TRUE when the implementation supports client modification of the ElementName property of the associated CIM_Sensor.
OtherSensorTypeDescription		Conditional	Conditional requirement: The OtherSensorTypeDescription property shall be mandatory when the SensorType property is set to a value of 1 (Other).The OtherSensorTypeDescription property shall be formatted as a free-formed string of variable length (pattern \.*\').'
EnabledState		Mandatory	This property may have the values 2 (Enabled), 3 (Disabled) or 5 (Not Applicable).
RequestedState		Mandatory	The RequestedState property shall have a value of 12 (Not Applicable), a value of 5 (No Change), or a value that is contained in the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated CIM_EnabledLogicalElementCapabilities instance (i.e., 2 (Enabled), 3 (Disabled) or 11(Reset)).
OperationalStatus		Mandatory	none.
HealthState		Mandatory	none.
RequestStateChange()		Conditional	Conditional requirement: When a CIM_EnabledLogicalElementCapabilities instance is associated with the Central Instance and the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property is a non-empty array, sensor state management shall be supported.

34.6.6 CIM_SystemDevice

CIM_SystemDevice is used to associate CIM_Sensor with CIM_ComputerSystem that the CIM_Sensor is a member of.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 350 describes class CIM_SystemDevice.

Table 350 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Key: shall be a reference to the CIM_ComputerSystem instance of which the current CIM_Sensor (or CIM_NumericSensor) instance is a member.
PartComponent		Mandatory	Key: shall be a reference to the current CIM_Sensor (or CIM_NumericSensor) instance.

EXPERIMENTAL

EXPERIMENTAL

35 Base Server Profile

35.1 Synopsis

Profile Name: Base Server (Autonomous Profile)

Version: 1.6.0

Organization: SNIA

CIM Schema Version: 2.20.0

Table 351 describes the related profiles for Base Server.

Table 351 - Related Profiles for Base Server

Profile Name	Organization	Version	Requirement	Description
Fan	DMTF	1.0.1	Optional	See DSP1004 version 1.0.1, section 5
Physical Asset	DMTF	1.0.0a	Mandatory	See DSP1004 version 1.0.1, section 5
Power Supply	DMTF	TBD	Optional	See DSP1004 version 1.0.1, section 5
Record Log	DMTF	1.0.1	Optional	See DSP1052 version 1.0.0, section 5
Sensors	DMTF	1.0.2	Optional	See DSP1052 version 1.0.0, section 5
Software Inventory	DMTF	TBD	Optional	See DSP1052 version 1.0.0, section 5
Software Update	DMTF	1.0.0	Optional	See DSP1052 version 1.0.0, section 5
Storage HBA	SNIA	1.6.1	Optional	Experimental.
Host Discovered Resources	SNIA	1.6.0	Optional	Experimental.
Disk Partition	SNIA	1.6.0	Optional	Experimental.
SCSI Multipath Management	SNIA	1.6.0	Optional	Experimental.
Host Hardware RAID Controller	SNIA	1.5.0	Optional	Experimental.
Storage Enclosure	SNIA	1.3.0	Optional	Experimental.
Launch In Context	DMTF	1.0.0	Optional	Experimental. See DSP1102 version 1.0.0
Host Filesystem	SNIA	1.6.0	Optional	Experimental.

Specializes: DMTF Base Server 1.0.0

Central Class: CIM_ComputerSystem

Scoping Class: CIM_ComputerSystem

The Base Server Profile models a customer server or storage system.

35.2 Description

The SNIA Base Server Profile models a customer server or storage system containing storage elements. This profile may be used to scope one or more HBAs (or other storage elements).

This profile may represent either a physical system or a virtual system.

35.3 Implementation

See DSP1004, DMTF Base Server Profile for details on the model.

In a storage context, there are several related deployment options.

35.3.1 HBA Instrumentation

If an HBA vendor wishes to create HBA instrumentation that can be used with CIM instrumentation from a server vendor, they would implement the component Storage HBA Profile and work with the server vendor(s) to assure it integrates effectively with their autonomous server profile. If an HBA vendor wishes to deliver a free-standing implementation that does not rely on server-vendor software, they could implement this profile along with the Storage HBA Profile. Note that the HBA vendor could support both approaches and let a customer or installation script determine which is most appropriate.

35.3.2 Host Hardware RAID Instrumentation

Host Hardware RAID vendors have the same deployment options as HBA vendors (see 35.3.1)

35.3.3 Storage Enclosure Instrumentation

In configurations where the Storage Enclosure Profile is not used with a single autonomous profile, the Base Server may be used as the referencing profile for the Storage Enclosure and other component profiles.

35.3.4 Health and Fault Management Consideration

Not defined in this standard

35.3.5 Cascading Considerations

None

35.4 Methods

See DSP1004, DMTF Base Server Profile.

35.5 Use Cases

See DSP1004, DMTF Base Server Profile.

35.6 CIM Elements

Table 352 describes the CIM elements for Base Server.

Table 352 - CIM Elements for Base Server

Element Name	Requirement	Description
35.6.1 CIM_ComputerSystem	Mandatory	The hosting system for the Storage Elements. Associated to RegisteredProfile.
35.6.2 CIM_ComputerSystemPackage	Mandatory	DSP1004 version 1.0.1, section 10.2.
35.6.3 CIM_ElementCapabilities	Optional	See DSP1052 version 1.0.0, section 10.2.

Table 352 - CIM Elements for Base Server

Element Name	Requirement	Description
35.6.4 CIM_EnabledLogicalElementCapabilities	Optional	See DSP1052 version 1.0.0, section 10.3 and DSP1004 version 1.0.1, section 10.2.
35.6.5 CIM_HostedService	Optional	See DSP1052 version 1.0.0, section 10.4.
35.6.6 CIM_PhysicalPackage	Mandatory	DSP1004 version 1.0.1, section 10.4 and DSP1011 version 1.0.1, section 10.16.
35.6.7 CIM_ServiceAffectsElement	Optional	See DSP1052 version 1.0.0, section 10.5.
35.6.8 CIM_TimeService	Optional	Experimental. See DSP1052 version 1.0.0, section 10.6.

35.6.1 CIM_ComputerSystem

The hosting system for the Storage Elements. The class definition specializes the CIM_ComputerSystem definition in the Base Server profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Base Server', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.6.0'.

Table 353 describes class CIM_ComputerSystem.

Table 353 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key.
Name (overridden)		Mandatory	Unique identifier for the hosting system.
ElementName (overridden)		Mandatory	User friendly name.
OtherIdentifyingInfo (overridden)		Mandatory	
IdentifyingDescriptions		Optional	See DSP1052 version 1.0.0, section 10.1.
Dedicated (overridden)		Mandatory	0 (Not Dedicated).
OperationalStatus		Mandatory	See DSP1052 version 1.0.0, section 10.1.
HealthState		Mandatory	See DSP1052 version 1.0.0, section 10.1.
EnabledState		Mandatory	See DSP1052 version 1.0.0, section 10.1 and DSP1004 version 1.0.1, section 10.1.
RequestedState		Mandatory	See DSP1052 version 1.0.0, section 10.1 and DSP1004 version 1.0.1, section 10.1.
NameFormat (added)		Mandatory	

Table 353 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description & Notes
OtherDedicatedDescriptions (added)		Optional	
RequestStateChange()		Conditional	Conditional requirement: The CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property contains at least one value.See DSP1052 version 1.0.0, section 10.1.

35.6.2 CIM_ComputerSystemPackage

One or more instances of CIM_ComputerSystemPackage associate the CIM_ComputerSystem instance with the CIM_PhysicalPackage instances in which it resides. The constraints specified in this Table are in addition to those specified in the Physical Asset Profile.

Requirement: Mandatory

Table 354 describes class CIM_ComputerSystemPackage.

Table 354 - SMI Referenced Properties/Methods for CIM_ComputerSystemPackage

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

35.6.3 CIM_ElementCapabilities

CIM_ElementCapabilities associates an instance of CIM_EnabledLogicalElementCapabilities with an instance of CIM_ComputerSystem.

Requirement: Optional

Table 355 describes class CIM_ElementCapabilities.

Table 355 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	This property shall be a reference to an instance of CIM_ComputerSystem.
Capabilities		Mandatory	This property shall be a reference to the instance of CIM_EnabledLogicalElementCapabilities.

35.6.4 CIM_EnabledLogicalElementCapabilities

CIM_EnabledLogicalElementCapabilities indicates support for managing the state of the system.

Requirement: Optional

Table 356 describes class CIM_EnabledLogicalElementCapabilities.

Table 356 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
RequestedStatesSupported		Mandatory	See DSP1052 version 1.0.0, section 10.3 and DSP1004 version 1.0.1, section 10.3.
ElementNameEditSupported		Mandatory	See DSP1052 version 1.0.0, section 10.3.
MaxElementNameLen		Optional	See DSP1052 version 1.0.0, section 10.3.

35.6.5 CIM_HostedService

CIM_HostedService relates the CIM_TimeService to its scoping CIM_ComputerSystem instance.

Requirement: Optional

Table 357 describes class CIM_HostedService.

Table 357 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	This property shall reference the Central Instance.
Dependent		Mandatory	This property shall reference CIM_TimeService.

35.6.6 CIM_PhysicalPackage

One or more instances of CIM_PhysicalPackage represent the physical packaging of the computer system. Other than the existence of at least one instance of CIM_PhysicalPackage, this profile does not specify any constraints for CIM_PhysicalPackage beyond those specified in the Physical Asset Profile.

Requirement: Mandatory

Table 358 describes class CIM_PhysicalPackage.

Table 358 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	Key.
CreationClassName		Mandatory	Key.
PackageType		Mandatory	See DSP1011 version 1.0.1, section 10.16.
Manufacturer		Conditional	Conditional requirement: The CIM_PhysicalAssetCapabilities.FRUInfoSupported has a value of TRUE.See DSP1011 version 1.0.1, section 10.16.
Model		Conditional	Conditional requirement: The CIM_PhysicalAssetCapabilities.FRUInfoSupported has a value of TRUE.See DSP1011 version 1.0.1, section 10.16.
SerialNumber		Conditional	Conditional requirement: The CIM_PhysicalAssetCapabilities.FRUInfoSupported has a value of TRUE.See DSP1011 version 1.0.1, section 10.16.

Table 358 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	Flags	Requirement	Description & Notes
PartNumber		Conditional	Conditional requirement: The CIM_PhysicalAssetCapabilities.FRUInfoSupported has a value of TRUE.See DSP1011 version 1.0.1, section 10.16.
SKU		Conditional	Conditional requirement: The CIM_PhysicalAssetCapabilities.FRUInfoSupported has a value of TRUE.See DSP1011 version 1.0.1, section 10.16.
VendorCompatibilityStrings		Optional	See DSP1011 version 1.0.1, section 10.16.
CanBeFRUed		Optional	This property should be implemented when the Physical Element can be replaced in the field.
Version		Optional	The property shall be the hardware version.
Name		Optional	
ElementName		Mandatory	This property shall be formatted as a free-form string of variable length.

35.6.7 CIM_ServiceAffectsElement

CIM_ServiceAffectsElement associates the CIM_TimeService instance with the Central Instance.

Requirement: Optional

Table 359 describes class CIM_ServiceAffectsElement.

Table 359 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement

Properties	Flags	Requirement	Description & Notes
ElementEffects		Mandatory	Matches 5 (Manages).
AffectedElement		Mandatory	This property shall be a reference to the Central Instance.
AffectingElement		Mandatory	This property shall be a reference to an instance of CIM_TimeService.

35.6.8 CIM_TimeService

Experimental. CIM_TimeService manages the current time on the system.

Requirement: Optional

Table 360 describes class CIM_TimeService.

Table 360 - SMI Referenced Properties/Methods for CIM_TimeService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
Name		Mandatory	Key.
ElementName		Mandatory	See DSP1052 version 1.0.0, section 10.6.
ManageTime()		Mandatory	See DSP1052 version 1.0.0, section 10.6.

EXPERIMENTAL

EXPERIMENTAL

36 Media Access Device Profile

36.1 Synopsis

Profile Name: Media Access Device (Component Profile)

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.11.0

Table 361 describes the related profiles for Media Access Device.

Table 361 - Related Profiles for Media Access Device

Profile Name	Organization	Version	Requirement	Description
Software Inventory	SNIA	1.0.0	Mandatory	
Software Update	DMTF	1.0.0	Optional	
Indication	SNIA	1.5.0	Optional	

The Media Access Device Profile models media access devices - such as tape and CD drives.

36.2 Description

The Media Access Device Profile models media access devices - such as tape and CD drives.

36.2.1 Location Indicator

The implementation may optionally support a drive location indicator (such as an LED) using CIM_MediaAccessDevice.LocationIndicator. The client may set this to 2 (On) or 3 (Off)). If the implementation does not support this feature, LocationIndicator shall have the value 4 (Not Supported).

36.2.2 Media Access Device Online/Offline

The drive may be started or stopped by setting the Starting and Stopping values in OperationalStatus using the RequestStateChange method. Figure 49 shows Media Access Device Class information.

See Table 362.

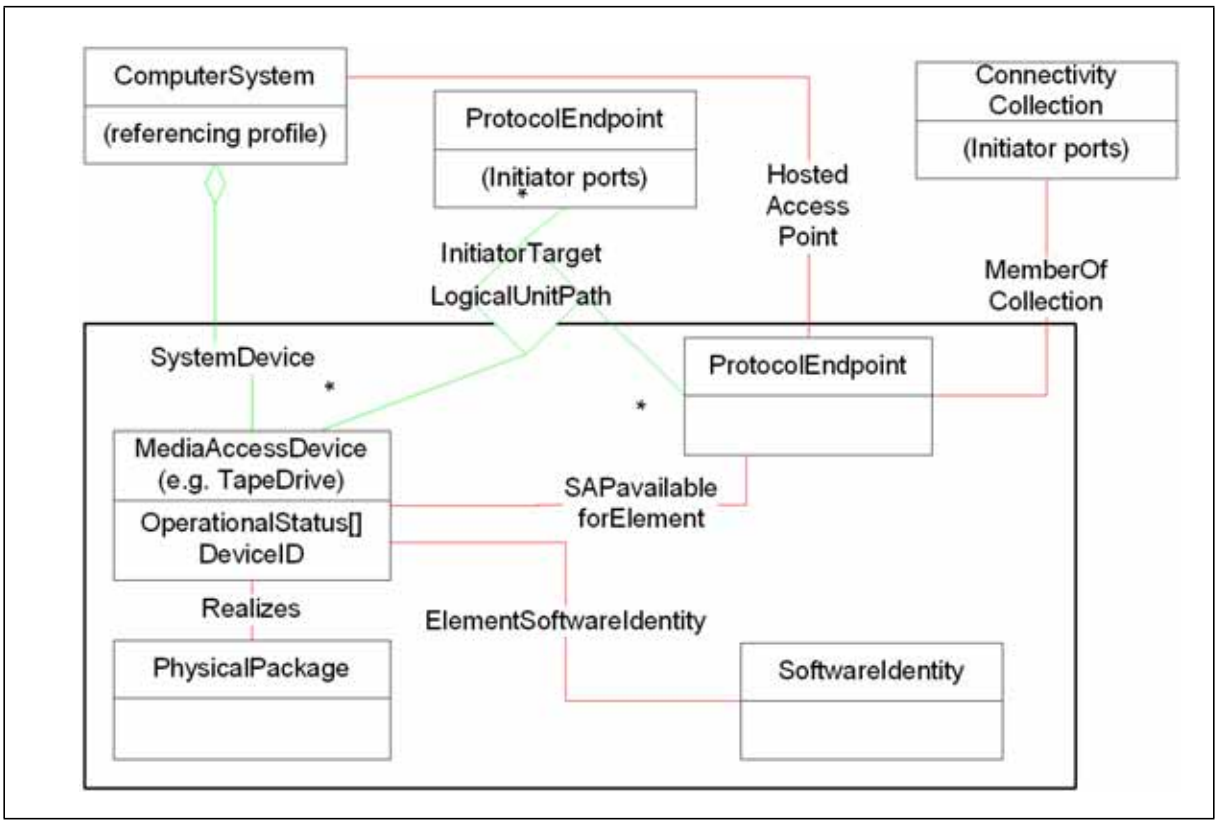

Figure 49 - Media Access Device Class Diagram

36.3 Implementation

36.3.1 Health and Fault Management Consideration

The MediaAccessDevice.OperationalStatus contains the overall status of the disk, summarized in Table 362.

Table 362 - OperationalStatus For MediaAccessDevice

Primary Operational Status	Subsidiary Operational Status	Description
2 “OK”		Media Access Device is enabled.
5 “Predictive Failure”		Media Access Device is functionality nominally but is predicting a failure
6 “Error”		Media Access Device is no longer functioning.
8 “Starting”		Media Access Device is becoming enabled.
9 “Stopping”		Media Access Device is being disabled.
10 “Stopped”		Media Access Device is disabled.

36.3.2 Cascading Considerations

Not defined in this standard.

36.3.3 Hot swap insertion or Removal of Drives

Insertion of a drive shall cause an InstCreation indication for the MediaAccessDevice instance. Similarly, hot-swap removal shall cause an InstDelete indication. ProtocolEndpoint, PhysicalPackage, SoftwareInventory, and related associations will also be created and deleted when a drive is inserted or removed, but no indications shall be produced for these other classes.

36.4 Methods

36.4.1 Request State Change

```
uint32 RequestStateChange(
  [In] uint16 RequestedState,
  [Out] CIM_ConcreteJob REF Job,
  [In] datetime TimeoutPeriod)
```

The allowed state changes are indicated by the RequestedStatesSupported property of EnabledLogicalElementCapabilities. A Job shall be returned if the operation takes longer than the TimeoutPeriod. The Requested State of Offline makes a drives extents unavailable to the dependent volume.

The Job may represent a drive rebuild if the RequestedState of the drive is Offline and a failover shall be complete before the offline operation can finish.

36.5 Use Cases

Not defined in this standard.

36.6 CIM Elements

Table 363 describes the CIM elements for Media Access Device.

Table 363 - CIM Elements for Media Access Device

Element Name	Requirement	Description
36.6.1 CIM_EnabledLogicalElementCapabilities	Mandatory	
36.6.2 CIM_HostedAccessPoint	Optional	ComputerSystem to storage ProtocolEndpoint.
36.6.3 CIM_MediaAccessDevice	Mandatory	Represents a tape or optical drive.
36.6.4 CIM_PhysicalPackage	Optional	The physical aspects of the drive. This is required when modeling physical drives and shall not be implemented for virtual drives in virtual system environments.
36.6.5 CIM_ProtocolEndpoint	Optional	
36.6.6 CIM_Realizes	Mandatory	Associates MediaAccessDevice and PhysicalPackage.
36.6.7 CIM_SAPAvailableForElement	Conditional	Conditional requirement: Support for ProtocolEndpoints. Associates MediaAccessDevice to ProtocolEndpoint.
36.6.8 CIM_SystemDevice	Mandatory	ComputerSystem to MediaAccessDevice.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_MediaAccessDevice	Optional	MediaAccessDevice Creation. See36.3.3 Hot swap insertion or Removal of Drives.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_MediaAccessDevice	Optional	MediaAccessDevice Removal. See36.3.3 Hot swap insertion or Removal of Drives.

36.6.1 CIM_EnabledLogicalElementCapabilities

Requirement: Mandatory

Table 364 describes class CIM_EnabledLogicalElementCapabilities.

Table 364 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description & Notes
RequestedStatesSupported		Mandatory	Possible states that can be requested when using the method RequestStateChange(). If RequestState and RequestStateChange are not implemented then RequestedStatesSupported would indicate none supported.

36.6.2 CIM_HostedAccessPoint

ComputerSystem to storage ProtocolEndpoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 365 describes class CIM_HostedAccessPoint.

Table 365 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

36.6.3 CIM_MediaAccessDevice

Represents a tape or optical drive.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 366 describes class CIM_MediaAccessDevice.

Table 366 - SMI Referenced Properties/Methods for CIM_MediaAccessDevice

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	

Table 366 - SMI Referenced Properties/Methods for CIM_MediaAccessDevice

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
OperationalStatus		Mandatory	Shall be 2|5|6|8|10|11 (Okay or Predictive Failure or Error or Starting or Stopping or Stopped).
LocationIndicator		Mandatory	
EnabledState		Mandatory	Possible values: 2 (Enabled - drive is Spun up and online), 3 (Disabled - drive is spun down, and offline), 4 (Shutting down - drive is spinning down), 6 (Enabled but Offline - drive is spun up but offline), 10 (Starting - drive is spinning up).
RequestedState		Optional	Possible RequestedStates: 2 Enabled (Spin up drive if it was spun down and Online the drive if it was offline), 4 (Shut down - spin down drive), 6 (Offline - offline drive).
RequestStateChange()		Conditional	Conditional requirement: Support for online/offline.

36.6.4 CIM_PhysicalPackage

The physical aspects of the drive.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 367 describes class CIM_PhysicalPackage.

Table 367 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Manufacturer		Mandatory	The name of the organization responsible for producing the PhysicalElement.
Model		Mandatory	The name by which the PhysicalElement is generally known.
Version		Mandatory	The version of the physical element - not necessarily the same as a software/firmware version.
SerialNumber		Mandatory	
PartNumber		Mandatory	

36.6.5 CIM_ProtocolEndpoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 368 describes class CIM_ProtocolEndpoint.

Table 368 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	

36.6.6 CIM_Realizes

Associates MediaAccessDevice and PhysicalPackage.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 369 describes class CIM_Realizes.

Table 369 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

36.6.7 CIM_SAPAvailableForElement

Associates MediaAccessDevice to ProtocolEndpoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ProtocolEndpoints.

Table 370 describes class CIM_SAPAvailableForElement.

Table 370 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

36.6.8 CIM_SystemDevice

ComputerSystem to MediaAccessDevice.

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 371 describes class CIM_SystemDevice.

Table 371 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

37 Storage Enclosure Profile

37.1 Synopsis

Profile Name: Storage Enclosure (Component Profile)

Version: 1.3.0

Organization: SNIA

CIM Schema Version: 2.15.0

Table 372 describes the related profiles for Storage Enclosure.

Table 372 - Related Profiles for Storage Enclosure

Profile Name	Organization	Version	Requirement	Description
Power Supply	SNIA	1.0.1	Optional	
Fan	SNIA	1.5.0	Optional	
Sensors	SNIA	1.5.0	Optional	
Disk Drive Lite	SNIA	1.6.0	Optional	
Media Access Device	SNIA	1.0.0	Optional	
Switch	SNIA	1.6.0	Optional	

The Storage Enclosure Profile describes an enclosure that houses storage components.

37.2 Description

The Storage Enclosure Profile describes an enclosure that contains storage elements (e.g., disk or tape drives) and enclosure elements (e.g., fans and power supplies). The logical aspects of the storage and enclosure elements are defined in other profiles; this profile specializes the DMTF Physical Asset Profile adding implementation details for storage enclosures. This profile supports enclosures with a single type of storage component (such as an enclosure of disks) or a mixture of different components.

The following terms apply to this profile:

- **storage elements** are CIM logical classes that relate to storage - CIM_DiskDrive, CIM_ComputerSystem (representing a disk array or switch), etc.
- **enclosure elements** are CIM logical elements that relate to enclosure service and baseboard management - fans, power supplies, sensors, etc.
- **physical elements** are CIM physical classes that map to storage or enclosure elements, and perhaps physical hardware with no logical mapping.

37.2.1 Guidelines related to Referencing Profiles

The Storage Enclosure Profile is a component profile. The autonomous referencing profile may be Array, Storage Virtualizer, or Host Hardware RAID controller. The following guidelines apply to how this profile is referenced by other profiles:

37.2.1.1 Guideline 1 - enclosure elements dedicated to a single top-level system

If the components of the enclosure are all dedicated to a single top-level System, then the profile defining that system shall be the referencing profile for the enclosure. All components (storage elements, enclosure elements, physical elements) need to be dedicated. For example, if the enclosure is used by a disk array, the CIM_ComputerSystem from the Array profile serves as the scoping instance for all the elements of the enclosure.

Note that the top-level system may be part of an autonomous profile that supports the SNIA Multiple Computer System Profile.

Note that other autonomous profiles may be dedicated as a component of another autonomous profile. For example, a Fibre Channel switch may share an enclosure with, and be dedicated as, a component of an Array.

37.2.1.2 Guideline 2 - enclosure elements shared by multiple top-level systems

If the elements of the enclosure support use by multiple top-level systems, then the referencing profile shall be the base system profile.

Examples include a JBOD array access by multiple servers or multiple switch blades sharing an enclosure.

37.2.1.3 Guideline 3 - enclosure elements need not be scoped by the system as storage elements

CIM requires instantiation of all weak associations whenever the referenced elements are instantiated. For example, every CIM_LogicalDevice instance shall be referenced by a CIM_SystemDevice association. But it is possible to have devices scope to different systems associated to each other by non-weak associations. In particular, when guideline 2 applies, enclosure elements scoped to the enclosure top-level system may be associated to storage elements scoped to a different top-level system. For example, CIM_AssociatedCooling can reference a CIM_Fan scoped to the enclosure system and a CIM_DiskDrive scoped to a server. In another example, CIM_SuppliesPower references a CIM_PowerSupply scoped to an Array within an enclosure and a CIM_ComputerSystem representing a switch.

Figure 50 is an example of two arrays that each have their own enclosure but share cooling. The two array enclosures are contained in an enclosure that provides a fan shared by the array elements.

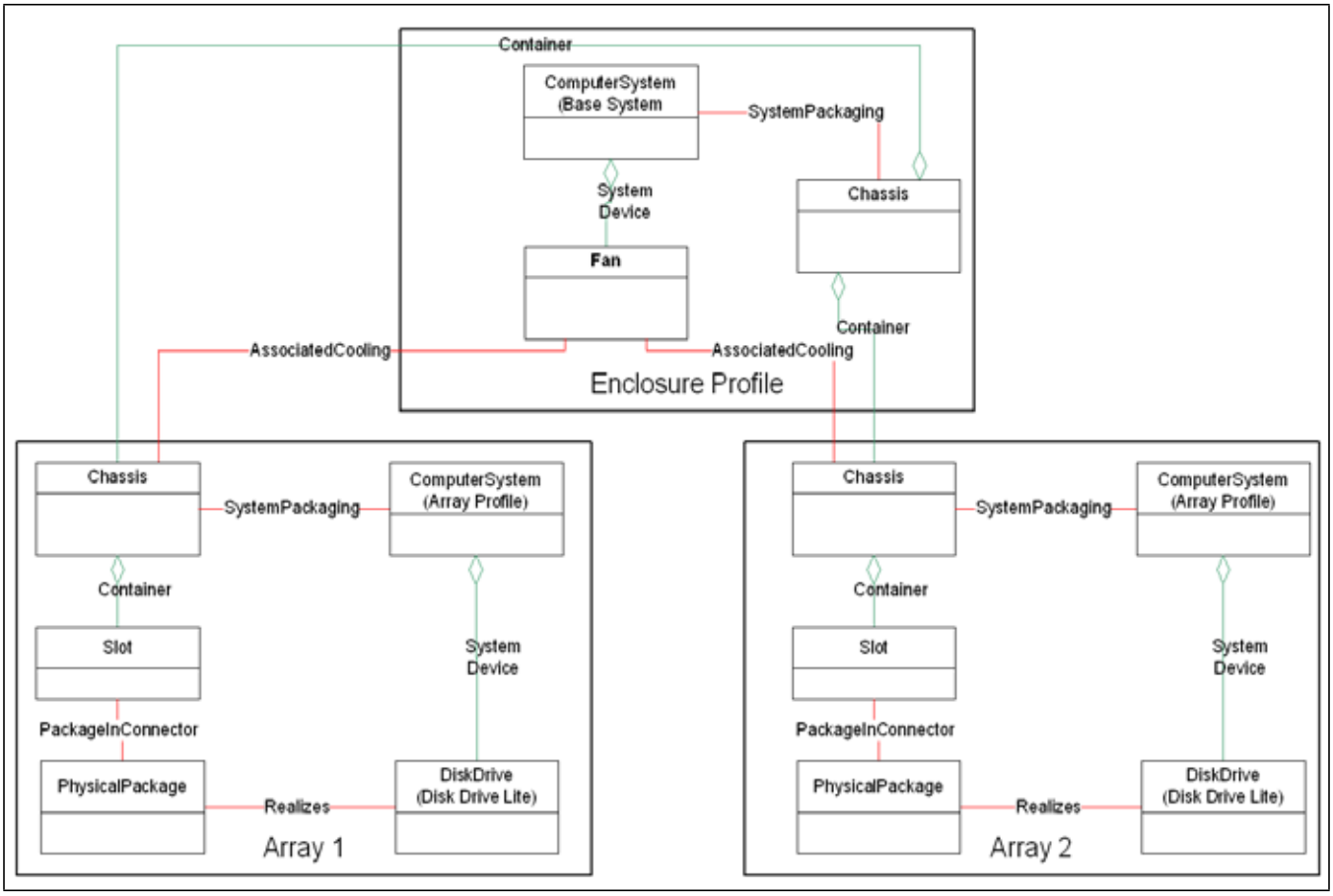

Figure 50 - Enclosure with Two Arrays

37.2.2 Examples of Storage Enclosure Configurations

37.2.2.1 Enclosure Dedicated to a Disk Array

The referencing profile is Array. Disk Drive Lite is a mandatory component profile. The physical model for disks as defined in 37.3.5.2 is mandatory.

37.2.2.2 Enclosure Dedicated to a RAID Host Controller

The referencing profile is the Host Hardware RAID Profile. Support for the Disk Drive Lite Profile is mandatory. The physical model for disks as defined in 37.3.5.2 is mandatory.

37.2.2.3 Enclosure Dedicated to non-RAID Controllers on a Single Server

The referencing profile is the Base System Profile referencing the Storage HBA Profile (or the FC HBA Profile).

37.2.2.4 Enclosure Dedicated to non-RAID Controllers on Multiple Servers

Guideline 2 applies. The referencing profile is the base system profile.

Guideline 3 may apply.

37.2.2.5 FC Switch as a Component of an Array

The Array and FC Switch share an enclosure, but the FC Switch is functionally a sub-component of the array receiving cooling and power from the enclosure. In this configuration, Array is the referencing profile to the Storage Enclosure. Guideline 3 may apply;

37.2.2.6 Enclosure containing multiple FC Switches (Director)

The enclosure is a director class switch which contains one or more switches and other devices including a FCIP Extenders and iSCSI Gateway. The referencing profile is the base system profile. Guideline 2 applies. Guideline 3 may apply.

37.3 Implementation

37.3.1 Health and Fault Management Consideration

Not defined in this standard.

37.3.2 Cascading Considerations

Not defined in this standard.

37.3.3 Enclosure Elements

37.3.3.1 Power Supplies

A storage enclosure may be modeled with one or more power supplies for device powering.

The CIM_SystemDevice association is used in the Power Supply Profile to connect the power supply to the managed system. The CIM_SuppliesPower association may be used to represent device powering to other enclosure elements of the top-level system as well as logical devices scoped to other systems.

37.3.3.2 Fans

A storage enclosure may be modeled with one or more fans for device cooling.

The CIM_SystemDevice association is used in the Fan Profile to connect the fan to the managed system. The CIM_AssociatedCooling association may be used to represent device powering to other enclosure elements of the top-level system as well as logical devices scoped to other systems.

37.3.3.3 Sensors

A storage enclosure may be modeled with one or more sensors for monitoring such factors as temperature or fan speed.

The CIM_SystemDevice association is used in the Sensors Profile to connect the sensor to the managed system. The CIM_AssociatedSensor association may be used to associate the sensor to other enclosure elements of the top-level system as well as logical devices scoped to other systems.

37.3.4 Storage Elements

37.3.4.1 Considerations for Media Access Devices in a Storage Enclosure

A storage enclosure may contain devices such as disk drives or switches. Each media access device is described by a corresponding device class as described in the corresponding profile. Each device may be associated to a physical bay or slot. The physical model for a disk drive describes a CIM_MediaAccessDevice associated to CIM_PhysicalPackage via CIM_Realizes, and CIM_Slot associated to the CIM_PhysicalPackage via CIM_PackageInConnector. If the implementation also supports hierarchical packaging, the CIM_Slot shall be associated to the CIM_PhysicalPackage realizing the referencing system or an enclosure nested in the system CIM_PhysicalPackage.

37.3.4.2 Disk Drive Considerations

If the implementation also supports the Disk Drive Lite Profile, the individual drives in the storage enclosure shall be described by an instance of CIM_DiskDrive subclassed from CIM_MediaAccessDrive. CIM_PhysicalPackage and CIM_Realizes from the Disk Drive Lite Profile shall provide the instances described in 37.3.4.1.

37.3.4.3 Media Access Devices and the Fan Profile

The Fan Profile describes fans used for device cooling and includes an AssociatedCooling association that references a CIM_ManagedSystemElement. If the implementation supports both the Fan and Disk Drive Lite Profiles, and utilizes the CIM_AssociatedCooling association, the CIM_AssociatedCooling association shall reference an instance of CIM_DiskDrive or an instance of CIM_Chassis.

37.3.4.4 Media Access Devices and the Power Supply Profile

The Power Supply Profile describes power supplies used for device powering and includes a CIM_SuppliesPower association that references a CIM_LogicalDevice. If the implementation supports both the Fan and Disk Drive Lite Profiles, and utilizes the CIM_SuppliesPower association, the CIM_SuppliesPower association shall reference an instance of CIM_DiskDrive or an instance of CIM_Chassis.

37.3.4.5 Configuration Reporting Service

The CIM_ConfigurationReportingService may be used to query for the CIM_MediaAccessDevice or CIM_LogicalPort subclasses supported within the enclosure, the supported total count and the currently installed count. In this way the total number of supported device slots, storage devices or connection ports may be retrieved. See the service method definitions in 37.4.1.

37.3.5 Physical Assets

The physical representation of the storage enclosure is mandatory. The core frame of the storage enclosure is described by CIM_Chassis.

37.3.5.1 Physical Package Hierarchy Considerations

A hierarchy of enclosures may be represented. The physical structure of a single enclosure, described by CIM_Chassis, may be associated with a variety of enclosure components and media devices. Any number of CIM_Packages may be used to group physical components. These packages may in turn be associated to one or more CIM_Chassis instances. In this case the CIM_PackageInChassis association shall be used.

37.3.5.2 Disk Drive or Media Access Device

If the implementation models slots within the enclosure, CIM_Slot shall be used to describe the slot. The instance of CIM_PhysicalPackage that describes the physical characteristics of the CIM_DiskDrive instance shall be associated to CIM_Slot by the CIM_PackageInConnector association. If the instance of

CIM_Slot is aggregated to an instance of CIM_Chassis, the CIM_ConnectorOnPackage association shall be used. Figure 51 illustrates the model.

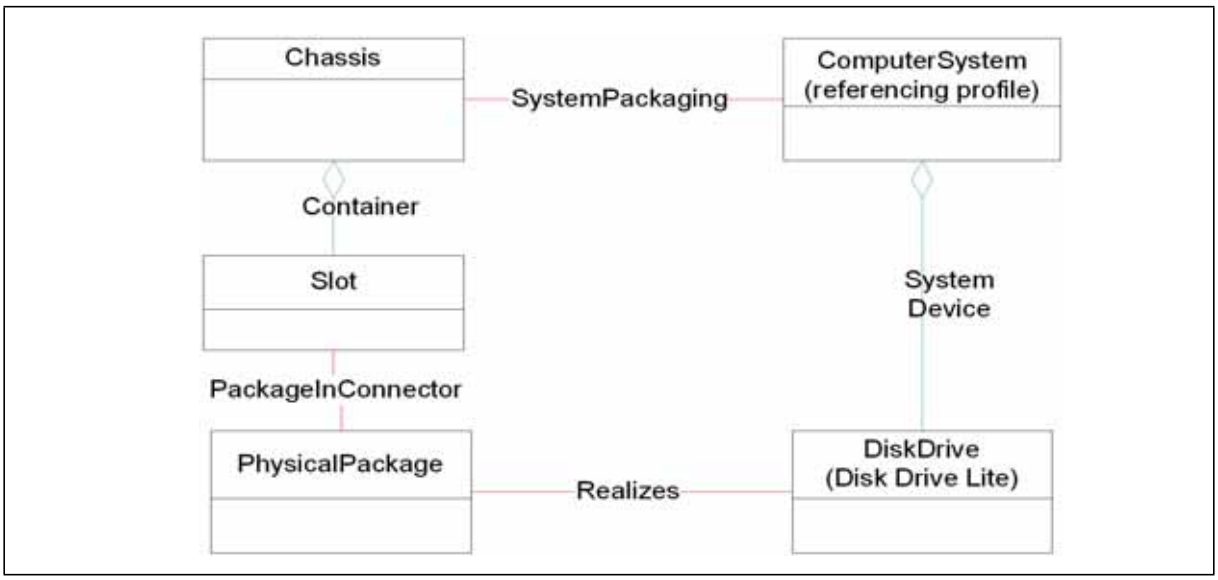

Figure 51 - Model for Disk in Enclosure

37.4 Methods

37.4.1 Extrinsic Methods of the Profile

37.4.1.1 CIM_ConfigurationReportingService GetClassTypes

GetClassTypes is used to query for the supported or currently installed device classes contained in the enclosure such as a CIM_DiskDrive or CIM_SASPort. Reporting of MediaAccessDevice derived classes directly contained within the enclosure (Recursive = False) is mandatory. Reporting of LogicalPort derived classes is optional.

The instrumentation shall support InquiryType parameter values of 2 (Supports) and 3 (Installed).

The instrumentation shall support a Recusive parameter value of false.

The instrumentation shall accept a reference to the top-level ComputerSystem in the Target parameter.

37.4.1.2 CIM_ConfigurationReportingService GetUnitTypes

GetUnitTypes is used to query for the supported or currently installed type of devices contained in the enclosure.

The instrumentation shall support InquiryType parameter values of 2 (Supports) and 3 (Installed).

The instrumentation shall support a Recusive parameter value of false.

The parameter UnitTypes may be set to "Contained", "StorageMediaLocation", "Front Side" or "Back Side". Support of the type "Contained" and "StorageMediaLocation" is mandatory. Support of "Front Side" or "Back Side" is optional. Types "Front Side" or "Back Side" are used to query for the count of the respective LogicalPorts.

37.4.1.3 CIM_ConfigurationReportingService ReportCapacity

ReportCapacity is used after GetClassTypes or GetUnitTypes is issued to find what subclasses and types are available in the enclosure, the ReportCapacity can be used to request the total supported or currently installed storage device slot count or data connection ports for the enclosure.

The instrumentation shall support InquiryType parameter values of 2 (Supports) and 3 (Installed).

The instrumentation shall support a Recusive parameter value of false.

37.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported

are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

37.5 Use Cases

37.6 CIM Elements

Table 373 describes the CIM elements for Storage Enclosure.

Table 373 - CIM Elements for Storage Enclosure

Element Name	Requirement	Description
37.6.1 CIM_ConfigurationReportingService	Mandatory	
37.6.2 CIM_HostedService	Mandatory	Associates the CIM_ConfigurationReportingService to the System in the referencing profile.

37.6.1 CIM_ConfigurationReportingService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 374 describes class CIM_ConfigurationReportingService.

Table 374 - SMI Referenced Properties/Methods for CIM_ConfigurationReportingService

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
Name		Mandatory	
CreationClassName		Mandatory	
GetClassTypes()		Mandatory	
GetUnitTypes()		Mandatory	
ReportCapacity()		Mandatory	

37.6.2 CIM_HostedService

Associates the CIM_ConfigurationReportingService to the System in the referencing profile.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 375 describes class CIM_HostedService.

Table 375 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The reference to the System.
Dependent		Mandatory	The reference to the Service.

EXPERIMENTAL

STABLE

38 Software Subprofile

38.1 Description

The Software Profile models software or firmware installed on a computer system.

Information on the installed software is given using the SoftwareIdentity class. This is linked to the system using a InstalledSoftwareIdentity association.

Software information may be associated with the “top” level ComputerSystem (if all components are using the same software) or a component ComputerSystem if the software loaded can vary by processor.

Firmware is modeled as SoftwareIdentity. InstalledSoftwareIdentity is used for firmware associated with a System.

Figure 52 contains the instance diagram for the Software Profile.

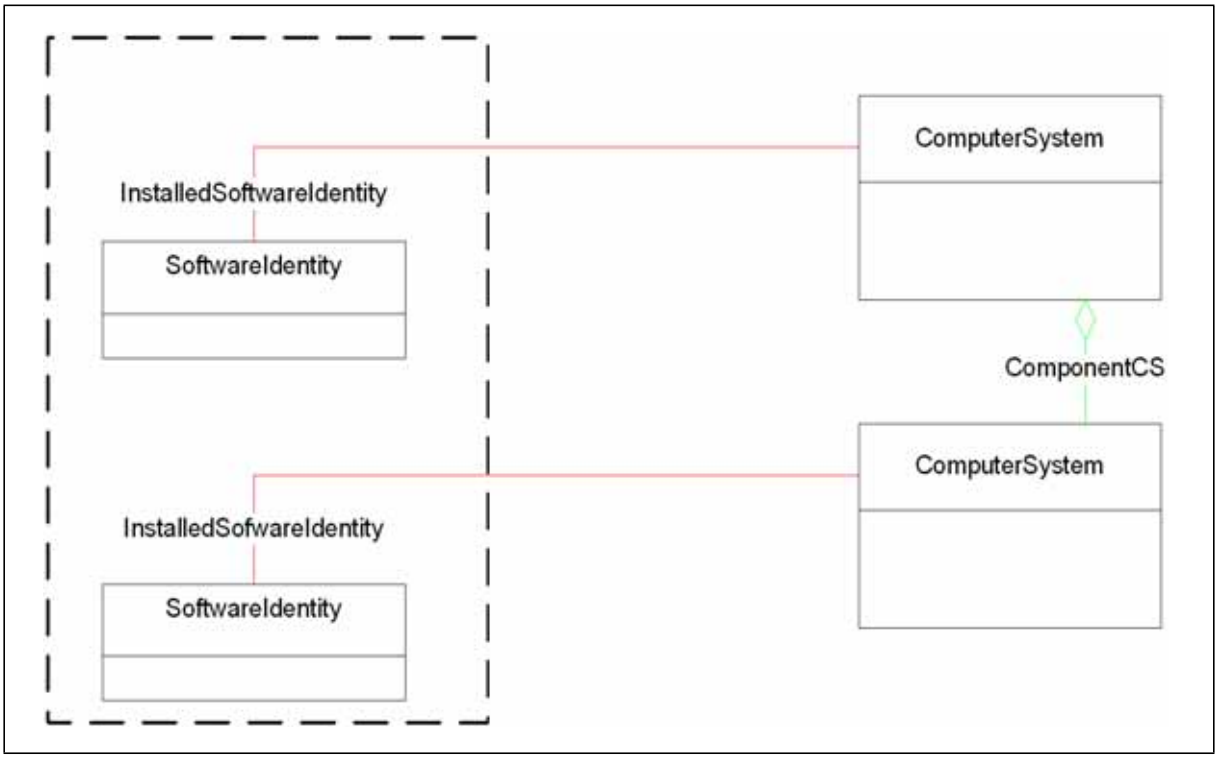

Figure 52 - Software Instance Diagram

38.2 Health and Fault Management Considerations

Not defined in this standard.

38.3 Cascading Considerations

Not defined in this standard.

38.4 Supported Subprofiles, and Packages

None

38.5 Methods of the Profile

None

38.6 Client Considerations and Recipes

None

38.7 Registered Name and Version

Software version 1.4.0 (Component Profile)

38.8 CIM Elements

Table 376 describes the CIM elements for Software.

Table 376 - CIM Elements for Software

Element Name	Requirement	Description
38.8.1 CIM_InstalledSoftwareIdentity	Mandatory	
38.8.2 CIM_SoftwareIdentity	Mandatory	

38.8.1 CIM_InstalledSoftwareIdentity

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 377 describes class CIM_InstalledSoftwareIdentity.

Table 377 - SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity

Properties	Flags	Requirement	Description & Notes
System		Mandatory	
InstalledSoftware		Mandatory	

38.8.2 CIM_SoftwareIdentity

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 378 describes class CIM_SoftwareIdentity.

Table 378 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
VersionString		Mandatory	
Manufacturer		Mandatory	
BuildNumber		Optional	
MajorVersion		Optional	
RevisionNumber		Optional	
MinorVersion		Optional	

STABLE

EXPERIMENTAL

39 Software Inventory Profile

39.1 Synopsis

Profile Name: Software Inventory (Component Profile)

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.15.0

Table 379 describes the related profiles for Software Inventory.

Table 379 - Related Profiles for Software Inventory

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.5.0	Mandatory	

Specializes: DMTF Software Inventory Profile 1.0.0

Central Class: CIM_SoftwareIdentity

Scoping Class: a CIM_System in a referencing autonomous profile

The Software Inventory Profile models installed and available software and firmware. The SNIA version specializes the DMTF profile in order to add indications.

39.2 Description

The Software Inventory Profile models installed and available software and firmware. The SNIA version specializes the DMTF profile in order to add indications.

39.2.1 Relationship to the SMI-S Software Profile

SMI-S defined a similar profile, the Software Subprofile (see 38 "Software Subprofile"). There are several differences between the two profiles:

- The Software Subprofile is limited to modeling software/firmware associated to a system and makes no provision for software/firmware associated to other elements (drives, ports,...)
- The DMTF Software Inventory Profile provides additional functionality:
  - software that is available on the system, but not installed - allowing the ability to model software/firmware that has been downloaded, but not activated.
  - collections of SoftwareIdentity instances
  - locations (such as URLS) associated with SoftwareIdentity instances

Also note that supporting this profile in SMI-S allows us to utilize the DMTF profiles which in turn use the Software Inventory Profile.

Note that although both profiles use InstalledSoftwareIdentity, the semantics are different. In the SMI-S Software Subprofile, InstalledSoftwareIdentity indicates that the software is both available and installed on the system. In the DMTF Software Inventory Profile, InstalledSoftwareIdentity indicates that the software is available (downloaded) on the system, and ElementSoftwareIdentity indicates that the software is active for the referenced element. ALso note that Software Inventory Profile has requirements for version properties beyond those in the SNIA Software Subprofile.

39.3 Implementation

See DSP1023, DMTF Software Inventory Profile.

39.3.1 Software Installation and Update

The CIM interface for Software Updates is described in the DMTF Software Update Profile (DSP1025). As a side effect of installation or updates, the inventory of software identities modeled in this profile is modified. This specialization adds indication filters:

- InstCreation of SoftwareIdentity represents a newly available software element (or new version)
- InstDeletion of SoftwareIdentity represents the deletion of an inactive SoftwareIdentity
- InstAlert with a Standard Message is used when a software (or firmware) version is updated “in-place” without installing a separate software/firmware package
- InstModification of ElementSoftwareIdentity.ElementSoftwareStatus (see 7.4.1.1 in DSP1023, DMTF Software Inventory Profile)

39.3.2 Health and Fault Management Consideration

None

39.3.3 Cascading Considerations

None

39.4 Methods

See DSP1023, DMTF Software Inventory Profile.

39.5 Use Cases

See DSP1023, DMTF Software Inventory Profile.

39.6 CIM Elements

Table 380 describes the CIM elements for Software Inventory.

Table 380 - CIM Elements for Software Inventory

Element Name	Requirement	Description
39.6.1 CIM_ElementSoftwareIdentity	Optional	
39.6.2 CIM_HostedAccessPoint	Optional	
39.6.3 CIM_HostedCollection	Conditional	Conditional requirement: Support for collection of SoftwareIdentity instances.
39.6.4 CIM_InstalledSoftwareIdentity	Optional	
39.6.5 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for collection of SoftwareIdentity instances.

Table 380 - CIM Elements for Software Inventory

Element Name	Requirement	Description
39.6.6 CIM_OrderedComponent	Optional	
39.6.7 CIM_OrderedDependency	Optional	
39.6.8 CIM_SAPAvailableForElement	Conditional	Conditional requirement: Support for SoftwareIdentityResource instances.
39.6.9 CIM_SoftwareIdentity	Mandatory	
39.6.10 CIM_SoftwareIdentityResource	Optional	
39.6.11 CIM_SystemSpecificCollection	Optional	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SoftwareIdentity	Mandatory	Creation of a SoftwareIdentity. See 39.3.1 Software Installation and Update.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SoftwareIdentity	Mandatory	Delete of a SoftwareIdentity. See 39.3.1 Software Installation and Update.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity=SNIA and MessageID=\Core1\"	Mandatory	In-place update of Software (or Firmware). See 39.3.1 Software Installation and Update.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SoftwareIdentity AND SourceInstance.CIM_SoftwareIdentity::ElementSoftwareStatus <> PreviousInstance.CIM_SoftwareIdentity::ElementSoftwareStatus	Optional	CQL -Change in ElementSoftwareStatus property of SoftwareIdentity. See 39.3.1 Software Installation and Update.

39.6.1 CIM_ElementSoftwareIdentity

CIM_ElementSoftwareIdentity is used to associate an instance of CIM_ManagedElement and an instance of CIM_SoftwareIdentity when the instance of CIM_ManagedElement is instrumented.

Requirement: Optional

Table 381 describes class CIM_ElementSoftwareIdentity.

Table 381 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity

Properties	Flags	Requirement	Description & Notes
ElementSoftwareStatus		Mandatory	
Antecedent		Mandatory	
Dependent		Mandatory	

39.6.2 CIM_HostedAccessPoint

CIM_HostedAccessPoint is used to associate CIM_System and CIM_SoftwareIdentityResource when an instance of CIM_SoftwareIdentityResource is instrumented.

Requirement: Optional

Table 382 describes class CIM_HostedAccessPoint.

Table 382 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

39.6.3 CIM_HostedCollection

CIM_HostedCollection is used to associate CIM_System and CIM_SystemSpecificCollection. CIM_HostedCollection is conditional and shall be implemented when an instance of CIM_SystemSpecificCollection is instrumented.

Requirement: Support for collection of SoftwareIdentity instances.

Table 383 describes class CIM_HostedCollection.

Table 383 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

39.6.4 CIM_InstalledSoftwareIdentity

CIM_InstalledSoftwareIdentity is used to associate an instance of CIM_System and an instance of CIM_SoftwareIdentity. CIM_InstalledSoftwareIdentity is conditional and shall be implemented when Installed Software is modeled.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 384 describes class CIM_InstalledSoftwareIdentity.

Table 384 - SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity

Properties	Flags	Requirement	Description & Notes
System		Mandatory	
InstalledSoftware		Mandatory	

39.6.5 CIM_MemberOfCollection

CIM_MemberOfCollection is used to associate an instance of CIM_SystemSpecificCollection and an instance of CIM_SoftwareIdentity. CIM_MemberOfCollection is conditional and shall be implemented when an instance of CIM_SystemSpecificCollection is instrumented.

Requirement: Support for collection of SoftwareIdentity instances.

Table 385 describes class CIM_MemberOfCollection.

Table 385 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

39.6.6 CIM_OrderedComponent

CIM_OrderedComponent is used to associate an instance of CIM_SoftwareIdentity that represents a Software Bundle and an instance of CIM_SoftwareIdentity that represents one of the discrete software images contained in the Software Bundle.

Requirement: Optional

Table 386 describes class CIM_OrderedComponent.

Table 386 - SMI Referenced Properties/Methods for CIM_OrderedComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

39.6.7 CIM_OrderedDependency

CIM_OrderedDependency is used to associate an instance of CIM_SoftwareIdentity that represents an Installation Dependency and an instance of CIM_SoftwareIdentity for which the Installation Dependencies are represented.

Requirement: Optional

Table 387 describes class CIM_OrderedDependency.

Table 387 - SMI Referenced Properties/Methods for CIM_OrderedDependency

Properties	Flags	Requirement	Description & Notes
AssignedSequence		Mandatory	
Antecedent		Mandatory	
Dependent		Mandatory	

39.6.8 CIM_SAPAvailableForElement

CIM_SAPAvailableForElement is used to associate CIM_SoftwareIdentityResource and CIM_SoftwareIdentity. CIM_SAPAvailableForElement is conditional and shall be implemented when the location information of CIM_SoftwareIdentity is represented.

Requirement: Support for SoftwareIdentityResource instances.

Table 388 describes class CIM_SAPAvailableForElement.

Table 388 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

39.6.9 CIM_SoftwareIdentity

CIM_SoftwareIdentity is used to represent either Installed Software or Available Software.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 389 describes class CIM_SoftwareIdentity.

Table 389 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
IsEntity		Mandatory	
VersionString		Optional	
BuildNumber		Optional	
MajorVersion		Conditional	Conditional requirement: No Support for SoftwareIdentity.VersionString.
MinorVersion		Conditional	Conditional requirement: No Support for SoftwareIdentity.VersionString.
RevisionNumber		Conditional	Conditional requirement: No Support for SoftwareIdentity.VersionString.
TargetOSTypes		Optional	

39.6.10CIM_SoftwareIdentityResource

CIM_SoftwareIdentityResource is used to represent the location of a Software Identity, which could be used as input to the software installation service.

Requirement: Optional

Table 390 describes class CIM_SoftwareIdentityResource.

Table 390 - SMI Referenced Properties/Methods for CIM_SoftwareIdentityResource

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	

Table 390 - SMI Referenced Properties/Methods for CIM_SoftwareIdentityResource

Properties	Flags	Requirement	Description & Notes
InfoFormat		Mandatory	
AccessInfo		Mandatory	
ResourceType		Optional	

39.6.11CIM_SystemSpecificCollection

CIM_SystemSpecificCollection is used to represent a collection of Available Software.

Requirement: Optional

Table 391 describes class CIM_SystemSpecificCollection.

Table 391 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	

EXPERIMENTAL

STABLE

40 Server Profile

40.1 Description

40.1.1 Model Overview

A CIM Server is anything that supports WBEM protocols. The Server Profile is mandatory for all compliant SMI-S servers.

The object manager part of the model, shown in Figure 53, defines the capabilities of a CIM object manager based on the communication mechanisms that it supports.

The namespace model of the Server Profile describes the namespaces managed by the object manager and the type information contained within the namespace. The main information provided in the namespace part of the model is the namespace itself and its association to the ObjectManager.

The InteropNamespace refers to the first namespace found in the InteropSchemaNamespace attribute of the SLP Template.

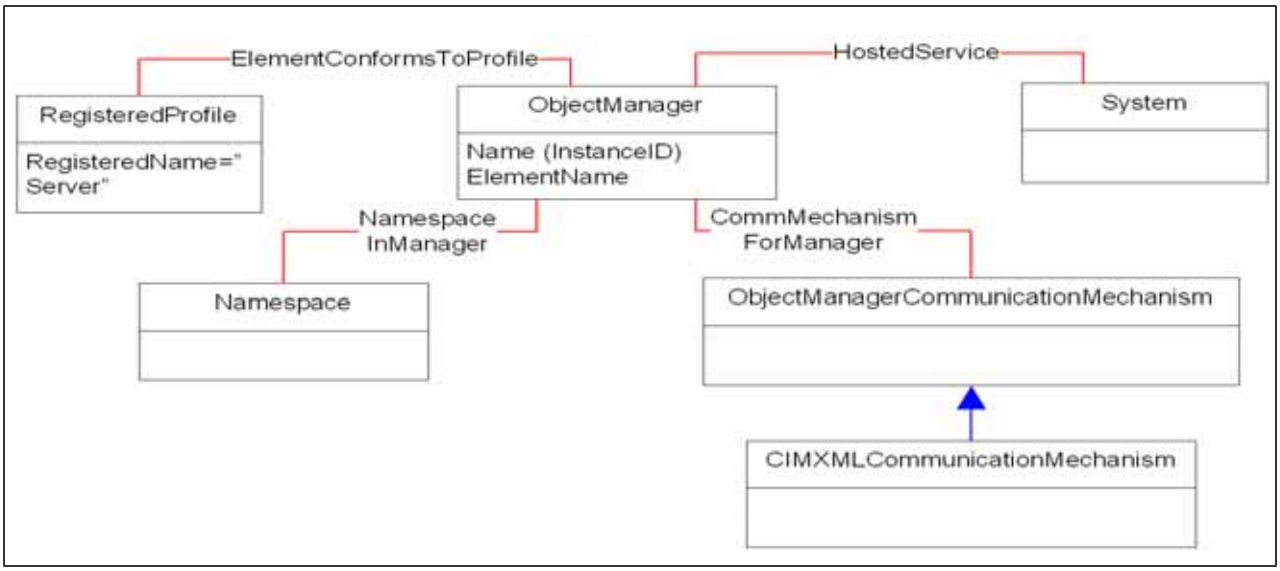

Figure 53 - Server Model

A Server is modeled as a System with a HostedService association to an ObjectManager. The ObjectManager is subclassed from Service. Implementations shall support an ElementConformsToProfile association referencing the RegisteredProfile for the Server Profile and referencing the ObjectManager (rather than CIM_System as is common in other profiles).

It is mandatory that all namespaces supported by the Server be identified (the Namespace class) and associated to the ObjectManager via the NamespaceInManager association.

The communication protocols supported by the ObjectManager should also be identified. Specifically, the CIMXMLCommunicationMechanism shall be present for standard communication support for clients. This class is associated to the ObjectManager via the CommMechanismForManager association.

The Profile Registration Profile describes the set of classes and associations deal with profiles supported by the ObjectManager. The Profile Registration Profile is required by the Server Profile.

40.1.2 Use of model fields to Populate the SLP template

The data used to populate the SLP template for advertising SMI-S profiles is found in the CIM Server profile. The SLP template fields are populated as follows:

template-url-syntax: =string
The following quotation is from the “WBEM SLP Template v1.0.0.
http://www.dmtf.org/standards/wbem/wbem.1.0.en

> “The template-url-syntax MUST be the WBEM URI Mapping 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.
>
> The WBEM URI Mapping is defined in the WBEM URI Mapping Specification 1.0.0 (DSP0207). Example: (template-url-syntax=https://localhost:5989)”

service-hi-name: ObjectManager.ElementName

service-hi-description: ObjectManager.Description

service-id: ObjectManager.Name

CommunicationMechanism: ObjectManagerCommunicationMechanism.CommunicationMechanism

OtherCommunicationMechanism:
ObjectManagerCommunicationMechanism.OtherCommunicationMechanism

InteropSchemaNamespace: Namespace.Name for the InteropNamespace

ProtocolVersion: ObjectManagerCommunicationMechanism.Version

MultipleOperationsSupported:
ObjectManagerCommunicationMechanism.MultipleOperationsSupported

AuthenticationMechanismSupported:
ObjectManagerCommunicationMechanism.AuthenticationMechanismsSupported

OtherAuthenticationDescription:
ObjectManagerCommunicationMechanism.AuthenticationMechanismDescriptions

Namespace: Namespace.Name for each Namespace instance supported

Classinfo: Namespace.Classinfo for each Namespace instance

RegisteredProfilesSupported:

A list of profiles supported by the CIM providers running in this CIM Server. Each entry is this list is separate by a comma and consists of two or three sub-fields, separated by colons. If an entry refers to a supported profile defined in a RegisteredProfile (and not RegisteredSubProfile) instance, the format shall be

```
Organization:Name
```

where organization is the name of the organization that defined the profile (e.g., SNIA or DMTF) and Name is the name of the profile. Note that this first format applies to autonomous or component profiles defined using RegisteredProfile. If an entry refers to a supported subprofile defined in a RegisteredSubProfile instance, the format shall be

```
Organization:Name:Subprofile-Name
```

where organization is the name of the organization that defined the profile (e.g., SNIA or DMTF), Name is the name of the profile, and Subprofile-Name is the name of the subprofile.

For either format, Organization shall be identical to the RegisteredOrganization attribute in the appropriate RegisteredProfile instance. For the first format, Name shall be identical to the RegisteredName attribute in the appropriate RegisteredProfile instance. For the second format:

- Subprofile-Name shall be identical to the RegisteredName attribute in the appropriate RegisteredSubProfile instance
- Name shall be identical to the RegisteredName attribute in the RegisteredProfile referenced by the RegisteredSubProfile

Implementations are required to include an entry for each supported autonomous profile. Implementations are required to include an entry for a component profile if the component profile definition in this standard states that the component profile shall be advertised via SLP. It is recommended that other subprofiles and component profiles be excluded from this list to minimize the size of the SLP template.

40.1.3 Support for Indications

An implementation of the Server Profile may optionally support an indication reporting that the Object Manager was started.

40.1.4 Security Background

See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5*, Security (13)

40.2 Health and Fault Management

Not defined in this standard.

40.3 Cascading Considerations

Not defined in this standard.

40.4 Supported Subprofiles and Packages

Table 392 describes the supported profiles for Server.

Table 392 - Supported Profiles for Server

Profile Name	Organization	Version	Requirement	Description
Object Manager Adapter	SNIA	1.3.0	Optional	
Experimental Indication	SNIA	1.5.0	Optional	
Profile Registration	SNIA	1.5.0	Mandatory	
Launch In Context	DMTF	1.0.0	Optional	Experimental. See DSP1102, version 1.0.0
Indication	SNIA	1.5.0	Support for at least one is mandatory.	Deprecated. See the SNIA Indications Profile
Indications	SNIA	1.6.0		Experimental.
Indications	DMTF	1.2.0		Experimental. See DSP1054, version 1.2.0

40.5 Methods of the Profile

The implementation may support ObjectManager.StopService. If implemented, this method shall shut down the CIM object manager. The method returns an integer value of 0 if the service was successfully stopped, 1 if the request is not supported, and any other number to indicate an error.

40.6 Client Considerations and Recipes

40.6.1 Segregate a SAN Device Type

```
// DESCRIPTION
// A management application wishes to manage a particular type of SAN
// device, but not other devices. So the management application needs to
// isolate the particular CIM Servers that support the type of device it
// wants to manage.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume CIM Servers have advertised their services (SrvReg)
// 2.Assume there are one or more Directory Agents in the subnet
// 3.Assume no security on SLP discovery
// 4.#DirectoryList[] is an array of directory URLs
// 5.#DirectoryEntries [] is an array of directory entry Structures.
//   The structure matches the “wbem” SLP Template (see ‘Standard
//   WBEM Service Type Templates).
// 6.Assume that the device is #DesiredProfile and the device is an
//   SMI-S device (a SNIA defined profile)

// Step 1:  Set the Previous Responders List to the Null String.
#PRList = ““

// Step 2: Multicast a Service Request for a Directory Server Service.
//   This is to find Directory Agents in the subnet.
//
SrvRqst (
     #PRList,        // The Previous Responders list
     service:directory-agent // Service type
     “DEFAULT”,          // The scope
     NULL,             // The predicate
     NULL)             // SLP SPI (security token)

// Step 3: Listen for Response from Directory Agent(s)
#DirectoryList[] = DAAdvert (
     BootTimestamp, // Time of last reboot of DA
     URL,      // The URL of the DA
     ScopeList,// The scopes supported by the DA
     AttrList,// The DA Attributes
     SLP SPI List,// SLP SPI (SPIs the DA can verify)
     Authentication Block)
// Iterate on Steps 2 & 3, until a response has been received or the client
// has reached a UA configured CONFIG_RETRY_MAX seconds.

// Step 4: Unicast a Service Request to each of the DAs specifying a
//   query predicate to select CIM Servers that support SNIA
//   #DesiredDevice profiles and listen for responses.
```

```
for #j in #DirectoryList[]
{

     SrvRqst (
        #DAPRList,        // The Previous Responders list
        “service:wbem”,    // Service type
        “DEFAULT”,          // The scope
        “RegisteredProfilesSupported=SNIA:”+#DesiredProfile+”*”,
                                          // The predicate
        NULL)             // SLP SPI (security token)
     #ServiceList [#j] = SrvRply  (
        Count,       // count of URLs
        #SAPRList[])
}

// Step 5: Next retrieve the attributes of each advertisement
For #i in #ServiceList[]  // for each url in list
{
     AttrRqst (
        #SAPRList,       // The Previous Responders list
        #ServiceList[#i ],// a url from #ServiceList[]
        “DEFAULT”, // The scope
        NULL,  // Tag list.  NULL means return all
               // attributes
        NULL)  // SLP SPI (security token)
     #DirectoryEntries [#i] = AttrRply (#attr-list)
}

// Step 7: Correlate the responses to the Service Request on unique
//   “service-id” to determine unique CIM Servers. The client will get
//   multiple responses (one for each access point) for each CIM
//   Server. At this point, the client has a list of CIM Servers that
//   claim to support SNIA #DesiredProfile profiles.
```

40.7 Registered Name and Version

Server version 1.6.0 (Autonomous Profile)

40.8 CIM Elements

Table 393 describes the CIM elements for Server.

Table 393 - CIM Elements for Server

Element Name	Requirement	Description
40.8.1 CIM_CIMXMLCommunicationMechanism	Mandatory	Deprecated. SMI-S 2.0 will use the superclass: CIM_ObjectManagerCommunicationMechanism. Represents CIM-XML Support.
40.8.2 CIM_CommMechanismForManager	Mandatory	This associates the ObjectManager and the communication classes it supports.
40.8.3 CIM_HostedAccessPoint	Mandatory	This associates the communication mechanisms with the hosting System.
40.8.4 CIM_HostedService	Mandatory	Connects the ObjectManager to the System that is hosting the ObjectManager.
40.8.5 CIM_Namespace	Mandatory	This is a namespace within the Object Manager.
40.8.6 CIM_NamespaceInManager	Mandatory	This associates the namespace to the ObjectManager.
40.8.7 CIM_ObjectManager	Mandatory	This is the Object Manager service of the CIM Server. Associated to RegisteredProfile.
40.8.8 CIM_ObjectManagerCommunicationMechanism	Optional	Instantiate to show optionalal support for WS-Man.
40.8.9 CIM_System	Mandatory	The System that is hosting the Object Manager (CIM Server).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ObjectManager AND SourceInstance.Started <> PreviousInstance.Started	Optional	Deprecated WQL -Start of object manager. See *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 40.1.3 Support for Indications.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ObjectManager AND SourceInstance.CIM_ObjectManager::Started <> PreviousInstance.CIM_ObjectManager::Started	Optional	CQL -Start of object manager. See *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 40.1.3 Support for Indications.

40.8.1 CIM_CIMXMLCommunicationMechanism

Deprecated. SMI-S 2.0 will use the superclass: CIM_ObjectManagerCommunicationMechanism. Represents CIM-XML Support.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 394 describes class CIM_CIMXMLCommunicationMechanism.

Table 394 - SMI Referenced Properties/Methods for CIM_CIMXMLCommunicationMechanism

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	

Table 394 - SMI Referenced Properties/Methods for CIM_CIMXMLCommunicationMechanism

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
CommunicationMechanism		Mandatory	Shall be 2 (CIM-XML).
Version		Mandatory	The 'CIM Operations over HTTP' version supported by the server.
CIMValidated		Mandatory	
FunctionalProfilesSupported		Mandatory	
MultipleOperationsSupported		Mandatory	
AuthenticationMechanismsSupported		Mandatory	
OperationalStatus		Mandatory	Should be 0 (Unknown), 2 (Okay), or 10 (Stopped).
StatusDescriptions		Optional	

40.8.2 CIM_CommMechanismForManager

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 395 describes class CIM_CommMechanismForManager.

Table 395 - SMI Referenced Properties/Methods for CIM_CommMechanismForManager

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to CommunicationMechanism.
Antecedent		Mandatory	Reference to ObjectManager.

40.8.3 CIM_HostedAccessPoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 396 describes class CIM_HostedAccessPoint.

Table 396 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to CommuicationMechanism.
Antecedent		Mandatory	Reference to CIM_System.

40.8.4 CIM_HostedService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 397 describes class CIM_HostedService.

Table 397 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to CIM_ObjectManager.
Antecedent		Mandatory	Reference to CIM_System.

40.8.5 CIM_Namespace

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 398 describes class CIM_Namespace.

Table 398 - SMI Referenced Properties/Methods for CIM_Namespace

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
ObjectManagerCreationClassName		Mandatory	
ObjectManagerName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ClassType		Mandatory	Should be 2 (CIM) for either CIM_ and SNIA_ classes.

Table 398 - SMI Referenced Properties/Methods for CIM_Namespace

Properties	Flags	Requirement	Description & Notes
DescriptionOfClassType		Conditional	Conditional requirement: CIM_Namespace.Namespace having value 1 (Other).
ClassInfo		Optional	Deprecated. Deprecated in the MOF, but required for 1.0 compatibility. Required only if the CIMOM is hosting profiles that were part of SMI-S 1.0.x.
DescriptionOfClassInfo		Optional	Deprecated. Deprecated in the MOF, but mandatory for 1.0 compatibility. Mandatory if ClassInfo is set to 'Other' and if the CIMOM is hosting profiles that were part of SMI-S 1.0.x.

40.8.6 CIM_NamespaceInManager

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 399 describes class CIM_NamespaceInManager.

Table 399 - SMI Referenced Properties/Methods for CIM_NamespaceInManager

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to CIM_Namespace.
Antecedent		Mandatory	Reference to CIM_ObjectManager.

40.8.7 CIM_ObjectManager

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Server', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.6.0'.

Table 400 describes class CIM_ObjectManager.

Table 400 - SMI Referenced Properties/Methods for CIM_ObjectManager

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	

Table 400 - SMI Referenced Properties/Methods for CIM_ObjectManager

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	The name (make and model) of the server for human interfaces. For example, "ACME CIM Server".
Description		Mandatory	The name (make and model) and version of the server for human interfaces. For example, "ACME CIM Server version 2.2".
OperationalStatus		Mandatory	Should be 0 (Unknown), 2 (Okay), or 10 (Stopped).
Started		Mandatory	
StopService()		Optional	This method shall shut down the CIM object manager. See *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* The implementation may support ObjectManager.StopService. If implemented, this method shall shut down the CIM object manager. The method returns an integer value of 0 if the service was successfully stopped, 1 if the request is not supported, and any other number to indicate an error..

40.8.8 CIM_ObjectManagerCommunicationMechanism

Instantiate to show optionalal support for WS-Man.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 401 describes class CIM_ObjectManagerCommunicationMechanism.

Table 401 - SMI Referenced Properties/Methods for CIM_ObjectManagerCommunicationMechanism

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ElementName		Mandatory	
CommunicationMechanism		Mandatory	Shall be 4 (WS Management).
FunctionalProfilesSupported		Mandatory	Shall be 0 (Unknown).
MultipleOperationsSupported		Mandatory	Shall be false.
AuthenticationMechanismsSupported		Mandatory	
OperationalStatus		Mandatory	Should be 0 (Unknown), 2 (Okay), or 10 (Stopped).
StatusDescriptions		Optional	
FunctionalProfileDescriptions		Optional	

40.8.9 CIM_System

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 402 describes class CIM_System.

Table 402 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
NameFormat		Mandatory	Shall be either 'IP', 'WWN', or 'Other'.
Name		Mandatory	If NameFormat is 'IP', Name shall be a valid Ipv4, Ipv6, or fully qualified domain name. If NameFormat, is 'WWN', Name shall be formatted as 16 unseparated upper case hex digits.
Description		Mandatory	
ElementName		Mandatory	
OperationalStatus		Mandatory	Should be 0 (Unknown), 2 (Okay), or 10 (Stopped).

STABLE

STABLE

41 Profile Registration Profile

41.1 Synopsis

Profile Name: Profile Registration

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.12.0

Specializes: DMTF Profile Registration 1.0.0

No included profiles are defined in this standard.

Profile Registration Profile models the profiles registered in the object manager and the associations between registration classes and the domain classes implementing the profile.

41.2 Description

The SNIA Profile Registration Profile specializes the DMTF Profile Registration Profile adding the following classes:

- CIM_RegisteredSubProfile (subclass of CIM_RegisteredProfile)
- CIM_SubProfileRequiresProfile (subclass of CIM_ReferencedProfile)
- CIM_SoftwareIdentity
- CIM_ElementSoftwareIdentity
- CIM_Product
- CIM_ProductSoftwareComponent

41.3 Implementation

In DMTF profiles, the term ‘component profile’ is used similarly to the way ‘subprofile’ was used in SMI-S 1.0.x and 1.1.x; and the term ‘autonomous profile’ is used similarly to the way ‘profile’ was used in SMI-S 1.0.x and 1.1.x. SNIA implementations may use the SNIA 1.0.x/1.1.x approach with the RegisteredSubProfile and SubProfileRequiresProfile subclasses) or the DMTF approach using

RegisteredProfile for component profiles and ReferencedProfile. Figure 54 shows the Profile Registration

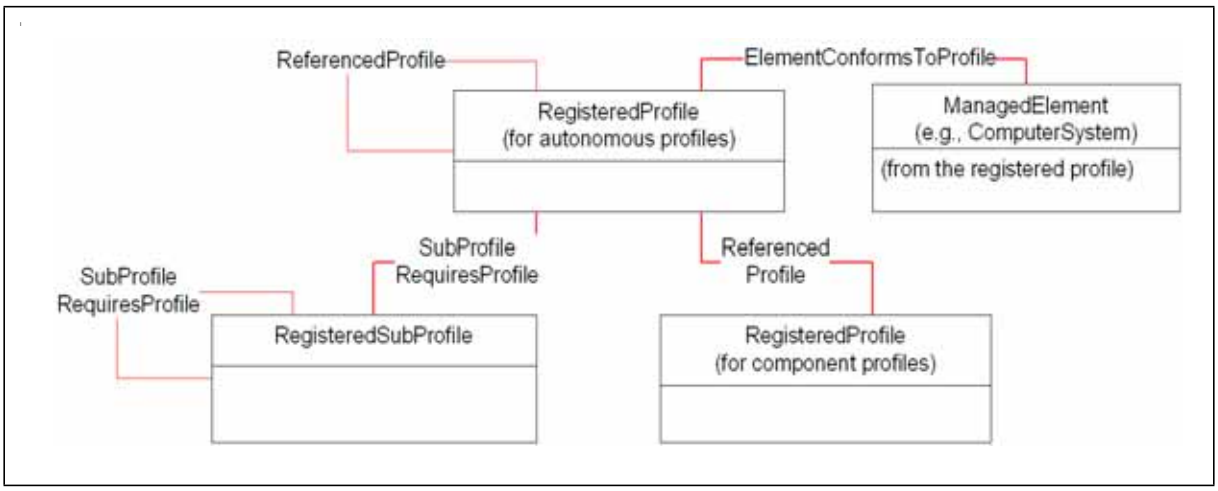

Figure 54 - Profile Registration Model

Model.

SMI-S clients should use the superclasses (RegisteredProfile and ReferencedProfile) in CIM operations to assure that implementations conforming to either SMI-S or DMTF profiles are discovered. ReferencedProfile associates two instances of RegisteredProfile. The DMTF Profile Registration Profile describes how the Antecedent and Dependent references should be used when one profile includes another in its supported/referenced profile list. Implementations are inconsistent in the use of these references and clients should be prepared for either approach; one technique to achieve this would be to specify NULL for Role and RemoteRole in Associator or AssociatorName operations.

The Scoping Class methodology defined in the DMTF Profile Registration Profile shall be implemented. The Central Class methodology may be implemented.

For each Profile instance, the supported component profiles should be identified via the SubprofileRequiresProfile or ReferencedProfile association. Subprofiles are modeled using RegisteredSubProfile (or ReferencedProfile).

Instances of RegisteredProfile, RegisteredSubProfile, SubProfileRequiresProfile, and ReferencedProfile are in the Interop namespace. The ManagedElement is in the implementation namespace.

For implementations conforming to SMI-S 1.0.x or 1.1.x, all RegisteredProfile/RegisteredSubprofile instances associated via SubProfileRequiresProfile shall have the version number of the SMI-S standard for the RegisteredVersion property. For implementations conforming to SMI-S 1.2.x or later, the RegisteredVersion property of associated RegisteredProfile/RegisteredSubprofile instances may have different values; these values shall be the same as those published in the *Registered Name and Version* subclause of the profiles. The version of the standard shall be expressed using the SMI-S RegisteredProfile (see 41.3.3 "The SMI-S Registered Profile").

RegisteredProfile instances are required for all SMI-S profiles, including those named as Subprofiles or Packages.

41.3.1 ElementConformsToProfile Association

In addition, the ElementConformsToProfile association ties the RegisteredProfile for SMI-S autonomous profiles to scoping managed elements (typically ComputerSystems); these are the “top-level” objects defined in SMI-S autonomous profiles. Implementations shall support an ElementConformsToProfile association from at least one RegisteredProfile instance to the scoping instance.

A single ManagedElement may have zero or more ElementConformsToProfile associations to RegisteredProfiles. Regardless of the number of associated RegisteredProfiles, the ManagedElement represents one set of resources. So for example, consider a ManagedElement that is a System that supports both the Array and Storage Virtualizer profiles. If one asks for the total amount of mapped capacity, the answer applies to both Array and Virtualizer and is not additive. See B.6 "Rules for Combining (Autonomous) Profiles" in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5*.

41.3.2 Associations between Autonomous and Component Profile

The DMTF Profile Registration Profile requires the RegisteredProfile instances representing a profile and its supported profiles be associated via ReferencedProfile (which may be subclassed as SubProfileRequiresProfile). SMI-S has the additional requirement, that all supported profiles (whether supported directly or indirectly), are associated directly to the “top-level” autonomous profile.

For example, as shown in Figure 55, the Array Profile supports the Disk Sparing Subprofile which supports the Job Control. SMI-S requires both of these component profiles to be directly attached to the Array Profile instance, even though Job Control is actually a component profile of Disk Sparing. DMTF Profile Registration Profile also requires a ReferencedProfile association between the RegisteredProfiles for Disk Sparing and Job Control.

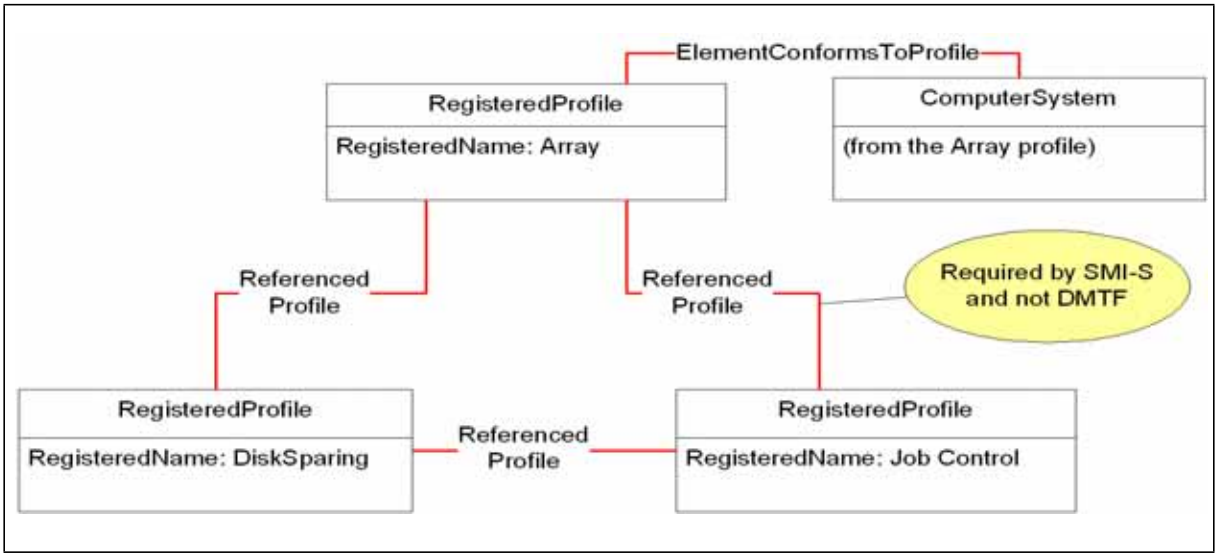

Figure 55 - Associations between RegisteredProfile instances

Each RegisteredProfile instance referenced by ElementConformsToProfile may have a set of supported profiles with RegisteredProfile instances associated using ReferencedProfile or SubProfileRequiresProfile. Typically the RegisteredProfile associated via ElementConformsToProfile is for an autonomous profile and the supported profiles are component profiles. If there are multiple ElementConformsToProfile associations between a single RegisteredProfile instance and multiple domain instances, the referenced domain implementations shall support all the profiles supported by the RegisteredProfile.

41.3.3 The SMI-S Registered Profile

SMI-S conformant implementations shall provide a technique that allows clients to determine which standard the implementation conforms to. This requirement is different for RegisteredProfile instances representing an profile from SMI-S versions before 1.2.0, which are required to use the standard’s version number (e.g., 1.0.3 or 1.1.0) in the RegisteredVersion property of each RegisteredProfile (or RegisteredSubprofile) instance.

Each RegisteredProfile instance representing a profile from SMI-S version 1.2.0 or later shall also be associated to a RegisteredProfile instance holding the SMI-S version number, as shown in Figure 56. The version number (RegisteredVersion) of SMI-S profiles may or may not be the same as the version number of the SMI-S Registered Profile. The RegisteredProfile instances are associated using ElementConformsToProfile where the RegisteredProfile representing SMI storage profiles (e.g., Array, Switch) is referenced from the ManagedElement role of the association. Figure 56 depicts the RegisteredProfile representing the SMI-S standard on the left, and RegisteredProfiles representing autonomous and component storage profiles in the middle.

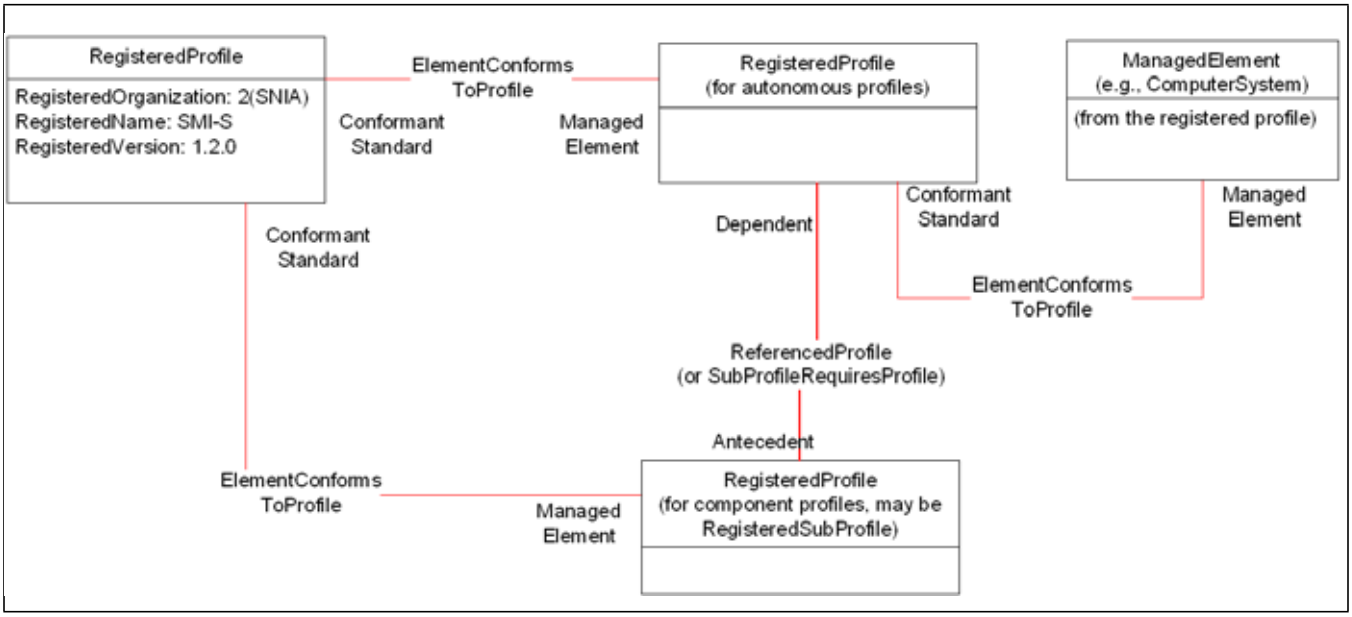

Figure 56 - Model for SMI-S Registered Profile

SMI-S class diagrams generally do not include the names of roles on associations. The requirements of roles (ConformantStandard and ManagedElement) of ElementConformsToProfile seemed critical to understand this model, so they are added to Figure 56. The role names are under the ends of the ElementConformsToProfile lines.

41.3.3.1 Provider Versions

Each RegisteredProfile and RegisteredSubprofile instance (from the Profile Registration Profile, except the SMI-S Profile) shall be associated to one (or more) SoftwareIdentity instances containing information about the software packages required to deploy the instrumentation (including providers). These are associated using ElementSoftwareIdentity. SoftwareIdentity instance may optionally be associated to

Product instances representing a software product. The model for Provider Versions is depicted in Figure 57.

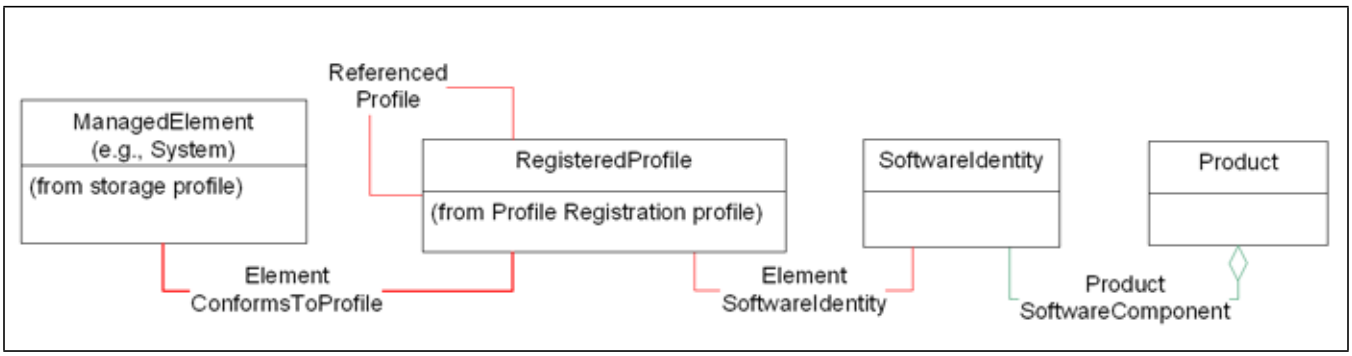

Figure 57 - Model for Provider Versions

41.3.3.2 Abstract Profile and Profile Registration

When profiles are defined for specialization, they may be defined as abstract and include this text in the Synopsis subclause:

> This abstract profile specification shall not be directly implemented; implementations shall be based on a profile specification that specializes the requirements of this profile.

RegisteredProfile instances shall not be instantiated for abstract profiles. Information about abstract profiles shall not be included in the SLP template.

41.3.3.3 Indications

The Profile Registration Profile supports optional indications for the creation and deletion of RegisteredProfile instances. These indications apply to autonomous and component profiles. The indications filters are defined in terms of the RegisteredProfile class and should be triggered for RegisteredSubprofile as well asRegisteredProfile instances. Indications should also be triggered for creation or deletion of the SMI-S profile.

These indications will sent to subscribers when profiles are added and removed from the CIM server. They might be added due to updates, new functionality, or enabling licensed features. If an implementation supports dynamic creation or removal of profiles, then these indications should be supported.

41.3.4 Health and Fault Management Consideration

None

41.3.5 Cascading Considerations

None

41.4 Methods

None

41.5 Use Cases

41.5.1 Using the CIM Server Model to Determine SNIA Profiles Supported

All SNIA profiles require the implementation of the Server Profile as part of the CIM Server. This allows a client to determine which SNIA profiles are supported by the a proxy, embedded or general purpose SMI-S Server. SMI-S clients can use SLP to search for services that support SNIA profiles. Indeed, a client

may restrict its search to specific types of SNIA profiles. The client would get a response for each CIM Server service that supports a SNIA profile. From the responses, the client should use the “service-id” to determine the unique CIM Servers it is dealing with.

For each CIM Server, the client can determine the types of entities supported by inspecting the RegisteredProfilesSupported attribute returned for the SLP entries. This identifies the types of entities (e.g., devices) supported by the CIM Server.

The client may determine more detail on the support for the profiles by going to the service advertised for the CIM Server and inspecting the RegisteredProfiles maintained in the server profile. This would be done by enumerating RegisteredProfiles and RegisteredSubprofiles within the interop namespace. By inspection of the actual profile instances, the client can determine the SNIA version (RegisteredVersion) of profile, associated namespaces and associated managed elements (e.g., systems).

From the RegisteredProfiles within the namespace of the ObjectManager, a client can determine other supported profiles by following the ReferencedProfile association (or its subclass SubProfileRequiresProfile). This returns a set of RegisteredProfile (or RegisteredSubProfile) instances that represent profiles supported by the specific autonomous profile instance. See individual profile descriptions in this specification for the specific list of “supported profiles”. For a given profile instance there may be zero, one or many supported profiles.

41.5.2 Recipe Assumptions

For discovery recipes, the following are assumed:

a) A top-level object (class instance) exists for each profile, and

b) the client knows what the top level object is.

The top-level object for each of the SMI-S profiles are:

- ComputerSystem: For Array, Storage (Media) Libraries, Virtualizers, Switches, and HBAs. This is the top-level ComputerSystem instance for the profile (not the component ComputerSystem or the member ComputerSystem);
- AdminDomain: For Fabric and HostDiscoveredResources;
- ObjectManager: For Server.

The top-level object (class instance) is associated to the RegisteredProfile instance for the profile via the ElementConformsToProfile association.

NOTE Other ManagedElement instances may be associated to the RegisteredProfile, but the meaning and behavior of such associations are not defined by SMI-S and are not mandatory.

41.5.3 Find Servers Supporting a Given Profile

```
// DESCRIPTION
// A management application wishes to find all CIM Servers on a
// particular subnet that support one or more SMI-S profiles.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume CIM Servers have advertised their services (SrvReg)
// 2.Assume there may (or may not) be Directory Agents in the subnet
// 3.Assume no security on SLP discovery
// 4.#DirectoryList[] is an array of directory URLs
// 5.#ServiceList[] is an array of service agent URLs
// 6.#DirectoryEntries [] is an array of directory entry Structures.
```

```
//   The structure matches the wbem SLP Template (see Clause 5,
//   section 10).

// Step 1:  Set the Previous Responders List to the Null String.
#PRList = ""

// Step 2: Multicast a Service Request for a Directory Server Service.
//   This is to find Directory Agents in the subnet.
//
SrvRqst (
     #PRList,        // The Previous Responders list
     "service:directory-agent" // Service type
     "DEFAULT",          // The scope
     NULL,             // The predicate
     NULL)             // SLP SPI (security token)

// Step 3: Listen for Response from Directory Agent(s)
#DirectoryList[] = DAAdvert (
     BootTimestamp, // Time of last reboot of DA
     URL,      // The URL of the DA
     ScopeList,// The scopes supported by the DA
     AttrList,// The DA Attributes
     SLP SPI List,// SLP SPI (SPIs the DA can verify)
     Authentication Block)
// Iterate on Steps 2 & 3, until a response has been received or the client has
// reached a UA configured CONFIG_RETRY_MAX seconds.  If no DA if found,
// proceed to step 4.  If a DA is found, proceed to step 7.

// Step 4: Set the Previous Responders List to the Null String.
#SAPRList = ""

// Step 5: Multicast a Service Request for Service Agent Services. This
//   is to find Service Agents in the subnet that are not advertised
//   in a Directory.

SrvRqst (
     #SAPRList,        // The Previous Responders list
     "service:service-agent" // Service type
     "DEFAULT",          // The scope
     "(Service-type=WBEM)",        // The predicate
     NULL)             // SLP SPI (security token)

// Step 6: Listen for Response from Service Agent(s)
#SAList[] = SAAdvert (
     URL,      // The URL of the SA
     ScopeList,// The scopes supported by the SA
     AttrList,// The SA Attributes
```

```
     Authentication Block)
// Iterate on Steps 5 & 6, until a response has been received or the client has
// reached a UA configured CONFIG_RETRY_MAX seconds.  If no SA if found,
// Then record an error.   There are NO WBEM SAs.  Otherwise proceed to
// Step 8.

//Step 7: Unicast a Service Request to each of the DAs specifying
//   a query predicate to select CIM Servers that support SNIA profiles
//   and listen for responses.
for #j in #DirectoryList[]
{
     SrvRqst (
        #PRList,        // The Previous Responders list
        “service:wbem”,    // Service type
        “DEFAULT”,          // The scope
        RegisteredProfilesSupported=”SNIA:*” // The predicate
        NULL)             // SLP SPI (security token)

     #ServiceList [#j] = SrvRply  (
        Count,       // count of URLs
        URL for each SA returned)
}
Go to Step 9.

//Step 8: Unicast a Service Request to each of the SAs specifying
//   a query predicate to select CIM Servers that support SNIA profiles
//   and listen for responses.
for #j in #SAList[]
{
     SrvRqst (
        #PRList,        // The Previous Responders list
        “service:wbem”,    // Service type
        “DEFAULT”,          // The scope
        RegisteredProfilesSupported=”SNIA:*”, // The predicate
        NULL)             // SLP SPI (security token)

     #ServiceList [#j] = SrvRply  (
        Count,       // count of URLs
        URL for each SA returned)
}


// Step 9: Next retrieve the attributes of each advertisement
For #i in #ServiceList[]  // for each url in list
{
     AttrRqst (
        #PRList,       // The Previous Responders list
```

```
        #ServiceList[#i],// a url from #ServiceList[]
        “DEFAULT”, // The scope
        NULL,  // Tag list.  NULL means return all attributes
        NULL)  // SLP SPI (security token)
     #DirectoryEntries [#i] = AttrRply (attr-list)
}

// Step 10: Correlate responses to the Service Request on unique
//   “service-id” to determine unique CIM Servers. The client will get
//   multiple responses (one for each access point) for each CIM
//   Server. At this point, the client has a list of CIM Servers that
//   claim to support SNIA profiles.
```

41.5.4 Enumerate Profiles Supported by a Given CIM Server

```
// DESCRIPTION
// A management application wishes to determine the Profiles supported by
// a particular CIM Server.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client only wants to know the “top level” profiles
//   supported by the CIM Server
// 2.Assume the client has used SLP to find the CIM Servers and has a
//   #DirectoryEntries [] structure
// 3.This recipe describes the operations for one of the entries in
//   the #DirectoryEntries [] structure.
// 4.    Assume the index into #DirectoryEntries[] for the CIM Server of
//        interest is #i.

// Step 1: Get the server url for the CIM Server.
#ServerName = #DirectoryEntries[#i].service-id

// Step 2: Get the Interop Namespace for the CIM Server.
#Inamespace = #DirectoryEntries[#i].InteropSchemaNamespace[1]

// Step 3:  Establish a connection to the CIM Server with
// #INameSpace. Note that the WBEM operations throughout the remainder
// of this recipe are performed with this client handle.
<Make client connection to this server using the interop namespace>

// Step 4:  Get the names of all the RegisteredProfiles in the
// Interop Namespace
#ProfileName[] = EnumerateInstances(“CIM_RegisteredProfile”,
TRUE, TRUE, FALSE, FALSE,
[“RegisteredName”])


// Step 5: Determine which RegisteredProfiles are autonomous.
```

```
// Subprofiles (aka component profiles) are associated to autonomous
// profiles via SubProfileRequiresProfile or its superclass,
// ReferencedProfile.  The autonomous profile is refered to
// as the ‘referencing profile’ and the component/sub profile
// is referred the referenced profile. There may be more than
// two tiers, so profile may be both referenced and referencing.
// In practice, component or sub profiles would only be registered
// when their referencing autonomous profile(s) are registered, so
// any profile not referenced by another profile is autonomous.

#k = 0;
for #i in #ProfileName[i] { // walk all profiles
    $ReferencingProfiles->[] = Associators(#ProfileName[i]->,
    CIM_ReferencedProfile”, “CIM_RegisteredProfile”, “Dependent”,
    “Antecedent”, FALSE, FALSE, NULL);
    if ($ReferencingProfiles[] != null && $ReferencingProfiles[].length > 0) {
      // if the profile is not referenced by another profile,
      // add it to the list of autonomous profiles
      #Autonomous[#k+1]=#ProfileName[#i]
    }
}
// #Autonomous[] now holds the autonomous RegisteredProfiles
```

41.5.5 Identify the ManagedElement Defined by a Profile

```
// DESCRIPTION
// A management application wishes to determine the ManagedElement that
// is defined by a particular Profile.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client has located the profile and has its object path
//   ($RegisteredProfile->)

// Step 1: Determine the ManagedElement (System) by traversing the
//    ElementConformsToProfile association from the RegisteredProfile
//    that is the top level Profile that applies to the System
$ManagedElement->[] = AssociatorNames (
     $RegisteredProfile->,
     “CIM_ElementConformsToProfile”,
     “CIM_System”,
     NULL,
     NULL)

// Step 2: The object name of more than one System may be contained
// in the array returned. Examine the contents of $ManagedElement[]
// and save the name of the System of interest as $Name.

// NOTE: “Top level” object for each profile will be returned.
```

```
// To accommodate other potential ManagedElements, then it may
// be necessary need to throw out the ones that are not top level objects.

// NOTE: The object path for the ManagedElement may be in a Namespace
//   that is different than the Interop Namespace. As a result, if the
//   client wishes to actually access the ManagedElement, the client
//   may get the namespace from the REF to the element:
#NameSpace=$Name.getNameSpace()
```

41.5.6 Determine the SNIA Version of a Profile

```
// DESCRIPTION
// A management application wishes to determine the SNIA version
// that a particular Profile supports.

//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client only wants to know version information
//   for a SNIA profile
// 2.Assume the client has already found the profile and has the
//   $RegisteredProfile-> reference


// Step 1: Get the Instance of the Profile name.
$Profile = GetInstance($RegisteredProfile->)

// Step 2: Look for an associated RegisteredProfile representing the
// SMI-S specification. This usage of RegisteredProfile was added in
// SMI-S 1.2.0, if none are found, then assume the implementation
// supports SMI-S 1.0.x or 1.1.x where the version of the profile
// matched the version of the specification. The use of ManagedElement
// and ConfomantStandard as the Role and ResultRoles asure that the
// returned list is restricted to RegisteredProfiles for SMI-S spec and
// does not include domain elements.
$SpecRegisteredProfiles->[] = Associators (
  $RegisteredProfile->,
  “CIM_ElementConformsToProfile”,
  “CIM_RegisteredProfile”,
  ManagedElement,
  ConformantStandard,
  false,
  false,
  [“RegisteredVersion”])

if  ($SpecRegisteredProfiles[] == null ||
        $SpecRegsisteredProfiles[].length == 0) {
  // no RegisteredProfile for specs were found; assume the
  // version of the profile is the spec version.
```

```
  #SNIAVersion = $Profile.RegisteredVersion
} else {
  // At least one $SpecRegisteredProfile was returned; an implmentation may
  // conform to multiple spec versions
  <Sort $SpecRegisteredProfile[] in reversed order of VersionNumbers>
  // The most recent supported SMI-S version is in element 0
  #SNIAVersion = $SpecRegisteredProfiles[0].RegisteredVersion
}
```

41.5.7 Find all Profiles on a Server

```
// DESCRIPTION
// A management application wishes to list all the SNIA profiles and
// their related profiles for a specific CIM Server.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client has already discovered the CIM Servers that
//   support SNIA profiles

// Step 1:  Get the names of all the RegisteredProfiles and their names
// in the Interop Namespace
$ProfileName[] = EnumerateInstances(“CIM_RegisteredProfile”
                            true, true, false, false, {“RegisteredName”})

// Step 2:  Get the ObjectName for the Profiles
for #i in #ProfileName[] {
    $Profile->[#i]=$Name.getObjectPath(#ProfileName[#i])
}

// Step 3: Get the (sub)profiles associated to the profiles.
// Since ReferencedProfile is the superclass for
// SubProfileRequiresProfile and RegisteredProfile is the
// supclass for RegsisteredSubProfile, this algorithm finds
// subprofiles and componement profiles referenced by a
// profile.
for #i in $ProfileName[]
{
    $Subprofile[] = Associators(
                            $ProfileName[#j].getObjectPath(),
                            “CIM_ReferencedProfile”,
                            “CIM_RegisteredProfile”,
                            NULL, NULL, false, false, NULL)
}
```

41.6 CIM Elements

Table 403 describes the CIM elements for Profile Registration.

Table 403 - CIM Elements for Profile Registration

Element Name	Requirement	Description
41.6.1 CIM_ElementConformsToProfile (Associates Domain object (e.g. System) to RegisteredProfile)	Mandatory	Ties managed elements (e.g., Systems representing devices) to the registered profile that applies.
41.6.2 CIM_ElementConformsToProfile (Associates RegisteredProfiles for SMI-S and domain profiles)	Mandatory	Associates RegisteredProfiles for SMI-S and domain profiles.
41.6.3 CIM_ElementSoftwareIdentity (Profile and SW identity)	Mandatory	Associates a domain RegisteredProfile and SoftwareIdentity instances.
41.6.4 CIM_ElementSoftwareIdentity (Subprofile and SW identity)	Conditional	Conditional requirement: Support for instances of RegisteredSubprofile. Associates the subprofile and SoftwareIdentity instances.
41.6.5 CIM_Product	Optional	Represents a software product aggregating SoftwareIdentity instances with provider versions.
41.6.6 CIM_ProductSoftwareComponent	Optional	Associates Product and SoftwareIdentity.
41.6.7 CIM_ReferencedProfile	Optional	Associates referenced profiles using the DMTF Profile Registration profile.
41.6.8 CIM_RegisteredProfile (Domain Registered Profile)	Mandatory	An object representing a domain (e.g. Array or Switch) profile.
41.6.9 CIM_RegisteredProfile (The SMI-S Registered Profile)	Mandatory	A registered profile that provides the version of the SMI-S standard.
41.6.10 CIM_RegisteredSubProfile	Optional	Specialization of RegisteredProfile for legacy SMI-S subprofiles.
41.6.11 CIM_SoftwareIdentity	Mandatory	A representation of some bundle of providers and supporting software that shares a version number.
41.6.12 CIM_SubProfileRequiresProfile	Optional	Specialization of ReferencedProfile referencing a SubProfile.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_RegisteredProfile	Optional	Creation of a registered profile instance. See 41.3.3.3 Indications.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_RegisteredProfile	Optional	Deletion of a registered profile instance. See 41.3.3.3 Indications.

41.6.1 CIM_ElementConformsToProfile (Associates Domain object (e.g. System) to RegisteredProfile)

The CIM_ElementConformsToProfile ties managed elements (e.g., Systems representing devices) to the registered profile that applies.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 404 describes class CIM_ElementConformsToProfile (Associates Domain object (e.g. System) to RegisteredProfile).

Table 404 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (Associates Domain object (e.g. System) to RegisteredProfile)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A element implementing a profile (e.g., top-level system).
ConformantStandard		Mandatory	RegisteredProfile instance describing the domain profile.

41.6.2 CIM_ElementConformsToProfile (Associates RegisteredProfiles for SMI-S and domain profiles)

Associates RegisteredProfiles for SMI-S and domain profiles.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 405 describes class CIM_ElementConformsToProfile (Associates RegisteredProfiles for SMI-S and domain profiles).

Table 405 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (Associates RegisteredProfiles for SMI-S and domain profiles)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The RegisteredProfile representing the domain profile.
ConformantStandard		Mandatory	The SMI-S RegisteredProfile.

41.6.3 CIM_ElementSoftwareIdentity (Profile and SW identity)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 406 describes class CIM_ElementSoftwareIdentity (Profile and SW identity).

Table 406 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (Profile and SW identity)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to SoftwareIdentity.
Dependent		Mandatory	Reference to domain RegisteredProfile.

41.6.4 CIM_ElementSoftwareIdentity (Subprofile and SW identity)

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Support for instances of RegisteredSubprofile.

Table 407 describes class CIM_ElementSoftwareIdentity (Subprofile and SW identity).

Table 407 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (Subprofile and SW identity)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to Software Identity.
Dependent		Mandatory	Reference to RegisteredSubProfile.

41.6.5 CIM_Product

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 408 describes class CIM_Product.

Table 408 - SMI Referenced Properties/Methods for CIM_Product

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	Commonly used product name.
IdentifyingNumber		Mandatory	Software serial number.
Vendor		Mandatory	Product supplier.
Version		Mandatory	Product version information.

41.6.6 CIM_ProductSoftwareComponent

Associates Product and SoftwareIdentity.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 409 describes class CIM_ProductSoftwareComponent.

Table 409 - SMI Referenced Properties/Methods for CIM_ProductSoftwareComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to Product.
PartComponent		Mandatory	Reference to SoftwareIdentity.

41.6.7 CIM_ReferencedProfile

Associates referenced profiles using the DMTF Profile Registration profile.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 410 describes class CIM_ReferencedProfile.

Table 410 - SMI Referenced Properties/Methods for CIM_ReferencedProfile

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

41.6.8 CIM_RegisteredProfile (Domain Registered Profile)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 411 describes class CIM_RegisteredProfile (Domain Registered Profile).

Table 411 - SMI Referenced Properties/Methods for CIM_RegisteredProfile (Domain Registered Profile)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	This is a unique value for the profile instance.
RegisteredOrganization		Mandatory	This is the official name of the organization that created the Profile. For SMI-S profiles, this would be SNIA. For DMTF profiles, this would be DMTF.
OtherRegisteredOrganization		Conditional	Conditional requirement: CIM_RegisteredProfile requires the OtherRegisteredOrganization property be populated if the RegisteredOrganization property has a value of 1 (\Other\).'Mandatory if RegisteredOrganization is 1 ('Other').
RegisteredName		Mandatory	This is the name assigned by the organization that created the profile.
RegisteredVersion		Mandatory	This is the version number assigned by the organization that defined the Profile.
AdvertiseTypes	N	Mandatory	Defines the advertisement of this profile. If the property is null then no advertisement is defined. A value of 1 is used to indicate 'other' and a 3 is used to indicate 'SLP'.
AdvertiseTypeDescriptions		Conditional	Conditional requirement: CIM_RegisteredProfile requires the AdvertiseTypeDescriptions property be populated if the AdvertiseTypes property has a value of 1 (\Other\).'This shall not be NULL if 1 ('Other') is identified in AdvertiseType.

41.6.9 CIM_RegisteredProfile (The SMI-S Registered Profile)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 412 describes class CIM_RegisteredProfile (The SMI-S Registered Profile).

Table 412 - SMI Referenced Properties/Methods for CIM_RegisteredProfile (The SMI-S Registered Profile)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	A unique value for the profile instance.
RegisteredOrganization		Mandatory	Shall be 11 (SNIA).
RegisteredName		Mandatory	Shall be 'SMI-S'.
RegisteredVersion		Mandatory	The version number of the SMI specification the associated profiles conform to.
AdvertiseTypes		Mandatory	Should be 2 (Not Advertised) or 3 (SLP). 2 is recommended to avoid increasing size of SLP template.
AdvertiseTypeDescriptions		Conditional	Conditional requirement: CIM_RegisteredProfile requires the AdvertiseTypeDescriptions property be populated if the AdvertiseTypes property has a value of 1 (\Other\).'This shall not be NULL if 'Other' is identified in AdvertiseType.

41.6.10CIM_RegisteredSubProfile

Specialization of RegisteredProfile for legacy SMI-S subprofiles.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 413 describes class CIM_RegisteredSubProfile.

Table 413 - SMI Referenced Properties/Methods for CIM_RegisteredSubProfile

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	This is a unique value for the subprofile instance.
RegisteredOrganization		Mandatory	This is the official name of the organization that created the subprofile. For SMI-S profiles, this would be 11 ('SNIA').
OtherRegisteredOrganization		Conditional	Conditional requirement: CIM_RegisteredProfile requires the OtherRegisteredOrganization property be populated if the RegisteredOrganization property has a value of 1 (\Other\).'Mandatory if RegisteredOrganization is 1 ('Other').
RegisteredName		Mandatory	This is the name assigned by the organization that created the subprofile.
RegisteredVersion		Mandatory	This is the version number assigned by the organization that defined the subprofile.

Table 413 - SMI Referenced Properties/Methods for CIM_RegisteredSubProfile

Properties	Flags	Requirement	Description & Notes
AdvertiseTypes	N	Mandatory	Should be 2 (Not Advertised) for subprofiles.
AdvertiseTypeDescriptions		Conditional	Conditional requirement: CIM_RegisteredProfile requires the AdvertiseTypeDescriptions property be populated if the AdvertiseTypes property has a value of 1 (\Other\').'This shall not be NULL if 1 ('Other') is identified in AdvertiseType.

41.6.11CIM_SoftwareIdentity

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 414 describes class CIM_SoftwareIdentity.

Table 414 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	A user-friendly name for the instrumentation software.
InstanceID		Mandatory	
VersionString		Mandatory	
Manufacturer		Mandatory	The name of the company associated with the instrumentation software.
Classifications		Mandatory	
ClassificationDescriptions		Conditional	Conditional requirement: CIM_SoftwareIdentity requires the ClassificationDescriptions property be populated if the Classifications property has a value of 1 (\Other\').'Mandatory if Classifications is set to 1 ('Other').

41.6.12CIM_SubProfileRequiresProfile

Specialization of ReferencedProfile referencing a SubProfile.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 415 describes class CIM_SubProfileRequiresProfile.

Table 415 - SMI Referenced Properties/Methods for CIM_SubProfileRequiresProfile

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to a RegisteredSubProfile.
Antecedent		Mandatory	Reference to a RegisteredProfile.

STABLE

DEPRECATED

42 Indication Profile

This profile is being deprecated in favor of the SNIA specialization of the DMTF Indications Profile version 1.2.0 (See 50 Indications Profile).

42.1 Description

The Indication Profile is a component profile of the Server Profile. It may also be a component profile of any other profile (e.g., Array Profile).

Refer to individual profile definitions to see whether or not the Indication Profile is mandatory. Figure 58 illustrates the structure of profiles, the Indication Profile and indication instances implied by an Array's support for the Indication Profile.

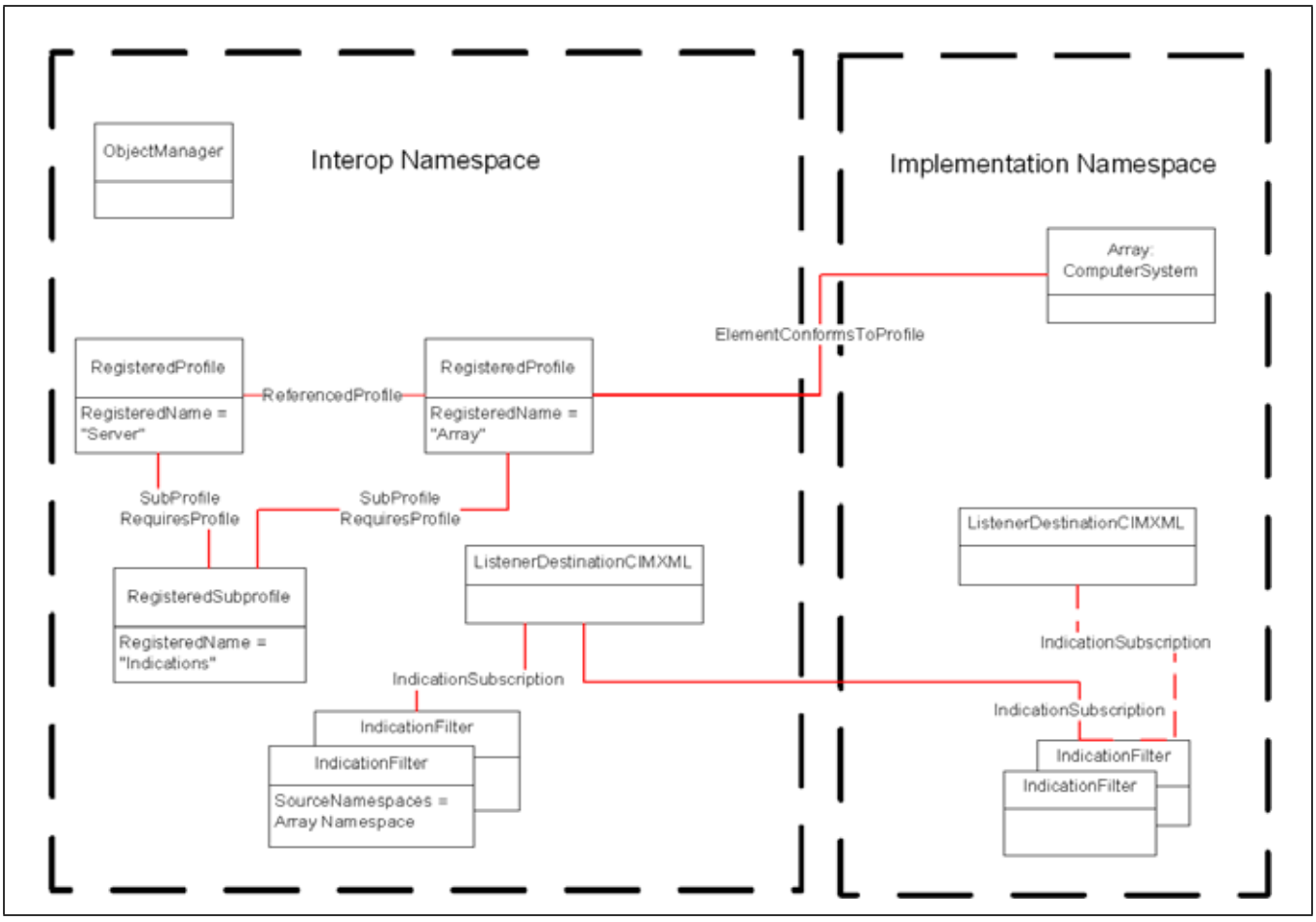

Figure 58 - Indication Profile and Namespaces

Indication filters are defined in the context of the namespace in which they are implemented. In Figure 58, this is shown as the implementation namespace. The indication filters shall be defined in two places: The Interop namespace and the namespace where the indications are intended to originate.

DEPRECATED

For the Filters defined in the InteropNamespace, the SourceNamespace property shall be filled out to indicate the implementation namespace where the indications are to originate. For the IndicationFilters defined in the implementation namespace, this property may be null (indicating the indications originate in the implementation namespace of the array).

DEPRECATED

EXPERIMENTAL

The SourceNamespaces property of an IndicationFilter indicates the list of namespaces from which indications for the Filter query are to originate (e.g., the implementation namespace for the device). For any filter, the SourceNamespaces property should be populated. For the Filters in the InteropNamespace, the Namespaces properties would indicate the implementation namespaces that support the indications defined by the filter. For filters defined in the implementation Namespace, the filters should contain the Implementation Namespace (as one of its source namespaces).

EXPERIMENTAL

The IndicationFilters may be populated by the Provider (or they may be created by a client). In either case, they are created in both the Interop Namespace and the implementation namespace of the array. The ListenerDestinationCIMXML class shall be in the Interop Namespace and may also be in the implementation namespace. And there would be two instantiations of the IndicationSubscription association: one in the Interop Namespace and one in the implementation namespace.

SMI-S profile implementations that support indications shall support either the use of “predefined” indications filters, “client defined” indication filters or both. In the case of an implementation that supports “predefined” filters, the SMI-S Server would populate its model with indication filters that it supports. SMI-S Clients would select the indication filters to which they wish to subscribe from the list supplied by the SMI-S Server (enumeration of IndicationFilters in the appropriate namespace). In the case of an implementation that supports “client defined” filters, the SMI-S Server shall support filter creation (and deletion) by clients and it shall support creation of at least the filters defined by the profile.

Creation of an IndicationFilter will cause the creation of instances in both the InteropNamespace and the implementation namespace. ListenerDestinationCIMXML instances should be created in the InteropNamespace, but may also be created in the implementation namespace (for SMI-S 1.0.x compatibility reasons). If a ListenerDestinationCIMXML instance is created in the implementation namespace, a duplicate instance will be instantiated in the InteropNamespace. However, if a ListenerDestinationCIMXML is created in the InteropNamespace, it may not be created in the implementation namespace.

NOTE An implementation may support both “predefined” filters and “Client Defined” filters.

SMI-S Clients would subscribe to the indications for the events to which they wish to be notified. They would also supply an address (Indication listener) in which the indications are to be sent. SMI-S Clients shall use the subclass ListenerDestinationCIMXML when creating subscriptions.

In any given implementation Indication Filters are scoped by NameSpace. That is, a subscription to the change of operational status for a ComputerSystem can result in reporting of any change of operational status for ANY ComputerSystem managed within a Namespace. A client should inspect any indication to see if it is for an element that it manages.

A vendor implementation may support additional indication filters beyond those identified in a profile specification, but all the filters identified in SMI-S shall be supported as specified by the profile.

NOTE Indication filters may correspond to optional or conditional features in a profile. When a provider supports an optional or conditional feature, the indications corresponding to the feature may be conditional on the feature. This means that the provider shall supply the filters or shall allow a client to define the filters. But optional indications that correspond to the feature need not be supported. Indications corresponding to the filter shall be generated by the provider when a corresponding event occurs. On the other hand, if a profile implementation does not support a component profile that defines mandatory indications, then the profile implementation does not need to support those indications.

EXPERIMENTAL

42.1.1 IndicationFilter Names

IndicationFilters have a Name property. The value of the Name property in instances defined by referencing profiles should be formatted as defined by the following ABNF rule:

OrgID ":" RegisteredName ":" UniqueID

Where OrgID identify the business entity owning the referencing profile. OrgID shall include a copyrighted, trademarked, or otherwise unique name that is owned by that business entity or that is a registered ID assigned to that business entity by a recognized global authority. In addition, to ensure uniqueness, OrgID shall not contain a colon (:).

For referencing profiles owned by the SNIA, OrgID should match "SNIA" for IndicationFilters defined by the standard. For client defined IndicationFilters, the OrgID should identify the client (application) organization.

The RegisteredName should be the registered name of the referencing profile, as defined by the value of its CIM_RegisteredProfile.RegisteredName property.

The UniqueID should uniquely identify the instance within the referencing profile.

EXPERIMENTAL

42.1.2 Basic Indication Classes and Association

Figure 59 illustrates the classes used in support of indications. Any given profile implementation may not include all of these classes. But they would at least support IndicationFilters (possibly predefined),

ListenerDestinationsCIMXML and IndicationSubscriptions. The actual types of indications supported can vary by profile (see 42.8 "CIM Elements" to determine the types of indications supported).

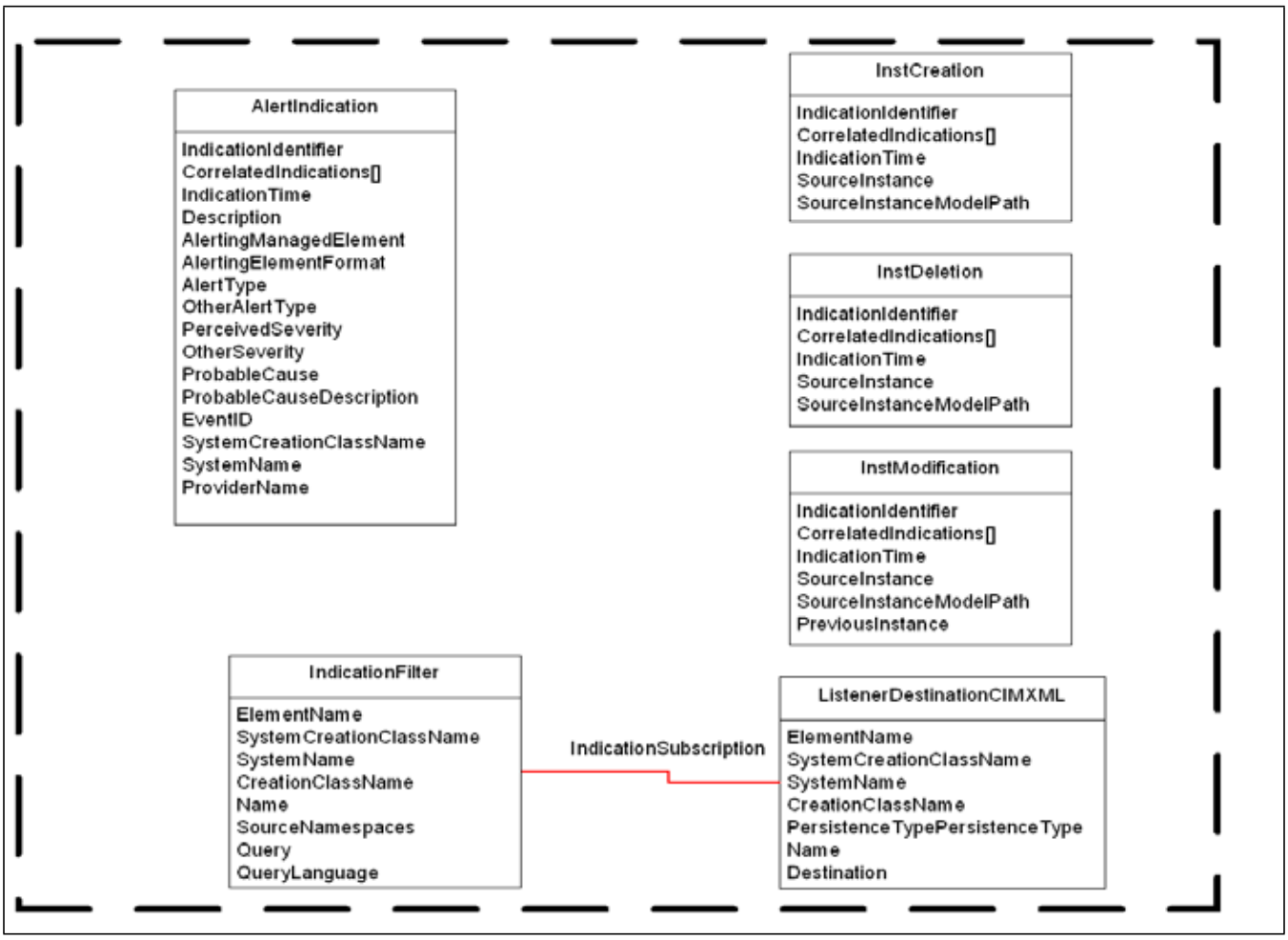

Figure 59 - Indication Profile Instance Diagram

Clients request indications to be sent to them by subscribing to the indication filters. Subscriptions are stored in the SMI-S Server. A Subscription is expressed by the creation of a IndicationSubscription association instance that references an IndicationFilter (a filter) instance, and an ListenerDestination (for the handler of the indications) instance. A Filter contains the query that selects an indication class or classes.

SMI-S Servers that support SMI-S profiles that provide CIM indications support shall populate their models with the filters as defined by the profile(s) or allow clients to create the filters that are defined for the profile(s). Additional filters may also be created by indication consumers (e.g., SMI-S Clients), but this is not mandatory with SMI-S. The client would create these filters using CreateInstance intrinsic method.

The query property of the IndicationFilter is a string that specifies which indications are to be delivered to the client. There is also a query language property that defines the language of the query string. Example query strings are:

“SELECT * FROM AlertIndication”

“SELECT * FROM InstModification WHERE SourceInstance ISA ComputerSystem”

AlertIndication and InstModification are types of indications. The first query says to deliver all alert type indications to the client, and the second query says to deliver all instance modification indications to the client, where the instance being modified is a ComputerSystem (or any subclass thereof).

See Annex C (normative) Indication Filter Strings in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* for information on the use of indications filter strings.

A ListenerDestination specifies the means of delivering indications to the client. The subclass ListenerDestinationCIMXML provides for XML encoded indications to be sent to a specific URL, which is specified as a property of that class.

When a client receives an indication, it will receive some information with the indication, and then it may need to do additional queries to determine all of the consequences of the event.

NOTE To avoid multiple calls to get additional data for an indication, profile designers (or clients, for client defined filters) should consider more elaborate Queries for Filters to return more information.

The instances of AlertIndications, InstCreation, InstDeletion and InstModification are temporary. They exist until they are delivered to the subscribing clients. The ListenerDestinationCIMXML, IndicationFilter and IndicationSubscription instance are permanent. That is, they persist until action is taken by client to delete them.

One final note on the indications supported. InstModification may or may not require the PreviousInstance property. A profile may be designed to require it or not. If the SMI-S profile defines an IndicationFilter on InstModification it shall specify whether or not PreviousInstance is required. It may always be recommended. If a profile defines PreviousInstance as optional, then an implementation may provide a previous instance (or not). However, if the SMI-S profile defines an IndicationFilter on InstModification with PreviousInstance required, then all implementations shall implement the PreviousInstance property.

42.1.3 Life Cycle Indications

A life cycle indication is used to convey changes in the model. It is represented by a subclass of InstIndication. Life cycle indications are concerned only with the creation, modification, or deletion of CIM Instances. The indication is a CIM class whose properties contain copies of CIM Instances that have been created, modified, or deleted (InstCreation, InstDeletion, InstModification). As such, life cycle indications can only report on classes that are supported. Profile designers use life cycle indications as means where clients can monitor for significant changes in a particular data model. The significant changes to the model are a reflection of changes in the managed element the CIM instance(s) represents. An event like component overheat is likely to affected several components. Therefore, in many cases the scope of the event can be observed through the telemetry communicated through life cycle indications.

The mandatory properties of a life cycle indication are:

- IndicationTime - The time and date of creation of the Indication.
- SourceInstance - A copy of the instance that changed to generate the Indication. SourceInstance contains the current values of the properties selected by the Indication Filter's Query.
- SourceInstanceModelPath - The Model Path of the SourceInstance.

In addition, the following properties are recommended, but not mandatory:

- IndicationIdentifier - An identifier for the Indication that may be used for correlated indications.
- CorrelatedIndications - IndicationIdentifiers whose notifications are correlated with this one.

In addition, for InstModification indications, the PreviousInstance property may be provided:

- PreviousInstance - A copy of the 'previous' instance whose change generated the Indication. PreviousInstance contains 'older' values of an instance's properties (as compared to SourceInstance), selected by the IndicationFilter's Query.

42.1.4 AlertIndications

An alert indication is another type of indication but with a different purpose. Alert Indications are used to drawn attention of subscribing client applications to the occurrence of an event. Alert Indication may describe aspects of an event. The event's characteristics may or may not be wholly or partially represented in the data model (as expressed through CIM). An Alert Indication is represented by a subclass of AlertIndication. The alert indication itself is considered a change in the instrumentation's model.

An alert indication describes the category, severity, and event specifics. However, the event specifics may not be understandable by an SMI-S Client. A standard message can convey the event specifics in a manner defined by SMI-S or another related standard. (See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 8 Standard Messages.) The interpretation for the alert indication is either contained within a standard message registry that is referenced by a profile, or defined by the profile to be produced for some reason and identifiable in some manner.

The mandatory properties of an AlertIndication are:

- IndicationTime - The time and date of creation of the Indication.
- AlertingManagedElement - The identifying information of the entity (i.e., the instance) for which this Indication is generated. The property contains the path of an instance, encoded as a string parameter - if the instance is modeled in the CIM Schema. If not a CIM instance, the property contains some identifying string that names the entity for which the Alert is generated.
- AlertingElementFormat - The format of the AlertingManagedElement property is interpretable based upon the value of this property. Values are defined as: “Unknown”, “Other”, “CIMObjectPath”
- AlertType - This is an integer property that is a value map. The values supported are: “Other”, “Communications Alert”, “Quality of Service Alert”, “Processing Error”, “Device Alert”, “Environmental Alert”, “Model Change”, “Security Alert”
- PerceivedSeverity - An enumerated value that describes the severity of the Alert Indication from the notifier's point of view. This is an integer property that is a value map. The values supported are: “Unknown”, “Other”, “Information”, “Degraded/Warning”, “Minor”, “Major”, “Critical”, “Fatal/Non Recoverable”.
- ProbableCause - This is an integer property that is a value map. There are many values that may be set (refer to the MOF for details).
- SystemCreationClassName - The scoping System's CreationClassName for the Provider generating this Indication.
- SystemName - The scoping System's Name for the Provider generating this Indication.

  The SystemName would typically be the same that for a system in the Implementation Namespace (unless the Indication is an indication generated for Server Profile).
- ProviderName - The name of the Provider generating this Indication.

In addition, the following properties are recommended, but not mandatory:

- IndicationIdentifier - An identifier for the Indication that can be used for identification when correlating Indications (see 42.1.6.2 “Indication Identification”).
- CorrelatedIndications[] - IndicationIdentifiers whose notifications are correlated with this one.

- Description - A short description of the Indication.
- OtherAlertType - This property is mandatory if the AlertType is 1 (for “other”).
- OtherSeverity - This property is mandatory if the PerceivedSeverity is 1 (for “other”)
- ProbableCauseDescription - Provides additional information related to the ProbableCause.
- EventID - An instrumentation or provider specific value that describes the underlying \”real-world\” event represented by the Indication.
- OwningEntity - A string that uniquely identifies the entity that owns the definition of the format of the Message. For messages owned by the SNIA, this shall be encoded as ‘SNIA’. However, for SMI-S, not all messages need be owned by SNIA.
- MessageID - A string that uniquely identifies, within the scope of the OwningEntity, the format of the Message.
- Message - The formatted message (including the MessageArguments).
- MessageArguments - An array of strings that contain the dynamic content of the message.

For descriptions of how these properties should be encoded, see the profile for specific alert indications that are supported. For encoding of the OwningEntity, MessageID, Message, and MessageArguments of SNIA messages, see section 8.3 in the *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5*.

42.1.5 Indication Delivery

Acceptance of a subscription, represented by an instance of the IndicationSubscription association between an instance of IndicationFilter and an instance of ListenerDestination, is a contract between the SMI-S Server and SMI-S Client that requires that the SMI-S Server shall produce indications when the conditions described by the associated indication filter are present. The SMI-S Server may not be able to deliver the indication for other reasons like authentication failures or network connectivity failures, but the SMI-S Server shall attempt to deliver the indication.

In some cases, the Client (ListenerDestination) may not be available when an event occurs that requires delivery to the client. In such cases, the CIM Server should attempt delivery to the listener destination 3 times. If the delivery cannot be made within 3 attempts, the indication may be considered delivered.

If the ListenerDestinationCIMXML.PersistenceType is set to "3" (transient), the IndicationSubscription may be deleted after 3 attempts that fail. If the ListenerDestinationCIMXML.PersistenceType is set to "2" (permanent) the IndicationSubscription shall be retained.

42.1.6 Instrumentation Requirements

42.1.6.1 General Instrumentation Considerations

A SMI-S Server may allow a client to create indications filters. If the SMI-S Server does not support this option, then the server shall send a return code indicating a request to create an instance of a filter is unsupported. This allows the provider to inform clients which types of indications the provider supports. For example, a provider that does not support SNMPTrapAlertIndications shall return unsupported for an indications filter create request.

42.1.6.2 Indication Identification

Not defined in this version of the Indication Profile.

42.1.6.3 SMI-S Dedicated Server Considerations

The dedicated server should supply more detailed queries as described in the profile sections.

A standard implementation of indications requires the server to accept client requests to create ListenerDestinations. The dedicated server implementation uses the Instance Manipulation functional group in addition to Basic Read.

42.1.6.4 Additional Indications

Most Indication Filters defined in the “CIM Elements” section of the specification are mandatory. However, a profile may also document additional Indication Filters as optional filters. A client can determine whether or not “additional” indication filters are supported by one of two techniques:

1) Enumerating Predefined Indication Filters – this will return all the indication filters that have been predefined by the provider for the Namespace.

2) CreateInstance of the desired “additional” Indication Filter – if the “additional” indication filter is supported, the CreateInstance will succeed.

42.1.6.5 Support for Query Languages

For versions of the standard prior to 1.3.0, CQL had not been approved as a standard and was treated as recommended and experimental. For those early versions, WQL (also referred to as the SMI-S query Language) was the non-experimental query language.

For versions of the standard starting at 1.3.0, CQL is mandatory for newly defined indication filters; WQL alternatives shall not be defined in the standard.

DEPRECATED

Support for the SMI-S 1.0.x Query Language is being deprecated.

DEPRECATED

See Annex C (normative) Indication Filter Strings in *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* for information on the use of indications filter strings.

42.1.6.6 Timing of Delivery of Indications

There are no standards for how quickly an implementation shall deliver an indication. All reasonable attempts should be made by the implementation to deliver all indications at the CIM Server’s earliest convenience.

There are also no standard guidelines on how long or how many attempts should be made to deliver an indication. As a general guideline an implementation should make at least 3 attempts to deliver an indication before giving up trying to deliver the indication. Similarly, delivery of indications should allow at least 30 seconds to elapse before giving up trying to deliver the indication. The intent is to allow sufficient time to allow any network problems to clear.

42.1.6.7 Handling of Indication Storms

Occasionally an event may occur that causes many indication filters to evaluate to true (an trigger many indications). This situation is referred to as an “indication storm.” These can be very expensive and degrade the performance of the environment. To contain the impact of this an implementation can employ the following technique:

- use the RepeatNotificationPolicy (and related properties) of the IndicationSubscription.

42.1.6.7.1 Use of RepeatNotificationPolicy

The RepeatNotificationPolicy property defines the desired behavior for handling Indications that report the occurrence of the same underlying event (e.g., the disk is still generating I/O errors and has not yet

been repaired). For SMI-S, this is extended to include multiple indications that are generated from a single IndicationFilter.

The related properties are RepeatNotificationCount, RepeatNotificationInterval, and RepeatNotificationGap. The defined semantics for these properties depend on the value of RepeatNotificationPolicy, but values for these properties shall be set if the property is defined for the selected policy.

If the value of RepeatNotificationPolicy is 2 (None), then the client will receive all indications. Special processing of repeat Indications shall not be performed.

If the value is 3 (Suppress) the first RepeatNotificationCount Indications, describing the same event, shall be sent and all subsequent Indications for this event suppressed for the remainder of the time interval RepeatNotificationInterval. A new interval starts when the next Indication for this event is received. That is, all indications after the first ‘n’ (where ‘n’ is defined by the RepeatNotificationCount) are not sent (within the RepeatNotificationInterval time).

If the value of RepeatNotificationPolicy is 4 (Delay), then indications are collected and notification is only sent after a certain number of events happen (as defined by RepeatNotificationCount) or the time interval (RepeatNotificationInterval) lapses. When an event happens, the Indication shall be suppressed if, including this Indication, RepeatNotificationCount or fewer Indications for this event have been received during the prior time interval defined by RepeatNotificationInterval. If this Indication is the RepeatNotificationCount + 1 Indication, this Indication shall be sent and all subsequent Indications for this event ignored until the RepeatNotificationGap has elapsed. A RepeatNotificationInterval may not overlap a RepeatNotificationGap time interval.

For SMI-S, a single indication filter that identifies a change in OperationalStatus on StorageVolumes would be subjected to the RepeatNotificationPolicy, even though the repeat notifications may be from multiple StorageVolumes.

The RepeatNotificationPolicy can vary by implementation (or even IndicationFilter). However, it shall be specified on any subscription. The valid values for an SMI-S implementation are:

- 2 (None),
- 3 (Suppress), or
- 4 (Delay)

An SMI-S profile may restrict this further for any given indication filter, but it cannot expand this to other policies without breaking interoperability. For example, a profile might restrict InstCreation filters for ComputerSystems to “None” and restrict InstModification filters on StorageVolume to “Suppress” or “Delay.” But an SMI-S profile shall not define “unknown” as a valid SMI-S setting for the RepeatNotificationPolicy.

NOTE RepeatNotificationPolicy set to 2 “none” is compatible with SMI-S 1.0.

42.1.6.8 Clarification of indication generation

42.1.6.8.1 General Requirements

To minimize the use of stale object references by WBEM Clients, a WBEM Server shall generate instance deletion indications, where defined as mandatory profile elements, whenever a MSE instance is removed while the WBEM Server is operational. The indication shall be generated for all causes of removal, which include but are not limited to, explicit WBEM instance manipulation by some WBEM Client, internal implementation of the WBEM Server outside the scope of SMI-S, and a side effect of invoking some WBEM extrinsic method.

A WBEM Server should generate instance deletion indications, where defined as mandatory profile elements, whenever a MSE instance that was present before a failure of the device or application is no longer present when the device or application recovers from the failure. Note: SMI-S already requires WBEM Servers to persist WBEM Client subscription for indications.

A WBEM Server shall generate instance creation indications, where defined as mandatory profile elements, whenever a MSE instance is created while the WBEM Server is operational. A WBEM Server shall also generate instance creation indications, where defined as mandatory profile elements, whenever a MSE instance that was not present before a failure of the device or application is present when the device or application recovers from the failure.

Almost universally in SMI-S profiles, all MSE's can be linked by association back to a specific “top-level” MSE. In most profiles this is either a ComputerSystem or a AdminDomain. A WBEM Server that is providing information on multiple devices will have multiple MSE instances, one for each of the devices. The behavior of WBEM Operations in the face of a failure of the device or applications differs.

42.1.6.8.2 Definition of “failed” MSE

A MSE instance is defined to be failed if any of the following conditions hold:

1) Failure status are contained in the OperationalStatus attribute, when present, and OperationalStatus array does not contain “OK”
2) EnumerateInstances, EnumerateInstanceNames, Associators, AssociatorNames, References, ReferenceNames WBEM Operations might return meaningless or no information for any mandatory profile element. OperationalStatus when present in the class will have meaningful data and will have a failure status. Explicit values for “unknown” or “undetermined” are completely meaningful when defined for a profile element.

3) WBEM extrinsic operations that ERR_FAILED may indicate that this instance is failed.

4) CIM Instances that were returned before the failure of the MSE might not be returned after the failure. Indications representing the OperationalStatus change to a failure status were produced for the this 'top-level' CIM Instance or 'top-level' parent CIM Instance. The combination of these two situations define failure in this case

A MSE with an OperationalStatus of “Lost Communications” or “No Contact” obviously shall be considered failed because no WBEM operations can succeed.

An OperationalStatus of “Starting”, “Stopping”, or “Stopped” does not mandate failure. The detailed behavior of the MSE with regard to the conditions given above, determines whether these status's indicate failure. The WBEM Client should be warned of a possible failure scenario when receiving these status.

42.1.6.8.3 Minimal function for failed MSEs

Any failed instance represented by any WBEM Server shall support the following functionality. If the WBEM Server is not able to support the functionality on a failed instance, it shall delete the instance.

1) EnumerateInstances, EnumerateInstanceNames, Associators, AssociatorNames, References, and ReferenceNames WBEM Operations that include the failed instance as part of the return set will complete without error. The Key and the OperationalStatus attributes, when present, shall be properly provided.

2) When a GetInstance WBEM Operation is attempted on the failed instance, CIM_ERR_FAILED shall be returned with a message describing or indicating the failure of the device or application.

3) Failed instance names shall be returned from WBEM Operations that return Object Names. Failed instances shall be returned for WBEM Operations that return Instances but only the keys and OperationalStatus, when present, are mandatory.

4) Method invocations on failed MSEs will fail with the CIM_ERR_FAILED error.

42.1.6.8.4 Isolation of failed top-level MSE's

For efficiency and consistency of navigation, a WBEM Client should not be able to retrieve false or meaningless information from the WBEM Server about a MSE instance.

A WBEM Server can take one of two actions in the Failed MSE case and top-level MSE instances. It shall set the OperationalStatus on the top-level MSE instance to reflect the failed state and forward the related CIM Indications as required. It may also remove all directly or indirectly associated instances, generating the corresponding indications.

A WBEM Client shall be prepared to deal with a WBEM Object CIM_ERR_NOT_FOUND error, indicating the use of a stale object reference not avoided by timely receipt and processing of an instance deletion indication. A WBEM Client shall also consider the OperationalStatus of any MSE for which OperationalStatus is a mandatory profile element before treating the other attributes and associations of the instance as meaningful.

42.1.6.9 HTTP Security

Not defined in this version of the specification.

42.2 Health and Fault Management Considerations

42.2.1 Elements Reporting Health

The Indication Profile has no classes that report health information. However, indications are a means available for reporting changes in health status.

42.2.2 Health State Transformations and Dependencies

No Indications class have OperationalStatus or HealthState properties.

42.2.3 Standard Errors Produced

All manipulation of Indication classes and associations are done using intrinsic methods. The errors produced are those listed for intrinsic methods.

42.2.4 Cause and effect associations

Cause and effect associations are defined as part of the Health and Fault Management Package.

EXPERIMENTAL

42.2.5 Indication Correlation

Not defined in this version of the specification.

EXPERIMENTAL

42.3 Cascading Considerations

Not Applicable.

42.4 Supported Profiles, Subprofiles and Packages

Related Profiles for Indication: Not defined in this standard.

42.5 Methods of the Profile

42.5.1 Extrinsic Methods of the Profile

No extrinsics are specified on the Indication Profile.

42.5.2 Intrinsic Methods of the Profile

The Indication Profile is mostly populated by providers and is accessible to clients using basic read and association traversal. However, there are two constructs that would be created by Clients. These are the ListenerDestinationCIMXML and the IndicationSubscription. In addition, a client may be able to create an IndicationFilter. In addition to being able to create them, client may delete them (except “pre-defined” filters which cannot be deleted), and a client may modify any IndicationFilter that was client created. These functions are performed using the intrinsics:

Table 416 shows Indication Profile methods that cause Instance Creation, Deletion or Modification.

Table 416 - Indication Profile Methods that cause Instance Creation, Deletion or Modification

Method	CreatedInstances	Deleted Instances	Modified Instances
CreateInstance	ListenerDestinationCIMXML	N/A	N/A
CreateInstance	IndicationSubscription	N/A	N/A
CreateInstance	IndicationFilter	N/A	N/A
DeleteInstance	N/A	ListenerDestinationCIMXML	N/A
DeleteInstance	N/A	IndicationSubscription	N/A
DeleteInstance	N/A	IndicationFilter	N/A
ModifyInstance	N/A	N/A	IndicationFilter

CreateInstance - for ListenerDestinationCIMXML, IndicationSubscription and IndicationFilter

```
<instanceName>CreateInstance (
        [IN] <instance> NewInstance
 )
```

EXPERIMENTAL

An implementation should populate all fields (scheme, hostname, port number, namespace, key) in object path for the instance being created.

The host name portion shall be set to a valid, client-resolvable host name (i.e., DNS) or IPv4 or IPv6 address. Internal names (e.g., /etc/hosts) are not valid. If hot name, then must be FQDN (not a short name). This implies the provider shall be configured as a valid DNS client to use host names and cannot rely on an administratively defined name.

If an implementation supports CreateInstance on IndicationFilters or ListenerDestinations, the implementation should allow NULL to be specified for key properties. If key properties are passed in on the CreateInstance, the implementation may ignore the keys.

For IndicationFilters and ListenerDestinations, if the client supplies an ElementName, the implementation shall persist the property for later use by the client.

For IndicationFilters the Query strings are case sensitive and the implementation shall preserve case.

EXPERIMENTAL

If successful, the return value defines the object path of the new CIM Instance relative to the target Namespace (i.e., the Model Path), created by the CIM Server.

Note that for CreateInstance of an IndicationSubscription requires that the ListenerDestinationCIMXML instance and the IndicationFilter exist.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_ALREADY_EXISTS (the CIM Instance already exists), CIM_ERR_FAILED (some other unspecified error occurred).

Note that a ListenerDestinationCIMXML instance should be created in the Interop namespace. However, they may be created in a “Source” namespace. If the client creates a ListenerDestinationCIMXML instance in a “Source” namespace, then a duplicate ListenerDestinationCIMXML instance shall be created in the Interop Namespace, if it does not already exist in the Interop Namespace.

NOTE The inverse is not true. If the client creates the ListenerDestinationCIMXML instance in the Interop Namespace, no instance will be created in other namespaces (there is nothing that would indicate which namespaces would be the Source namespaces).

IndicationFilters may be created in either the Interop Namespace or an implementation namespace in which the indications are to originate. In either case, the Client only needs to create one instance (and providers will automatically create the corresponding instance in the other namespaces).

NOTE If a client attempts to create an IndicationFilter that already exists (has the same key fields), but other properties are different, then the request will fail. If the Client attempts to create an IndicationFilter that has identical properties to an existing IndicationFilter instance, it will succeed and CreateInstance need not treat the instance as a separate instance.

When a client creates an IndicationSubscription the client only needs to create a subscription to one of the IndicationFilters (the provider will automatically generate the corresponding subscription to the other filter instance). Even though there are two instance of the IndicationFilter created (and two instances of the subscription) duplicate indications will not be sent to the ListenerDestination.

Indeed, in general, redundant subscriptions need not produce duplicate indications (that is, if the same listener subscribes to two filters that are equivalent, then an implementation need not produce two indications).

DeleteInstance - for ListenerDestinationCIMXML, IndicationSubscription and IndicationFilter

```
void DeleteInstance (
        [IN] <instanceName> InstanceName
)
```

The InstanceName input parameter defines the name (model path) of the Instance to be deleted.

If successful, the specified Instance (ListenerDestinationCIMXML, IndicationSubscription or IndicationFilter) shall have been removed by the CIM Server.

The deletion of a ListenerDestinationCIMXML or an IndicationFilter instance will cause the automatic deletion of any associated IndicationSubscription instances. Deletion of an IndicationSubscription will not cause the deletion of any corresponding ListenerDestinationCIMXML or IndicationFilter instances. For example, the deletion of an instance may cause the automatic deletion of all associations that reference that instance. Or the deletion of an instance may cause the automatic deletion of instances (and their associations) that have a Min(1) relationship to that instance.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class does not exist in the specified namespace), CIM_ERR_NOT_FOUND (the CIM Class does exist, but the requested CIM Instance does not exist in the specified namespace), CIM_ERR_FAILED (some other unspecified error occurred).

DEPRECATED

NOTE Deleting the instance of an IndicationFilter in the Interop Namespace will cause the corresponding IndicationFilter in the “SourceNamespace” to also be deleted (and vice versa). Deletion of an indication filter will also cause all subscriptions to that filter to be deleted. However, deletion of a filter will not cause the deletion of any listener destination.

NOTE Deleting the instance of an IndicationSubscription in the InteropNamespace will cause the corresponding IndicationSubscription in the “SourceNamespace” to also be deleted (and vice versa). However, deleting a subscription will not delete filters or listener destinations.

NOTE Deleting the instance of ListenerDestinationCIMXML in either the InteropNamespace or the “source” namespace will cause the corresponding instance (if one exists) to be deleted.

DEPRECATED

EXPERIMENTAL

NOTE Deleting the instance of an IndicationFilter in the Interop Namespace will cause the corresponding IndicationFilters in the "SourceNamespaces" to also be deleted. Deletion of an IndicationFilter in one of the SourceNamespaces will not cause the deletion of the IndicationFilter in the InteropNamespace, unless it is the "last entry" in SourceNamespaces of the IndicationFilter in the InteropNamespace. Deletion of an indication filter will also cause all subscriptions to that filter to be deleted. Deletion of a filter will not cause the deletion of any listener destination.

NOTE Deleting the instance of an IndicationSubscription in the InteropNamespace will cause the corresponding IndicationSubscription in the "SourceNamespaces" to also be deleted. Deletion of an IndicationSubscription in a source namespace will not affect IndicationSubscriptions to filters in the Interop namespace (or other implementation namespaces). Deleting a subscription will not delete filters or listener destinations.

NOTE Deleting the instance of ListenerDestinationCIMXML in the InteropNamespace will cause the corresponding instance (if one exists) to be deleted. Deleting an instance of ListenerDestinationCIMXML in any "source" namespace will not affect the corresponding instance in the InteropNamespace to be deleted.

EXPERIMENTAL

ModifyInstance - for IndicationFilters

```
void ModifyInstance (
        [IN] <namedInstance> ModifiedInstance,
        [IN, Optional, NULL] string propertyList[] = NULL
 )
```

The ModifiedInstance input parameter identifies the name of the Instance to be modified, and defines the set of changes to be made to the current Instance definition.

The only Property that may be specified in the PropertyList input parameter is the Query property. Modification of all other properties is not specified by SMI-S.

If successful, the specified Instance shall have been updated by the CIM Server.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_NOT_FOUND (the CIM Instance does not exist), CIM_ERR_FAILED (some other unspecified error occurred)

EXPERIMENTAL

NOTE Modifying the SourceNamespaces property of an IndicationFilter is not defined in this version of the specification. The results may vary depending on the implementation.

EXPERIMENTAL

42.6 Client Considerations and Recipes

42.6.1 Use of Profile Specific Recipes

This profile only defines the interfaces for creating indication filters, listener destinations and subscriptions to receive indications. For information related to indications defined by profiles, that information is document in the CIM Elements tables for the profile in question. For example, Array indications are documented in the CIM Elements table in the Array Profile (see *Storage Management Technical Specification, Part 4 Block Devices, 1.6.1 Rev 5*, 4.8 “CIM Elements”).

42.6.2 General Client Considerations

The indication filters that a client subscribes to are either “predefined” and populated by the profile, or they are created by the client. If the profile supports “predefined” indication filters the client can find them via an enumeration. If the client cannot find the filter it is looking for, it may attempt to create the desired indication filter. If this fails, the client should fall back to creating a filter exactly as it exists in SMI-S. This shall work. The “predefined” indication filters in this specification shall be populated in the profile or it shall be possible to create it.

42.6.3 Discovery of Implementation variations

A client will need to discovery the variations that are allowed in SMI-S profile implementations. A profile implementation has the following degrees of variability:

- Client defined IndicationFilters, pre-defined IndicationFilters or both
- InstModification, with or without PreviousInstance
- Additional Indications

To determine if an implementation supports Client Defined filters, the client should attempt to create an SMI-S specified filter. If it succeeds, the implementation supports client defined filters. At this point, the

client can attempt to create a filter of its own choice or making (e.g., using the client's desired query). If it fails, this means the implementation does not support an indication based on the query used.

If the attempt to create an SMI-S specified indication filter fails, this means client defined queries are not supported. At this point, the client should look for pre-defined filters. This can be done by enumerating filters in the namespace of the profile the client wishes to monitor.

An implementation may (or may not) support PreviousInstance, when the SMI-S specification for the profile identifies InstModification as the indication filter and PreviousInstance is identified as optional. If a client wishes to determine whether or not the implementation actually supports PreviousInstance, it can only tell by receiving an InstModification indication.

Additional Indications are IndicationFilters that are supported by the implementation, but not mandatory with SMI-S. If the implementation supports pre-defined Filters, these can easily be discovered in the enumeration of IndicationFilters. If the implementation does not support pre-defined filters, then the only way a client can discover these is through trial and error (or specific knowledge of the implementation).

42.6.4 Client Defined Filters

Clients need to avoid Filters that generate excessive events. Subscriptions to a general-purpose Server should be specific to the provider – for example "select * from CompanyCorp_InstCreation" rather than "select * from CIM_InstCreation".

EXPERIMENTAL

42.6.5 Creation of IndicationFilter and ListenerDestination Instances

If an implementation populates scheme, hostname, and port in object path, the client should not change anything. If not, the client should accept namespace (and everything to the right) from the implementation and populate scheme, hostname, and port.

When creating IndicationFilters or ListenerDestination instances, clients should be prepared for CIM Servers expecting no key properties to be set and also be prepared for CIM Servers expecting all key properties to be set and valid. If key properties are specified, clients should be aware that the CIM Server may change the keys as specified in the CreateInstance request.

The recommended use of ElementName in these classes is that it is set by client to allow client to easily find instances in later client operations.

42.6.6 Creation of IndicationSubscription Instances

When creating IndicationSubscriptions, a client shall provide the references to the IndicationFilter and the ListenerDestination. But the client should not populate properties that it does not explicitly need set.

EXPERIMENTAL

42.6.7 Determine if the indication subscription requested already exists

```
// DESCRIPTION
// Determine if the indication subscription requested already exists. If
// not, then attempt to create the indication subscription passed in. If
// the CIM Server does not support the addition of indication, then the
// CIM Client will need to poll for these instance changes. This recipes
// does not handle the issue of providing the target URL for indications.
//
```

```
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The namespace of interest has previously been identified and
//   defined in the #SomeNameSpace variable
// 2.The list of filters of interest has been previously built in the
//   #filters[] array. Each element is this array is the WQL filter itself

// FUNCTION: createIndication
sub createIndication ($Filter)
{
     try {
        <create indications as per SMIS specification>
     } catch(CIM_ERROR_NOT_SUPPORTED) {
        <setup this class of instances to be polled for>
     }
}

// MAIN
$ExistingInstances[] = EnumerateInstances(#SomeNameSpace, “CIM_IndicationFilter”)
for #i in $ExistingInstances
{
     for #j in #filters
     {
        if(!compare($ExistingInstances[#j].Query, #filters[#j])
        {
           &createIndiciation(#filters[#j])
        }
     }
}
```

42.6.8 Listenable Instance Notification

```
// DESCRIPTION
// Create an indication subscription for every indication that is
// required by the profile.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.The namespace of interest has previously been identified and
//   defined in the #SomeNameSpace variable

#filters[] = <array of SMIS filters for the target profile>
@{Determine if Indications already exist or have to be created} #filters
```

42.6.9 Life Cycle Event Subscription Description

```
// DESCRIPTION
// Create an indication subscription for the operational status for a
// computer systems defined within a given CIM agent and namespace. This
// subscription   will only be made in those CIM agents that have SAN
// devices or applications of   interest defined in them. The client will
// have to determine once having received the indication, whether the
```

```
// computer system related to this indication (AlertingManagedElement
// attribute) is of interest. This recipe does not handle the target URL
// for the indication.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// None

#filter[0] = “SELECT * FROM CIM_InstModification
    WHERE SourceInstance ISA CIM_ComputerSystem
      AND SourceInstance.OperationalStatus[0] <>
          PreviousInstance.OperationalStatus[0]”
@{Determine if Indications already exist or have to be created} #filter
```

42.6.10Subscription for alert indications

```
// DESCRIPTION
// Create an indication subscription for the alert indications defined
// within a given CIM agent and namespace. This subscription will only be
// made in those CIM agents that have SAN devices or applications of
// interest defined in them. The client will have to determine once having
// received the indication, whether the computer system related to this
// indication (AlertingManagedElement attribute) is of interest.   Each
// specific alert indication will have also specific handling required
// for it by the CIM Client.
// NOTE: This recipe does not handle the target URL for the indication.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// None

#filter[0] = “SELECT * FROM CIM_AlertIndication”
@{Determine if Indications already exist or have to be created} #filter
```

42.6.11Listenable Interface Modification Notification

```
// DESCRIPTION
// Create an indication subscription for every indication
// that isrequired by the profile
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The namespace of interest has previously been identified and
//   defined in the #SomeNameSpace variable

#filters[] = <array of SMIS filters for the target profile>
@{Determine if Indications already exist or have to be created} #filters
```

42.6.12Subscribe for Lifecycle Events where OperationalStatus Changes

```
// DESCRIPTION
// Create an indication subscription for the operational status for a
// computer systems defined within a given CIM agent and namespace. This
// subscription   will only be made in those CIM agents that have SAN
```

```
// devices or applications of   interest defined in them. The client will
// have to determine once having received the indication, whether the
// computer system related to this indication (AlertingManagedElement
// attribute) is of interest. This recipe does not handle the target URL
// for the indication.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// None

#filter[0] = “SELECT * FROM CIM_InstModification
    WHERE SourceInstance ISA CIM_ComputerSystem
      AND SourceInstance.OperationalStatus[0] <>
          PreviousInstance.OperationalStatus[0]”
@{Determine if Indications already exist or have to be created} #filter
```

42.7 Registered Name and Version

Indication version 1.5.0 (Component Profile)

42.8 CIM Elements

Table 417 describes the CIM elements for Indication.

Table 417 - CIM Elements for Indication

Element Name	Requirement	Description
42.8.1 CIM_AlertIndication	Optional	This Indication is used to capture events that occur in the profile, but may not be related to a specific part of the model.
42.8.2 CIM_IndicationFilter (client defined)	Optional	This is for 'client defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the value and format of an indication filter string.
42.8.3 CIM_IndicationFilter (pre-defined)	Optional	This is for 'pre-defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the value and format of an indication filter string.
42.8.4 CIM_IndicationSubscription	Mandatory	This association defines a subscription to a specific IndicationFilter instance by a specific indication handler (as represented by a ListenerDestinationCIMXML instance).
42.8.5 CIM_InstCreation	Optional	CIM_InstCreation is an indication of the creation of a CIM instance. It would be generated when an instance of the SourceInstance class is created (either explicitly or implicitly).
42.8.6 CIM_InstDeletion	Optional	CIM_InstDeletion is an indication of the Deletion of a CIM instance. It would be generated when an instance of the SourceInstance class is deleted from the model (either explicitly or implicitly).

Table 417 - CIM Elements for Indication

Element Name	Requirement	Description
42.8.7 CIM_InstModification	Optional	CIM_InstModification is an indication of the modification or change to a CIM instance. It would be generated when an instance of the SourceInstance class is modified or changed (either explicitly or implicitly).
42.8.8 CIM_ListenerDestinationCIMXML (Indication Handler)	Mandatory	A CIM_ListenerDestinationCIMXML describes the destination for CIM Export Messages to be delivered via CIM-XML. ListenerDestinationCIMXML is subclassed from ListenerDestination.

42.8.1 CIM_AlertIndication

A CIM_AlertIndication is a specialized type of CIM_Indication that contains information about the severity, cause, recommended actions and other data of a real world event.

CIM_AlertIndication is subclassed from CIM_ProcessIndication.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 418 describes class CIM_AlertIndication.

Table 418 - SMI Referenced Properties/Methods for CIM_AlertIndication

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Optional	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime	N	Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
Description		Optional	A free form text description.
AlertingManagedElement		Mandatory	The identifying information of the entity for which this Indication is generated.
AlertingElementFormat		Mandatory	Valid SMI-S values are 0|1|2 ('Unknown' | 'Other' | 'CIMObjectPath').
AlertType		Mandatory	This shall be 1|2|3|4|5|6|7|8 ('Other' | 'Communications Alert' | 'Quality of Service Alert' | 'Processing Error' | 'Device Alert' | 'Environmental Alert' | 'Model Change' | 'Security Alert').
OtherAlertType		Optional	
PerceivedSeverity		Mandatory	This shall be 0|1|2|3|4|5|6|7 ('Unknown', 'Other' | 'Information' | 'Degraded/ Warning' | 'Minor' | 'Major' | 'Critical' | 'Fatal/NonRecoverable').
OtherSeverity		Optional	
ProbableCause		Mandatory	Many possible values in a value map. See MOF.
ProbableCauseDescription		Optional	
EventID		Optional	
SystemCreationClassNam e		Mandatory	

Table 418 - SMI Referenced Properties/Methods for CIM_AlertIndication

Properties	Flags	Requirement	Description & Notes
SystemName		Mandatory	The scoping System's Name for the Provider generating this Indication. The SystemName would typically be the name of the system that generates the indication.
ProviderName		Mandatory	
OwningEntity	N	Optional	A string that uniquely identifies the entity that owns the definition of the format of the Message.
MessageID	N	Optional	A string that uniquely identifies, within the scope of the OwningEntity, the format of the Message.
Message	N	Optional	The formatted message (including the MessageArguments).
MessageArguments	N	Optional	An array of strings that contain the dynamic content of the message.
OtherAlertingElementFormat	N	Optional	Not Specified in this version of the Profile.
Trending	N	Optional	Not Specified in this version of the Profile.
RecommendedActions	N	Optional	Not Specified in this version of the Profile.
EventTime	N	Optional	Not Specified in this version of the Profile.

42.8.2 CIM_IndicationFilter (client defined)

CIM_IndicationFilter instances that are 'client defined' are IndicationFilters that are be created by a client using CreateInstance. If a profile implementation can support client defined IndicationFilters, the implementation would support 'client defined' IndicationFilter instances. The implementation shall support 'client defined' filters that are defined by SMI-S profile as mandatory, but may also support additional filters supported by the implementation (See QueryCapabilities).

CIM_IndicationFilter is subclassed from CIM_ManagedElement.

Created By: CreateInstance
Modified By: ModifyInstance
Deleted By: DeleteInstance
Requirement: Optional

Table 419 describes class CIM_IndicationFilter (client defined).

Table 419 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This should take the form OrgID ":" RegisteredName ":" UniqueID. For more details, see section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 42.1.1 IndicationFilter Names.

Table 419 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

Properties	Flags	Requirement	Description & Notes
SourceNamespace	N	Optional	Deprecated. For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Mandatory	This should be all the namespaces where the indications may originate.
Query		Mandatory	A string that specifies (in QueryLanguage terms) which indications are to be delivered to the ListenerDestinations.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName		Optional	A Client Defined user friendly string that identifies the Indication Filter.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

42.8.3 CIM_IndicationFilter (pre-defined)

CIM_IndicationFilter instances that are 'pre-defined' are IndicationFilters that are populated automatically by the profile provider. If a profile implementation cannot support client defined IndicationFilters, the implementation can populate its model with 'pre-defined' IndicationFilter instances. 'Pre-defined' filters shall include those that are required by the profile, but may also contain additional filters supported by the implementation.

CIM_IndicationFilter is subclassed from CIM_ManagedElement.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 420 describes class CIM_IndicationFilter (pre-defined).

Table 420 - SMI Referenced Properties/Methods for CIM_IndicationFilter (pre-defined)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This should take the form OrgID ":" RegisteredName ":" UniqueID. For more details, see section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 42.1.1 IndicationFilter Names.
SourceNamespace	N	Optional	Deprecated. For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Mandatory	This should be all the namespaces where the indications may originate.

Table 420 - SMI Referenced Properties/Methods for CIM_IndicationFilter (pre-defined)

Properties	Flags	Requirement	Description & Notes
Query		Mandatory	A string that specifies (in QueryLanguage terms) which indications are to be delivered to the ListenerDestinations.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	SMI-S does not specify this property for pre-defined IndicationFilters.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

42.8.4 CIM_IndicationSubscription

A CIM_IndicationSubscription is not subclassed from anything.

Created By: CreateInstance
Modified By: Static
Deleted By: DeleteInstance
Requirement: Mandatory

Table 421 describes class CIM_IndicationSubscription.

Table 421 - SMI Referenced Properties/Methods for CIM_IndicationSubscription

Properties	Flags	Requirement	Description & Notes
RepeatNotificationPolicy		Mandatory	SMI-S supports a restricted set of values. This shall be 2|3|4 ('None' | 'Suppress' | 'Delay').
RepeatNotificationInterval		Optional	Mandatory if the RepeatNotificationPolicy is 'Suppress' or 'Delay'.
RepeatNotificationGap		Optional	Mandatory if the RepeatNotificationPolicy is 'Delay'.
RepeatNotificationCount		Optional	Mandatory if the RepeatNotificationPolicy is 'Suppress' or 'Delay'.
LastIndicationIdentifier		Optional	The IndicationIdentifier of the last indication produced for this subscription regardless if that indication were delivered.
LastIndicationProductionDateTime		Optional	The date and time of the production of the last indication produced for this subscription regardless if that indication were delivered.
OnFatalErrorPolicy	N	Optional	Not Specified in this version of the Profile.
OtherOnFatalErrorPolicy	N	Optional	Not Specified in this version of the Profile.
FailureTriggerTimeInterval	N	Optional	Not Specified in this version of the Profile.
SubscriptionState	N	Optional	Not Specified in this version of the Profile.
OtherSubscriptionState	N	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	N	Optional	Not Specified in this version of the Profile.
SubscriptionDuration	N	Optional	Not Specified in this version of the Profile.
SubscriptionStartTime	N	Optional	Not Specified in this version of the Profile.
SubscriptionTimeRemaining	N	Optional	Not Specified in this version of the Profile.

Table 421 - SMI Referenced Properties/Methods for CIM_IndicationSubscription

Properties	Flags	Requirement	Description & Notes
OtherRepeatNotificationPolicy	N	Optional	Not Specified in this version of the Profile.
AlertOnStateChange	N	Optional	Not Specified in this version of the Profile.
Filter		Mandatory	
Handler		Mandatory	

42.8.5 CIM_InstCreation

CIM_InstCreation notifies a handler when a new instance (of a class defined in the Filter QueryString) is created.

CIM_InstCreation is subclassed from CIM_InstIndication.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 422 describes class CIM_InstCreation.

Table 422 - SMI Referenced Properties/Methods for CIM_InstCreation

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Optional	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime		Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
SourceInstance		Mandatory	A copy of the instance that changed to generate the Indication. SourceInstance contains the current values of the properties selected by the Indication Filter's Query.
SourceInstanceModelPath		Mandatory	The Model Path of the SourceInstance.
PerceivedSeverity	N	Optional	Not Specified in this version of the Profile.
OtherSeverity	N	Optional	Not Specified in this version of the Profile.
SourceInstanceHost	N	Optional	Not Specified in this version of the Profile.

42.8.6 CIM_InstDeletion

CIM_InstDeletion notifies a handler when a new instance (of a class defined in the Filter QueryString) is deleted.

CIM_InstDeletion is subclassed from CIM_InstIndication.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 423 describes class CIM_InstDeletion.

Table 423 - SMI Referenced Properties/Methods for CIM_InstDeletion

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Optional	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime		Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
SourceInstance		Mandatory	A copy of the instance that changed to generate the Indication. SourceInstance contains the current values of the properties selected by the Indication Filter's Query.
SourceInstanceModelPath		Mandatory	The Model Path of the SourceInstance.
PerceivedSeverity	N	Optional	Not Specified in this version of the Profile.
OtherSeverity	N	Optional	Not Specified in this version of the Profile.
SourceInstanceHost	N	Optional	Not Specified in this version of the Profile.

42.8.7 CIM_InstModification

CIM_InstModification notifies a handler when a new instance (of a class defined in the Filter QueryString) is modified or changed. To avoid undue effort on Providers, the select list (in the query filter) for this indication should only call for properties that are needed.

CIM_InstModification is subclassed from CIM_InstIndication.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 424 describes class CIM_InstModification.

Table 424 - SMI Referenced Properties/Methods for CIM_InstModification

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Optional	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime		Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
SourceInstance		Mandatory	A copy of the instance that changed to generate the Indication. SourceInstance contains the current values of the properties selected by the Indication Filter's Query.
SourceInstanceModelPath		Mandatory	The Model Path of the SourceInstance.
PreviousInstance		Optional	A copy of the 'previous' instance whose change generated the Indication. PreviousInstance contains 'older' values of an instance's properties (as compared to SourceInstance), selected by the IndicationFilter's Query.
PerceivedSeverity	N	Optional	Not Specified in this version of the Profile.
OtherSeverity	N	Optional	Not Specified in this version of the Profile.
SourceInstanceHost	N	Optional	Not Specified in this version of the Profile.

42.8.8 CIM_ListenerDestinationCIMXML (Indication Handler)

CIM_ListenerDestinationCIMXML is subclassed from CIM_ListenerDestination.

Created By: CreateInstance
Modified By: Static
Deleted By: DeleteInstance
Requirement: Mandatory

Table 425 describes class CIM_ListenerDestinationCIMXML (Indication Handler).

Table 425 - SMI Referenced Properties/Methods for CIM_ListenerDestinationCIMXML (Indication Handler)

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A client defined user friendly string that identifies the CIMXML Listener destination.
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
PersistenceType		Mandatory	For SMI-S, this shall be 2|3 ('permanent' | 'transient').
Destination		Mandatory	The destination URL to which CIM-XML Export Messages are to be delivered. The scheme prefix shall be consistent with the DMTF CIM-XML specifications.If a scheme prefix is not specified, the scheme \http:\'shallbeassumed.'
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
OtherPersistenceType	N	Optional	Not Specified in this version of the Profile.

DEPRECATED

DEPRECATED

43 Experimental Indication Profile

This profile is being deprecated in favor of the SNIA specialization of the DMTF Indications Profile version 1.2.0 (See 50 Indications Profile).

43.1 Description

The Experimental Indication Profile supports the Indication Profile and extends it with the following experimental content:

- Client Certificates for securing indications delivery
- Use of a Typed WBEM URI for identifying the AlertingManagedElement
- Scheme for encoding of the IndicationIdentifier
- The use of CQL as the Query Language for IndicationFilters
- Use of Bellwether events to control indication storms
- Use of batching to control indication storms
- Use and encoding of CorrelatedIndications
- Semi-fixed Client Specific Indications
- Filter Collections (both predefined and client defined)
- Indication Configuration Service (extrinsic methods)

43.1.1 Basic Indication Classes and Association

Figure 60 illustrates the classes used in support of indications.

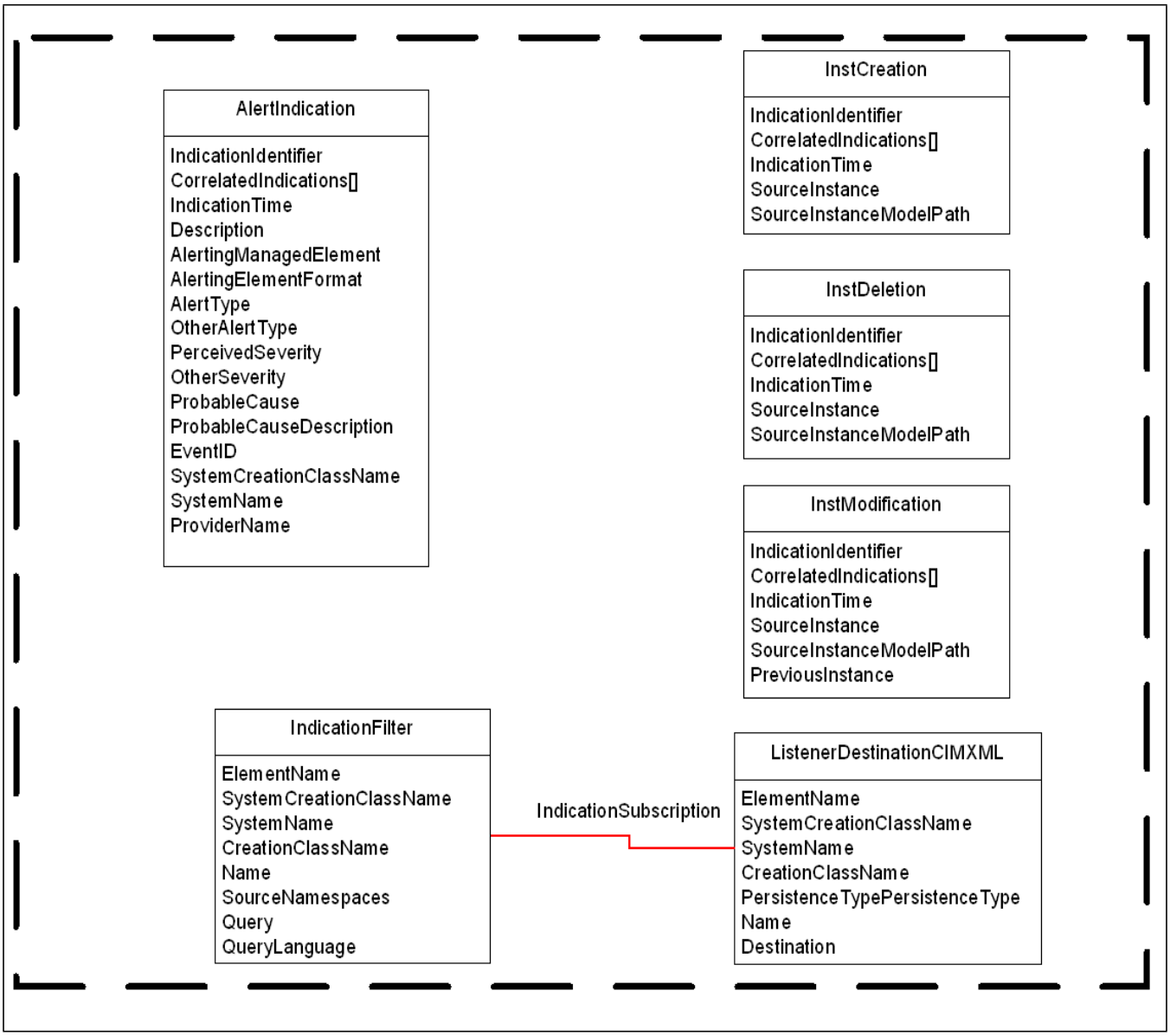

Figure 60 - Indication Profile Instance Diagram

43.1.2 AlertingManagedElement encoding in AlertIndication Instances

When encoding the mandatory property “AlertingManagedElement” of an AlertIndication the following rules apply:

- If the element in question is modeled by the profile implementation, then the format for this property should be as a Typed WBEM URI as defined in DSP0207.
- If the element in question is not modeled by the profile implementation, then the encoding for this property should be as meaningful to clients as possible

43.1.3 Instrumentation Requirements

EXPERIMENTAL

43.1.3.1 Indication Identification

Indications are identified through the IndicationIdentifier property. An indication can be correlated to previously produced indications through the use of the CorrelatedIndication property. Generally, the identity of the indication is only meaningful as a correlatable ID within the CorrelatedIndication property or in its relevancy to the LastIndicationIdentifier property in the IndicationSubscription class.

The LastIndicationIdentifier property on the IndicationSubscription association should record the identity of the last indication produced for the combination of IndicationFilter and IndicationDestination that the association instance represents.

NOTE The LastIndicationIdentifier property will become mandatory in a future release of SMI-S as WBEM infrastructures are enhanced to support the property.

The client can determine if it did get delivery of any indication destined for it by comparing the last indication it received, or the last indication it received for a particular indication subscription, with the LastIndicationIdentifier. It is important for clients to be able to determine if there are interruptions in the indication telemetry. Confidence in the indication delivery combined with the ability to determine the extent of the failure to receive indications, provides clients with a mechanism to gauge appropriately the response to the failures and avoid having to flush state and explore the SMI-S Server's model again.

NOTE In future release of SMI-S, the modeling of the health of the indication delivery system or service will help clients determine if there are problems in the configuration of the subscription and related credentials, or in the indication delivery configuration of the SMI-S Server. This design will require the logging of the errors produced in the delivery of the indications.

The naming algorithm for the IndicationIdentifier property, shown in Figure 61, includes the population of the two subcomponents of the property, OrgID and LocalID, as separated by a colon ":". The OrgID shall contain a registered trademark for the developer of the implementation producing the indication. The LocalID shall contain the combination of the CIM Object Name of the IndicationFilter that produced the Indication, production sequence number, and a delivery sequence number. These sequence numbers are in the form of an unsigned integer. These three elements within the LocalID are separated by a hash "#". The omission of the Handler key property of the indication subscription, which is the object name of the indication destination, means that the client should assume that the indication was correctly delivered to it.

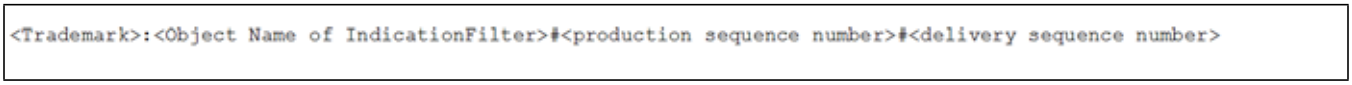

Figure 61 - Anatomy of IndicationIdentifier

The production sequence number is a count of all indications produced by this SMI-S implementation. The sequence shall be unique by device or application instrumented through SMI-S. The production sequence number shall not be unique by indication filter, but instead shall represent the count of indications produced for this device or application. Any gaps in the production sequence number represent indications that were produced but were not delivered because there is no indications subscribers or a SMI-S Client did not receive the indication because it was not subscribed to it.

The delivery sequence number range shall be unique and independent by indication subscription. The delivery sequence number reported shall increase by one and only one with every indication produced for that subscription. In other words, this delivery sequence number can be viewed as a count of indications produced for a particular indication subscription. Any gaps in the delivery sequence number represent indications that were produced for a particular destination (e.g., a SMI-S Client) but were not delivered for some reason. Since an SMI-S Server, the infrastructure, is normally in charge of forwarding indications

delivered to it by CIM Providers, it is best able to able to produce this sequence number. SMI-S Servers should produce this sequence number, but may omit it if unsupported by the CIM Server.

It is recommend that the sequence numbers have a 16-bit range. It other words, the sequence numbers should start at 1 and iterate to 65,536. The implementation may use a larger range, like 1 to 262,144, and should do so if there is a possibility that 65,536 indications per a given indication subscription can be produced within a twenty-four hour period. Regardless, the maximum sequence number shall be a power of two.

The implementation may roll-over the sequence number and start again at one. The requirement that the sequence number shall be a power of two allows a client to determine what the maximum the sequence could be, like 65,536, in order to determine if the last sequence number received for an indication subscription, e.g., 65,533, is the last one it should have received. Clients can not be certain how many indications were missed when the sequence number rolls over given the unknown frequency of indication production and the unknown maximum value of the sequence.

Conformance to the indication identifier naming algorithm is mandatory.

However, an indicator that many indications for a given subscription may have been missed is contained in the LastIndicationProductionDateTime property. The difference between now and the date and time value of LastIndicationProductionDateTime is significant, then the possibility exists.

EXPERIMENTAL

43.1.3.2 Handling of Indication Storms

To contain the impact of indication storms an implementation can employ additional techniques:

- Use of Bellwether events (if they are defined by the profile)

43.1.3.2.1 Use of Bellwether Events

There are many state changes in the model for a device or application that results in changes in many CIM instances. For example, the addition of a device or application representation to a CIMOM should result in creation indications for every single member instance of that device or application. The activation of a ZoneSet from one of the member Switches in a fabric should result to indication listeners on another Switch's namespace creation indications for every instance of the new ZoneSet.

The worse case risk is that several of this type of situation may occur simultaneously and result in network storms and the sudden saturation of the LAN. Additionally, the use of computing resources of the device or application producing the indication or client receiving the indications may be unacceptably high.

Indications provide the most value when they are used by a client as a mechanism to pick a significant or small number of changes in CIMOMs of interest. In order to capture a wide variety of changes, any of which may be pertinent to the client application, the client is likely to create many indication subscriptions and keep them all active simultaneously. This approach is not problematic because the number of management related changes to any device or application in the network is usually very small.

As mentioned previously, there are several potential situations where an excessive number of indications can be produced, thereby potentially overloading the network, originating CIMOM, and receiving client's resources. There is no need to occur such a risk because it is likely that the client is not going to be interested in all things at all times. The interest of the client in instance changes usually follows the needs of the current users of that client application.

Bellwether indications are used by SMI-S designers and individual implementation to signal many instance changes with one event. A client can assume that some previously defined graph of associated CIM instances are affected when it receives a bellwether indication. It can then choose, if

warranted, to fetch all or some of these instances. This design prevent the previously mentioned adverse side effects.

Some rules being considered are:

- When a device or application is added to a namespace and there are indication subscription that cover some or all of the graph of instances added by side effect of the addition, then only a create indication is produced for the top level object for the device or application, like ComputerSystem, provided that there is an indication subscription for changes in the top-level object. Similarly, if a device or application is deleted in the same situation, then only a delete indication will be produced.
- Bellwether indication are mandatory if they exist in SMI-S and will be easily identified as being bellwether events.
  - The classes associated to the bellwether indication will be part of the definition of the indication. The client can assume that instances of these classes will have been affected and can choose to harvest that data. The implementation is not required to produce instances of every class listed as per the requirements defined elsewhere in SMI-S.
- SMI-S Designer's are encouraged to define bellwether indications, which can be of any class of indication, for major state changes of a model. In the previous examples, the device creation could be a life cycle indication where changes in ZoneSet change may be best communicated by an Alert Indication.

43.1.3.2.2 Bellwether Indications for ComputerSystem

It is important to not overload a SMI-S client when device or applications are added or removed from CIM Object Managers. The addition or removal of the representation of a device or application is attributed to the creation or deletion of a top-level computer system instance. This overloading would arise from a SMI-S Agent sending creation or deletion indications to every indication destination for all component or dependent instances to the top-level computer system. For this profile, when a top-level computer system instance is created in the model, the SMI-S agent shall not produce indications for indication subscriptions, on indications that do not reference the top-level computer system, that would otherwise receive InstCreation indications. Likewise, for this profile, when a top-level computer system is deleted from the model, the SMI-S agent shall not produce indications for indication subscriptions, on indications that do not reference the top-level computer system, that would otherwise receive InstDeletion indications.

43.1.4 Semi-Fixed Client Specific Indication Filters

Semi-fixed Client specific IndicationFilters extend the support for indications in the following classes:

- SNIA_IndicationFilterTemplate

  This class mirrors the CIM_IndicationFilter, but the Query property supports a query with the string 'SUBSTITUTION_STRING' included in a CQL query. This is a template that may be used by a client application to create an instance of CIM_IndicationFilter with a client application supplied string in place of the string 'SUBSTITUTION_STRING'.

- SNIA_IndicationConfigurationCapabilities (and the SupportedFeatures property)

  The SupportedFeatures property of SNIA_IndicationConfigurationCapabilities includes the value ‘7’ (“Semi-fixed IndicationFilters”) that indicates semi-fixed IndicationFilters are supported.

When an implementation sets this enumeration, the implementation shall support creation (CreateInstance) of IndicationFilters that follow a pattern that includes substitution strings as defined in the

SNIA_IndicationFilterTemplate Query property. The client is allowed to replace the substitution string with any simple expression (including constants).

Semi-fixed IndicationFilterTemplates are documented in this standard with the substitution string identified with the string ‘SUBSTITUTION_STRING’ in the Query property. For example, the following Query values would indicate a Semi-fixed IndicationFilterTemplate:

```
SELECT * FROM CIM_InstDeletion
WHERE SourceInstance ISA CIM_StorageSynchronized AND
OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH(‘SUBSTITUTION_STRING’)
```

or

```
SELECT * FROM CIM_InstModification
WHERE SourceInstance ISA CIM_StorageVolume AND
SourceInstance.OperationalStatus = ‘SUBSTITUTION_STRING’
```

When a semi-fixed IndicationFilterTemplate is defined in this standard, the description column for the CIM Element for the Indication will identify valid substitutions or will reference an implementation section that identifies the valid values.

43.1.4.1 Naming Conventions for IndicationFilterTemplates and IndicationFilters

Both IndicationFilters and IndicationFilterTemplates have a Name property. The value of the Name property shall be formatted as defined by the following ABNF rule:

```
OrgID ":" RegisteredName ":" UniqueID
```

Where OrgID identify the business entity owning the referencing profile. OrgID shall include a copyrighted, trademarked, or otherwise unique name that is owned by that business entity or that is a registered ID assigned to that business entity by a recognized global authority. In addition, to ensure uniqueness, OrgID shall not contain a colon (:).

For referencing profiles owned by the SNIA, OrgID shall match "SNIA" for IndicationFilterTemplates defined by the standard. For vendor unique IndicationFilterTemplates, the OrgID should be a unique name for the vendor. For client defined IndicationFilters that are based on IndicationFilterTemplates, the OrgID should identify the client (application) organization.

The RegisteredName shall be the registered name of the referencing profile, as defined by the value of its CIM_RegisteredProfile.RegisteredName property.

The UniqueID shall uniquely identify the instance within the referencing profile.

43.1.5 Filter Collections

The standard supports two types of filter collections:

- Predefined Filter Collections - The predefined filter collections augments the Indication Profile support for predefined indication filters and indication filter templates by collecting them into a structure of collections (one per profile) that are hosted on the top level system of the autonomous profile. This provides a convenient means of finding an implementations support for predefined filters.
- Client Defined Filter Collections - The client defined filter collections are collections that are defined by applications of the standard. Client defined filter collections may include predefined and/or client defined indication filters and allow the application to collect a set of related indication filters to which the application wishes to subscribe.

43.1.5.1 Predefined Filter Collections

Predefined filter collections are an optional feature of the standard. Support would be indicated via the IndicationConfigurationCapabilities (see section 43.1.6) in the SupportedFeatures property. If the SupportedFeatures array includes the value ‘5’ it means the implementation supports predefined filter collections.

Figure 62 illustrates the classes associated with predefined FilterCollection support.

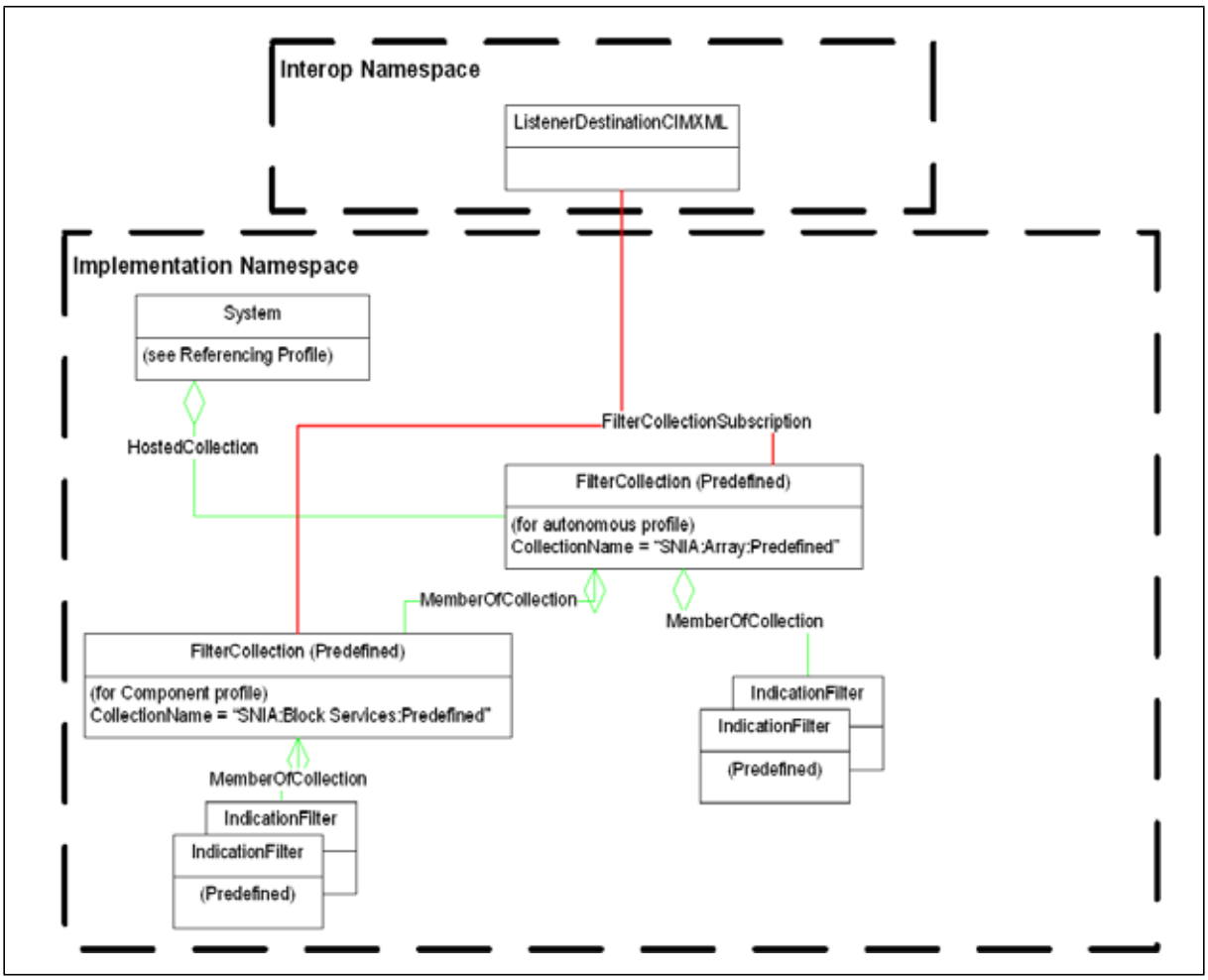

Figure 62 - Predefined Filter Collections

Support for predefined filter collections includes instantiations of the following classes and associations to the model:

- FilterCollection (Predefined)

  A predefined FilterCollection collects a set of predefined IndicationFilters. The primary purpose of a predefined FilterCollection is for an implementation to declare the IndicationFilters that it supports. Minimally this should include all IndicationFilters that are defined as mandatory for the profile. However, it may also include optional, conditional or vendor extension IndicationFilters supported by the implementation.

  One predefined FilterCollection is defined for each profile supported by the implementation. The FilterCollections are organized in a 2 level hierarchy. The top most FilterCollection is the FilterCollection for the autonomous profile. The name of the FilterCollection (CollectionName property) is of the form "SNIA:<profile name>. In Figure 62 the autonomous profile is an Array profile. In addition to collecting predefined IndicationFilters of the autonomous profile, the top level FilterCollection would also collect FilterCollections for each of the component profiles supported by the in implementation. In Figure 62 the component profile FilterCollection shown is for the Block Services Package (CollectionName = "SNIA:Block Services").

- HostedCollection

This associates each filter collection with the top level system for which the collection applies (e.g., the top level system of the autonomous profile).

- FilterCollectionSubscription

  This associates (subscribes) a ListenerDestination (Handler) to a FilterCollection (collection). All indication filters in the collection will be reported to the ListenerDestination (without the need for individual subscriptions on the indication filters).

  Note: A FilterCollectionSubscription will not return any indications for any IndicationFilterTemplates in a pre-defined FilterCollection. Templates are only used for creation of client defined IndicationFilters.

  A client may subscribe to the collection of indications rather than subscribing to each individual filter in the collection. However, a client might prefer to form its own collection of filters that it wants to subscribe to (see section 43.1.5.2).

- MemberOfCollection

  This associates predefined IndicationFilters and predefined component FilterCollections to the FilterCollections in which they belong. MemberOfCollection associations are static and established by the implementation.

43.1.5.2 Client Defined Filter Collections

Client defined filter collections are an optional feature of the standard. Support would be indicated via the IndicationConfigurationCapabilities (see section 43.1.6) in the SupportedFeatures property. If the SupportedFeatures array includes the value ‘6’ it means the implementation supports client defined filter collections.

Figure 63 illustrates the classes associated with client defined FilterCollection support.

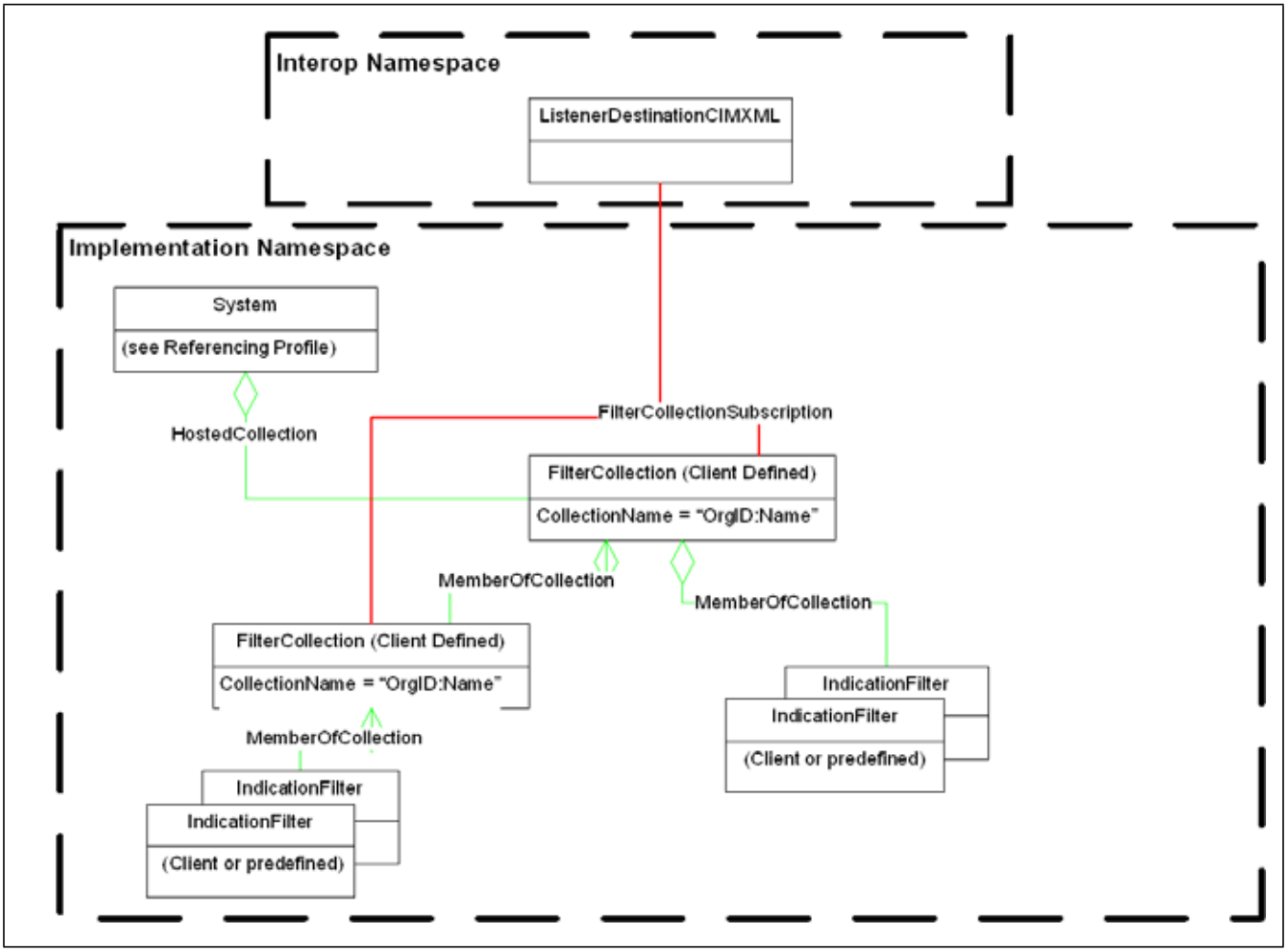

Figure 63 - Client Defined Filter Collections

Support for client defined filter collections includes support of the following classes and associations to the model:

- FilterCollection (Client Defined)

  A client defined FilterCollection collects a set of IndicationFilters (or other FilterCollections). The primary purpose of a client defined FilterCollection is to allow a client to establish a set of indication filters in which it wishes to subscribe to as a group. The indication filters collected may be either client defined or predefined. They may include any IndicationFilters supported by the implementation.

  Unlike predefined FilterCollections client defined FilterCollections may be organized for the convenience of the client application. The FilterCollections may be organized in a hierarchy of any number of levels and a any one FilterCollection need not correspond to a profile.

  The name of the FilterCollection (CollectionName property) is of the form “<OrgID:<Unique Name>. In Figure 63 a top level collection is defined with a lower level FilterCollection. The <OrgID> component should be a company indicator (e.g., stock ticker). The <Unique Name> part of the name should uniquely identify the collection within that company. It may be desirable to make the <Unique Name> part of the CollectionName a compound construction (e.g., <Product Name:Name within Product>). But all that this standard dictates is that the <OrgID> cannot be “SNIA”.

A set of methods for creating and maintaining client defined FilterCollections are provided by the IndicationConfigurationService (see section 43.1.6).

- HostedCollection

  This associates each filter collection with the top level system for which the collection applies (e.g., the top level system of the autonomous profile).

- FilterCollectionSubscription

  This associates (subscribes) a ListenerDestination (Handler) to a FilterCollection (collection). All indication filters in the collection will be reported to the ListenerDestination (without the need for individual subscriptions on the indication filters.

  A client may subscribe to the collection of indications rather than subscribing to each individual filter in the collection.

- MemberOfCollection

  This associates IndicationFilters (predefined or client defined) and other FilterCollections (client defined or predefined) to higher level FilterCollections. MemberOfCollection is established using methods of the IndicationConfigurationService (see section 43.1.6).

43.1.6 Indication Configuration Services

Figure 64 illustrates the classes associated with support for methods for configuring and testing indications support.

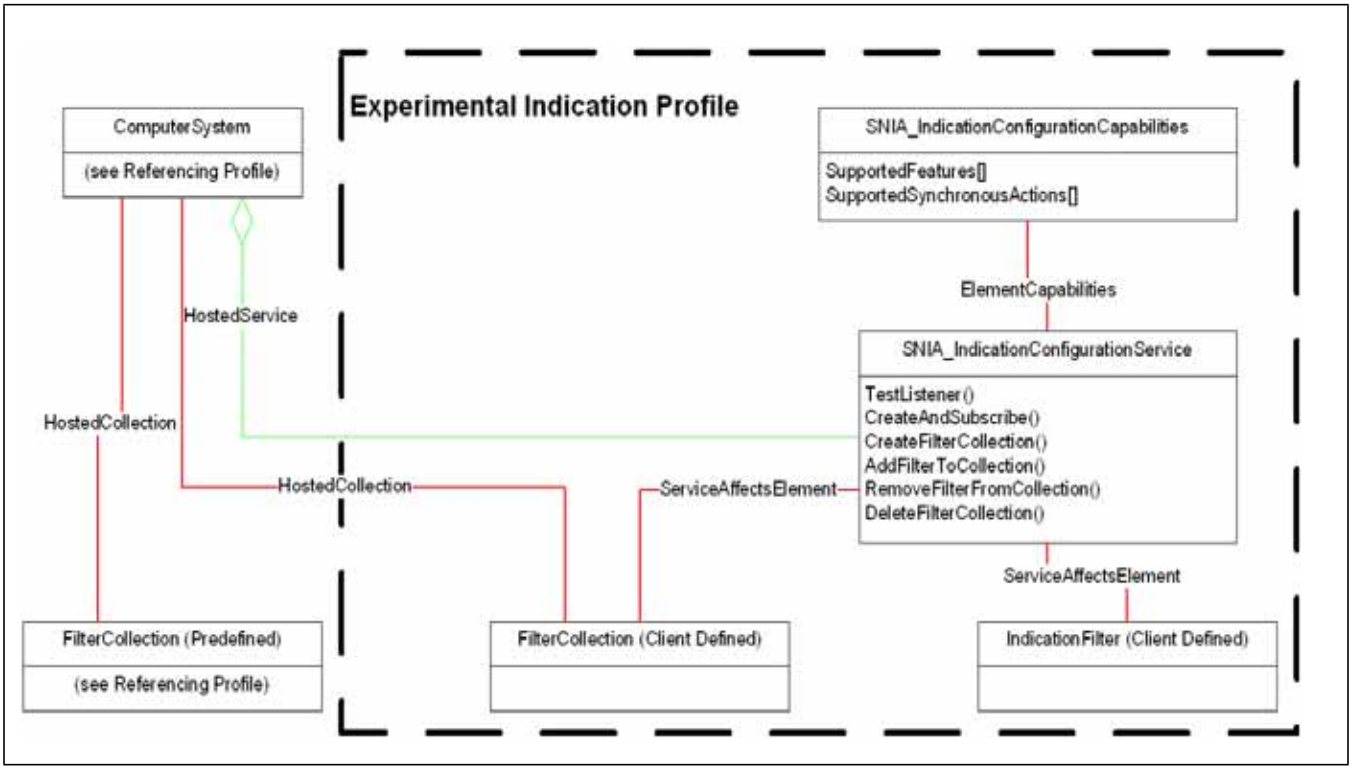

Figure 64 - Indication Configuration Service Classes

Support for the Indication Configuration Service add the following classes and associations to the model:

- SNIA_IndicationConfigurationService

  This service includes methods for testing a listener, handling the creation and subscription to indications as one extrinsic method (rather than a set of CreateInstances) and methods for managing client defined filter collections. The SNIA_IndicationConfigurationService is a service that is specific to a particular profile implementation and the classes managed are instantiated in the implementation namespace. The SNIA_IndicationConfigurationService shall be instantiated when the Experimental Indication Profile is supported, but support for the individual methods are conditional (See the SNIA_IndicationConfigurationCapabilities).

  Note that predefined FilterCollections are not managed by the methods of the indication configuration services. They are, by their nature, static and maintained by the implementation.

- HostedService

  This associates the SNIA_IndicationConfigurationService to the system of the autonomous profile (referencing profile) for which the service applies. For example, for the Array Profile, the indication configuration service would be hosted on the top level computer system for the Array. An autonomous profile shall have exactly one indication configuration service and it shall be hosted on the top level system. It may not be hosted on non-top-level systems (e.g., component computer systems defined in the Multiple Computer System Profile).

- SNIA_IndicationConfigurationCapabilities

  There is one instance of SNIA_IndicationConfigurationCapabilities for an SNIA_IndicationConfigurationService. These capabilities define the extrinsic methods supported by the implementation and a set of SupportedFeatures. The possible SupportedSynchronousActions values and their definitions are as follows:

  - ‘2’ (None) - None of the IndicatonConfigurationService methods are supported.
  - ‘3’ (Test Listener) - The TestListener method is supported.
  - '4' ("Create and Subscribe) - The CreateAndSubscribe method is supported.
  - ‘5’ (Filter Collection Methods) - The methods for managing client defined FilterCollections are supported.

  The possible SupportedFeatures values and their definitions are as follows:

  - ‘2’ (None) - None of the optional features are supported. Specifically, FilterCollections (either predefined or client defined) are not supported. Filters (either predefined or client defined) are not supported. And semi-fixed Indication Filters are not supported.

    Note: In SMI-S, ‘none’ is only valid for profiles that don’t support indications. Any profile that supports indications shall support either or both predefined or client defined indication filters.

  - ‘3' (Predefined Filters) - The implementation has populated a set of predefined IndicationFilters for indications that it supports. These should include those specified by SMI-S, but may include vendor specific IndicationFilters that the implementation supports.
  - ‘4' (Client Defined Filters) - The implementation supports client defined IndicationFilters through CreateInstance (and possibly through the CreateAndSubscribe method). If SupportedFeatures includes ‘4’, but SupportedSynchronousActions does not include ‘4’, it means that only CreateInstance is supported for creation of client defined filters.
  - ‘5‘ (Predefined Filter Collections) - The implementation has collected its predefined IndicationFilters into FilterCollections. Each predefined FilterCollection corresponds to an SMI-S profile (autonomous or

component) and is structured as a two level hierarchy. The top FilterCollection contains the predefined IndicationFilters of the autonomous profile and FilterCollections for the component profiles.

- '6' (Client Defined Filter Collections) - The implementation supports client defined FilterCollections through CreateInstance (and possibly through the CreateAndSubscribe method). If SupportedFeatures includes '6', but SupportedSynchronousActions does not include '4', it means that only CreateInstance is supported for creation of client defined filter collections.
- '7' (Semi-fixed Indication Filters) - The implementation supports semi-fixed IndicationFilters, which means that it can accept client definitions of filters that fit the pattern defined by the semi-fixed IndicationFilter.

- ElementCapabilities

  This associates the SNIA_IndicationConfigurationService instance to its SNIA_IndicationConfigurationCapabilities instance.

- ServiceAffectsElement

  This associates the IndicationConfigurationService to client defined FilterCollections it manages.

43.2 Fault Management Considerations

EXPERIMENTAL

43.2.1 Indication Correlation

There are cases where many indications are produced in response to a single event. In fact, the indications themselves are correctly viewed as presenting an aspect or view of the event itself and not as a comprehensive representation of the event. AlertIndications provide a means of notification that is direct to the point than life cycle indications, even though the production of life cycle indications are also important. The subtleties of the effect of the event are better communicated through life cycle indications.

A given event, like a network port communication failure, can itself be reported as an AlertIndication. It is also important to communicate the change in status of the port itself through life cycle indications. It is probable that the network port communication failure will cause some function of the device which contains the point to also fail or become degraded. The impact of the failure (or significant state or status change) is of great interest to management clients as it assist in the triage of the error and potentially can also assist HFM aware clients to contain the failure, fence off the failing component, or even prevent a more serious failure of the system in which the component participates, like the failure of business function (like closing the book at quarter end or dropping transactions at Christmas time).

SMI-S provides the mechanism where storage management can be affected without requiring a priori knowledge of the device or application being managed. In this world, the overall system or service component that is most able to assess and report the impact of the failure (or significant state or status change) is the managed device or application itself. Indication correlation provides the mechanism that can be used to asynchronously report the changes brought about by the event.

The mechanism requires that a single indication be the first reporter of the event. This first reporter may be an AlertIndication or a life cycle indication. This indication should report the state or status change caused by the event in the simplest and most direct manner. All other indications that report state or status change and are associated directly to the first reporter indications should correlated to the first reported indication. Indication correlation shall be done by the implementation through reporting the IndicationIdentifier of the correlated and previously produced indications in the CorrelatedIndications array. The elements in the CorrelatedIndications may be in any order. The linkage of indication thusly correlated is like a one-way linked list. The beginning of the correlation link is indicated by the nullness of the CorrelatedIndications property.

Indication correlation shall be accomplished in the path of cause and effect or scoping relationships. If indication B is to correlated to indication A, then the model change reported by B is caused by or is a side-effect of the model change reported by A. Indication correlation shall not be accomplished by sorting the indications to be correlated by PerceivedSeverity. That being said, Indication correlation should not be used to report secondary events, themselves caused by the primary event, and side-effects of the secondary event.

Indication correlation provides important information about the onset of the condition and its immediate impact that may not be retrievable when the client can react. The spread of the effects of the event within a device or application can certainly be faster than maximum speed of the management network.

Indication correlation shall be accomplished through scoping relationships, like the part to group component or dependent to antecedent relationships, or across direct cause and effect relationships for peer components. For example, given that a network port communication failure within a given device causes changes to the status of port, the scoping computer system, the port communications statistics, the status of the network pipe, and the overall communication statistics of the device, then indication correlation shall not report correlation of the network port communication failure to the changes in the overall communications statistics of the device. This requirement is necessary to limit the potentially lengthy correlation and impose undue burden on the implementation without value to the client.

EXPERIMENTAL

43.3 Cascading Considerations

Not Applicable.

43.4 Supported Profiles, Subprofiles and Packages

Related Profiles for Experimental Indication: Not defined in this standard.

43.5 Methods of the Profile

43.5.1 Extrinsic Methods of the Profile

43.5.1.1 TestListener Method

The TestListener method allows a client to test that the ListenerDestination is actually reachable from the implementation. It also provides information, in error cases, on why the test failed (e.g., destination not resolvable, Port not reachable, certificate errors, etc.).

```
uint32 IndicationConfigurationService.TestListener(
   [IN] CIM_ListenerDestination REF Destination );
```

The TestListener method takes as input a reference to a CIM_ListenerDestination class.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the indication was sent. The implementation will send an AlertIndication with the standard message MP22 (Listener Destination Test).

1 - The method is not supported (e.g., the provider does not support the function)

4 - Failed. If the return code is 4, it means that the provider was not able to send the indication.

5 - Invalid parameter. The protocol specified is not a recognized protocol or the destination was not in a valid URI format.

43.5.1.2 CreateAndSubscribe Method

The CreateAndSubscribe method allows a client to create IndicationFilters, ListenerDestinations and Subscriptions in a single extrinsic call. If the instance supplied as input already exist (e.g., an IndicationFilter, FilterCollection and ListenerDestination), then the only element created is the subscription.

```
uint32 IndicationConfigurationService.CreateAndSubscribe(
   [IN, EmbeddedInstance(“CIM_IndicationFilter")]
       string IndicationFilter,
   [IN, EmbeddedInstance(“CIM_FilterCollection")]
       string FilterCollection,
   [IN, EmbeddedInstance(“CIM_ListenerDestination")]
       string ListenerDestination,
   [IN, EmbeddedInstance(“CIM_AbstractIndicationSubscription")]
       string SubscriptionData,
   [OUT] CIM_AbstractIndicationSubscription REF Subscription);
```

The CreateAndSubscribe method takes as input a set of 4 embedded Instances provided by the client application:

- CIM_IndicationFilter - This would be filled in if the desired output is an instance of CIM_IndicationSubscription. The method will use properties of the embedded instance to determine if the IndicationFilter exists. If it does not exist and IndicationConfigurationService.SupportedFeatures includes ‘4’, then the method will create the instance.
- CIM_FilterCollection - This would be filled in if the desired output is an instance of CIM_FilterCollectionSubscription. The method will use properties of the embedded instance to determine if the FilterCollection exists. If it does not exist and IndicationConfigurationService.SupportedFeatures includes ‘6’, the method will create the instance (but it will be an empty collection).
- CIM_ListenerDestination - The method will use properties of the embedded instance to determine if the ListenerDestination exists. If it does not exist, the method will create the instance.
- CIM_AbstractIndicationSubscription - As an input embedded instance, the references should be NULL. The other properties of the instance will be used to establish the properties of the subscription created.

The CreateAndSubscribe method output (Subscription) is a reference to the created subscription which can be either an IndicationSubscription or a FilterCollectionSubscription.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the subscription (and any related instances) has been created.

1 - Not Supported. The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘4’.

4 - Failed. If the return code is 4, it means that the provider was not able to create the subscription.

5 - Invalid Parameter. The implementation does not recognize the value of a parameter.

43.5.1.3 CreateFilterCollection

The CreateFilterCollection method allows a client to create a client specific collection of IndicationFilters (and/or other filter collections) for the purpose of subscribing to all collected indications with one subscription to the FilterCollection.

```
uint32 IndicationConfigurationService.CreateFilterCollection(
[IN] string FilterCollectionName,
```

```
        [IN] CIM_ManagedElement REF Members[],
        [OUT] CIM_FilterCollection REF FilterCollection );
```

The CreateFilterCollection method takes as input the name of the collection and a list of members to be added to the collection:

- FilterCollectionName - FilterCollectionName takes the form ‘OrgID:CollectionID’ where OrgID is the client vendor and product is part of CollectionID. Predefined collections have an OrgID of ‘SNIA’, so a client defined filter collection shall not use the prefix ‘SNIA’.
- Members[] - This is a list of references to instances of either CIM_IndicationFilter or CIM_FilterCollection. A Member may be an IndicationFilter (a ManagedElement) or another FilterCollection (a Collection).

The CreateFilterCollection method output (FilterCollection) is a reference to the created FilterCollection.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the FilterCollection has been created and the specified members have been added to the collection.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to create the FilterCollection.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, the FilterCollectionName has a prefix of ‘SNIA’ or one of the Members is not a CIM_IndicationFilter or CIM_FilterCollection.

43.5.1.4 AddFilterToCollection

The AddFilterToCollection method allows a client to add members to an existing client defined FilterCollection.

```
        uint32 IndicationConfigurationService.AddFilterToCollection(
           [IN] CIM_ManagedElement REF Members[],
           [IN] CIM_FilterCollection REF FilterCollection );
```

The AddFilterToCollection method takes as input

- Members[] - This is a list of references to instances of either CIM_IndicationFilter or CIM_FilterCollection to be added to the FilterCollection.
- FilterCollection - This is a reference to the (Client Defined) FilterCollection to which the new members are to be added.

The AddFilterToCollection method output is simply success or failure.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the Members have been added to the FilterCollection.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to add the members to the FilterCollection and none of the additions were done.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, one of the Members is not a CIM_IndicationFilter or CIM_FilterCollection.

43.5.1.5 RemoveFilterFromCollection

The RemoveFilterFromCollection method allows a client to remove filters from client defined collections.

```
uint32 IndicationConfigurationService.RemoveFilterFromCollection(
   [IN] CIM_ManagedElement REF Members[],
   [IN] CIM_FilterCollection REF FilterCollection );
```

The RemoveFilterFromCollection method takes as input

- Members[] - This is a list of references to instances of either CIM_IndicationFilter or CIM_FilterCollection to be removed to the FilterCollection.
- FilterCollection - This is a reference to the (Client Defined) FilterCollection to which the members are to be removed.

The RemoveFilterFromCollection method output is simply success or failure.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the Members have been removed to the FilterCollection.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to remove the members to the FilterCollection and none of the removals were done.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, one of the Members is not a CIM_IndicationFilter or CIM_FilterCollection.

43.5.1.6 DeleteFilterCollection

The DeleteFilterCollection method allows a client to remove a client defined FilterCollection.

```
uint32 IndicationConfigurationService.DeleteFilterCollection(
   [IN, Required] CIM_FilterCollection REF FilterCollection.
   [IN] boolean RemoveMembers );
```

The DeleteFilterCollection method takes as input

- FilterCollection - This is a reference to the (Client Defined) FilterCollection to which the members are to be removed.
- RemoveMembers - This boolean, when “true” means the method should imply removal of existing members of the collection. When “false”, the method will fail if members exist in the collection.

The DeleteFilterCollection method output (FilterCollection) is simply success or failure.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the FilterCollection has been deleted.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to delete the FilterCollection. For example, this method will return this error if RemoveMembers is ‘false’ and there are members in the FilterCollection.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, the FilterCollectionName has a prefix of 'SNIA'. That is, this method will return this error if the application attempts to delete a predefined FilterCollection.

43.5.2 Intrinsic Methods of the Profile

The Experimental Indication Profile is mostly populated by providers and is accessible to clients using basic read and association traversal. However, there are three constructs that may be created by Clients. These are the FilterCollection, MemberOfCollection and the FilterCollectionSubscription. In addition to being able to create them, client may delete them (except "pre-defined" filters which cannot be deleted), and a client may modify any IndicationFilter that was client created.

NOTE The IndicationConfigurationService provides extrinsic methods for creating and managing client defined FilterCollections. The intrinsic methods for creating and maintaining FilterCollections is provided for clients that have a preference to use of intrinsic methods.

43.5.2.1 FilterCollection

The CreateInstance and DeleteInstance operations are supported for client defined FilterCollections.

CreateInstance

```
<instanceName>CreateInstance (
        [IN] <instance> NewInstance
 )
```

On CreateInstance on FilterCollections, the implementation should allow NULL to be specified for key properties (the InstanceID). If key properties are passed in on the CreateInstance, the implementation may ignore the keys.

If the client supplies an CollectionName, the implementation shall persist the property for later use by the client.

If successful, the return value defines the object path of the new CIM Instance relative to the target Namespace (i.e., the Model Path).

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_ALREADY_EXISTS (the CIM Instance already exists), CIM_ERR_FAILED (some other unspecified error occurred).

NOTE If a client attempts to create an FilterCollection that already exists (has the same InstanceID), but other properties are different, then the request will fail. If the Client attempts to create an FilterCollection that has identical properties to an existing FilterCollection instance, it will succeed and CreateInstance need not treat the instance as a separate instance.

DeleteInstance

```
void DeleteInstance (
        [IN] <instanceName> InstanceName
)
```

The InstanceName input parameter defines the name (model path) of the Instance to be deleted.

If successful, the specified FilterCollection Instance shall have been removed.

The deletion of a FilterCollection instance will cause the automatic deletion of any associated MemberOfCollection and FilterCollectionSubscription instances.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED (the filter collection is a pre-defined filter collection), CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class does not exist in the specified namespace), CIM_ERR_NOT_FOUND (the CIM Class does exist, but the requested CIM Instance does not exist in the specified namespace), CIM_ERR_FAILED (some other unspecified error occurred).

43.5.2.2 MemberOfCollection

The CreateInstance and DeleteInstance operations are supported on MemberOfCollection for associating client defined FilterCollections and their members.

CreateInstance

```
<instanceName>CreateInstance (
        [IN] <instance> NewInstance
 )
```

An implementation should populate all fields of references (scheme, hostname, port number, namespace, key) in object path for the instance being created.

The host name portion shall be set to a valid, client-resolvable host name (i.e., DNS) or IPv4 or IPv6 address. Internal names (e.g., /etc/hosts) are not valid. If host name, then must be FQDN (not a short name). This implies the provider shall be configured as a valid DNS client to use host names and cannot rely on an administratively defined name.

If successful, the return value defines the object path of the new CIM Instance relative to the target Namespace (i.e., the Model Path).

Note that for CreateInstance of a MemberOfCollection requires that the FilterCollection instance and the member instances (FilterCollection or IndicationFilter) exist.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED (the FilterCollection referenced is a pre-defined filter collection), CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_ALREADY_EXISTS (the CIM Instance already exists), CIM_ERR_FAILED (some other unspecified error occurred).

DeleteInstance

```
void DeleteInstance (
         [IN] <instanceName> InstanceName
)
```

The InstanceName input parameter defines the name (model path) of the Instance to be deleted.

If successful, the specified MemberOfCollection Instance shall have been removed.

Deletion of a MemberOfCollection will not cause the deletion of any corresponding FilterCollection or IndicationFilter instances.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED (the FilterCollection referenced is a pre-defined filter collection), CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class does not exist in the specified namespace), CIM_ERR_NOT_FOUND (the CIM Class does exist, but the requested CIM Instance does not exist in the specified namespace), CIM_ERR_FAILED (some other unspecified error occurred).

43.5.2.3 FilterCollectionSubscription

The CreateInstance and DeleteInstance operations are supported on FilterCollectionSubscription for associating client defined FilterCollections with ListenerDestinations.

CreateInstance

```
<instanceName>CreateInstance (
        [IN] <instance> NewInstance
 )
```

An implementation should populate all fields of references (scheme, hostname, port number, namespace, key) in object path for the instance being created.

The host name portion shall be set to a valid, client-resolvable host name (i.e., DNS) or IPv4 or IPv6 address. Internal names (e.g., /etc/hosts) are not valid. If host name, then must be FQDN (not a short name). This implies the provider shall be configured as a valid DNS client to use host names and cannot rely on an administratively defined name.

If successful, the return value defines the object path of the new CIM Instance relative to the target Namespace (i.e., the Model Path).

Note that for CreateInstance of an FilterCollectionSubscription requires that the ListenerDestinationCIMXML instance and the FilterCollection exist.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_ALREADY_EXISTS (the CIM Instance already exists), CIM_ERR_FAILED (some other unspecified error occurred).

DeleteInstance

```
void DeleteInstance (
         [IN] <instanceName> InstanceName
)
```

The InstanceName input parameter defines the name (model path) of the Instance to be deleted.

If successful, the specified FilterCollectionSubscription Instance shall have been removed.

Deletion of a FilterCollectionSubscription will not cause the deletion of any corresponding ListenerDestinationCIMXML or FilterCollection instances.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class does not exist in the specified namespace), CIM_ERR_NOT_FOUND (the CIM Class does exist, but the requested CIM Instance does not exist in the specified namespace), CIM_ERR_FAILED (some other unspecified error occurred).

43.6 Client Considerations and Recipes

The use cases in the following sections illustrate some of the features (and particularly the methods) of this profile.

43.6.1 Testing a Listener Destination

Table 426 identifies the elements of the use case to test whether a listener destination can receive indications from a CIM Server.

Table 426 - Test that a Listener Destination if Functioning Properly

Use Case Element	Description
Summary	Given an application that listens for indications, test whether or not a CIM_Server can successfully communicate with the application.
Basic Course of Events	1. Start the application that is listening for indications 2. Tell the CIM_Server to send an indication to the listener application 3. Verify the CIM_Server sent the indication 4. Verify that the listening application received the indication
Alternative Paths	None
Exception Paths	None
Triggers	Installing a new or upgraded application that listens for indications.
Assumptions	None
Preconditions	The CIM Server is operational and supports a profile with support for the TestListener function.
Postconditions	The listener application receives the indication and displays it (or logs it).

43.6.2 Discovering predefined IndicationFilters of an implementation

Table 427 identifies the elements of the use case to discover predefined indication filters supported by a profile implementation.

Table 427 - Discovery of Predefined IndicationFilters

Use Case Element	Description
Summary	Given an implementation of an autonomous Profile and its top level ComputerSystem, determine any predefined IndicationFilters it has.

Table 427 - Discovery of Predefined IndicationFilters

Use Case Element	Description
Basic Course of Events	1. Determine if the implementation has IndicationConfigurationCapabilities 2. If it does, verify that it supports Predefined Indications and/or Predefined FilterCollections 2a. If Predefined FilterCollections are supported, then look for the Top Level FilterCollection and determine the predefined IndicationFilters supported 2b. If FilterCollections are not supported (or the IndicationConfigurationService is not supported), then simply enumerate CIM_IndicationFilter in the namespace of the top level ComputerSystem
Alternative Paths	None
Exception Paths	None
Triggers	The administrator (or application) wants to inspect filters that are declared to be supported by an implementation.
Assumptions	None
Preconditions	The top level system of the profile has been discovered from profile registration and ElementConformsToProfile.
Postconditions	A list of predefined IndicationFilters (possibly by Profile) is produced.

43.6.3 Creating a subscription to a predefined IndicationFilter

Table 428 identifies the elements of the use case to create a subscription to a predefined indication filter.

Table 428 - Create a subscription to a predefined indication filter

Use Case Element	Description
Summary	Given a ListenerDestination and a predefined indication filter, subscribe to the filter
Basic Course of Events	1. See if the implementation supports the CreateAndSubscribe method 2. Retrieve the predefined IndicationFilter 2a. If CreateAndSubscribe is supported, copy properties of the predefined indication filter into and embedded instance and invoke CreateAndSubscribe passing the Destination of the Listener 2b. If CreateAndSubscribe is not supported, Create the ListenerDestination and Create the IndicationSubscription
Alternative Paths	None
Exception Paths	None
Triggers	Set up a listener to get indications for an indication in the SMI-S Specification.
Assumptions	None
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The subscription is recorded in the CIM Server.

43.6.4 Creating a client defined indication and subscription

Table 429 identifies the elements of the use case to create an indication filter and subscribe to it.

Table 429 - Create an IndicationFilter and subscribe to it

Use Case Element	Description
Summary	Given a top level system of an autonomous profile and a URI for an indication listener create a client defined indication and subscribe to it.
Basic Course of Events	1. Get the IndicationConfigurationService if it exists 1b. If not try to do a CreateInstance for the IndicationFilter (if it succeeds, then continue) 2. If the service exists, get the IndicationConfigurationCapabilities to find out if the implementation supports client defined IndicationFilters 3. If the capability exists, then do a CreateAndSubscribe for the indication 3b. Do a CreateInstance on the ListenerDestination and another CreateInstance on the IndicationSubscription
Alternative Paths	1. Create a FilterCollection and put the client defined indication filter in that collection
Exception Paths	None
Triggers	The administrator wants to listen for a specific indication of his/her choosing.
Assumptions	The implementation supports client defined IndicationFilters
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The IndicationFilter is created and a subscription to it is recorded in the CIM Server.

43.6.5 Creating a semi-fixed indication filter

Table 430 identifies the elements of the use case to create a semi-fixed indication filter.

Table 430 - Creation of a semi-fixed Indication filters

Use Case Element	Description
Summary	A client application wants to create an indication filter that has application specific information to include in the filter.
Basic Course of Events	1. Determine if the implementation supports semi-fixed IndicationFilters 2. Find the semi-fixed IndicationFilter 2a. Look in the 'SNIA' FilterCollections if they exist 2b. Enumerate predefined indication filters if not 3. Do a CreateAndSubscribe for the IndicationFilter substituting the application specific information in the query 3a. Do a CreateInstance on the filter and the subscription if CreateAndSubscribe is not supported.
Alternative Paths	1. If none of the new functions are supported try to create the filter using a CreateInstance
Exception Paths	None
Triggers	The application wants to restrict the number of indications it receives by adding application specific information to the filter query.
Assumptions	The implementation supports semi-fixed IndicationFilters
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The IndicationFilter is created and a subscription to it is recorded in the CIM Server.

43.6.6 Creating a FilterCollection

Table 431 identifies the elements of the use case to create a client defined Filter Collection.

Table 431 - Creation of a client defined FilterCollection

Use Case Element	Description
Summary	Given a top level system of an autonomous profile and a URI for an indication listener and a list of IndicationFilter that the administrator wants to listen for create a client defined FilterCollection and subscribe to it.
Basic Course of Events	1. Determine if the implementation supports client defined FilterCollections 1a. If not quit, you have no options. 2. Determine if the implementation supports FilterCollection Methods 2a. If not go to 3a 3. Do a CreateFilterCollection to create the FilterCollection 3a. Do a CreateInstance on the FilterCollection and a bunch of CreateInstances for the MemberOfCollection associations to each of the IndicationFilters.Then do a CreateInstance on FilterCollectionSubscription.
Alternative Paths	None
Exception Paths	None
Triggers	The client was subscriptions to IndicationFilters in a client specific list of filters.
Assumptions	None
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The FilterCollection is created and a subscription to it is recorded in the CIM Server.

43.7 Registered Name and Version

Experimental Indication version 1.5.0 (Component Profile)

Specializes SNIA Indication version 1.5.0

43.8 CIM Elements

Table 432 describes the CIM elements for Experimental Indication.

Table 432 - CIM Elements for Experimental Indication

Element Name	Requirement	Description
43.8.1 CIM_AlertIndication	Optional	This is a specialization of the CIM_AlertIndication class in the Indication Profile.
43.8.2 CIM_ElementCapabilities (Indication Config Service to Capabilities)	Mandatory	Experimental. This associates the IndicationConfigurationService to the IndicationConfigurationCapabilities.
43.8.3 CIM_FilterCollection (Client Defined)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections). This is a client defined collection of IndicationFilters to which a client may subscribe.

Table 432 - CIM Elements for Experimental Indication

Element Name	Requirement	Description
43.8.4 CIM_FilterCollectionSubscription (Filter Collection Subscription)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections) or '6' (Client Defined Filter Collections). This associates the FilterCollection to the system in the referencing profile.
43.8.5 CIM_HostedCollection (Hosted Filter Collection)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections). This associates a client defined FilterCollection to the system in the referencing profile.
43.8.6 CIM_HostedService (Indication Config Service to System)	Mandatory	Experimental. This associates the IndicationConfigurationService to the System in the referencing profile.
43.8.7 CIM_IndicationFilter (client defined)	Optional	This is for 'client defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the value and format of an indication filter string.
43.8.8 CIM_IndicationFilter (pre-defined)	Optional	This is for 'pre-defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the value and format of an indication filter string.
43.8.9 CIM_IndicationSubscription	Mandatory	This association is a specialization of the IndicationSubscription as defined in the Indication Profile.
43.8.10 CIM_InstCreation	Optional	This is a specialization of the CIM_InstCreation class in the Indication Profile.
43.8.11 CIM_InstDeletion	Optional	This is a specialization of the CIM_InstDeletion class in the Indication Profile.
43.8.12 CIM_InstModification	Optional	This is a specialization of the CIM_InstModification class in the Indication Profile.
43.8.13 CIM_ListenerDestinationCIMXML (Indication Handler)	Mandatory	A CIM_ListenerDestinationCIMXML describes the destination for CIM Export Messages to be delivered via CIM-XML. ListenerDestinationCIMXML is subclassed from ListenerDestination.
43.8.14 CIM_ListenerDestinationWSManagement (WS-Man Indication Handler)	Optional	Experimental. A CIM_ListenerDestinationWSManagement describes the destination for CIM Export Messages to be delivered via WS-Man.
43.8.15 CIM_MemberOfCollection (Filter Collection to Filters)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections). This associates a client defined FilterCollection to the Filters in the collection.
43.8.16 SNIA_IndicationConfigurationCapabilities	Mandatory	Experimental. This is the capabilities of the implementation of indications.
43.8.17 SNIA_IndicationConfigurationService	Mandatory	Experimental. This is the indication services of the implementation.

Table 432 - CIM Elements for Experimental Indication

Element Name	Requirement	Description
43.8.18 SNIA_IndicationFilterTemplate (semi-fixed)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='7' (Semi-fixed Indication Filters). This is a template for 'semi-fixed' IndicationFilter instances.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FilterCollectionSubscription	Optional	Experimental. CQL -This indicates that a subscription to a FilterCollection has been deleted by either explicit user action (DeleteInstance) or by provider clean up.

43.8.1 CIM_AlertIndication

This is a specialization of the CIM_AlertIndication class in the Indication Profile. The class definition specializes the CIM_AlertIndication definition in the Indication profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 433 describes class CIM_AlertIndication.

Table 433 - SMI Referenced Properties/Methods for CIM_AlertIndication

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier (overridden)		Mandatory	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime	N	Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
Description		Optional	A free form text description.
AlertingManagedElement (overridden)		Mandatory	The identifying information of the entity for which this Indication is generated. If the element in question is modeled by the profile implementation, then the format for this property should be as a Typed WBEM URI as defined in DSP0207.
AlertingElementFormat		Mandatory	Valid SMI-S values are 0|1|2 ('Unknown' | 'Other' | 'CIMObjectPath').
AlertType		Mandatory	This shall be 1|2|3|4|5|6|7|8 ('Other' | 'Communications Alert' | 'Quality of Service Alert' | 'Processing Error' | 'Device Alert' | 'Environmental Alert' | 'Model Change' | 'Security Alert').
OtherAlertType		Optional	
PerceivedSeverity		Mandatory	This shall be 0|1|2|3|4|5|6|7 ('Unknown', 'Other' | 'Information' | 'Degraded/Warning' | 'Minor' | 'Major' | 'Critical' | 'Fatal/NonRecoverable').
OtherSeverity		Optional	
ProbableCause		Mandatory	Many possible values in a value map. See MOF.
ProbableCauseDescription		Optional	
EventID		Optional	

Table 433 - SMI Referenced Properties/Methods for CIM_AlertIndication

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	The scoping System's Name for the Provider generating this Indication. The SystemName would typically be the name of the system that generates the indication.
ProviderName		Mandatory	
OwningEntity (overridden)	N	Mandatory	A string that uniquely identifies the entity that owns the definition of the format of the Message.
MessageID (overridden)	N	Mandatory	A string that uniquely identifies, within the scope of the OwningEntity, the format of the Message.
Message (overridden)	N	Mandatory	The formatted message (including the MessageArguments).
MessageArguments	N	Optional	An array of strings that contain the dynamic content of the message.
OtherAlertingElementFormat	N	Optional	Not Specified in this version of the Profile.
Trending	N	Optional	Not Specified in this version of the Profile.
RecommendedActions	N	Optional	Not Specified in this version of the Profile.
EventTime	N	Optional	Not Specified in this version of the Profile.
IndicationFilterName (added)		Mandatory	The list of IndicationFilter.Name values for IndicationFilters or FilterCollections that this indication supports.

43.8.2 CIM_ElementCapabilities (Indication Config Service to Capabilities)

Experimental. This associates the IndicationConfigurationService to the IndicationConfigurationCapabilities.

Requirement: Mandatory

Table 434 describes class CIM_ElementCapabilities (Indication Config Service to Capabilities).

Table 434 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Indication Config Service to Capabilities)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The indication capabilities instance associated with the indication configuration service.
ManagedElement		Mandatory	The indication configuration service.

43.8.3 CIM_FilterCollection (Client Defined)

Experimental. This is a client defined collection of IndicationFilters to which a client may subscribe. An implementation would indicate support for client defined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '6' (Client Defined Filter Collections).

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections).

Table 435 describes class CIM_FilterCollection (Client Defined).

Table 435 - SMI Referenced Properties/Methods for CIM_FilterCollection (Client Defined)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be constructed using the following algorithm: OrgID:CollectionID where OrgID and CollectionID are separated by a colon ':'.

43.8.4 CIM_FilterCollectionSubscription (Filter Collection Subscription)

Experimental. This associates the FilterCollection to the system in the referencing profile.

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections) or '6' (Client Defined Filter Collections).

Table 436 describes class CIM_FilterCollectionSubscription (Filter Collection Subscription).

Table 436 - SMI Referenced Properties/Methods for CIM_FilterCollectionSubscription (Filter Collection Subscription)

Properties	Flags	Requirement	Description & Notes
OnFatalErrorPolicy		Mandatory	
OtherOnFatalErrorPolicy		Optional	This should contain a description of the fatal error policy if OnFatalErrorPolicy=1 (Other).
FailureTriggerTimeInterval		Mandatory	Specifies minimum delay before OnFatalErrorPolicy is implemented.
SubscriptionState		Mandatory	
OtherSubscriptionState		Optional	This should contain a description of the subscription state if SubscriptionState=1 (Other).
RepeatNotificationPolicy		Mandatory	This shall be 2 (None), 3 (Suppress), or 4 (Delay).
RepeatNotificationInterval		Optional	This should be provided if the value of RepeatNotificationPolicy is 3 (Suppress) or 4 (Delay).
RepeatNotificationGap		Optional	This should be provided if the value of RepeatNotificationPolicy is 4 (Delay).
RepeatNotificationCount		Optional	This should be provided if the value of RepeatNotificationPolicy is 3 (Suppress) or 4 (Delay).
Filter		Mandatory	Reference to the FilterCollection.
Handler		Mandatory	Reference to the ListenerDestination.

43.8.5 CIM_HostedCollection (Hosted Filter Collection)

Experimental. This associates a client defined FilterCollection to the system in the referencing profile.

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections).

Table 437 describes class CIM_HostedCollection (Hosted Filter Collection).

Table 437 - SMI Referenced Properties/Methods for CIM_HostedCollection (Hosted Filter Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the FilterCollection.
Antecedent		Mandatory	Reference to the 'Top level' System that hosts the collection.

43.8.6 CIM_HostedService (Indication Config Service to System)

Experimental. This associates the IndicationConfigurationService to the System in the referencing profile.

Requirement: Mandatory

Table 438 describes class CIM_HostedService (Indication Config Service to System).

Table 438 - SMI Referenced Properties/Methods for CIM_HostedService (Indication Config Service to System)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting system.
Dependent		Mandatory	The Indication configuration service hosted on the system.

43.8.7 CIM_IndicationFilter (client defined)

CIM_IndicationFilter instances that are 'client defined' are IndicationFilters that are be created by a client using CreateInstance. If a profile implementation can support client defined IndicationFilters, the implementation would support 'client defined' IndicationFilter instances. The implementation shall support 'client defined' filters that are defined by SMI-S profile as mandatory, but may also support additional filters supported by the implementation (See QueryCapabilities).

CIM_IndicationFilter is subclassed from CIM_ManagedElement.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Optional

Table 439 describes class CIM_IndicationFilter (client defined).

Table 439 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This should take the form OrgID ":" RegisteredName ":" UniqueID. For more details, see section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 42.1.1 IndicationFilter Names.

Table 439 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

Properties	Flags	Requirement	Description & Notes
SourceNamespace	N	Optional	Deprecated. For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Mandatory	This should be all the namespaces where the indications may originate.
Query		Mandatory	A string that specifies (in QueryLanguage terms) which indications are to be delivered to the ListenerDestinations.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName		Optional	A Client Defined user friendly string that identifies the Indication Filter.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

43.8.8 CIM_IndicationFilter (pre-defined)

CIM_IndicationFilter instances that are 'pre-defined' are IndicationFilters that are populated automatically by the profile provider. If a profile implementation cannot support client defined IndicationFilters, the implementation can populate its model with 'pre-defined' IndicationFilter instances. 'Pre-defined' filters shall include those that are required by the profile, but may also contain additional filters supported by the implementation.

CIM_IndicationFilter is subclassed from CIM_ManagedElement.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 440 describes class CIM_IndicationFilter (pre-defined).

Table 440 - SMI Referenced Properties/Methods for CIM_IndicationFilter (pre-defined)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This should take the form OrgID ":" RegisteredName ":" UniqueID. For more details, see section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 42.1.1 IndicationFilter Names.
SourceNamespace	N	Optional	Deprecated. For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Mandatory	This should be all the namespaces where the indications may originate.

Table 440 - SMI Referenced Properties/Methods for CIM_IndicationFilter (pre-defined)

Properties	Flags	Requirement	Description & Notes
Query		Mandatory	A string that specifies (in QueryLanguage terms) which indications are to be delivered to the ListenerDestinations.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	SMI-S does not specify this property for pre-defined IndicationFilters.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

43.8.9 CIM_IndicationSubscription

A CIM_IndicationSubscription is not subclassed from anything. The class definition specializes the CIM_IndicationSubscription definition in the Indication profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: CreateInstance

Modified By: Static

Deleted By: DeleteInstance

Requirement: Mandatory

Table 441 describes class CIM_IndicationSubscription.

Table 441 - SMI Referenced Properties/Methods for CIM_IndicationSubscription

Properties	Flags	Requirement	Description & Notes
RepeatNotificationPolicy		Mandatory	SMI-S supports a restricted set of values. This shall be 2|3|4 ('None' | 'Suppress' | 'Delay').
RepeatNotificationInterval		Optional	Mandatory if the RepeatNotificationPolicy is 'Suppress' or 'Delay'.
RepeatNotificationGap		Optional	Mandatory if the RepeatNotificationPolicy is 'Delay'.
RepeatNotificationCount		Optional	Mandatory if the RepeatNotificationPolicy is 'Suppress' or 'Delay'.
LastIndicationIdentifier (overridden)		Optional	The IndicationIdentifier of the last indication produced for this subscription regardless if that indication were delivered.
LastIndicationProductionDateTime (overridden)		Optional	The date and time of the production of the last indication produced for this subscription regardless if that indication were delivered.
OnFatalErrorPolicy	N	Optional	Not Specified in this version of the Profile.
OtherOnFatalErrorPolicy	N	Optional	Not Specified in this version of the Profile.
FailureTriggerTimeInterval	N	Optional	Not Specified in this version of the Profile.
SubscriptionState	N	Optional	Not Specified in this version of the Profile.
OtherSubscriptionState	N	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	N	Optional	Not Specified in this version of the Profile.
SubscriptionDuration	N	Optional	Not Specified in this version of the Profile.
SubscriptionStartTime	N	Optional	Not Specified in this version of the Profile.
SubscriptionTimeRemaining	N	Optional	Not Specified in this version of the Profile.

Table 441 - SMI Referenced Properties/Methods for CIM_IndicationSubscription

Properties	Flags	Requirement	Description & Notes
OtherRepeatNotificationPolicy	N	Optional	Not Specified in this version of the Profile.
AlertOnStateChange	N	Optional	Not Specified in this version of the Profile.
Filter (overridden)		Mandatory	
Handler (overridden)		Mandatory	

43.8.10CIM_InstCreation

This is a specialization of the CIM_InstCreation class in the Indication Profile. The class definition specializes the CIM_InstCreation definition in the Indication profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 442 describes class CIM_InstCreation.

Table 442 - SMI Referenced Properties/Methods for CIM_InstCreation

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier (overridden)		Mandatory	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime		Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
SourceInstance		Mandatory	A copy of the instance that changed to generate the Indication. SourceInstance contains the current values of the properties selected by the Indication Filter's Query.
SourceInstanceModelPath		Mandatory	The Model Path of the SourceInstance.
PerceivedSeverity	N	Optional	Not Specified in this version of the Profile.
OtherSeverity	N	Optional	Not Specified in this version of the Profile.
SourceInstanceHost	N	Optional	Not Specified in this version of the Profile.
IndicationFilterName (added)		Mandatory	The list of IndicationFilter.Name values for IndicationFilters or FilterCollections that this indication supports.

43.8.11CIM_InstDeletion

This is a specialization of the CIM_InstDeletion class in the Indication Profile. The class definition specializes the CIM_InstDeletion definition in the Indication profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 443 describes class CIM_InstDeletion.

Table 443 - SMI Referenced Properties/Methods for CIM_InstDeletion

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier (overridden)		Mandatory	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime		Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
SourceInstance		Mandatory	A copy of the instance that changed to generate the Indication. SourceInstance contains the current values of the properties selected by the Indication Filter's Query.
SourceInstanceModelPath		Mandatory	The Model Path of the SourceInstance.
PerceivedSeverity	N	Optional	Not Specified in this version of the Profile.
OtherSeverity	N	Optional	Not Specified in this version of the Profile.
SourceInstanceHost	N	Optional	Not Specified in this version of the Profile.
IndicationFilterName (added)		Mandatory	The list of IndicationFilter.Name values for IndicationFilters or FilterCollections that this indication supports.

43.8.12CIM_InstModification

This is a specialization of the CIM_InstModification class in the Indication Profile. The class definition specializes the CIM_InstModification definition in the Indication profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 444 describes class CIM_InstModification.

Table 444 - SMI Referenced Properties/Methods for CIM_InstModification

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier (overridden)		Mandatory	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
IndicationTime		Mandatory	The time and date of creation of the Indication. The property may be set to NULL if it cannot be determined.
SourceInstance		Mandatory	A copy of the instance that changed to generate the Indication. SourceInstance contains the current values of the properties selected by the Indication Filter's Query.
SourceInstanceModelPath		Mandatory	The Model Path of the SourceInstance.
PreviousInstance		Optional	A copy of the 'previous' instance whose change generated the Indication. PreviousInstance contains 'older' values of an instance's properties (as compared to SourceInstance), selected by the IndicationFilter's Query.
PerceivedSeverity	N	Optional	Not Specified in this version of the Profile.

Table 444 - SMI Referenced Properties/Methods for CIM_InstModification

Properties	Flags	Requirement	Description & Notes
OtherSeverity	N	Optional	Not Specified in this version of the Profile.
SourceInstanceHost	N	Optional	Not Specified in this version of the Profile.
IndicationFilterName (added)		Mandatory	The list of IndicationFilter.Name values for IndicationFilters or FilterCollections that this indication supports.

43.8.13CIM_ListenerDestinationCIMXML (Indication Handler)

CIM_ListenerDestinationCIMXML is subclassed from CIM_ListenerDestination.

Created By: CreateInstance
Modified By: Static
Deleted By: DeleteInstance
Requirement: Mandatory

Table 445 describes class CIM_ListenerDestinationCIMXML (Indication Handler).

Table 445 - SMI Referenced Properties/Methods for CIM_ListenerDestinationCIMXML (Indication Handler)

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A client defined user friendly string that identifies the CIMXML Listener destination.
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
PersistenceType		Mandatory	For SMI-S, this shall be 2|3 ('permanent' | 'transient').
Destination		Mandatory	The destination URL to which CIM-XML Export Messages are to be delivered. The scheme prefix shall be consistent with the DMTF CIM-XML specifications.If a scheme prefix is not specified, the scheme \http:\'shallbeassumed.'
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
OtherPersistenceType	N	Optional	Not Specified in this version of the Profile.

43.8.14CIM_ListenerDestinationWSManagement (WS-Man Indication Handler)

Experimental. CIM_ListenerDestinationWSManagement is subclassed from CIM_ListenerDestination.

Created By: CreateInstance
Modified By: Static
Deleted By: DeleteInstance
Requirement: Optional

Table 446 describes class CIM_ListenerDestinationWSManagement (WS-Man Indication Handler).

Table 446 - SMI Referenced Properties/Methods for CIM_ListenerDestinationWSManagement (WS-Man Indication Handler)

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A user-friendly string that describes the destination.
SystemCreationClassName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
SystemName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
CreationClassName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
Name		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
PersistenceType		Mandatory	For SMI-S, this shall be 2|3 ('permanent' | 'transient').
Destination		Mandatory	The value shall be a valid IETF Uniform Resource Identifier (URI) value.
ProtocolType		Mandatory	For WS-Man, this shall be '4' (WS-Management).

43.8.15CIM_MemberOfCollection (Filter Collection to Filters)

Experimental. This associates a client defined FilterCollection to the Filters in the collection.

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections).

Table 447 describes class CIM_MemberOfCollection (Filter Collection to Filters).

Table 447 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Filter Collection to Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the FilterCollection.
Member		Mandatory	Reference to the IndicationFilter.

43.8.16SNIA_IndicationConfigurationCapabilities

Experimental. This is the capabilities of the implementation of indications.

Requirement: Mandatory

Table 448 describes class SNIA_IndicationConfigurationCapabilities.

Table 448 - SMI Referenced Properties/Methods for SNIA_IndicationConfigurationCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Optional	This is a user friendly name of the capabilities instance.

Table 448 - SMI Referenced Properties/Methods for SNIA_IndicationConfigurationCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedFeatures		Mandatory	This may be any or all of the following values: '2' (none), '3' (Predefined Filters), '4' (Client Defined Filters), '5' (Predefined Filter Collections), '6' (Client Defined Filter Collections) or '7' (Semi-fixed Indication Filters).
SupportedSynchronousActions		Mandatory	This shall be '2' (none), '3' (Test Listener), '4' ("Create and Subscribe) or '5' (Filter Collection Methods).

43.8.17SNIA_IndicationConfigurationService

Experimental. This is the indication services of the implementation.

Requirement: Mandatory

Table 449 describes class SNIA_IndicationConfigurationService.

Table 449 - SMI Referenced Properties/Methods for SNIA_IndicationConfigurationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	
TestListener()		Optional	A method for testing if the listener can receive indications.
CreateAndSubscribe()		Optional	A method for creating a Filter and subscribing to it.
CreateFilterCollection()		Optional	A method for creating a FilterCollection and adding initial members.
AddFilterToCollection()		Optional	A method for adding members to a client defined FilterCollection.
RemoveFilterFromCollection()		Optional	A method for removing members from a client defined FilterCollection.
DeleteFilterCollection()		Optional	A method for Deleting a FilterCollection.

43.8.18SNIA_IndicationFilterTemplate (semi-fixed)

Experimental. IndicationFilter instances that are 'semi-fixed' are IndicationFilters that are be created by a client using CreateInstance, but they follow a pattern defined by an IndicationFilterTemplate. If a profile implementation can support semi-fixed IndicationFilters, the implementation would support 'semi-fixed' IndicationFilterTemplate instances. The implementation shall support 'semi-fixed' filters that are defined by SMI-S profile as mandatory.

SNIA_IndicationFilterTemplate is subclassed from CIM_ManagedElement.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='7' (Semi-fixed Indication Filters).

Table 450 describes class SNIA_IndicationFilterTemplate (semi-fixed).

Table 450 - SMI Referenced Properties/Methods for SNIA_IndicationFilterTemplate (semi-fixed)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This shall take the form OrgID ":" RegisteredName ":" UniqueID. For more details, see section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 43.1.4.1 Naming Conventions for IndicationFilterTemplates and IndicationFilters.
SourceNamespace	N	Optional	Deprecated. For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Optional	Experimental. For instances in the InteropNamespace, this should be all the namespaces where the indications may originate. For instances in the implementation namespaces where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
Query		Mandatory	A string that specifies a template (in QueryLanguage terms with SUBSTITUTION_STRINGs) what IndicationFilters may be created from this template.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL' for CQL queries with substitution.
ElementName		Optional	A Client Defined user friendly string that identifies the Indication Filter.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

DEPRECATED

STABLE

44 Object Manager Adapter Subprofile

44.1 Description

The ObjectManagerAdapter model defines the protocol adapters that are supported for a CIM Server. This model is optional for the CIM Server Profile. If implemented, the ObjectManagerAdapterModel shall adhere to Table 451, “CIM Elements for Object Manager Adapter”.

44.1.1 Instance Diagram

ObjectManagerAdapter Subprofile is not advertised. Figure 65 illustrates the model.

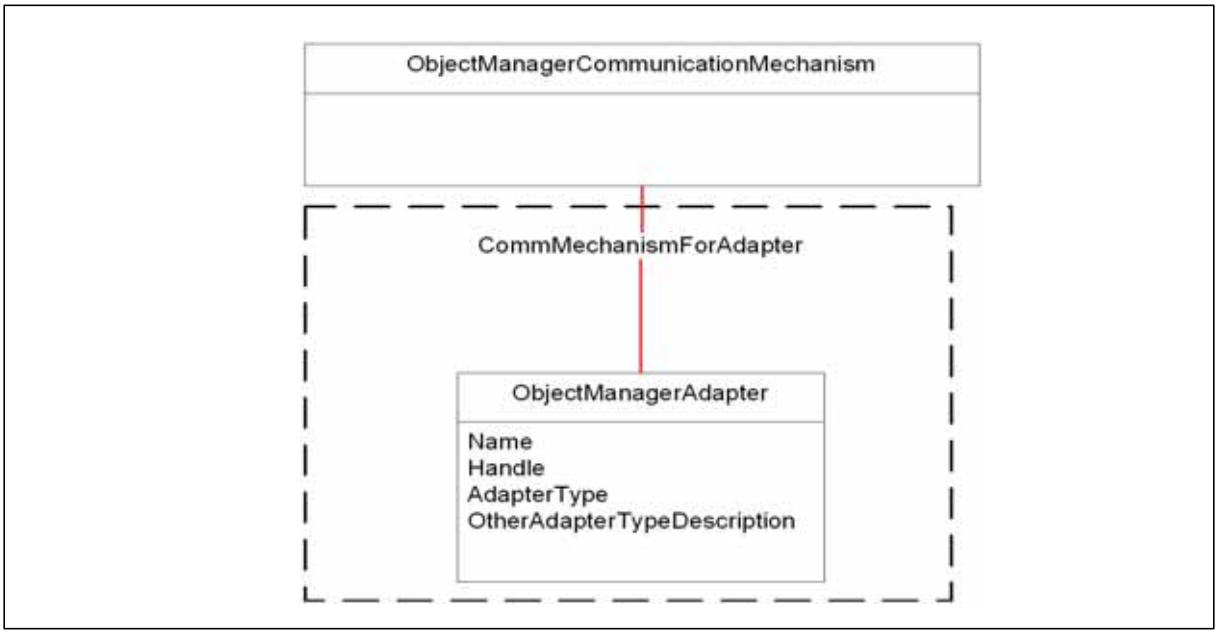

Figure 65 - ObjectManagerAdapter Subprofile Model

44.2 Health and Fault Management

Not defined in this standard.

44.3 Cascading Considerations

Not defined in this standard.

44.4 Supported Subprofiles and Packages

None.

44.5 Methods of the Profile

None.

44.6 Client Considerations and Recipes

None.

44.7 Registered Name and Version

Object Manager Adapter version 1.3.0 (Component Profile)

44.8 CIM Elements

Table 451 describes the CIM elements for Object Manager Adapter.

Table 451 - CIM Elements for Object Manager Adapter

Element Name	Requirement	Description
44.8.1 CIM_CommMechanismForObjectManagerAdapter	Mandatory	
44.8.2 CIM_ObjectManagerAdapter	Mandatory	

44.8.1 CIM_CommMechanismForObjectManagerAdapter

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 452 describes class CIM_CommMechanismForObjectManagerAdapter.

Table 452 - SMI Referenced Properties/Methods for CIM_CommMechanismForObjectManagerAdapter

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The encoding/protocol/set of operations that may be used to communicate between the Object Manager and the referenced ObjectManagerAdapter.
Antecedent		Mandatory	The specific ObjectManagerAdapter whose communication mechanism with the CIM Object Manager is described.

44.8.2 CIM_ObjectManagerAdapter

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 453 describes class CIM_ObjectManagerAdapter.

Table 453 - SMI Referenced Properties/Methods for CIM_ObjectManagerAdapter

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	

Table 453 - SMI Referenced Properties/Methods for CIM_ObjectManagerAdapter

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
Handle		Mandatory	
AdapterType		Mandatory	
OtherAdapterTypeDescription		Optional	
OperationalStatus		Mandatory	
StatusDescriptions		Conditional	Conditional requirement: CIM_ObjectManagerAdapter requires the StatusDescriptions property be populated if the OperationalStatus property has a value of 1 (\Other\).'This shall not be NULL if 'Other' is identified in OperationalStatus.
Started		Mandatory	
StartService()		Mandatory	
StopService()		Mandatory	

STABLE

EXPERIMENTAL

45 Proxy Server System Management Subprofile

45.1 Description

This subprofile addresses the question of how an SMI-S server can discover the devices it is going to manage. Knowledge of the external devices must be set by the client and retained in some fashion by the SMI-S server. Note that the mechanics of storing that information is beyond the scope of this profile. Typically, in order for the SMI-S server to discover and manage these devices, the client will need to provide some connection information, such as IP addresses, and authorization/authentication information (e.g., user name and password) to allow access to the device. SMI-S defines the two roles of SMI-S servers -- Embedded and Proxy. Proxy servers (see 10 SMI-S Roles in *Storage Management Technical Specification, Part 2 Common Architecture*) typically manage one or more devices that are separate from the computer system the proxy is running on. Embedded servers are internal to the device being managed. While it is more likely that the Proxy server will need client help to determine which devices to manage, it may also be the case that Embedded services may also take advantage of this capability. Therefore this profile does not distinguish between Proxy and Embedded servers.

This subprofile defines a new service, SystemRegistrationService with three methods, AddSystem, DiscoverSystems, and RemoveSystem. AddSystem will supply all the parameters, such as IP Address, that will allow the proxy to add the device. DiscoverSystems is similar to AddSystem but relies on the SMI-S server to go out and discover devices that it can manage. The Credential Management and Device Credentials subprofiles will be used to support the passing of the security credentials to the device to be added or discovered. RemoveSystem will remove the device from management by the proxy.

When a system is added to the proxy, it will result in the creation of the top-level system and all the other objects needed to correctly model that system. Similarly, when a system is removed, it results in the deletion of the top-level computer system and corresponding objects. The Client Considerations section below will cover this in more detail.

45.1.1 Relationship to Server Profile

This profile is a component profile (or subprofile) and extends the functionality of the Server Profile, which in turn references this as a component profile. This profile introduces a new Service that is associated to the Server System.

45.1.2 Model

The service shall be modeled as an instance of SNIA_SystemRegistrationService associated to the System that is associated to the ObjectManager via HostedService as defined in the Server Profile. Figure 66 shows the **Proxy Server System Management** model. The service shall have an associated Capabilities object, SNIA_SystemRegistrationCapabilities, that is associated to the service via ElementCapabilities.

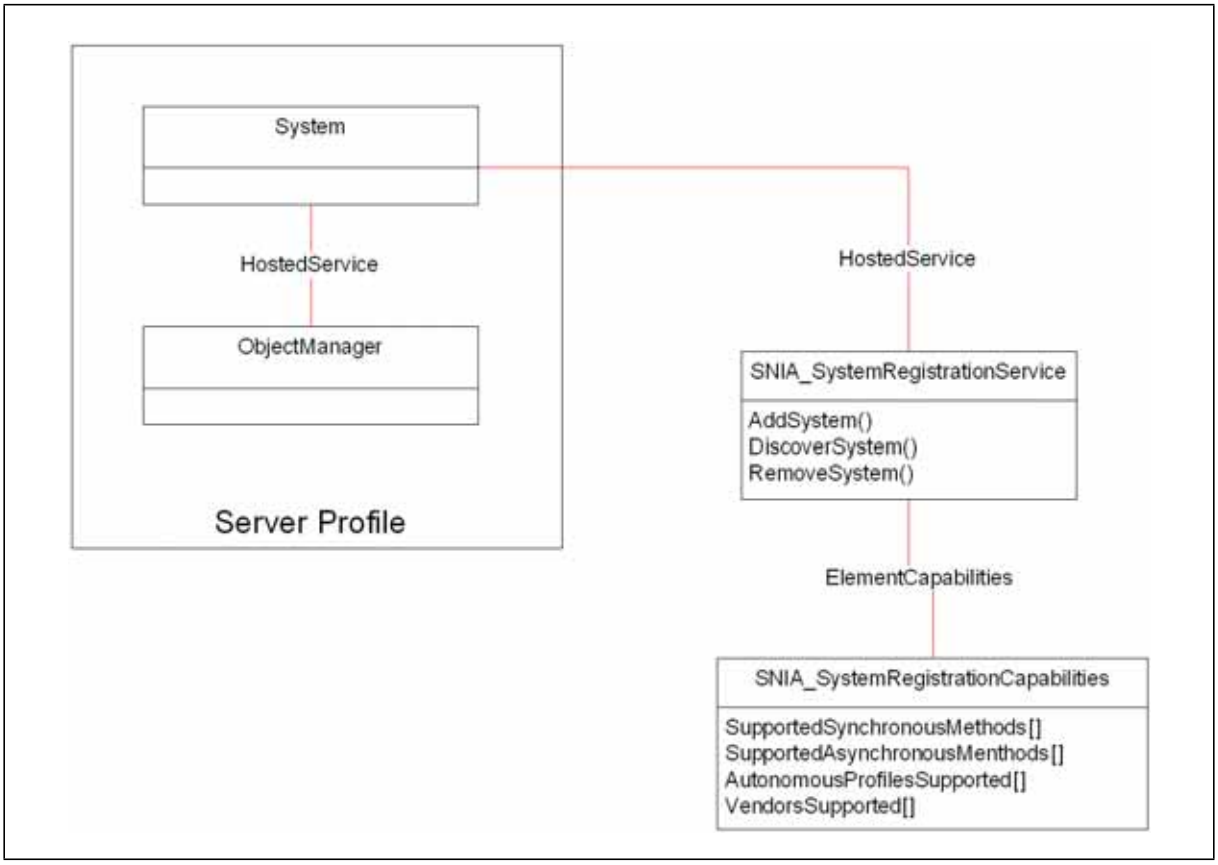

Figure 66 - Proxy Server System Management Model

Table 454 describes each associated capability.

Table 454 - Capabilities

Capability	Description
SupportedSynchronousMethods[]	Lists methods of the profile that do not result in a job being created
SupportedAsynchronousMenthods[]	Lists methods of the profile that do result in a job being created. If a method is listed in both, the client needs to check the Job parameter to see if a job was created
AutonomousProfilesSupported[]	This property identifies the profiles that this service is capable of discovering and managing. For example, a block device could potentially list “Array” and “Storage Virtualizer”. An attempt to discover a different kind of device, like a Fibre Channel switch would fail.
VendorsSupported[]	This property identifies the vendors whose devices this service can discover. For example, if the list contains “Vendor A” then only “Vendor A” devices with the supported autonomous profiles listed in AutonomousProfilesSupported[] can be discovered. Attempting to discover other vendors devices would result in an error. This should include at least the instrumentation vendor, and may include other vendors. (e.g., due to OEM relationships)

45.1.3 Creation Considerations

The methods in this profile shall not support the creation of new namespaces. The namespace supplied to the method shall already exist. The SMI-S server may restrict the namespaces that can be used. An instance of the SNIA_SystemRegistrationService shall be created in each namespace supported and shall be associated to the System in the interop namespace. For some SMI-S servers, addition of a device in one namespace may result in the device being accessible from other namespace. One specific use case for this is where a proxy supports namespaces for prior versions of SMI-S for backwards compatibility with clients. The same code may support these multiple namespaces by default. There is no mechanism in this profile for a client to determine if this is indeed the case.

45.2 Health and Fault Management Consideration

Not defined in this standard.

45.3 Cascading Considerations

Not applicable

45.4 Supported Profiles, Subprofiles, and Packages

Related Profiles for Proxy Server System Management: Not defined in this standard.

45.5 Methods of the Profile

This subprofile defines three new methods, AddSystem, DiscoverSystem, and RemoveSystem. AddSystem will have parameters such as IP Address that will allow the proxy to discover the device. The security aspect needs some refinement. It may take advantage of the Security profiles.

45.5.1 AddSystem

The AddSystem method shall result in the SMI-S server contacting the device and creating the instances necessary to model that device in the requested namespace. If the device has already been added to the SMI-S, then calling this method shall result in an update to the instances already in existence. Note that this may result in the creation of new and deletion of old instances. See 45.6 "Client Considerations and Recipes" for more details on this method call.

Method signature:

uint32 AddSystem(CIM_Job REF Job, String Namespace, String Addresses[], uint32 PortNumbers[], uint16 AddressTypes[], String ElementName, String Description, SharedSecret REF Secret, OUT CIM_System REF AddedSystem)

Table 455 describes the AddSystem Method Parameters.

Table 455 - AddSystem Method Parameters

Parameters	Description
Job	Reference to a Job, if one is created
UseNamespace	Name of the Namespace to create the system in. Namespace must already exist
Addresses[]	Address of the device (e.g., IP address(es) of array controller). Shall have the same number of elements as AddressTypes[]
PortNumbers	Port number to use for each address given. Shall either be null if not applicable or shall ave one entry per entry in the Addresses array
AddressTypes[]	Type of address (valid values are URL, IPAddress, DeviceName, WWN), Each entry in AddressTypes[] is matched with the entry in Addresses[]

Table 455 - AddSystem Method Parameters

Parameters	Description
ElementName	User-friendly name to give to the system
Description	Description to use for the system
Secret	Reference to previously created SharedSecret to pass along to device
AddedSystem	Reference to system added

Table 456 presents the valid return codes.

Table 456 - AddSystem Return Codes

Value	Description
0: Success	Job completed with no error.
1: Not Supported	Not supported
2: Unknown	Unknown error occurred
3: Timeout	Timeout
4: Failed	Method failed.
5: Invalid Parameter	
6: In Use	Element is in use and cannot be modified
Invalid namespace	Namespace supplied does not exist or service not supported for that namespace
Device profile not supported	Device at the address specified does not support any of the autonomous profiles supported
Vendor not supported	Device at the address specified from a vendor that is not supported
Device not found	No device found at address given
Communication error	Unable to communicate with device
Invalid credentials	Invalid credentials for device
4096: Method Parameters Checked - Job started	Job was started

45.5.2 DiscoverSystems

The DiscoverSystems method shall result in the SMI-S server attempting to discover devices that are available. The difference between AddSystem and DiscoverSystems is that DiscoverSystems does not need connection information. Upon discovery, the SMI-S server shall create the instances necessary to model that device in the requested namespace. If the device has already been added to the SMI-S server, then calling this method shall result in an update to the instances already in existence. Note that this may result in the creation of new and deletion of old instances. See 45.6 "Client Considerations and Recipes" for more details on what happens with this method call.

Method signature:

uint32 DiscoverSystems(CIM_Job REF Job, String Namespace, SharedSecret REF Secret, OUT CIM_System REF DiscoveredSystems[])

Table 457 describes parameters of DiscoverSystem.

Table 457 - DiscoverSystem Parameters

Parameter	Description
Job	Reference to a Job, if one is created
UseNamespace	Name of the Namespace to create the system in. Namespace must already exist
Secret	Reference to previously created SharedSecret to pass along to device
DiscoveredSystems[]	System discovered

Table 458, “DiscoverSystem Return Codes,” presents return codes for the DiscoverSystem method.

Table 458 - DiscoverSystem Return Codes

Value	Description
0: Success	Job completed with no error.
1: Not Supported	Not supported
2: Unknown	Unknown error occurred
3: Timeout	Timeout
4: Failed	Method failed.
5: Invalid Parameter	
6: In Use	Element is in use and cannot be modified
Invalid namespace	Namespace supplied does not exist or service not supported for that namespace
Device profile not supported	Device at the address specified does not support any of the autonomous profiles supported
Vendor not supported	Device at the address specified from a vendor that is not supported
Device not found	No device found at address given
Communication error	Unable to communicate with device
Invalid credentials	Invalid credentials for device
4096: Method Parameters Checked - Job started	Job was started

45.5.3 RemoveSystem

The RemoveSystem method shall result in the removal of all instances related to that device from the proxy server.

Method signature:

uint32 RemoveSystem(CIM_Job REF Job, String Namespace, CIM_System REF Device)

Table 459 describes each of the RemoveSystem Parameters.

Table 459 - RemoveSystem Parameters

Parameters	Description
Job	Reference to a Job, if one is created
Namespace	Name of the Namespace to create the system in. Namespace must already exist
Device	Reference to device currently managed

45.6 Client Considerations and Recipes

One of the key client considerations is indications. Because adding and removing a device will result in the creation or deletion of a large number of instances, care must be taken to avoid “indication storms” that overwhelm clients with large numbers of indications. To this end, the proxy shall only send an InstCreation or InstDeletion indication for the creation or deletion of the top-level ComputerSystem, respectively. Note that these are exactly the indications specified in the autonomous device profiles Fabric, Array, and Storage Virtualizer.

Another consideration is what happens if the proxy does not support the device being added. For example, a proxy for an array is asked to discover a switch, or vendor A’s proxy is asked to discover vendor B’s device. To prevent clients from having to try-and-fail a request, the SNIA_SystemRegistrationCapabilities class provides AutonomousProfilesSupported and VendorsSupported arrays. When adding a device, the client will probably have enough information at hand about that device to know, based on these arrays, whether or not the AddSystem call would succeed for that device.

The following are the anticipated uses cases that will drive development of the functionality.

45.6.1 Use Case 1: Add Device

In this use case, the client wishes to discover a new device just installed

Pseudo-code:

Assume IP address, user name and password are known

Step 1: Create SharedSecret

Step 2: Create indication listener

Step 3: Call AddSystem

Step 4: If Job created, wait for indication

Step 5: Remove indication listener

45.6.2 Use Case 2: Remove Device

In this use case, the client wishes to delete a device that has just been replaced.

Assume IP address, user name and password are known

Step 1: Create indication listener

Step 2: Call RemoveSystem

Step 3: If Job created, wait for indication

Step 4: Remove indication listener

45.7 Registered Name and Version

Proxy Server System Management version 1.3.0 (Component Profile)

45.8 CIM Elements

Table 460 describes the CIM elements for Proxy Server System Management.

Table 460 - CIM Elements for Proxy Server System Management

Element Name	Requirement	Description
45.8.1 CIM_HostedService	Mandatory	
45.8.2 SNIA_SystemRegistrationCapabilities	Mandatory	
45.8.3 SNIA_SystemRegistrationService	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Addition of a device (ComputerSystem).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Deletion of a device (ComputerSystem).

45.8.1 CIM_HostedService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 461 describes class CIM_HostedService.

Table 461 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

45.8.2 SNIA_SystemRegistrationCapabilities

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 462 describes class SNIA_SystemRegistrationCapabilities.

Table 462 - SMI Referenced Properties/Methods for SNIA_SystemRegistrationCapabilities

Properties	Flags	Requirement	Description & Notes
AutonomousProfilesSupported		Mandatory	This property identifies the profiles that this service is capable of discovering and managing. For example, a block device could potentially list "Array" and "Storage Virtualizer".
VendorsSupported		Mandatory	This property identifies the vendors whose devices this service can discover. For example, if the list contains "Vendor A" then only "Vendor A" devices can be discovered. This should include at least the instrumentation vendor, and may include other vendors. (e.g. due to OEM relationships).
SupportedAsynchronousActions		Mandatory	Indicates which methods are executed asynchronously.
SupportedSynchronousActions		Mandatory	Indicates which methods are executed synchronously.

45.8.3 SNIA_SystemRegistrationService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 463 describes class SNIA_SystemRegistrationService.

Table 463 - SMI Referenced Properties/Methods for SNIA_SystemRegistrationService

Properties	Flags	Requirement	Description & Notes
AddSystem()		Mandatory	
DiscoverSystems()		Optional	
RemoveSystem()		Mandatory	

EXPERIMENTAL

STABLE

46 Device Credentials Subprofile

46.1 Description

Many devices require a shared secret to be provided to access them. This shared secret is different that the credentials used by the SMI-S Client for authentication with the CIM Server. This subprofile is used to change this device shared secrets.

The SMI-S Client shall not be provided with the password, only the principle. The SMI-S Client can use the principle to change the shared secret appropriately.

The device credentials can be exposed throughout the CIM model such that a CIM Client may manipulate them. The credentials are modeled as shared secrets.

46.1.1 Instance Diagram

Figure 67 provides a sample instance diagram.

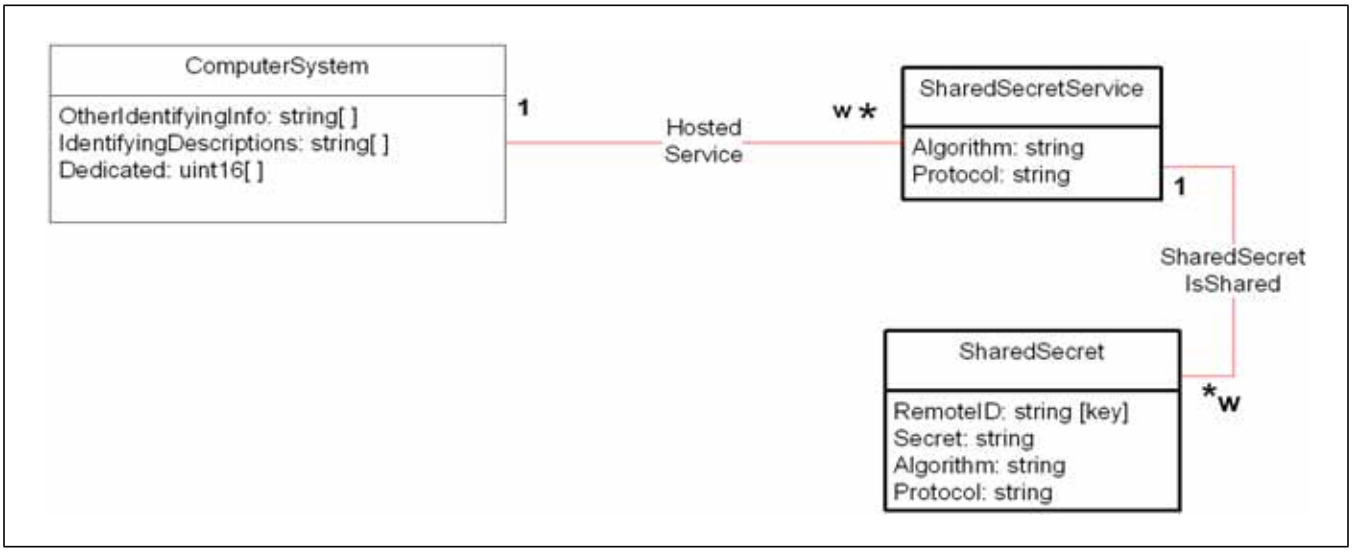

Figure 67 - DeviceCredentials Subprofile Model

46.2 Health and Fault Management Considerations

Not defined in this standard.

46.3 Cascading Considerations

Not defined in this standard.

46.4 Supported Subprofiles and Packages

Not defined in this standard.

46.5 Extrinsic Methods of this Profile

Not defined in this standard.

46.6 Client Considerations and Recipes

None.

46.7 Registered Name and Version

Device Credentials version 1.3.0 (Component Profile)

46.8 CIM Elements

Table 464 describes the CIM elements for Device Credentials.

Table 464 - CIM Elements for Device Credentials

Element Name	Requirement	Description
46.8.1 CIM_HostedService	Mandatory	
46.8.2 CIM_SharedSecret	Mandatory	
46.8.3 CIM_SharedSecretIsShared	Mandatory	
46.8.4 CIM_SharedSecretService	Mandatory	

46.8.1 CIM_HostedService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 465 describes class CIM_HostedService.

Table 465 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

46.8.2 CIM_SharedSecret

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 466 describes class CIM_SharedSecret.

Table 466 - SMI Referenced Properties/Methods for CIM_SharedSecret

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	

Table 466 - SMI Referenced Properties/Methods for CIM_SharedSecret

Properties	Flags	Requirement	Description & Notes
ServiceCreationClassName		Mandatory	
ServiceName		Mandatory	
RemoteID		Mandatory	
Secret		Mandatory	

46.8.3 CIM_SharedSecretIsShared

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 467 describes class CIM_SharedSecretIsShared.

Table 467 - SMI Referenced Properties/Methods for CIM_SharedSecretIsShared

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

46.8.4 CIM_SharedSecretService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 468 describes class CIM_SharedSecretService.

Table 468 - SMI Referenced Properties/Methods for CIM_SharedSecretService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ElementName		Mandatory	

STABLE

DEPRECATED

47 Miscellaneous Security Profiles

The functionality of several experimental SMI-S security profiles has been subsumed by emerging profile development in DMTF. The following experimental SMI-S profiles have been removed from this standard:

- Security Profile
- Authorization Subprofile
- Credential Management Subprofile
- Security Resource Ownership Subprofile
- Security Role Based Access Control Subprofile
- Identity Management Subprofile
- 3rd Party Authentication Subprofile

The final experimental versions of these profiles are defined in *Storage Management Technical Specification*, *Part 2 Common Profiles,* Version 1.3.0. The emerging profiles from DMTF do not map exactly to the SNIA profile, but DSP1034 Simple Identity Management Profile and DSP1039 Role Based Authorization Profile are related. These and other DMTF profiles may be downloaded from http://www.dmtf.org/standards/profiles/

DEPRECATED

EXPERIMENTAL

48 Operational Power Profile

48.1 Synopsis

Profile Name: Operational Power (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.23.0

Table 469 describes the related profiles for Operational Power.

Table 469 - Related Profiles for Operational Power

Profile Name	Organization	Version	Requirement	Description
Disk Drive Lite	SNIA	1.6.0	Optional	
Fan	SNIA	1.5.0	Optional	
Power Supply	SNIA	1.0.1	Optional	
Multiple Computer System	SNIA	1.2.0	Optional	
CPU	DMTF	1.0.0	Optional	
Device Tray	DMTF	1.0.0	Optional	

Central Class: OperationalPowerStatisticsService

Scoping Class: ComputerSystem

48.2 Description

48.2.1 Overview

The Operational Power Profile defines classes and methods for viewing system power usage information in real time. It is a component profile supported by autonomous profiles such as the Array and Self-Contained NAS Profile.

Emerging data center best practices require close monitoring of the energy used by various system components. These include the CPUs, chipsets, fans, power supplies, disks and PDUs (Power Distribution Units) used in storage systems.

Systems have various capabilities with respect to the granularity of information they are able to provide. For this reason, nearly all the classes and properties in this profile, except for whole-system info, are listed as optional. However, implementers must understand that the more granular the information that a data center manager can obtain, the more specifically they can tune their power and air conditioning systems. So it is advantageous to implement every property on which the underlying system supports reporting.

This profile is specifically patterned after the Block Server Performance and the Filesystem Performance Profiles in order to ease implementation by both client and server-side developers already experienced with those profiles.

48.3 Implementation

48.3.1 Model Overview

Figure 68 provides an overview of the model. The ComputerSystem is that of the autonomous profile (e.g., a NAS Head or a Self-Contained NAS) which utilizes the Operational Power Profile. This set of classes indicates the system supports power statistics for the entire system (OperationalPowerStatistics with ElementType = 2) and disks (OperationalPowerStatistics with ElementType = 10). There may be multiple instances of OperationalPowerStatistics with ElementType = 2 providing power statistics for multiple disks. There may be power statistics for other element types (see 48.3.2 Element Types).

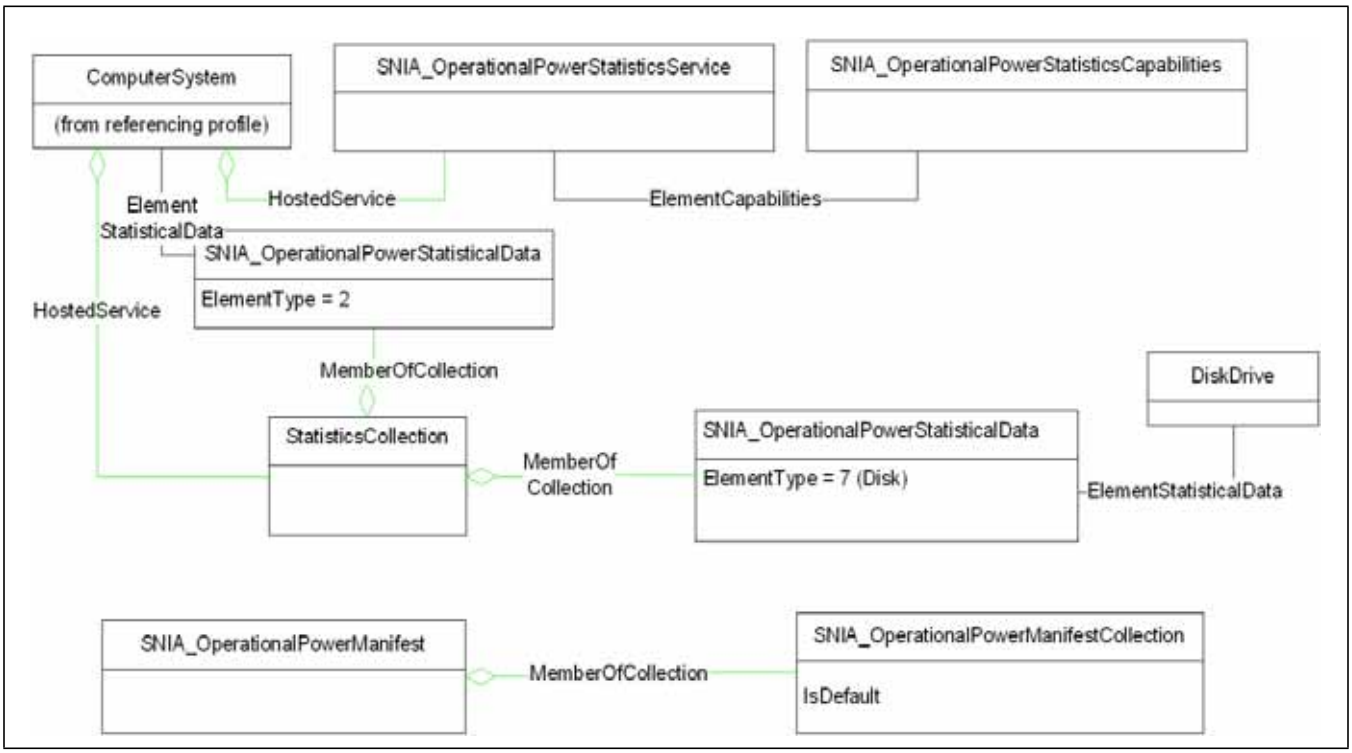

Figure 68 - Operational Power Profile Summary

The StatisticsCollection is the anchor point from which all statistics being kept by the profile can be found. Statistics are defined as an OperationalPowerStatisticalData class, instances of which hold the statistics for particular metered elements (i.e., whole systems, PDUs, power supplies, disks, disk trays, RAID groups, and fans). The particular type of metered element is recorded in the instance of OperationalPowerStatisticalData within the ElementType property.

All of the statistics instances are related to the elements that they meter via the ElementStatisticalData association (e.g., OperationalPowerStatisticalData for a DiskDrive can be found from the DiskDrive by traversing the ElementStatisticalData association).

All of the statistics instances kept within the profile are associated to the one StatisticsCollection instance. Access to all of the statistics for the profile is through the StatisticsCollection. The StatisticsCollection has a HostedCollection association to the "top level" computer system of the profile.

48.3.2 Element Types

Statistics may be kept for a number of elements within the profile, including elements within other component profiles. For example, power statistics may be associated with an instance of CIM_DiskDrive implemented as part of the SMI-S DIsk Drive Lite Profile. If the implementation does not support the Disk Drive Live Profile, then it may chose to support CIM_DiskDrive instances purely for this profile.

Figure 69, “Model for Element Types,” provides an overview of the models for various element types.

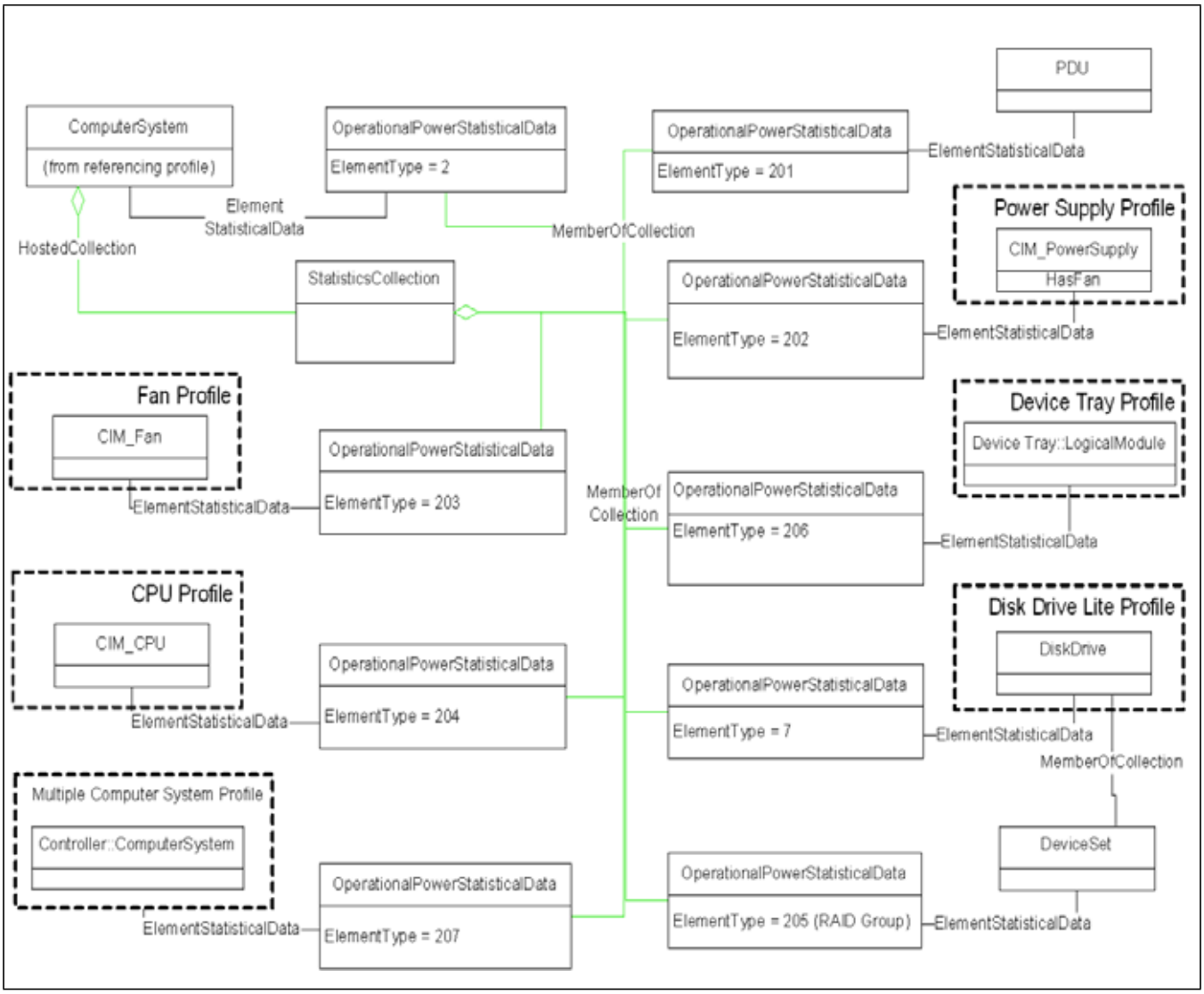

Figure 69 - Model for Element Types

OperationalPowerStatisticsCapabilities.ElementsSupported holds a list of the types of elements supported by this profile. See 48.3.7.1 ElementsSupported.

48.3.2.1 "Top Level" System

The top-level system is the ComputerSystem instance defined in an autonomous profile (e.g., Array, NAS Head) that supports the Operational Power Profile. This top-level system represents the entire storage system; power statistics for the top-level system represent the aggregation of values from all elements within the system. In other words, the milliwatt value reported for the top-level statistics should equal the sum of the Milliwatts reported for other elements. Power statistics for the top-level system are mandatory.

Figure 70, “Classes related to Top-level System Power Statistics,” which is a subset of the model from Figure 69, “Model for Element Types,” shows the mandatory classes for power statistics for the top-level system.

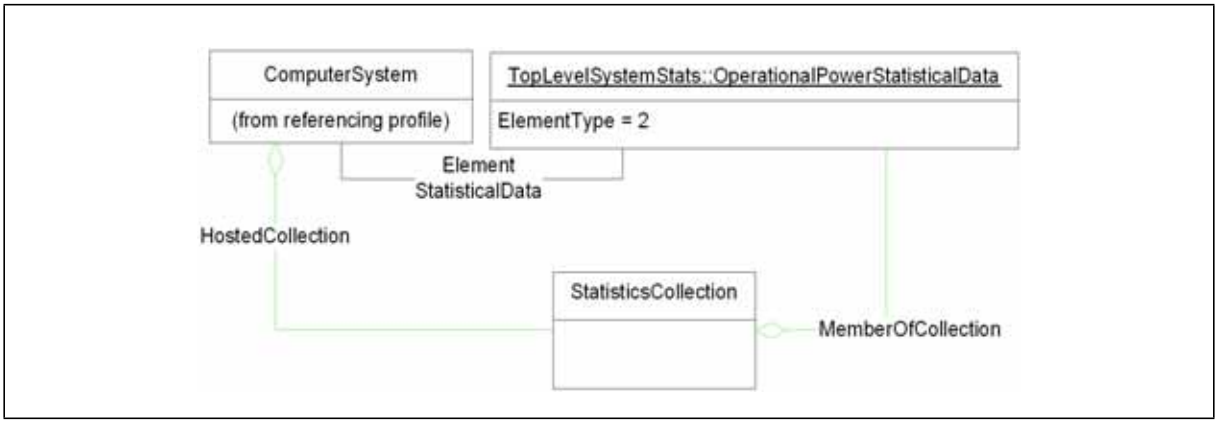

Figure 70 - Classes related to Top-level System Power Statistics

48.3.2.2 Power Source Statistics

Within this profile, a power source that provides statistics and provides some source of power conversion (for example, from AC to DC) is known as a Power Supply and is modeled using CIM_PowerSupply. A device that provides power statistics without information about conversions is known as a Power Distribution Unit (PDU) and is modeled using CIM_PowerSource. Any given subset of power may be represented by either a PDU (CIM_PowerSource) or Power Supply (CIM_PowerSupply), but shall not be resented by both.

The power statistics for Power Supplies or PDUs are kept within the OperationalPowerStatisticalData instances, with one for each Power Supply or PDU within the system. The Milliwatts attribute of OperationalPowerStatisticalData associated to PowerSource or PowerSupply represents the power dedicated to that power source, not the power supplied by it. In other words, it represents the overhead of the power source. There may be multiple instances per disk tray or controller. The implementation may use the DMTF Power Supply Profile (DSP1015) or model key classes as follows:

Each PowerSupply or PowerSource instance shall be associated to the ComputerSystem using SystemDevice.

If no instance of CIM_SuppliesPower references the instance of CIM_PowerSupply (or PowerSource), the power supply represented by CIM_PowerSupply or PowerSource supplies power to the whole managed system. In this case, the CIM_ComputerSystem instance and the CIM_PowerSupply or PowerSource instance shall only be associated through an instance of CIM_SystemDevice.

If at least one instance of CIM_SuppliesPower references the instance of CIM_PowerSupply (or PowerSource), all of the power-receiving elements shall be associated with the CIM_PowerSupply (or PowerSource) instance through an instance of CIM_SuppliesPower.

The ElementType associated with a PDU is 201; the ElementType associated with a power supply is 202.

48.3.2.3 Disk tray

Each disk tray is represented by a separate OperationalPowerStatisticalData instance. The tray itself is modeled by an instance of CIM_LogicalModule. The implementation may use the DMTF Device Tray Profile (DSP1019) or may model key classes as follows:

Each LogicalModule instance shall be associated to the ComputerSystem using SystemDevice. There shall be ConcreteComponent associations between LogicalModule and contained disks. The Milliwatts

value in the OperationalPowerStatisticalData instance associated with LogicalModule shall be the sum of values of disks residing in the tray.

The ElementType associated with a disk tray is 206.

48.3.2.4 Fan

The power statistics for each fan may be presented in a separate OperationalPowerStatisticalData instance. There may be multiple instances per disk tray or controller head. The implementation may use the DMTF Fan Profile (DSP1013) or may model key classes as follows:

Each Fan instance shall be associated to the ComputerSystem using SystemDevice.

The ElementType associated with a fan is 203.

48.3.2.5 CPU

The power statistics for each CPU in a system may be modeled by a separate OperationalPowerStatisticalData instance. At the provider's discretion, each core on the system may have its own instance. While the naming of these instances is not specified, there may be a need to indicate whether cores or entire CPU units are being reported. The implementation may use the DMTF CPU Profile (DSP1022) or may model key classes as follows:

Each CPU (CIM_Processor) instance shall be associated to the ComputerSystem using SystemDevice.

The ElementType associated with a CPU is 204.

48.3.2.6 Disk Drive

The statistics for each disk drive in a system is represented by a separate OperationalPowerStatisticalData instance. The implementation may use the SNIA Disk Drive Lite Profile or may model key classes as follows:

Each Disk Drive (CIM_DiskDrive) instance shall be associated to the ComputerSystem using SystemDevice.

The ElementType associated with a disk drive is 10. This value is consistent with the value in the Block Server Performance Profile.

48.3.2.7 RAID Groups

The statistics for each RAID Group in a system is represented by a separate OperationalPowerStatisticalData instance. This OperationalPowerStatisticalData instance shall have an association to the SNIA_DeviceSet instance representing the RAID Group. Each DIskDrive instance which is a member of the RAID group shall be associated to the SNIA_DeviceSet via MemberOfCollection. The Milliwatts value in OperationalPowerStatisticalData representing the RAID group shall be the sum of the values of the associated disks.

The ElementType associated with a RAID Group is 205.

48.3.2.8 Controller

If the implementation supports the SNIA Multiple Computer System Profile, it may model controller statistics with OperationalPowerStatisticalData instances associated to ComputerSystem instances representing controllers.

The ElementType associated with a RAID Group is 207.

48.3.3 Power Metric Attributes

At this time, the following attributes are defined for all Elements described in 48.3.2 "Element Types":

- **Milliwatt**: the current value in Milliwatts of the power consumed by the element referenced. Note that in this standard, a milliwatt is defined to be one thousandth of a watt.
- **Precision**: the number of decimal places of accuracy of the Milliwatt attribute.
- **StatisticTime**: the date and time the Milliwatt value was acquired from the device. This is stored in CIM datetime format.

All three of these values are mandatory for the OperationalPowerStatisticalData instance reference the top-level ComputerSystem and optional for other elements.

48.3.4 Bulk Retrieval

Figure 68, “Operational Power Profile Summary,” illustrates the OperationalPowerStatisticsService for bulk retrieval of all the statistics data and the creation of manifest collections. Associated with the OperationalPowerStatisticsService is an OperationalPowerStatisticsCapabilities instance that identifies the specific capabilities implemented by the operational power statistics support. Specifically, it includes an “ElementsSupported” property that identifies the elements for which statistics are kept; the OperationalPowerStatisticsCapabilities instance also identifies the various retrieval mechanisms (e.g., Extrinsic or Association Traversal) that are implemented (i.e., supported) by the provider’s statistics support.

48.3.5 Default Manifest Collection

Associated with the instances of the StatisticsCollection shall be a provider-supplied (Default) SNIA_ManifestCollection that represents the statistics properties that are kept by the profile. The default manifest collection is indicated by the IsDefault property (=True) of the SNIA_OperationalPowerManifestCollection. For each metered object (element) of the profile implementation, the default manifest collection will have exactly one manifest that will identify which properties are included for that metered object. If an object is not metered, then there shall not be a manifest for that element type. If an element type (e.g., CPU) is metered, then there shall be a manifest for that element type.

EXPERIMENTAL

Each default manifest in the default manifest collection identifies the properties included by default by the implementation. The CSVSequence property of the manifest shall identify the default sequence in which the implementation will return properties within each record for the ElementType on a GetStatisticsCollection request. For each property included in the manifest by the value “true”, there should be an entry in the CSVSequence array identifying the OperationalPowerStatisticalData property by name. The first three values of CSVSequence shall be "InstanceID", "ElementType" and "StatisticsTime" to allow correlation of the Manifest with the CSV record based on the ElementType.

EXPERIMENTAL

48.3.6 Client Defined Manifest Collection

Manifest collections are either provider-supplied (SNIA_OperationalPowerManifestCollection.IsDefault=True) or client-defined collections (SNIA_OperationalPowerManifestCollection.IsDefault=False). Client-defined collections are used to indicate the specific statistics properties that the client would like to retrieve using the GetStatisticsCollection method. For a discussion of provider-supplied manifest collections, see 48.3.5.

Client-defined manifest collections are a mechanism for restricting the amount of data returned on a GetStatisticsCollection request. A client-defined manifest collection is identified by the IsDefault property of the collection set to False. For each element type of the operational power statistics class (e.g., PDU,

Fan, etc.), a manifest can be defined that identifies which specific properties of the particular statistics class element type are to be returned on a GetStatisticsCollection request. Each of the element types of the operational power statistics class may have no or one manifest in any given manifest collection. This is illustrated in Figure 68.

In Figure 68, manifest classes are defined for PDUs, power supplies, fans, CPUs, disks and disk trays. Each property of the manifest is a Boolean that indicates whether the property is to be returned (true) or omitted (false).

Multiple client-defined manifest collections can be defined in the profile. Consequently, different clients or different client applications can define different manifests for different application needs. A manifest collection can completely omit a whole set of statistics pertaining to one or more element types; for example, one manifest might collect information on disk power only. Since manifest collections are "client objects", they are named (ElementName) by the client for the client's convenience. The CIM server will generate an instance ID to uniquely identify the manifest collection in the CIM Server.

Client-defined manifest collections are created using the CreateManifestCollection method. Manifests are added or modified using the AddOrModifyManifest method. A manifest may be removed from the manifest collection by using the RemoveManifests method.

NOTE Use of manifest collections is optional with the GetStatisticsCollection method. If NULL for the manifest collection is passed on input, then all statistics instances are assumed (i.e., all available statistics will be returned).

48.3.7 Capabilities Support for Operational Power Profile

To determine what is supported with a Operational Power Profile implementation, inspect the RegisteredSubprofiles supported by the autonomous profile (i.e., an Array, NAS Head or Self-Contained NAS Profile) that utilizes the Operational Power Profile. In order to support statistics for a particular class of metered element, the corresponding object must be modeled. This profile requires support for all classes used by the profile, regardless of whether statistics on them are available or not. To find out whether statistics are reported, examine the ElementsSupported property in the SNIA_OperationalPowerStatisticsCapabilities instance associated to the SNIA_OperationalPowerStatisticsService. This capabilities class instance is not created nor modified by clients; rather, it is populated by the provider and has three properties of interest (as discussed within the following sections).

For the methods-supported properties described below (namely, SynchronousMethodsSupported and AsynchronousMethodsSupported), any or all of the respective values can be missing (e.g., the arrays can be NULL). If all of the methods supported are NULL, then manifest collections are not supported and GetStatisticsCollection is not supported for the retrieval of statistics. This leaves enumerations or association traversals as the only methods for retrieving the statistics.

48.3.7.1 ElementsSupported

This property within the OperationalPowerStatisticsCapabilities class defines a list of element types for which statistical data is available. The valid values are "PDU", "Disk tray", "Disk", "CPU", "Fan", "Power Supply", "RAID Group", and "Controller".

To be a valid implementation of the Operational Power Profile, at least one of the values listed for ElementsSupported shall be supported. ElementsSupported is an array, such that all of the values can be identified.

Some of these elements are available in hybrid configurations. Two common examples are multi-core CPUs and power supplies with integrated fans. Clients must examine the corresponding class instances to find out whether or not they are so configured.

48.3.7.2 SynchronousMethodsSupported

This property within the OperationalPowerStatisticsCapabilities class defines the synchronous mechanisms that are supported for retrieving statistics and for defining and modifying filters for statistics retrieval. For this release of SMI-S, the values of interest are "GetStatisticsCollection", "Manifest Creation", "Manifest Modification", and "Manifest Removal".

48.3.7.3 AsynchronousMethodsSupported

This property within the OperationalPowerStatisticsCapabilities class defines the asynchronous mechanisms that are supported for retrieving statistics. For this release of SMI-S, this should be NULL.

48.3.7.4 ClockTickInterval

An internal clocking interval for all timer counters kept in the system implementation, measured in microseconds (i.e., the unit of measure in the timers, measured in microseconds). Time counters are considered to be monotonically increasing counters that contain "ticks". Each tick represents one clock tick interval.

For example, if ClockTickInterval contained a value of 32, then each time counter tick would represent 32 microseconds.

48.3.8 Health and Fault Management Consideration

Not defined in this version of the specification.

48.3.9 Cascading Considerations

Not applicable

48.4 Methods of the Profile

48.4.1 Extrinsic Methods of the Profile

48.4.1.1 Overview

The methods supported by this profile are summarized in Table 470 and detailed within the sections that follow it.

Table 470 - Creation, Deletion and Modification Methods

Method	Created Instances	Deleted Instances	Modified Instances
GetStatisticsCollection	None	None	None
CreateManifestCollection	OperationalPowerStatisticsManifestCollection AssociatedOperationalPowerStatisticsManifestCollection	None	None
AddOrModifyManifest	OperationalPowerStatisticsManifest(subclass) MemberOfCollection	None	OperationalPowerStatisticsManifest(subclass)
RemoveManifest	None	OperationalPowerStatisticsManifest(subclass) MemberOfCollection	None

48.4.1.2 GetStatisticsCollection

This extrinsic method retrieves statistics in a well-defined bulk format. The set of statistics returned by this method is determined by the list of element types passed into the method and the manifests for those

types contained in the supplied manifest collection. The statistics are returned through a well-defined array of strings that can be parsed to retrieve the desired statistics as well as limited information about the elements that those metrics describe.

```
GetStatisticsCollection(

        [OUT, Description(“Reference to the job(shall be null in this version of SMI-S.)”]
        CIM_ConcreteJob REF Job = NULL,

        [IN, Description("Element types for which statistics should be returned")
         ValueMap {"1", "2", "10", "201", "202", "203", "204", "205", "206", "207", "..",
                           "0x8000.."},
         Values {"Other", "Top-Level System", "Disk Drive", "PDU", "Power Supply", "Fan",
                                   "CPU", "RAID Group", "Disk tray", "Controller", "DMTF
                                   Reserved", "Vendor Specific"}]
        uint16 ElementTypes[],

        [IN, Description ("An array of strings that specify the particular ’Other’"
                     "element(s) when the ElementType property above includes the"
                     "ElementType value of 1 (i.e., "Other"). Each string within"
                     "this array identifies a separate "Other" element and duplicate"
                     "string values are NOT allowed.   This property should be set"
                     "to NULL when the ElementType property does not include the" "value
                     of 1.")]
        string OtherElementTypeDescriptions[],

        [IN, Description("The manifest collection that contains the manifests which list"
                                   "the metrics that should be returned for each element
                                   type")]
        SNIA_OperationalPowerStatisticsManifestCollection REF ManifestCollection,

        [IN, Description("Specifies the format of the Statistics output parameter"),
         ValueMap {"2", "..", "0x8000.."},
         Values ("CSV", "DMTF Reserved", "Vendor Specific")]
        uint16 StatisticsFormat,

        [OUT, Description("The statistics for all the elements as determined by the"
                                   "Elements and ManifestCollection parameters")]
        string Statistics[]
);
```

Error returns are:

{"Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter", "Method Reserved", "Method Parameters", "Element Not Supported", "Statistics Format Not Supported", "Method Reserved", "Vendor Specific"}

NOTE In this version of the standard, Job Control is not supported for the GetStatisticsCollection method. This method should always return NULL for the Job parameter, and no job related errors shall be returned.

If the ElementTypes[] array is empty, then no data shall be returned. If the ElementTypes[] array is NULL, then the ElementTypes[] parameter shall be ignored and all data specified in the manifest collection shall be returned.

If the manifest collection is empty, then no data shall be returned. If the manifest collection parameter is NULL, then the default manifest collection is used. (Note: In SMI-S, a default manifest collection shall exist if the GetStatisticalCollection method is supported).

NOTE The ElementTypes[] and ManifestCollection parameters may identify different sets of element types. The effect of this will be for the implementation to return statistics for the element types that are in both lists (that is, the intersection of the two lists). This intersection could be empty. In this case, no data will be returned.

For the current version of SMI-S, the only recognized value for StatisticsFormat is "CSV". The method may support other values, but they are not specified by SMI-S (i.e., they would be vendor specific).

Given a client has an inventory of the metered objects with Statistics InstanceIDs that may be used to correlate with the OperationalPowerStatisticalData instances, a simple CSV format is sufficient and the most efficient human-readable format for transferring bulk statistics. More specifically, the following rules constrain that format and define the content of the String[] Statistics output parameter to the GetStatisticsCollection() method:

- The Statistics[] array may contain multiple statistics records per array entry. In such cases, the total length of the concatenated record strings shall not exceed 64K bytes. And a single statistics record shall not span Array entries.
- There shall be exactly one statistics record per line in the bulk Statistics parameter. A line is terminated by:
  - a line-feed character
  - the end of a String Array Element (i.e., a statistics record cannot overlap elements of the String[] Statistics output parameter).
- Each statistics record shall contain the InstanceID of the OperationalPowerStatisticalData instance, the value map (number) of the ElementType of the metered object, and one value for each property that the relevant OperationalPowerStatisticsManifest specifies as "true".
- Each value in a record shall be separated from the next value by a Semi-colon (";"). This is to support internationalization of the CSV format. A provider creating a record in this format should not include white space between values in a record, though it may be included for human readability if so desired. A client reading a record it has received shall ignore white-space between values.
- The InstanceID value is an opaque string that shall correspond to the InstanceID property from OperationalPowerStatisticalData instance.
  - For the convenience of client software that needs to be able to correlate InstanceIDs between different GetStatisticsCollection method invocations, the InstanceID for OperationalPowerStatisticalData instance shall be unique across all instances of the OperationalPowerStatisticalData class. It is not sufficient that InstanceID is unique across subclasses of OperationalPowerStatisticalData.
- The ElementType value shall be a decimal string representation of the Element Type number (e.g., "201" for PDU). The StatisticTime shall be a string representation of DateTime. All other values shall be decimal string representations of their statistical values.
- Null values shall be included in records for which a statistic is returned (specified by the manifest or by a lack of manifest for a particular element type) but there is no meaningful value available for the statistic. A NULL statistic is represented by placing a semi-colon (;) in the record without a value at the position where the value would have otherwise been included. A record in which the last statistic has a NULL value shall end in a semi-colon (;). Clients shall ignore whitespace between semicolons.
- The first three values in a record shall be the InstanceID, ElementType and StatisticTime values from the OperationalPowerStatisticalData instance. The remaining values shall be returned in the order in which they are defined by the MOF for the OperationalPowerStatisticsManifest class or subclass the record describes.

As an additional convention, a provider should return all the records for a particular element type in consecutive String elements, and the order of the element types should be the same as the order in which the element types were specified in the input parameter to GetStatisticsCollection().

48.4.1.3 CreateManifestCollection

This extrinsic method creates a new manifest collection whose members serve as a filter for metrics retrieved through the GetStatisticsCollection method.

```
CreateManifestCollection(
        [IN, Description("The collection of statistics that will be filtered using the new
                                manifest collection")]
        CIM_StatisticsCollection REF Statistics,

        [IN, Description("Client-defined name for the new manifest collection)"]
        string ElementName,

        [OUT, Description("Reference to the new manifest collection")]
        SNIA_OperationalPowerStatisticsManifestCollection REF ManifestCollection
);
```

Error returns are:

```
        {"Ok", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter",
                                "Method Reserved", "Vendor Specific"}
```

48.4.1.4 AddOrModifyManifest

This is an extrinsic method that either creates or modifies a statistics manifest for this statistics service. A client supplies a manifest collection within which the new manifest collection will be placed or an existing manifest will be modified, the element type of the statistics that the manifest will filter, and a list of statistics that should be returned for that element type using the GetStatisticsCollection method.

```
AddOrModifyManifest(
        [IN, Description("Manifest collection that the manifest is or should be a member
                                of")]
        SNIA_OperationalPowerStatisticsManifestCollection REF ManifestCollection,

        [IN, Description("The element type whose statistics the manifest will filter")
        ValueMap {"1", "2", "10", "201", "202", "203", "204", "205", "206", "207", "..",
                                "0x8000.."},
         Values {"Other", "Disk Drive", "PDU", "Power Supply", "Fan", "CPU", "RAID Group",
                                "Disk tray", "Controller", "DMTF Reserved", "Vendor
                                Specific"}]
        uint16 ElementType,

        [IN, Description ("A string describing the type of element when the ElementType"
                             "property above is set to 1 (i.e., 'Other'). This property"
                             "should be set to NULL when the ElementType property is any"
                             "value other than 1.")]
              string OtherElementTypeDescription,

        [IN, Description("The client-defined string that identifies the manifest created
                                or modified by this method")]
```

```
            string ElementName,

            [IN, Description("The statistics that will be included by the manifest filter;"
                             "that is, the statistics that will be supplied through
                             the" "GetStatisticsCollection method")
            string StatisticsList[],

            [OUT, Description("The Manifest that is created or modified on the successful"
                             "execution of this method")]
            SNIA_OperationalPowerStatisticsManifest REF Manifest
    );
```

Error returns are:

```
            {"Success", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter",
                             "Method Reserved", "Element Not Supported", "Metric not
                             supported", "ElementType Parameter Missing", "Method
                             Reserved", "Vendor Specific"}
```

If the StatisticsList[] array is empty, then only InstanceID and ElementType will be returned when the manifest is referenced. If the StatisticsList[] array parameter is NULL, then all supported properties is assumed (i.e., all supported properties shall be included).

NOTE This would be the OperationalPowerStatisticsManifest from the default manifest collection.

48.4.1.5 RemoveManifests

This is an extrinsic method that removes manifests from the manifest collection.

```
            RemoveManifests(
            [IN, Description("Manifest collection from which the manifests will be removed")]
            SNIA_OperationalPowerStatisticsManifestCollection REF ManifestCollection,

            [IN, Description("List of manifests to be removed from the manifest collection")
            SNIA_OperationalPowerStatisticsManifest REF Manifest[]);
```

Error returns are:

```
            {"Success", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter",
                             "Method Reserved", "Manifest not found", "Method
                             Reserved", "Vendor Specific"}
```

48.4.2 Intrinsic Methods of this Profile

48.4.2.1 DeleteInstance (of a OperationalPowerStatisticsManifestCollection)

This will delete the OperationalPowerStatisticsManifestCollection where IsDefault=False, the AssociatedOperationalPowerStatisticsManifestCollection association to the StatisticsCollection and all manifests collected by the manifest collection (and the MemberOfCollection associations to the OperationalPowerStatisticsManifestCollection).

48.4.2.2 Association Traversal

One of the ways of retrieving statistics is through association traversal from the StatisticsCollection to the individual Statistics following the MemberOfCollection association. This shall be supported by all

implementations of the Operational Power Profile and would be available to clients if the provider does not support the GetStatisticsCollection approach.

48.5 Use Cases

48.5.1 Client Considerations and Recipes

48.5.1.1 Operations Using SNIA_DeviceSet

The purpose of SNIA_DeviceSet is to organize a set of LogicalDevices that support some management operation, for example the Operational Power management that instrumentation might provide to a client. The LogicalDevices associated to the SNIA_DeviceSet would typically be related by some hardware constraint, like being in the same RAID Group that supports power operations.

48.6 CIM Elements

Table 471 describes the CIM elements for Operational Power.

Table 471 - CIM Elements for Operational Power

Element Name	Requirement	Description
48.6.1 CIM_ElementCapabilities	Mandatory	This associates the OperationalPowerStatisticsCapabilities to the OperationalPowerStatisticsService.
48.6.2 CIM_ElementStatisticalData (Component System Stats)	Conditional	Conditional requirement: Component Systems statistics support. This is mandatory if OperationalPowerStatisticsCapabilities.ElementTypesSupported = '3', '4' or '5'. This associates an OperationalPowerStatisticalData instance to the component ComputerSystem for which the statistics are collected.
48.6.3 CIM_ElementStatisticalData (Top Level System Stats)	Conditional	Conditional requirement: Top level system statistics support. This is mandatory if OperationalPowerStatisticsCapabilities.ElementTypesSupported = '2'. This associates an OperationalPowerStatisticalData instance to the Top Level ComputerSystem for which the statistics are collected.
48.6.4 CIM_ElementStatisticalData (Volume Stats)	Optional	This is mandatory if OperationalPowerStatisticsCapabilities.ElementTypesSupported = '8', and the parent profile supports Storage Volumes. This associates an OperationalPowerStatisticalData instance to the volume for which the statistics are collected.
48.6.5 CIM_HostedCollection (Client Defined)	Conditional	Conditional requirement: Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported. This would associate a client defined OperationalPowerStatisticalManiestCollection to the top level system for the profile (e.g., array).

Table 471 - CIM Elements for Operational Power

Element Name	Requirement	Description
48.6.6 CIM_HostedCollection (Default)	Mandatory	This would associate a default OperationalPowerStatisticsManifestCollection to the top level system for the profile (e.g., array).
48.6.7 CIM_HostedCollection (Systemto StatisticsCollection)	Mandatory	This would associate the StatisticsCollection to the top level system for the profile (e.g., array).
48.6.8 CIM_HostedService	Mandatory	This associates the OperationalPowerStatisticsService to the ComputerSystem that hosts it.
48.6.9 CIM_MemberOfCollection (DeviceSet)	Optional	This would associate LogicalDevices to SNIA_DeviceSets.
48.6.10 CIM_MemberOfCollection (Member of client defined collection)	Conditional	Conditional requirement: Clients can modify manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported. This would associate Manifests to client defined manifest collections.
48.6.11 CIM_MemberOfCollection (Member of pre-defined collection)	Mandatory	This would associate pre-defined Manifests to default manifest collection.
48.6.12 CIM_MemberOfCollection (Member of statistics collection)	Mandatory	This would associate all statistics instances to the StatisticsCollection.
48.6.13 CIM_StatisticsCollection	Mandatory	This would be a collection point for all Statistics that are kept for a storage system.
48.6.14 SNIA_DeviceSet (Provider Defined)	Optional	An instance of this class defines a grouping of LogicalDevices that support some management operation.
48.6.15 SNIA_OperationalPowerManifest (Client Defined)	Conditional	Conditional requirement: Clients can modify manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported. An instance of this class defines the statistics properties of interest to the client for one element type.
48.6.16 SNIA_OperationalPowerManifest (Provider Support)	Mandatory	An instance of this class defines the statistics properties supported by the profile implementation for one element type.
48.6.17 SNIA_OperationalPowerManifestCollection (Client Defined)	Conditional	Conditional requirement: Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported. An instance of this class defines one client defined collection of statistics manifests (one manifest for each element type).
48.6.18 SNIA_OperationalPowerManifestCollection (Provider Defined)	Mandatory	An instance of this class defines the predefined collection of default statistics manifests (one manifest for each element type).
48.6.19 SNIA_OperationalPowerStatisticalData	Mandatory	This is a Subclass of CIM_StatisticalData for Operational Power statistics. It is instantiated to provide specific statistics for particular components.
48.6.20 SNIA_OperationalPowerStatisticsCapabilities	Mandatory	This defines the statistics capabilities supported by the implementation of the profile.
48.6.21 SNIA_OperationalPowerStatisticsService	Mandatory	This is a Service that provides (optional) services of bulk statistics retrieval and manifest set manipulation methods.

48.6.1 CIM_ElementCapabilities

CIM_ElementCapabilities represents the association between ManagedElements (i.e., OperationalPowerStatisticsService) and their Capabilities (e.g., OperationalPowerStatisticsCapabilities). Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementCapabilities association for the referenced instance of Capabilities. ElementCapabilities describes the existence requirements and context for the referenced instance of ManagedElement. Specifically, the ManagedElement shall exist and provides the context for the Capabilities.

CIM_ElementCapabilities is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 472 describes class CIM_ElementCapabilities.

Table 472 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The managed element (OperationalPowerStatisticsService).
Capabilities		Mandatory	The Capabilities instance associated with the OperationalPowerStatisticsService.

48.6.2 CIM_ElementStatisticalData (Component System Stats)

CIM_ElementStatisticalData is an association that relates a component ComputerSystem to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of OperationalPowerStatisticalData. ElementStatisticalData describes the existence requirements and context for the OperationalPowerStatisticalData, relative to a specific component ComputerSystem.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Component Systems statistics support.

Table 473 describes class CIM_ElementStatisticalData (Component System Stats).

Table 473 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Component System Stats)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A reference to a component ComputerSystem for which the Statistics apply.
Stats		Mandatory	A reference to the OperationalPowerStatisticalData that hold the statistics for the ComputerSystem.

48.6.3 CIM_ElementStatisticalData (Top Level System Stats)

CIM_ElementStatisticalData is an association that relates a top level ComputerSystem to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of OperationalPowerStatisticalData. ElementStatisticalData describes the existence requirements and context for the OperatioanlPowerStasticalData, relative to a specific ComputerSystem.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Top level system statistics support.

Table 474 describes class CIM_ElementStatisticalData (Top Level System Stats).

Table 474 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Top Level System Stats)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A reference to the top level ComputerSystem for which the Statistics apply.
Stats		Mandatory	A reference to the OperationalPowerStatisticalData that hold the statistics for the ComputerSystem.

48.6.4 CIM_ElementStatisticalData (Volume Stats)

CIM_ElementStatisticalData is an association that relates a StorageVolume to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of OperationalPowerStatisticalData. ElementStatisticalData describes the existence requirements and context for the OperationalPowerStatisticalData, relative to a specific volume.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 475 describes class CIM_ElementStatisticalData (Volume Stats).

Table 475 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Volume Stats)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A reference to a StorageVolume for which the Statistics apply.
Stats		Mandatory	A reference to the OperationalPowerStatisticalData that hold the statistics for the StorageVolume.

48.6.5 CIM_HostedCollection (Client Defined)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Operational Power profile, it is used to associate a client defined OperationalPowerStatisticalManiestCollections to the top level Computer System.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported.

Table 476 describes class CIM_HostedCollection (Client Defined).

Table 476 - SMI Referenced Properties/Methods for CIM_HostedCollection (Client Defined)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	A client defined OperationalPowerStatisticalManiestCollection.

48.6.6 CIM_HostedCollection (Default)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Operational Power profile, it is used to associate the default OperationalPowerStatisticsManifestCollection to the top level Computer System.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 477 describes class CIM_HostedCollection (Default).

Table 477 - SMI Referenced Properties/Methods for CIM_HostedCollection (Default)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	The provider defined OperatonalPowerManifestCollection.

48.6.7 CIM_HostedCollection (Systemto StatisticsCollection)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Operational Power profile, it is used to associate the StatisticsCollection to the top level Computer System.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 478 describes class CIM_HostedCollection (Systemto StatisticsCollection).

Table 478 - SMI Referenced Properties/Methods for CIM_HostedCollection (Systemto StatisticsCollection)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	The StatisticsCollection.

48.6.8 CIM_HostedService

CIM_HostedService is an association between a Service (OperationalPowerStatisticsService) and the System (ComputerSystem) on which the functionality resides. Services are weak with respect to their hosting System. Heuristic: A Service is hosted on the System where the LogicalDevices or SoftwareFeatures that implement the Service are located.

CIM_HostedService is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 479 describes class CIM_HostedService.

Table 479 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Service hosted on the System.

48.6.9 CIM_MemberOfCollection (DeviceSet)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 480 describes class CIM_MemberOfCollection (DeviceSet).

Table 480 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (DeviceSet)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The SNIA_DeviceSet.
Member		Mandatory	The individual LogicalDevice that is part of the set.

48.6.10CIM_MemberOfCollection (Member of client defined collection)

This use of MemberOfCollection is to Collect all Manifests instances in a client defined manifest collection.

Created By: Extrinsic: AddOrModifyManifest

Modified By: Static

Deleted By: Extrinsic: RemoveManifest

Requirement: Clients can modify manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported.

Table 481 describes class CIM_MemberOfCollection (Member of client defined collection).

Table 481 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of client defined collection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	A client defined manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

48.6.11CIM_MemberOfCollection (Member of pre-defined collection)

This use of MemberOfCollection is to Collect all Manifests instances in the default manifest collection.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 482 describes class CIM_MemberOfCollection (Member of pre-defined collection).

Table 482 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of pre-defined collection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The provider defined default manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

48.6.12CIM_MemberOfCollection (Member of statistics collection)

This use of MemberOfCollection is to collect all OperationalPowerStatisticalData instances (in the StatisticsCollection). Each association is created as a side effect of the metered object getting created.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 483 describes class CIM_MemberOfCollection (Member of statistics collection).

Table 483 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of statistics collection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The default manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

48.6.13CIM_StatisticsCollection

The CIM_StatisticsCollection collects all statistics kept by the profile. There is one instance of the CIM_StatisticsCollection class and all individual element statistics can be accessed by using association traversal(using MemberOfCollection) from the StatisticsCollection.

CIM_StatisticsCollection is subclassed from CIM_SystemSpecificCollection.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 484 describes class CIM_StatisticsCollection.

Table 484 - SMI Referenced Properties/Methods for CIM_StatisticsCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SampleInterval		Mandatory	Minimum recommended polling interval for an array, storage virtualizer system or volume manager. It is set by the provider and cannot be modified.
TimeLastSampled		Mandatory	Time statistics table by object was last updated (Time Stamp in SMI 2.2 specification format).

48.6.14SNIA_DeviceSet (Provider Defined)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 485 describes class SNIA_DeviceSet (Provider Defined).

Table 485 - SMI Referenced Properties/Methods for SNIA_DeviceSet (Provider Defined)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Unique identifier for the collection.
ElementName		Mandatory	User friendy name for the collection.
Purpose		Mandatory	This property shows what kind of management operation is the purpose of the grouping.
GroupType		Mandatory	This property shows the unit of grouping.

48.6.15SNIA_OperationalPowerManifest (Client Defined)

The SNIA_OperationalPowerManifest class is concrete class that defines the OperationalPowerStatisticalData properties that should be returned on a GetStatisticsCollection request.

SNIA_OperationalPowerManifest is subclassed from CIM_ManagedElement.

In order for a client defined instance of the SNIA_OperationalPowerManifest class to exist, the all the manifest collection manipulation functions shall be identified in the 'SynchronousMethodsSupported' property of the SNIA_OperationalPowerStatisticsCapabilities (OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported = '6') instance, AND a client must have created at least ONE instance of SNIA_OperationalPowerManifestCollection.

Created By: Extrinsic: AddOrModifyManifest
Modified By: Extrinsic: AddOrModifyManifest
Deleted By: Extrinsic: RemoveManifest

Requirement: Clients can modify manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported.

Table 486 describes class SNIA_OperationalPowerManifest (Client Defined).

Table 486 - SMI Referenced Properties/Methods for SNIA_OperationalPowerManifest (Client Defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A Client defined string that identifies the manifest.
InstanceID		Mandatory	The instance Identification. Within the scope of the instantiating Namespace, InstanceID opaquely and uniquely identifies an instance of this class.
ElementType		Mandatory	This value is required AND the current version of SMI-S specifies the following values: ValueMap {'2', '3', '4', '5', '6','7', '8', '9','10', '11', '12'} Values { 'Computer System', 'Front-end Computer System', 'Peer Computer System', 'Back-end Computer System', 'Front-end Port', 'Back-end Port', 'Volume', 'Extent', 'Disk Drive', 'Arbitrary LUs' , 'Remote Replica Group'}.
IncludeStatisticTime		Mandatory	
CSVSequence		Mandatory	An array of strings that define a sequence of OperationalPowerStatisticalData property names. The sequence is the sequence that data is to be returned on a GetStatisticsCollection request using this manifest. The first three elements of this array should be "InstanceID", "ElementType" and "StatisticsTime" to allow applications to match the ElementType of the Manifest with the OperationalPowerStatisticalData CSV record. For OperationalPowerManifest (Client Defined) this shall be the sequence desired by the client.

48.6.16SNIA_OperationalPowerManifest (Provider Support)

The SNIA_OperationalPowerManifest class is concrete class that defines the OperationalPowerStatisticalData properties that supported by the Provider. These Manifests are established by the Provider for the default manifest collection.

SNIA_OperationalPowerManifest is subclassed from CIM_ManagedElement.

At least one Provider supplied instance of the SNIA_OperationalPowerManifest class shall exist, if the Operational Power profile is supported.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 487 describes class SNIA_OperationalPowerManifest (Provider Support).

Table 487 - SMI Referenced Properties/Methods for SNIA_OperationalPowerManifest (Provider Support)

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A Provider defined string that identifies the manifest in the context of the Default Manifest Collection.
InstanceID		Mandatory	The instance Identification. Within the scope of the instantiating Namespace, InstanceID opaquely and uniquely identifies an instance of this class.

Table 487 - SMI Referenced Properties/Methods for SNIA_OperationalPowerManifest (Provider Support)

Properties	Flags	Requirement	Description & Notes
ElementType		Mandatory	This value is required AND the current version of SMI-S specifies the following values: ValueMap {'2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'} Values { 'Computer System', 'Front-end Computer System', 'Peer Computer System', 'Back-end Computer System', 'Front-end Port', 'Back-end Port', 'Volume', 'Extent', 'Disk Drive', 'Arbitrary LUs' , 'Remote Replica Group'}.
IncludeStatisticTime		Mandatory	
CSVSequence		Mandatory	An array of strings that define a sequence of OperationalPowerStatisticalData property names. The sequence is the sequence that data is to be returned on a GetStatisticsCollection request using this manifest. The first three elements of this array shall be "InstanceID", "ElementType" and "StatisticsTime" to allow applications to match the ElementType of the Manifest with the OperationalPowerStatisticalData CSV record. For OperationalPowerManifest (Provider Support) this shall be the default sequence provided by the provider.

48.6.17SNIA_OperationalPowerManifestCollection (Client Defined)

An instance of a client defined SNIA_OperationalPowerManifestCollection defines the set of Manifests to be used in retrieval of statistics by the GetStatisticsCollection method.

SNIA_OperationalPowerManifestCollection is subclassed from CIM_SystemSpecificCollection.

In order for a client defined instance of the SNIA_OperationalPowerManifestCollection class to exist, then all the manifest collection manipulation functions shall be identified in the 'SynchronousMethodsSupported' property of the SNIA_OperationalPowerStatisticsCapabilities instance and a client must have created a Manifest Collection..

Created By: Extrinsic: CreateManifestCollection

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported.

Table 488 describes class SNIA_OperationalPowerManifestCollection (Client Defined).

Table 488 - SMI Referenced Properties/Methods for SNIA_OperationalPowerManifestCollection (Client Defined)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A client defined user-friendly name for the manifest collection. It is set during creation of the Manifest Collection through the ElementName parameter of the CreateManifestCollection method.
IsDefault		Mandatory	Denotes whether or not this manifest collection is a provider defined default manifest collection. For the client defined manifest collections this is set to 'false'.

48.6.18SNIA_OperationalPowerManifestCollection (Provider Defined)

An instance of a default SNIA_OperationalPowerManifestCollection defines the set of Manifests that define the properties supported for each ElementType supported for the implementation. It can also be used by clients in retrieval of statistics by the GetStatisticsCollection method.

SNIA_OperationalPowerManifestCollection is subclassed from CIM_SystemSpecificCollection.

At least ONE SNIA_OperationalPowerManifestCollection shall exist if the Operational Power profile is implemented. This would be the default manifest collection that defines the properties supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 489 describes class SNIA_OperationalPowerManifestCollection (Provider Defined).

Table 489 - SMI Referenced Properties/Methods for SNIA_OperationalPowerManifestCollection (Provider Defined)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	For the default manifest collection, this should be set to 'DEFAULT'.
IsDefault		Mandatory	Denotes whether or not this manifest collection is a provider defined default manifest collection. For the default manifest collection this is set to 'true'.

48.6.19SNIA_OperationalPowerStatisticalData

The OperationalPowerStatisticalData class defines the statistics properties that may be kept for an metered element of the storage entity.

Instances of this class will exist for each of the metered elements if the 'ElementTypesSupported' property of the SNIA_OperationalPowerStatisticsCapabilities indicates that the metered element is supported. For example, 'Computer System' is identified in the 'ElementTypesSupported' property, then this indicates support for metering of the Top level computer system or 'Component Computer System'.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 490 describes class SNIA_OperationalPowerStatisticalData.

Table 490 - SMI Referenced Properties/Methods for SNIA_OperationalPowerStatisticalData

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	The InstanceID for OperationalPowerStatisticalData instance shall be unique across all instances of the OperationalPowerStatisticalData class.
StatisticTime		Mandatory	Time statistics table by object was last updated (Time Stamp in CIM 2.2 specification format).
ElementType		Mandatory	This value is required AND current version of SMI-S specifies the following values: ValueMap {'2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'} Values { 'Computer System', 'Front-end Computer System', 'Peer Computer System', 'Back-end Computer System', 'Front-end Port', 'Back-end Port', 'Volume', 'Extent', 'Disk Drive', 'Arbitrary LUs' , 'Remote Replica Group'}.
Milliwatts		Mandatory	

48.6.20SNIA_OperationalPowerStatisticsCapabilities

An instance of the OperationalPowerStatisticsCapabilities class defines the specific support provided with the statistics implementation. Note: There would be zero or one instance of this class in a profile. There would be none if the profile did not support the Operational Power profile. There would be exactly one instance if the profile did support the Operational Power Statistics profile.

SNIA_OperationalPowerStatisticsCapabilities class is subclassed from CIM_Capabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 491 describes class SNIA_OperationalPowerStatisticsCapabilities.

Table 491 - SMI Referenced Properties/Methods for SNIA_OperationalPowerStatisticsCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
ElementTypesSupported		Mandatory	ValueMap { '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12' }, Values {'Computer System', 'Front-end Computer System', 'Peer Computer System', 'Back-end Computer System', 'Front-end Port', 'Back-endPort', 'Volume', 'Extent', 'Disk Drive', 'Arbitrary LUs' , 'Remote Replica Group'}.
SynchronousMethodsSupported	N	Mandatory	This property is mandatory, but the array may be empty. ValueMap { '2', '3', '4', '5', '6', '7', '8'}, Values {'Exec Query', 'Indications', 'QueryCollection', 'GetStatisticsCollection', 'Manifest Creation', 'Manifest Modification', 'Manifest Removal' }.
AsynchronousMethodsSupported		Optional	Not supported in current version of SMI-S.
ClockTickInterval		Mandatory	An internal clocking interval for all timers in the subsystem, measured in microseconds (Unit of measure in the timers, measured in microseconds). Time counters are monotonically increasing counters that contain 'ticks'. Each tick represents one ClockTickInterval. If ClockTickInterval contained a value of 32 then each time counter tick would represent 32 microseconds.

48.6.21SNIA_OperationalPowerStatisticsService

The OperationalPowerStatisticsService class provides methods for statistics retrieval and Manifest Collection manipulation.

There shall be an instance of the OperationalPowerStatisticsService, if the Operational Power profile is implemented. It is not necessary to support any methods of the service, but the service shall be populated.

The methods that are supported can be determined from the SynchronousMethodsSupported and AsynchronousMethodsSupported properties of the OperationalPowerStatisticsCapabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 492 describes class SNIA_OperationalPowerStatisticsService.

Table 492 - SMI Referenced Properties/Methods for SNIA_OperationalPowerStatisticsService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
GetStatisticsCollection()		Conditional	Conditional requirement: Clients can get statistics collections using the GetStatisticsCollection as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or Clients can get statistics collections using the GetStatisticsCollection as identified by SNIA_OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported containing '5' (GetStatisticsCollection). This method retrieves all statistics kept for the profile as directed by a manifest collection.
CreateManifestCollection()		Conditional	Conditional requirement: Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported containing '6' (Manifest Creation). This method is used to create client defined manifest collections.
AddOrModifyManifest()		Conditional	Conditional requirement: Clients can modify manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or Clients can modify manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported containing '7' (Manifest Modification). This method is used to add or modify statistics manifests in a client defined manifest collection.
RemoveManifests()		Conditional	Conditional requirement: Clients can remove manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or Clients can remove manifests as identified by SNIA_OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on OperationalPowerStatisticsCapabilities.SynchronousMethodsSupported or OperationalPowerStatisticsCapabilities.AsynchronousMethodsSupported containing '8' (Manifest Removal). This method is used to remove a statistics manifest from a client defined manifest collection.

49 Cross Profile Considerations

49.1 Overview

Many applications access data from multiple profiles to perform operations. Figure 71 shows a client application communicating with multiple SMI-S agents.

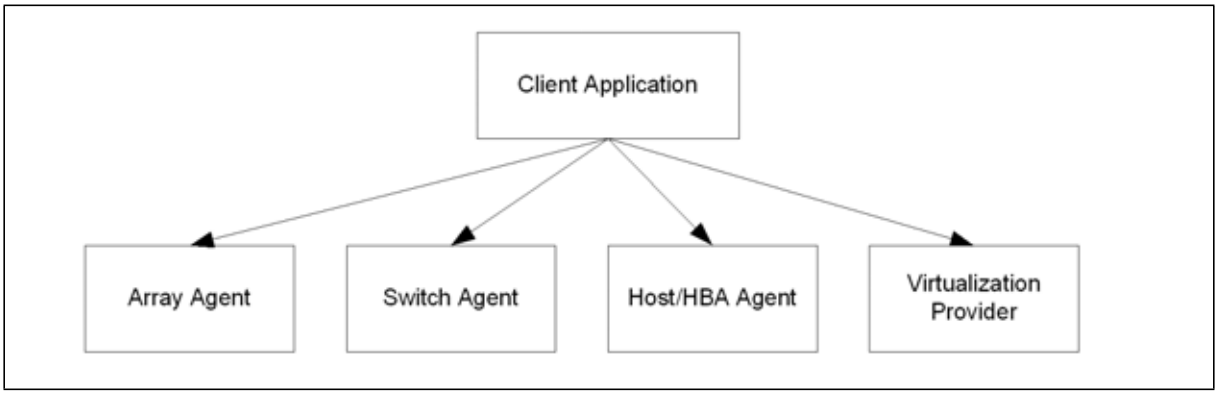

Figure 71 - System Diagram

This section describes algorithms that can be used to associate objects from different profiles to understand connections between the profiles. The algorithms use Durable Names to match objects from different profiles. Figure 72 and Figure 73 are simplified instance diagrams that are used to illustrate the algorithms.

49.2 HBA model

Figure 72 represents a simple “Host Bus Adapter”. The model includes objects that represent a single port Fibre channel HBA. The model also includes a storage volume being accessed through the HBA.

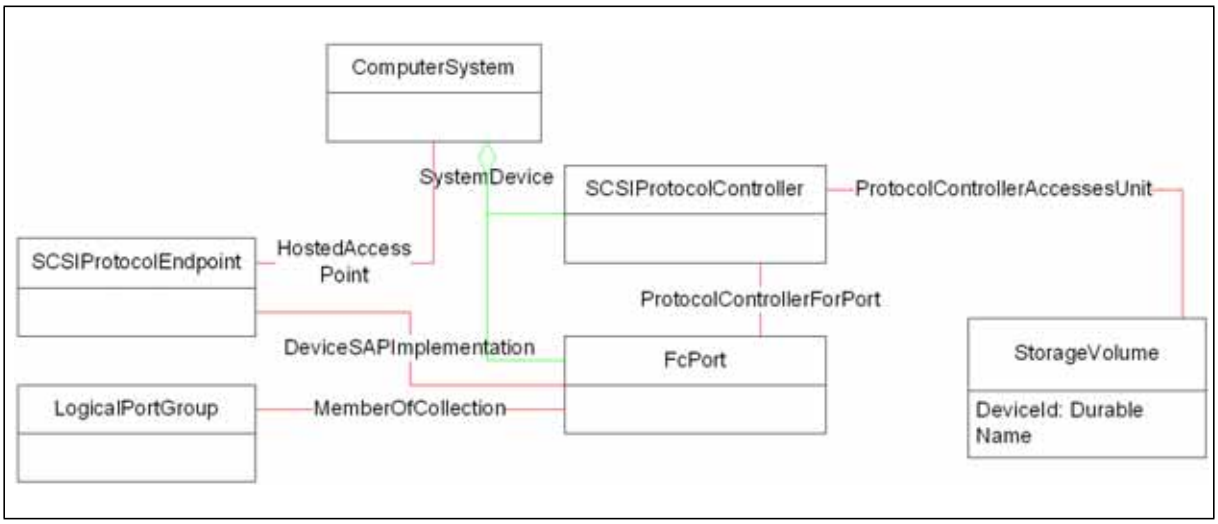

Figure 72 - Host Bus Adapter Model

49.3 Switch Model

Figure 73 represents a two-port Fibre channel switch. The model also includes objects representing links to remote ports the switch agent knows about, and ComputerSystems.

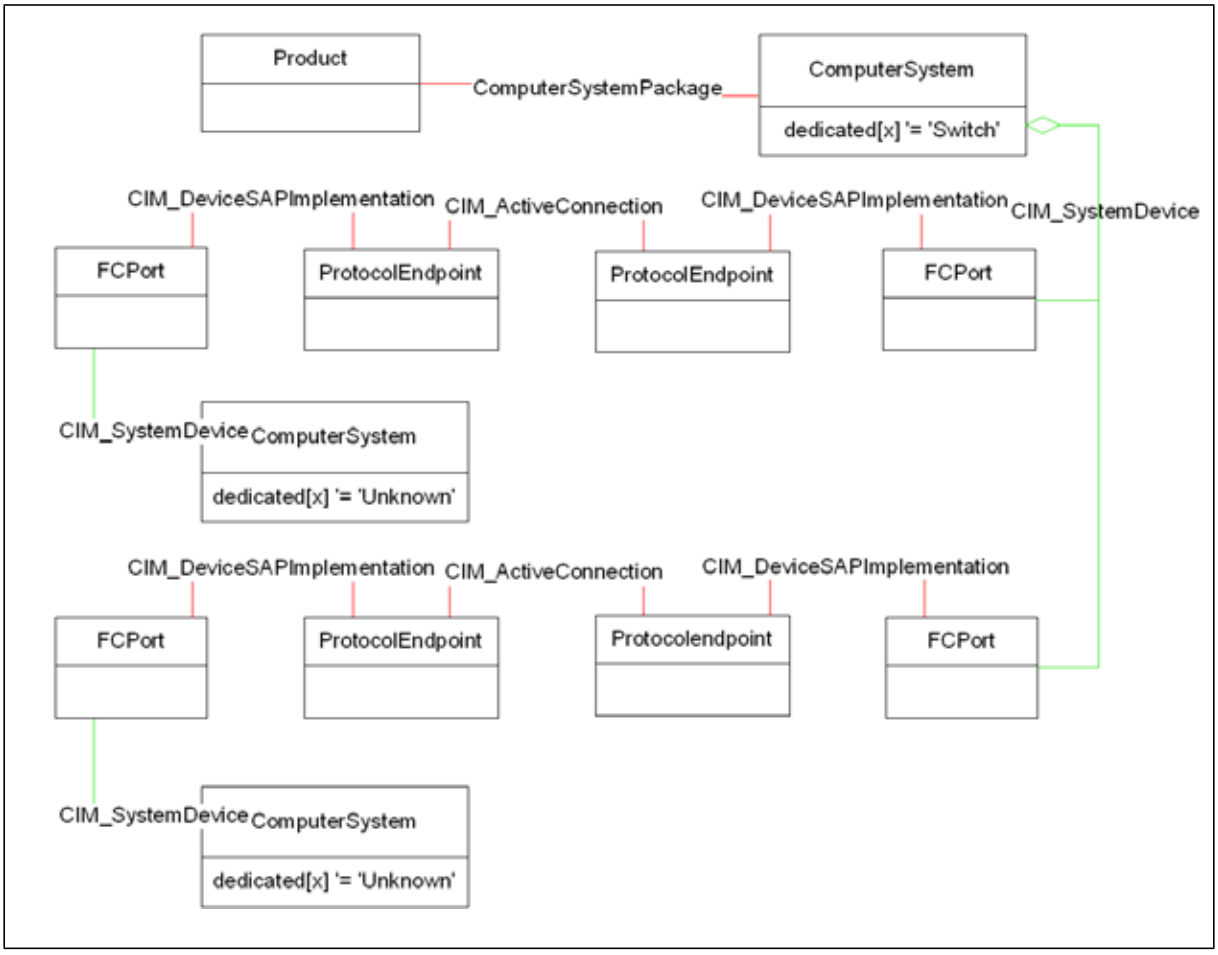

Figure 73 - Switch Model

49.3.1 Recipes

49.3.1.1 Disclaimer

The recipes in this section are included for illustrative purposes only. These recipes are not part of CTP and may not have been validated.

49.3.1.2 Create MAP

```
// DESCRIPTION
// Create a map of how elements in a SAN are connected together via Fibre-Channel
                            ports
//
// The map is built in array $attachedFcPorts->[], where the index is a
// WWN of any device port on the SAN, and the value at that index is
// the object path of the connected switch port.
//
// First find all the switches in a SAN.  Get all the FCPorts for each
// switch and get the Attached FCPorts for each Switch FCPort.  Save
```

```
// these device ports in the map described above.

// PREEXISTING CONDITIONS AND ASSUMPTIONS
// 1.  All agents/namespaces supporting Fabric Profile previously identified using
                          SLP

// Do this for each CIMOM supporting Fabric Profile

switches[] = enumerateInstances("CIM_ComputerSystem", true, false, true, true,
                          null)

for #i in $switches[]
{
    if (!contains(5, $switches[#i].Dedicated))
        continue // only process switches, not other computer systems

    $fcPorts->[] = AssociatorNames(
        $switches[#i].getObjectPath(),
        "CIM_SystemDevice",
        "CIM_FCPort",
        "GroupComponent",
        "PartComponent")

    for #j in $fcPorts->[]
    {
        $protocolEndpoints->[] = AssociatorNames(
             fcPorts->[#j],
             "CIM_DeviceSAPImplementation",
             "CIM_ProtocolEndpoint",
             "Antecedent",
             "Dependent");

       // NOTE - It is possible for this collection to be empty (ports that are not
        // connected).  It is NOT possible for this collection to have more than
                          one
        // element
        if ($protocolEndpoints->[].length == 0)
            continue

        $attachedProtocolEndpoints->[] = AssociatorNames(
            $protocolEndpoints->[0],
         "CIM_ActiveConnection",
            "CIM_ProtocolEndpoint",
            null, null) // NOTE: role & resultRole are null as the
                        // direction of the association is not
                        // dictated by the specification

        for #k in $attachedProtocolEndpoints->[] {
```

```
            // $attachedFcPort is either a device port or an ISLÂ’d
          // switch port from another switch. We store this result
          // (i.e. which device FCPort is connected to which switch
          // FCPort) in a suitable data structure for subsequent
          // correlation to ports discovered on devices.
             $attachedFcPorts->[] = Associators(
         $attachedProtocolEndpoints->[#k],
         “CIM_DeviceSAPImplementation”,
         “CIM_FCPort”,
                “Dependent”,
                “Antecedent”,
                false,
                false,
                [“PermanentAddress”])

            $attachedFcPort = $attachedFcPorts[0] // Exactly one member guaranteed
                                by model
             #wwn = $attachedFcPort.PermanentAddress
          $attachedFcPorts->[#wwn] = $fcPorts->[#j]
         }
    }
}
```

49.3.1.3 HBA to Switch Physical Path

```
// DESCRIPTION
// Determine physical path from HBA to switch.
//
// For each HBA port on every host, determine the connected switch
// port.  NOTE: Not every HBA port will be connected to a switch port,
// and not every switch port will be connected to a device port.  Only
// the connections between HBA ports and switch ports are discovered
// by this recipe
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  All agents/namespaces supporting HBA Profile previously identified using
                           SLP
// 2.  Array $attachedFcPorts->[] is a map of how elements in a SAN are
// connected together via Fibre-Channel ports. Each index is a WWN of
// any device port on the SAN, and the value at that index is the
// connected switch port.

// Do this for each CIMOM supporting HBA Profile

$hosts[] = enumerateInstances(“CIM_ComputerSystem”)

for #i in $hosts->[]
{
```

```
    if (!contains(0, $hosts[#i].Dedicated))
        continue // only process systems that are “not dedicated”

    $fcPorts[] = Associators(
        $hosts[#i].getObjectPath(),
        “CIM_SystemDevice”,
        “CIM_FCPort”,
        “GroupComponent”,
        “PartComponent”,
        false,
        false,
        [“PermanentAddress”])

    for #j in $fcPorts[]
    {
        // Get the FCPort WWN
        #wwn = $fcPorts[#j].PermanentAddress

        // Match this device port WWN to one (or less) switch
        // ports, by using the mapping table
        $attachedSwitchPort-> = $attachedFcPorts->[#wwn]

       // Note - if there is no entry in the mapping array, this
       // port is not connected to any switch
    }
}
```

49.3.1.4 Array to Switch Physical Path

```
// DESCRIPTION
// Determine physical path from Storage Arrays to Switches
//
// For each fibre-channel port on every array, determine the connected
// switch port.  NOTE: This identifies the FrontEnd I/O Controllers
// (and Storage Arrays) whose ports are physically connected to
// some of the ports of some of the Switches. This recipe does not
// distinguish and does not filter the front-end FC Port from the
// back-end FC Ports.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  All agents/namespaces supporting Array Profile previously identified using
                          SLP
// 2.  Array $attachedFcPorts[] is a map of how elements in a SAN are
// connected together via Fibre-Channel ports. Each index is a WWN of
// any device port on the SAN, and the value at that index is the
// connected switch port.
```

```
// Do this for each CIMOM supporting the Array Profile

$storageArrays[] = enumerateInstances(“CIM_ComputerSystem”);


// NOTE: Some of the ports contained will be back-end ports, but they will
//   have no connectivity to switches, so we won’t distinguish them
//   from unconnected front-end ports

for #i in $storageArrays[]
{
    if (!contains(3, $storageArrays[#i].Dedicated))
        continue // only process systems that are dedicated “storage”

    if (!contains(15, $storageArrays[#i].Dedicated))
        continue // only process systems that are dedicated “block server”

    $fcPorts[] = Associators(
        $storageArrays[#i].getObjectPath(),
     “CIM_SystemDevice”,
     “CIM_FCPort”,
        “GroupComponent”,
        “PartComponent”,
        false,
        false,
        [“PermanentAddress”])

    for #j in $fcPorts[]
    {
     // Get the FCPort WWN
     #wwn = $fcPorts[#j].PermanentAddress

     // Match this device port WWN to one (or less) switch
     // ports, by using the mapping table

        $attachedSwitchPort-> = $attachedFcPorts->[#wwn]

       // Note - if there is no entry in the mapping array, this
       // port is not connected to any switch
    }
}
```

49.4 Array Model

Figure 74 is a simple model of a disk array. The array has a single controller with a single Fibre channel port on the front end and a single parallel SCSI port for the disks. The model shows two disks that are

members of a single redundancy group. Part of the redundancy group is made available over the Fibre channel as a single volume.

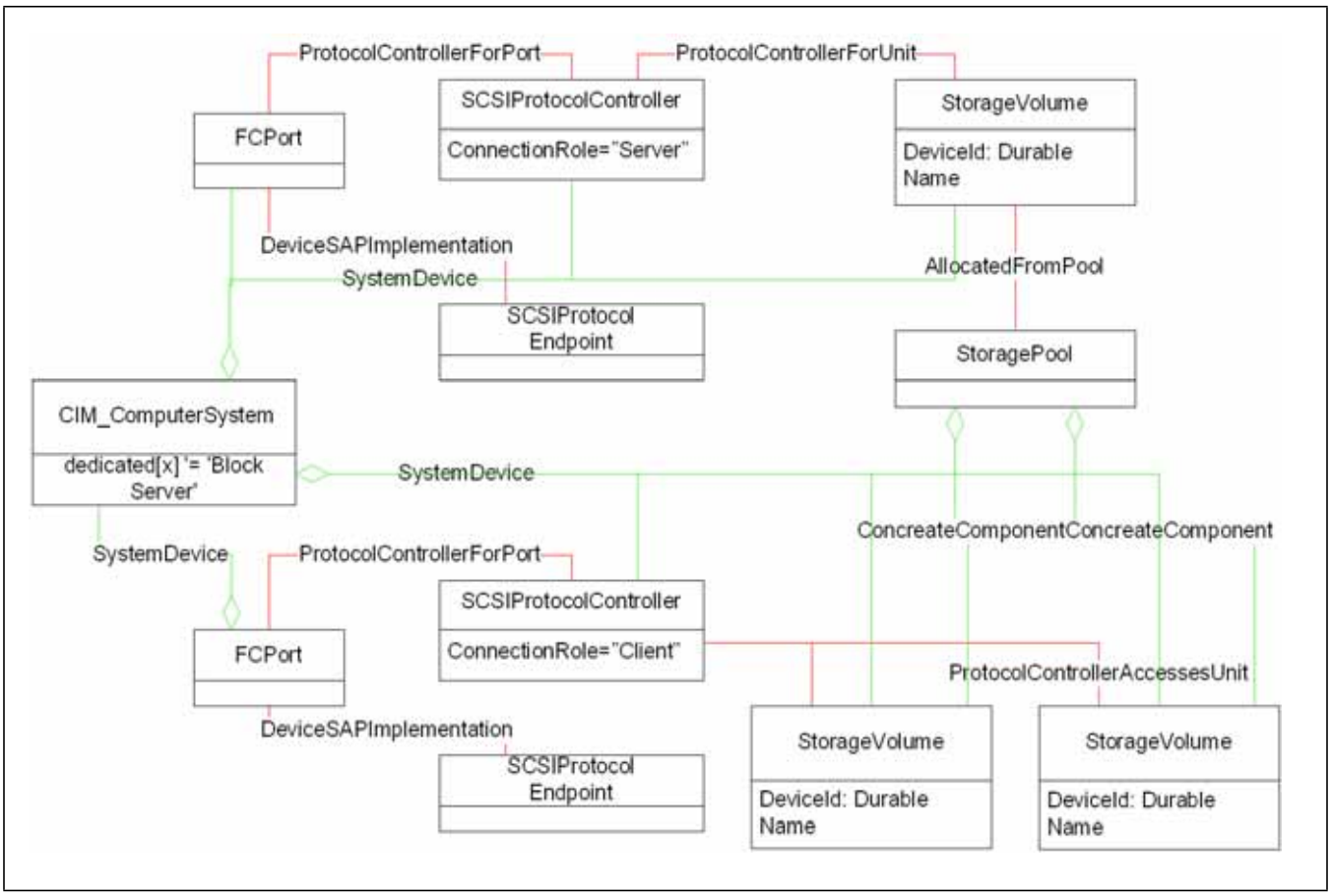

Figure 74 - Array Instance

49.5 Storage Virtualization Model

Figure 75 is a simple model of a Storage Virtualizer. The model shows the basic controller and pool. The model also shows a single volume being used and a single volume being served to a host.

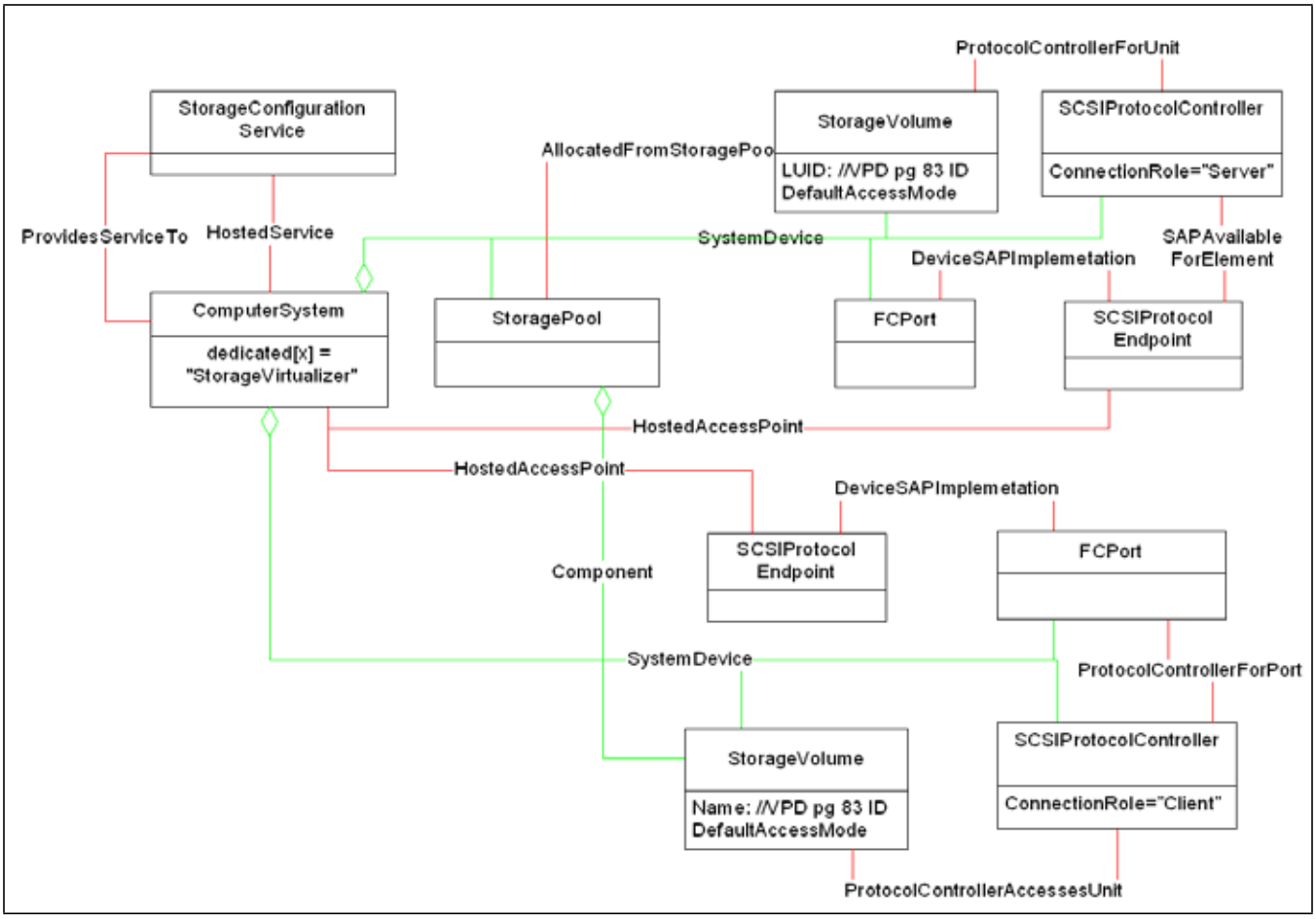

Figure 75 - Virtualization Instance

49.6 Fabric Topology (HBA, Switch, Array)

A map of a SAN that shows all the elements and the connections between them is very useful. To create the map all the elements in the SAN with their Fibre channel ports are first located. Next the ports are linked together. Figure 76 shows how a SAN map can be constructed.

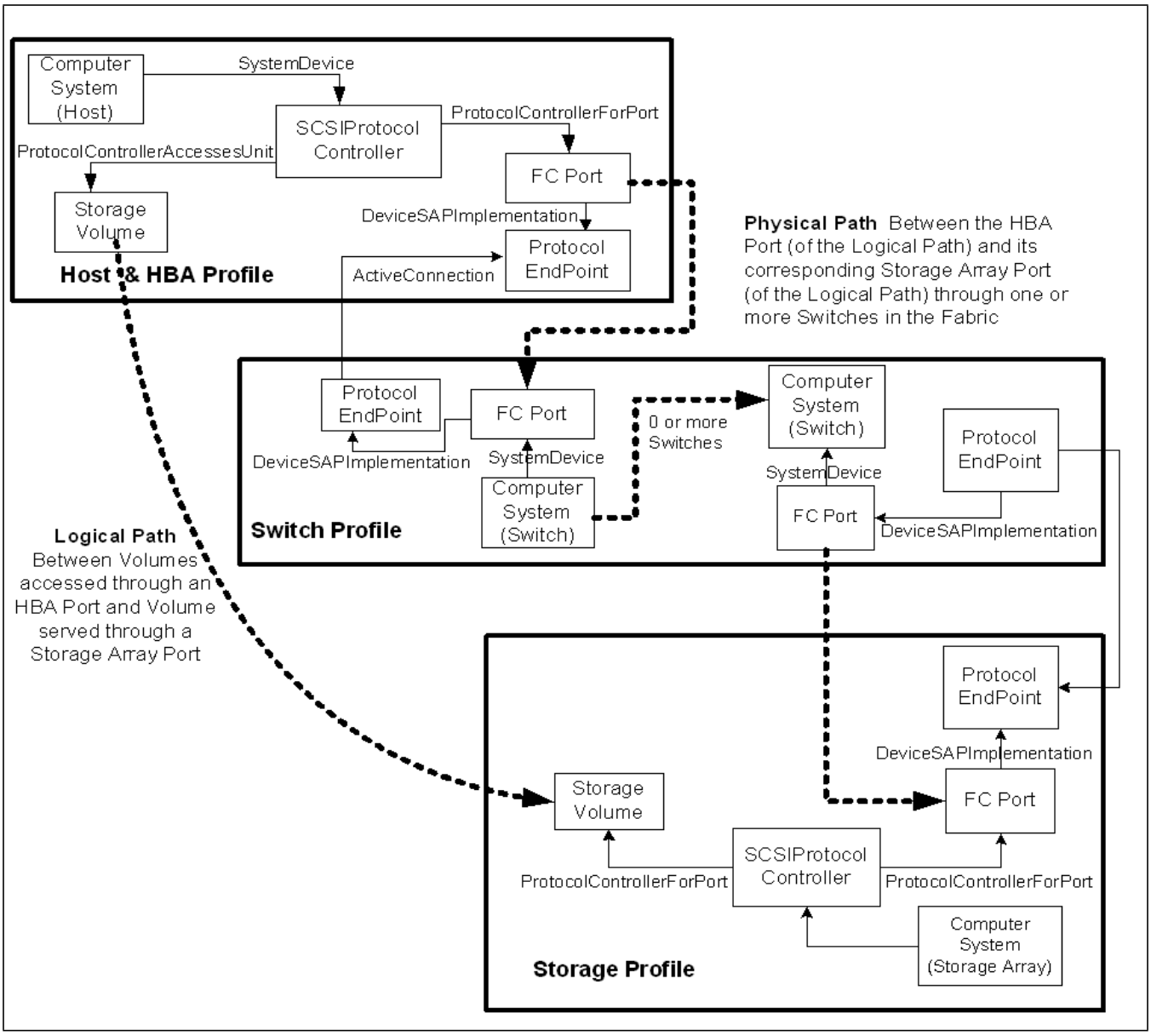

Figure 76 - Fabric Topology

To locate all the elements in a SAN, you start by locating the agents. SMI-S agents are located using SLP. Once the agents are located, intrinsic methods are used to enumerate ComputerSystem objects. Each ComputerSystem object represents an element in the SAN. The ComputerSystem object's "Dedicated" attribute can be used to identify the type of the element.

After the elements are located, Fibre channel ports for each element are discovered. For each ComputerSystem object follow SystemDeviceFCPort objects and ProtocolController objects. For each ProtocolController object follow the ProtocolControllerForPort associations to FCPort objects. Use the

information in the FCPort objects found to determine the Durable Name for the FCPort object. The Durable Name is used to match the ports to objects in other profiles.

Now to link the elements' ports together find the Switch elements. Switches know about ports on elements logged into their ports. To find this information start by locating the ComputerSystem objects that represents switches. Switches can be identified by the "Dedicated" attribute of the ComputerSystem object being set to "Switch". For each switch follow the SystemDeviceFCPort objects that represent the ports of the switch. Next look for ActiveConnection ActiveConnectionFCPort objects. These FCPort objects represent the ports on the other side of a link. Use attributes from the FCPort object to determine the Durable Name. These identifiers are then matched to identifiers found in other profiles to complete the connections.

49.6.1 Logical Device Composition

The Logical Device Composition Recipe traces the objects and associations that make up a LogicalDevice across profile boundaries. It serves performance and fault identification use-cases by allowing the user to map out all the objects in the I/O stack that may contribute to the storage services a LogicalDevice provides to applications. It covers the Disk Partition, Volume Manager, Disk, Multipath, Host Discovered Resources, Common Initiator, Fabric, iSCSI Target, Storage Virtualizer, Array, and Storage Library Profiles and Subprofiles. This recipe also shows how Correlatable Naming conventions may be used to identify and correlate instances of objects within, and across profiles.

49.6.1.1 Main Recipe

```
Logical Device Composition Recipe

// This main recipe is profile-independent.  It
// uses subroutines which are profile-dependent.
//
// DESCRIPTION:
//
// By stitching together information from
// multiple profiles, determine the logical composition
// a host LogicalDevice in terms of its constituent
// LogicalDevices, ProtocolControllers, Ports, StoragePools,
// etc. and the associations between them.  This host LogicalDevice
// would typically either be a disk volume or tape device.
// Collect sufficient information to allow a graph to be drawn.
// Where possible, allow network topologies to be attached.
//
// PREEXISTING CONDITIONS AND ASSUMPTIONS:
//
// That all providers relevant to the logical composition
// of the device have been discovered (see the Server Profile
// recipe "Find Servers Supporting a Given Profile),
// can be queried for the information they have to contribute,
// and follow SMI-S 1.2
//
// Durable Names naming conventions to allow stitching
// across profiles and providers. Correlatable, unique
// and durable names are assumed if this is to work.
// In particular, this must be true of instances of:
```

```
//
// CIM_LogicalDisk
// CIM_TapeDrive
// CIM_StorageExtent
// CIM_SCSIProtocolEndpoint  // From Host Discovered Resources.
// CIM_iSCSIProtocolEndpoint  // From iSCSI Initiator
//
// Which are node objects, included in multiple profiles.
//
// Other CIM Classes to be added as nodes are:
//
// CIM_SCSIProtocolController
// CIM_ProtocolEndpoint
// CIM_LogicalPort
// CIM_ComputerSystem
// CIM_iSCSISession
// CIM_EthernetPort
// CIM_StoragePool
// CIM_DiskDrive
// CIM_GenericDiskPartition


// SUBROUTINES:
//
// Each subroutine of this recipe has access to all
// providers relevant to the path under consideration.


// Add $node to $nodes[] if it has not already been added.
// If a new node was added, set #new_added to true.
sub AddIfNotAlreadyAdded(IN     CIM_LogicalElement   $node,
                         IN/OUT CIM_LogicalElement[] $nodes[],
                         OUT boolean #new_added,
                         OUT int #error_code);

// Add $link to $links[] if it has not already been added.
// If a new link was added, set #new_added to true.
sub AddLinkIfNotAlreadyAdded(IN     CIM_Dependency   $link,
                             IN/OUT CIM_Dependency[] $links[],
                             OUT boolean #new_added,
                             OUT int #error_code);

// Compare two LogicalElement references to determine if
// they represent the same modelled object. The two nodes
// may come from entirely different providers/profiles.
// This method uses correlatable naming conventions defined
// for the classes in question by the Profiles/SubProfiles.
```

```
sub RepresentsTheSameObject(IN     CIM_LogicalElement   $node1,
                            IN     CIM_LogicalElement   $node2,
                            OUT int #error_code);

// Compare two Dependency references to determine if
// they represent the same modelled association. The two links
// may come from entirely different providers/profiles.
// This method uses correlatable naming conventions defined
// for the endpoints in question by the Profiles/SubProfiles.
sub RepresentsTheSameAssociation(IN     CIMObjectPath   $link1->,
                                 IN     CIMObjectPath   $link2->,
                                 OUT int #error_code) {

// Given the Names and NameFormats of two object instances,
// determine if the two instances represent the same modelled object
// unambiguously according to correlatable names semantics.
// Return true only if the match is unambiguous.
sub MatchUnambiguouslyByNameNameFormat(string name1,
                                       string nameFormat1,
                                       string name2,
                                       string nameFormat2
                                      );

// Given the Names and ConnectionTypes of two SCSIProtocolEndpoint instances,
// determine if the two instances represent the same modelled object
// unambiguously according to correlatable names semantics for
// SCSIProtocolEndpoint.
// Return true only if the match is unambiguous.
sub MatchUnambiguouslyByNameAndConnectionType(string name1,
                                              int16 conType1,
                                              string name2,
                                              int16 conType2
                                             );

// Given the IdentifyingDescriptions and OtherIDentifyingInfo
// arrays of two object instances,
// determine if the two instances represent the same object
// unambiguously according to Correlatable names semantics
// for ComputerSystem names.
// Return true only if the match is unambiguous.
sub MatchUnambiguouslyByIdentifyingInfo(string info1[],
                                        string desc1[],
                                        string info2[],
                                        string desc2[]
                                       );

// Given an instance of an object from one provider,
```

```
// find the instance of the same object in the current
// provider.  Return null if not found.
sub GetProviderInstanceOf(IN CIM_LogicalElement  $that_node,
                          OUT CIM_LogicalElement $this_node,
                          OUT int #error_code);


// In this function, the layer is passed references
// to the working graph.  It is expected that the layer
// will search the structures for objects it recognizes
// and can add new objects and associations to the graph.
// If the layer does not exist or does not recognize
// any of the objects or associations in the graph
// as objects it manages or knows about, it adds nothing.
// Set #new_added to true if the layer contributed anything new to
// contribute to the graph.

sub AddToGraphFromLayerXXX( IN/OUT CIM_LogicalElement $nodes[],
                            IN/OUT CIM_Dependency     $links[],
                            OUT boolean #new_added,
                            OUT int #error_code
                          );

// This fictitious function would draw a node in
// this logical composition on a canvas.  The net effect
// of drawing all the nodes would be
// an arrangement of boxes containing CIM class names
// and identifiers of those objects.
sub DrawNode($node);

// This fictitious function would draw a line representing
// the specified association between two nodes.  The net
// result would be a graph directed graph of the nodes
// with their associations.
sub DrawLinkBetweenNodes($link);


// ------ Main Recipe ------


// Begin with a CIM_LogicalDevice reference representing a
// volume on which a filesystem has been placed or is
// being used “raw” by an application managing its own
// block structures.  The CIM_LogicalDeivce could also
// represent a tape device such as (/dev/rmtX or \\.\\TAPEX)

$logicaldevice;
```

```
// The goal is to build two arrays: An array of objects
// representing nodes in the logical topology graph, and
// an array of Associations linking those objects to form
// a directed graph.

// CIM_LogicalElement[] $nodes[];
// CIM_Dependency[]     $links[];

// Define some other flow control variables.
boolean #new_objects_added = true;
int #error_code = 0;

// Start by adding the top level volume.

$node[0] = $logicaldevice;
#new_objects_added = true;

// Now, build down through the layers building what
// should be a breadth-first traversal of the tree graph.
// Repeatedly cycle through the layers until no new objects
// have been added. This allows for multiple layers of
// virtualization and network to kick in if new objects are found
// from the layers above added in previous passes.

while (#new_objects_added) {

   boolean #added;
   #new_objects_added = false;

   &AddToGraphFromLayerVolumeManager($nodes[],
                                     $links[],
                                     #added,
                                     #error_code);
   #new_objects_added |= #added;
   if (0 != #error_code) { return #error_code; }

   &AddToGraphFromLayerDiskPartitioning($nodes[],
                                        $links[],
                                        #added,
                                        #error_code);
   #new_objects_added |= #added;
   if (0 != #error_code) { return #error_code; }

   &AddToGraphFromLayerLocalDiskDrive($nodes[],
                                      $links[],
                                      #new_objects_added,
```

```
                                    #error_code);
if (0 != #error_code) { return #error_code; }

&AddToGraphFromLayerMultipath($nodes[],
                              $links[],
                              #added,
                              #error_code);
#new_objects_added |= #added;
if (0 != #error_code) { return #error_code; }

&AddToGraphFromLayerHostDiscoveredResources($nodes[],
                                            $links[],
                                            #added,
                                            #error_code);
#new_objects_added |= #added;
if (0 != #error_code) { return #error_code; }

&AddToGraphFromLayerCommonInitiator($nodes[],
                                    $links[],
                                    #added,
                                    #error_code);
#new_objects_added |= #added;
if (0 != #error_code) { return #error_code; }

&AddToGraphFromLayerFabric($nodes[],
                           $links[],
                           #added,
                           #error_code);
#new_objects_added |= #added;
if (0 != #error_code) { return #error_code; }

&AddToGraphFromLayerIPNetwork($nodes[],
                              $links[],
                              #added,
                              #error_code);
#new_objects_added |= #added;
if (0 != #error_code) { return #error_code; }

&AddToGraphFromLayerStorageVirtualizer($nodes[],
                                       $links[],
                                       #added,
                                       #error_code);
#new_objects_added |= #added;
if (0 != #error_code) { return #error_code; }

&AddToGraphFromLayerArray($nodes[],
                          $links[],
```

```
                              #added,
                              #error_code);
   #new_objects_added |= #added;
   if (0 != #error_code) { return #error_code; }

   &AddToGraphFromLayerStorageLibrary($nodes[],
                                      $links[],
                                      #added,
                                      #error_code);
   #new_objects_added |= #added;
   if (0 != #error_code) { return #error_code; }

} // while.

// Now “draw” the logical device composition. In reality these functions
// would need to rather sophisticated with geometric constraints
// to draw a nice looking graph.


for #i in $nodes[] {
   &DrawNode($nodes[#i]);
}

for #i in $links[] {
   &DrawLinkBetweenNodes($link[#i]);
}


// ------ Supporting Subroutines ------


sub AddIfNotAlreadyAdded(IN     CIM_LogicalElement   $node,
                         IN/OUT CIM_LogicalElement[] $nodes[],
                         OUT boolean #new_added,
                         OUT int #error_code) {
    boolean #wasFound = false;
    boolean #new_added = false;

    // Search through the nodes looking for a match.
    // Not a particularly efficient way of doing it, but functional.
    for #i in $nodes[] {
        if (&RepresentsTheSameObject($node, $nodes[i], #error_code)) {
            if (compare(#error_code, 0)) {
               #wasFound = true;
            }
            break;
        }
```

```
    }

    // If we did not find a match, and there were no errors, add it.
    if ( (!#wasFound) && (compare(#error_code, 0))) {
        $nodes[].add($node);
        #new_added = true;
    }
} // AddIfNotAlreadyAdded.


// We are not being too picky about strong typing here.
// This function will take associations that are not subclasses
// of CIM_Dependency ( such as the trinary CIM_SCSIInitiatorLogicalUnitPath)
sub AddLinkIfNotAlreadyAdded(IN     CIM_Dependency   $link,
                             IN/OUT CIM_Dependency[] $links[],
                             OUT int #error_code) {
    boolean #wasFound = false;
    #new_added = false;

    // Search through the nodes looking for a match.
    // Not a particularly efficient way of doing it, but functional.
    for #i in $links[] {
        if (&RepresentsTheSameAssociation($link.getObjectPath(),
                                          $links[#i].getObjectPath(),
                                          #error_code)) {
            if (compare(#error_code,0)) {
               #wasFound = true;
            }
            break;
        }
    }

    // If we did not find a match, and there were no errors, add it.
    if ( (!#wasFound) && (compare(#error_code, 0))) {
        $links[].add($link);
        #new_added = true;
    }
} // AddLinkIfNotAlreadyAdded.

sub RepresentsTheSameAssociation(IN     CIMObjectPath   $link1->,
                                 IN     CIMObjectPath   $link2->,
                                 OUT int #error_code) {

    // Determine if the links are the same by comparing thier class

    if (
```

```
// Now compare the correlatable identifiers of their endpoints.
if !compare($link1->getObjectClass(), $link2->getObjectClass()) return false;

// Handle descendents of CIM_Dependency.
if (($link1-> ISA CIM_Dependency) && ($link2-> ISA CIM_Dependency)) {

   if (
        (&RepresentsTheSameObject($link1->Antecedent,
                                  $link2->Antecedent, #error_code) &&
        (&RepresentsTheSameObject($link1->Dependent,
                                  $link2->Dependent, #error_code)
       ) {
       return true;
   } else {
       return false;
   }

// Handle the trinary association here.
} else if ( ($link1-> ISA CIM_SCSIInitiatorLogicalUnitPath) &&
            ($link2-> ISA CIM_SCSIInitiatorLogicalUnitPath)       ) {

   if (
        (&RepresentsTheSameObject($link1->Initiator,
                                  $link2->Initiator, #error_code) &&
        (&RepresentsTheSameObject($link1->Target, $link2->Target,
                                  #error_code)                    &&
        (&RepresentsTheSameObject($link1->LogicalUnit,
                                  $link2->LogicalUnit, #error_code)
       ) {
       return true;
   } else {
       return false;
   }

// Handle the CIM_SAPAvailableForElement association here.
} else if ( ($link1-> ISA CIM_SAPAvailableForElement) &&
            ($link2-> ISA CIM_SAPAvailableForElement)       ) {

   if (
        (&RepresentsTheSameObject($link1->AvailableSAP,
                                  $link2->AvailableSAP, #error_code) &&
        (&RepresentsTheSameObject($link1->ManagedElement, i
                                  $link2->ManagedElement, #error_code)
       ) {
       return true;
   } else {
       return false;
```

```
       }

    } else {
        return false;
    }
}


sub RepresentsTheSameObject(IN     CIM_LogicalElement   $node1,
                            IN     CIM_LogicalElement   $node2,
                            OUT int #error_code) {

    int #error_code = 0;
    boolean #result;

    // First, check if this is the same instance by checking object path.
    if (compare($node1.getObjectPath(), $node2.getObjectPath())) {
       return true;
    }

    // SCSIProtocolEndpoint is handled by Name and ConnectionType.
    if ($node1 ISA CIM_SCSIProtocolEndpoint) &&
       ($node2 ISA CIM_SCSIProtocolEndpoint)) {
       ) {
       #result = &MatchUnambiguouslyByNameAndConnectionType(
                                        $node1.Name, $node1.ConnectionType,
                                        $node2.Name, $node2.ConnectionType);
    // LogicalDevice and its subclasses StorageExtent and
    // LogicalDisk are handled
    // by IdentifyingInfo.
    } else if (($node1 ISA CIM_LogicalDevice) &&
               ($node2 ISA CIM_LogicalDevice)) {
         #result = &MatchUnambiguouslyByIdentifyingInfo(
                                       $node1.OtherIdentifyingInfo[],
                                       $node1.IdentifyingDescriptions[],
                                       $node2.OtherIdentifyingInfo[],
                                       $node2.IdentifyingDescriptions[]);
    // ComputerSystems are compared by two methods.
    } else if ($node1 ISA CIM_ComputerSystem) &&
              ($node2 ISA CIM_ComputerSystem)) {
         #result = &MatchUnambiguouslyByNameNameFormat(
                                       $node1.Name, $node1.NameFormat,
                                       $node2.Name, $node2.NameFormat);
       if (!#result) {
         #result = &MatchUnambiguouslyByIdentifyingInfo(
                                       $node1.OtherIdentifyingInfo[],
                                       $node1.IdentifyingDescriptions[],
```

```
                                        $node2.OtherIdentifyingInfo[],
                                        $node2.IdentifyingDescriptions[]);
      }

   // These objects are compared by name.
   } else if (($node1 ISA CIM_GenericDiskPartition) &&
              ($node2 ISA CIM_GenericDiskPartition)) {
      #result = (compare($node1.Name, $node2.Name));
   } else if (($node1 ISA CIM_FCPort) && ($node2 ISA CIM_FCPort)) {
      #result = (compare($node1.Name, $node2.Name));

   // These DiskDrive and StoragePool have their own monikers.
   } else if (($node1 ISA CIM_DiskDrive) && ($node2 ISA CIM_DiskDrive)) {
      #result = (compare($node1.DeviceID, $node2.DeviceID));
   } else if (($node1 ISA CIM_StoragePool) && ($node2 ISA CIM_StoragePool)) {
      #result = (compare($node1.InstanceID, $node2.InstanceID));
   } else {
      < this method can't handle this type >
      #error_code = 1;
      return false;
   }
   return #result;

} // RepresentsTheSameObject.


sub MatchUnambiguouslyByNameAndConnectionType(string name1,
                                              int16 conType1,
                                              string name2,
                                              int16 conType2
                                             ) {

   if (conType1 != conType2) {
       return false;
   } else {
       if (compare(name1, name2)) {
          return true;
       }
   }
   return false;
}

sub MatchUnambiguouslyByNameNameFormat(string name1,
                                       string nameFormat1,
                                       string name2,
                                       string nameFormat2
                                      ) {
```

```
    if (nameFormat1 != nameFormat2) {
        return false;
    } else {
        if (compare(name1, name2)) {
           return true;
        }
    }
    return false;
}

sub MatchUnambiguouslyByIdentifyingInfo(string info1[],
                                        string desc1[],
                                        string info2[],
                                        string desc2[]
                                       ) {
    boolean #matchFound = false;

    // Loop through both arrays looking for a match.
    for (#i=0; #i<info1[].length; #i++) {
        for (#j=0; #j<info2[].length; #j++) {
            if (MatchUnambiguouslyByNameNameFormat(desc1[#i],
                                                   info1[#i],
                                                   desc2[#j],
                                                   info2[#j]
                                                  )
               ) {
                #matchFound = true;
                break;
        }
        if (#matchFound) {
           break;
        }
    }
    return #matchFound;
}

sub GetProviderInstanceOf(IN CIM_LogicalElement  $that_node,
                          OUT CIM_LogicalElement $this_node,
                          OUT int #error_code) {

    CIM_LogicalElement $possible_matches[];

    $this_node = null;

    // Enumerate through all the instances of this class in this provider
    // looking for a match to $that_node.
```

```
    $possible_matches = EnumInstances($that_node.getClass(), false, false);
    for #i in $possible_matches[] {
        if ( &RepresentsTheSameObject($that_node, $possible_matches[#i],
                                      #error_code)
             && !#error_code) {
           $this_node = $possible_matches[#i];
        )
    }
}  // GetProviderInstanceOf.
```

49.6.1.2 Array paths

```
// Array layer piece of the Logical Device Composition Recipe

// This is based on the
// Array Profile, which uses the Target Port Subprofile.
// It connects LogicalDevices left by the SCSI initiator
// side to StorageVolumes and their LogicalPorts on the array side
// to allow network and logical disk topologies to be correlated.
// Further analysis of the topology inside the array
// will be left fo the next release of SMI-S.

sub AddToGraphFromLayerArray(IN/OUT CIM_LogicalElement $nodes[],
                             IN/OUT CIM_Dependency $links[],
                             OUT    boolean #new_added,
                             OUT    int     #error_code) {

   // CIM_SCSIProtocolController      $found_protocol_controllers[];
   // CIM_ProtocolControllerForUnit   $found_for_unit_associations[];
   // CIM_ProtocolEndpoint            $found_protocol_endpoints[];
   // CIM_DeviceSAPImplementation     $found_sap_associations[];
   // CIM_LogicalPort                 $found_ports[];
   // CIM_SAPAvailableForElement      $found_available_associations[];

   boolean #added = false;

   #new_added = false;

   for #i in $nodes[] {

      if ($nodes[#i] ISA CIM_LogicalDevice) {

       &GetProviderInstanceOf($nodes[#i], $node, #error_code);
```

```
if (#error_code) { return;}

if ($node != null) {

  // Work up the path to include the network ports
  // for stitching in the network topology.

  // Follow an ProtocolControllerForUnit to a SCSIProtocolController.
  $found_protocol_controllers[] = Associators(
                          $node.getObjectPath(),
                          “CIM_SCSIProtocolControllerForUnit”,
                          “CIM_SCSIProtocolController”,
                          “Dependent”,
                          “Antecedent”
                                );

  $found_for_unit_associations[] = References(
                                       $node.getObjectPath(),
                                       “CIM_SCSIProtocolController”,
                                       “Dependent”
                                             );

  // Each LogicalDevice may be handled by multiple controllers.
  for #j in $found_protocol_controllers[] {

     &AddIfNotAlreadyAdded($found_protocol_controllers[#j],
                          $nodes[], #added, #error_code);

     #new_added |= #added;
     &AddLinkIfNotAlreadyAdded($found_for_unit_associations[#j],
                               $links[], #added, #error_code);
     #new_added |= #added;

     // Follow an SAPAvailableForElement to a SCSIProtocolEndpoint.
     $found_protocol_endpoints[] = Associators(
                $found_protocol_controllers[#j].getObjectPath(),
                “CIM_SAPAvailableForElement”,
                “CIM_ProtocolEndpoint”,
                “ManagedElement”,
                “AvailableSAP”
                                          );

     $found_available_associations[] = References(
                $found_protocol_controllers[#j].getObjectPath(),
                “CIM_ProtocolEndpoint”,
                “ManagedElement”
                                             );
```

```
                // Each controller may multipath through multiple ports.
                for #k in $found_protocol_endpoints[] {

                   &AddIfNotAlreadyAdded($found_protocol_endpoints[#k],
                                         $nodes[], #added, #error_code);
                   #new_added |= #added;
                   &AddLinkIfNotAlreadyAdded($found_available_associations[#k],
                                             $links[], #added, #error_code);
                   #new_added |= #added;

                   // Follow the DeviceSAPImplementation to a LogicalPort.
                   // This is a 1:1 relationship.
                   $found_ports[] = Associators(
                                    $found_protocol_endpoints[#k].getObjectPath(),
                                    “CIM_DeviceSAPImplementation”,
                                    “CIM_LogicalPort”,
                                    “Dependent”,
                                    “Antecedent”
                                              );

                   $found_sap_associations[] = References(
                              $found_protocol_endpoints.getObjectPath(),
                              “CIM_LogicalPort”,
                              “Dependent”
                                                    );
                   &AddIfNotAlreadyAdded($found_ports[0],
                                         $nodes[], #added, #error_code);
                   #new_added |= #added;
                   &AddLinkIfNotAlreadyAdded($found_sap_associations[0],
                                             $links[], #added, #error_code);
                   #new_added |= #added;

                } // for #k.

             } // for #j.

          } // for #i.


       } // AddToGraphFromLayerArray.
```

49.6.1.3 Host Discovered Resource

```
       // Host Discovered Resources layer piece of the Logical Device Composition Recipe
```

```
// It uses the Host Discovered Resources Profile.

sub AddToGraphFromLayerHostDiscoveredResources(
                                IN/OUT CIM_LogicalElement $nodes[],
                                IN/OUT CIM_LogicalElement $links[],
                                OUT boolean #new_added,
                                OUT int     #error_code) {
boolean #added = false;
#new_added = false;

for #i in $nodes[] {

   // CIM_SCSIInitiatorTargetLogicalUnitPath $scsi_paths[];
   // CIM_SCSIProtocolEndpoint $initiator_endpoint;
   // CIM_SCSIProtocolEndpoint $target_endpoint;

   #i =0;

   if ($nodes[#i] ISA CIM_LogicalDevice) {

       &GetProviderInstanceOf($nodes[#i], $node, #error_code);
       if (#error_code) { return; }

       if ($node != null) {

          // Find all CIM_SCSIInitiatorTargetLogicalUnitPath
          // with $node as the LogicalUnit reference.
          $scsi_paths[] = References(
                           $node.getObjectPath(),
                           “CIM_SCSIInitiatorLogicalUnitPath”,  //ResultClass
                           “LogicalUnit”                        // Role
                                   );

          for (#j=0; #j<$scsi_paths.length; #j++) {
              &AddLinkIfNotAlready($scsi_paths[#j], $links[],
                                   #added, #error_code);
              #new_added |= #added;
              $initiator_endpoint = $scsi_paths[#j].Initiator;
              $target_endpoint = $scsi_paths[#j].Target;
              &AddIfNotAlreadyAdded($initiator_endpoint, $nodes[],
                                   #added, #error_code);
              #new_added |= #added;
              &AddIfNotAlreadyAdded($target_endpoint, $nodes[],
                                   #added, #error_code);
              #new_added |= #added;
          }
```

```
      } // if $node != null.
   } // if $node ISA.

}  // For #i.


} // AddToGraphFromLayerHostDiscoveredResources.
```

49.6.1.4 Common Initiator Port

```
// Common Initiator layer piece of the Logical Device Composition Recipe

// It uses one of the initiator port subprofiles (eg. FibreChannel or iSCSI).

sub AddToGraphFromCommonInitiator(IN/OUT CIM_LogicalElement $nodes[],
                                  IN/OUT CIM_LogicalElement $links[],
                                  OUT boolean #new_added,
                                  OUT int     #error_code) {
boolean #added = false;
#new_added = false;

// The Goal is to start with SCSIProtocolEndpoints and add
// the associated port objects.

for #i in $nodes[] {

   // CIM_LogicalPort $ports[];
   // CIM_DeviceSAPImplementation $sap_associations[];

   if ($nodes[#i] ISA CIM_SCSIProtocolEndpoint) {

      &GetProviderInstanceOf($nodes[#i], $node, #error_code);
      if (#error_code) { return; }

      if ($node != null) {

      // Follow the DeviceSAPImplementation assocation
      // to the LogicalPort object
      $ports[] = Associators($node.getObjectPath(),
                             “CIM_DeviceSAPImplementation”,
                             “CIM_LogicalPort”,
                             “Dependent”,
                             “Antecedent”
                             );
```

```
      $sap_associations[] = References($node.getObjectPath(),
                                       “CIM_LogicalPort”,
                                       “Dependent”
                                      );

      // Add the port objects and associations to the graph.
      for #j in $ports[] {
         if ((null != $sap_associations[#j]) &&
             (null != $ports[#j])
            ) {
            &AddLinkIfNotAlreadyAdded($sap_associations[#j], $links[],
                                      #added, #error_code);
            #new_added |= #added;
            &AddIfNotAlreadyAdded($ports[#j], $nodes[], #added, #error_code);
            #new_added |= #added;
         }
      } for #j.

      } // if $node != null.

   } // if $nodes[#i] ISA.

} // AddToGraphFromLayerCommonInitiator.
```

49.6.1.5 Fabric Layer

```
Fibre Channel Fabric layer piece of the Logical Disk Composition Recipe

It uses the Fabric profile.

Sub AddToGraphFromLayerFabric($nodes, $links, #error_code){

// This function does the following
//
// 1. Identifies all the Switches and adds their objects paths and the object
// paths of the FC Ports belonging to these Switches to the $nodes array
//
// 2. Creates a suitable Association instance (e.g. a SystemDevice Association
// instance between a Switch and a FC Port), setting its GroupComponent and
// PartComponent. Adds the object path of the Association to the $links array
//
// 3. Creates a map of all connected FC Ports (i.e., belonging to Switches
// that are ISL’d together and to Host HBAs and Storage System Front End
// Controllers)
//
// In this map, the FC Ports (i.e., the ones that are connected) are
// cross-connected.
//
```

```
// e.g., For a pair of FC Ports, one belonging to a Switch and the other
// belonging to a Host (HBA), the map indexed by the Switch Port WWN returns
// the Host (HBA) FC Port object path and the map indexed by the Host (HBA) FC //
                          Port WWN returns the Switch FC Port object path.
//
// The Object stored in this Map is a composite of five objects and four
// associations. They are Switch, Switch FC Port, Switch end Protocol End Point,
// Attached Protocol End Point and Attached FC Port. The Associations are
// System Device, Device SAPImplementation, ActiveConnection, The attached side
// DeviceSAPImplementation.
// This information is kept in the Map. While traversing the Host-HBA part of
// the topology, the HBA FC Ports are matched in this Map to find out if there
// is a corresponding Switch side FC Port. If yes, only then all the objects
// are that lie on that path are saved in the Nodes Array and the corresponding
// Associations that lie on the path are stored in the Links Array.
//
// Similar relationship exists between the pairs of FC Ports where one belongs //
                          to a Switch and the other belonging belongs to a Storage
                          System Front End
// Controller and for FC Ports each of which belongs to a Switch.
//
// 4. Identifies all the Hosts and adds their objects paths to the $nodes array.
// Note that the object paths of the FC Ports (HBA Ports) belonging to these
// Hosts are already added to the $nodes array in step-3.
//
// 5. Creates a suitable Association instance (e.g. a SystemDevice Association
// instance between a Host and a FC Port), setting its GroupComponent and
// PartComponent. Adds the object path of the Association to the $links array.
//
// 6. Identifies all the Storage Systems and adds their objects paths to the
// $nodes array.
// Note that the object paths of the FC Ports (i.e., Front End Controller FC
// Ports) belonging to these Storage Systems are already added to the $nodes
// array in step-3.
//
// 7. Creates a suitable Association instance (e.g. a SystemDevice Association
// instance between a Storage System and a FC Port), setting its GroupComponent //
                          and PartComponent. Adds the object path of the
                          Association to the $links
// array.
//
// First find all the switches in a SAN. Get all the FC Ports for each
// switch and get the Attached FC Ports for each Switch FC Port. Save these
// device FC ports in the map described above.

// PREEXISTING CONDITIONS AND ASSUMPTIONS
// 1. All agents/namespaces supporting Fabric Profile previously identified
// using SLP. Do this for each CIMOM supporting Fabric Profile
```

```
// A composite elementsOnPath object is created. This object will be populated
// as we go along and will be stored in elementsOnPathMap with the index
// of attached FC Port WWN

ElementsOnPath #elementsOnPath = new ElementsOnPath();
ElementsOnPathMap #elementsOnPathMap = new ElementsOnPathMap();

switches[] = enumerateInstances(“CIM_ComputerSystem”, true, false, true,
true, null)
for #i in $switches[]
{
if (!contains(5, $switches[#i].Dedicated))
continue // only process switches, not other computer systems

// Add the switch to the elementsOnPath object

#elementsOnPath.switch = $switches[#i];

// Get all the SystemDevice associations between this switch and its FC Ports

$sysDevAssoc[] = ReferenceNames($switches[#i],
                                “CIM_FCPort”,
                                “GroupComponent”);

// Add the system device associations to the links array

for #a in $sysDevAssoc-[]
$links.addIfNotAlreadyAdded ($sysDevAssoc[#a];

$fcPorts->[] = AssociatorNames(
$switches[#i].getObjectPath(),
“CIM_SystemDevice”,
“CIM_FCPort”,
“GroupComponent”,
“PartComponent”)
for #j in $fcPorts->[]
{
// Add the FC Port to the elementsOnPathObject

#elementsOnPath.swFCPort = fcPorts->[#j];

$protocolEndpoints->[] = AssociatorNames(
fcPorts->[#j],
“CIM_DeviceSAPImplementation”,
“CIM_ProtocolEndpoint”,
“Antecedent”,
“Dependent”);
```

```
// NOTE - It is possible for this collection to be empty (i.e., ports that are not
// connected). It is NOT possible for this collection to have more than one
// element

if ($protocolEndpoints->[].length == 0)
continue

// Add the Protocol End Point to the elementsOnPathObject

#elementsOnPath.prorEP = protocolEndpoints[0];

// Add the associations between the fcPort and the Protocol end point to the
// links array

$devSAPImplassoc[]  = ReferenceNames($fcPorts->[#j],
                                     “CIM_ProtocolEndpoint”,
                                     “Antecedent”);
for #a in $devSAPImplassoc->[]
$links.addIfNotAlreadyAdded ($devSAPImplassoc->[#a];

$attachedProtocolEndpoints->[] = AssociatorNames(
$protocolEndpoints->[0],
“CIM_ActiveConnection”,
“CIM_ProtocolEndpoint”,
 null, null)

//Add the AttachedProtocolEndPoint to the elementsOnPath object

elementsOnPath.attachedPEP = attachedProtocolEndpoints->[0];

// Get the associations between the Protocol end point and the Attached
// protocol endpoint

$actConnassoc[]  = ReferenceNames($protocolEndpoint->[#0],
                                  “CIM_ActiveConnection”,
                                   “Antecedent”);

// Add it to the elementsOnPath object
elementsOnPath.actConn = actConnAssoc->[0];


// NOTE: role & resultRole are null as the direction of the association is not
// dictated by the specification

// $attachedFcPort is either a device FC port or an ISL’d switch FC port from
// another switch. We store this result is stored (i.e., which device
```

```
// FC Port is connected // to which switch FC Port) in a suitable data
// structure for subsequent correlation to ports discovered on devices.

for #k in $attachedProtocolEndpoints->[] {
$attachedFcPorts->[] = Associators(
$attachedProtocolEndpoints->[#k],
“CIM_DeviceSAPImplementation”,
“CIM_FCPort”,
“Dependent”,
“Antecedent”,
false,
false,
[“PermanentAddress”]);

$attachedFcPort = $attachedFcPorts[0] // Exactly one member guaranteed by model

// Add the attached FC Port to the elementsOnPath object

if $attachedFcPort != null
  #elementsOnPath.attFCPort = $attachedFcPort);

// Save the elementsOnPath object in elementsOnPath Map with the index of
// wwn of the attached fc port

elementsOnPathMap.put ($attachedfcPort.PermanentAddress, elementsOnPath);
}
}
}

 //   HBA to switch paths
// DESCRIPTION
// Determine physical path from HBA to switch.
//
// For each HBA FC port on every host, determine the connected switch
//FC port. NOTE: Not every HBA FC port will be connected to a switch FC port,
// and not every switch FC port will be connected to a device FC port. Only
// the connections between HBA FC ports and switch FC ports are discovered
// by this recipe
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. All agents/namespaces supporting HBA Profile previously identified
// using SLP
// 2. Array $attachedFcPorts->[] is a map of how elements in a SAN are
// connected together via Fibre-ChannelFC ports. Each index is a WWN of
// any device port on the SAN, and the value at that index is the
// connected switch FC port.
```

```
// Do this for each CIMOM supporting HBA Profile

$hosts[] = enumerateInstances(“CIM_ComputerSystem”)
for #i in $hosts->[]
{
if (!contains(0, $hosts[#i].Dedicated))
continue // only process systems that are “not dedicated”
$fcPorts[] = Associators(
$hosts[#i].getObjectPath(),
“CIM_SystemDevice”,
“CIM_FCPort”,
“GroupComponent”,
“PartComponent”,
false,
false,
[“PermanentAddress”])

// If the Host has FC Ports, add the Host to the $nodes array

if $fcPorts[] != null
$nodes.addIfNotAlreadyAdded ($hosts[#i]);

// Get all the SystemDevice associations between this host and its FC Ports

$sysDevAssoc[] = ReferenceNames($hosts[#i],
                                “CIM_FCPort”,
                                “GroupComponent”);

// Add these associations to the $links array

for #a in $sysDevAssoc-[]
$links.addIfNotAlreadyAdded ($sysDevAssoc[#a];

for #j in $fcPorts[]
{
// Get the FCPort WWN

#wwn = $fcPorts[#j].PermanentAddress

// Match this device port WWN to one (or less) switch FC ports, by using the
// mapping table built above

$elementsOnPath = elementsOnPathMap.get(#wwn);

// If a match is found, then add all the elements from the elementsOnPath
// object to nodes and links array.
```

```
// This will ensure that only those Switches and Switch FC Ports etc that are on a
                          path will be entered in the nodes and links array

if elementsOnPath != null
{
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getSwitch());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getswFCPort());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getPEP());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.geAttPEP());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getAttFCPort());

   $links.addIfNotAlreadyAdded (elementsOnPath.getDevSAPImpl());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getActConn());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getAttDevSAPImpl());
}

}
//    Determine physical path from Switch to Storage Arrays
// DESCRIPTION
// Determine physical path from Storage Arrays to Switches
//
// For each fibre-channelFC port on every array, determine the connected
// switch FC port. NOTE: This identifies the FrontEnd I/O Controllers
// (and Storage Arrays) whose FC ports are physically connected to
// some of the FC ports of some of the Switches. This recipe does not
// distinguish and does not filter the front-end FC Port from the
// back-end FC Ports.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. All agents/namespaces conforming to the Array profile previously
// identified
// 2. Array $attachedFcPorts[] is a map of how elements in a SAN are
// connected together via Fibre-ChannelFC ports. Each index is a WWN of
// any device FC port on the SAN, and the value at that index is the
// connected switch FC port.

// Do this for each CIMOM supporting the Array Profile:
// First identify upper-level computer systems for storage arrays -
// see 7.3.7.9.4 for how to use the Server profile to do this,
// or (as here) enumerate all systems within a conforming namespace
$computerSystems[] = enumerateInstances(“CIM_ComputerSystem”);
#n = 0
for #i in $computerSystems[]
{
if (!contains(3, $computerSystems[#i].Dedicated))
continue // only process systems that are dedicated “storage”
if (!contains(15, $computerSystems[#i].Dedicated))
continue // only process systems that are dedicated “block server”
```

```
$storageSystems[#n++] = $computerSystems[#i]
}
// Now accumulate all subsidiary computerSystems (cluster members or
// storage controllers) - treat $storageSystems[] as a queue and stuff
// newly discovered subsidiaries onto the end, so that ComponentCS
// associations are followed to arbitrary depth
#i = 0
while (#i < #n)
{
$subsidiaries[] = Associators(
$storageSystems[#i].getObjectPath(),
“CIM_ComponentCS”,
“CIM_ComputerSystem”,
“GroupComponent”,
“PartComponent”,
false,
false,
null)
for #j in $subsidiaries[]
{
$storageSystems[#n++] = $subsidiaries[#j]
}
#i++;
}
// Now get scoped FC ports for all the systems that have been accumulated
// NOTE: Some of the FC ports contained will be back-end ports, but they will
// have no connectivity to switches, so we won’t distinguish them
// from unconnected front-end FC ports

for #i in $storageSystems[]
{
$fcPorts[] = Associators(
$storageSystems[#i].getObjectPath(),
“CIM_SystemDevice”,
“CIM_FCPort”,
“GroupComponent”,
“PartComponent”,
false,
false,
[“PermanentAddress”])
for #j in $fcPorts[]
{
// Get the FCPort WWN
#wwn = $fcPorts[#j].PermanentAddress

// If the Storage System has FC Ports, add the storage system to the $nodes array
```

```
if $fcPorts[] != null
$nodes.addIfNotAlreadyAdded ($storageSystems[#i]);

// Get all the SystemDevice associations between this host and its FC Ports

$sysDevAssoc[] = ReferenceNames($storageSystems[#i],
                                “CIM_FCPort”,
                                “GroupComponent”);

// Add these associations to the $links array

for #a in $sysDevAssoc-[]
$links.addIfNotAlreadyAdded ($sysDevAssoc[#a];

for #j in $fcPorts[]
{
// Get the FCPort WWN

#wwn = $fcPorts[#j].PermanentAddress

// Match this device port WWN to one (or less) switch FC ports, by using the
// mapping table built above

$elementsOnPath = elementsOnPathMap.get(#wwn);

// If a match is found, then add all the elements from the elementsOnPath
// object to nodes and links array.

// This will ensure that only those Switches and Switch FC Ports etc that are on a
                           path will be entered in the nodes and links array

if elementsOnPath != null
{
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getSwitch());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getswFCPort());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getPEP());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.geAttPEP());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getAttFCPort());

   $links.addIfNotAlreadyAdded (elementsOnPath.getDevSAPImpl());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getActConn());
   $nodes.addIfNotAlreadyAdded (elementsOnPath.getAttDevSAPImpl());
}

}
}
}
```

49.6.1.6 IP Network Layer

```
// IP Network piece of the Logical Device Composition Recipe

// It uses the iSCSI Target Ports Subprofile.

// This subroutine tries to account for the logical topology
// of the IP network between an iSCSI Initiator and Target
// by adding an object representing the iSCSISession (NetworkPipe)
// between them.

sub AddToGraphFromLayerIPNetwork(IN/OUT CIM_LogicalElement $nodes[],
                                 IN/OUT CIM_Dependency $links[],
                                 OUT boolean #new_added,
                                 OUT int     #error_code) {

// CIM_EnpointOfNetworkPipe    $found_endpoints_of_pipe[];
// CIM_iSCSISession            $found_sessions[];
// CIM_EthernetPort            $found_ports[];
// CIM_DeviceSAPImplementation $found_sap_associations[];

boolean #added;

for #i in $nodes[] {

    if ($nodes[#i] instanceof iSCSIProtocolEndpoint) {

       // Find the iSCSIProtocolEndpoints left for us by the iSCSI
       // Initiator Port subprofile. These are correlated by Name-NameFormat.
       &GetProviderInstanceOf($nodes[#i], $node, #error_code);

       if ($node != null) {

       // Using the EndpointOfNetworkPipe, follow the association
       // to an iSCSISession.  This represents the topology contribution
       // if the IP Network.

         $found_sessions[] = Associators(
                                $node.getObjectPath(),
                                “CIM_EndpointOfNetworkPipe”,
                                “CIM_iSCSISession”,
                                “Antecedent”,
                                “Dependent”
                                      );
```

```
            $found_endpoints_of_pipe[] = References($node.getObjectPath(),
                                                    “CIM_iSCSISession”,
                                                    “Antecedent”
                                                   );

            &AddIfNotAlreadyAdded($found_sessions[0],
                                  $nodes[], #added, #error_code);
            #new_added |= #added;
            &AddLinkIfNotAlreadyAdded($found_endpoints_of_pipe[0],
                                      $links[], #added, #error_code);
            #new_added |= #added;

            // Also follow the DeviceSAPImplementation association
            // from the protocol endpoint to the EthernetPort for completeness.

            $found_ports[] = Associators($node.getObjectPath(),
                                 “CIM_DeviceSAPImplementation”,
                                 “CIM_EthernetPort”,
                                 “Dependent”,
                                 “Antecedent”
                                 );

            $found_sap_associations[] = References($node.getObjectPath(),
                                           “CIM_EthernetPort”,
                                           “Dependent”
                                          );

            // Add the ports and sap associations. There should only be one
            &AddIfNotAlreadyAdded($found_ports[0],
                                  $nodes[], #added, #error_code);
            #new_added |= #added;
            &AddLinkIfNotAlreadyAdded($found_sap_associations[0],
                                      $links[], #added, #error_code);
            #new_added |= #added;

          } // if $node != null.

        } // if $nodes[#i] instanceof.

    } // for #i.

    } // AddToGraphFromLayerIPNetwork.
```

49.6.1.7 Local Disk Layer

```
    // Local Disk layer piece of the Logical Device Composition Recipe
```

```
// It uses the Disk Subprofile.

sub AddToGraphFromLayerLocalDiskDrive(IN/OUT CIM_LogicalElement $nodes[],
                                      IN/OUT CIM_Dependency     $links[],
                                      OUT boolean               #new_added,
                                      OUT int                   #error_code) {

// Make sure we've recursively tracked down all the StorageExtents.

boolean           #added = false;
// CIM_StorageExtent $found_extents[];
// CIM_BasedOn       $found_associations[];

#new_added = false;

// Now see if there are any local disk drives making
// up those extents through the MediaPresent association.

// CIM_DiskDrive $disk_media[];
// CIM_MediaPresent $mediapresent_associations[];

for #i in $nodes[] {

   if ($nodes[#i] ISA CIM_StorageExtent) {

      &GetProviderInstanceOf($nodes[#i], $node, #error_code);
      if (#error_code) { return;}

      if ($node != null) {

         $disk_media[] = Associators($node.getObjectPath(),
                                     "CIM_MediaPresent",
                                     "CIM_DiskDrive",
                                     "Dependent",
                                     "Antecedent"
                                    );

         $mediapresent_associations[] = References($node.getObjectPath(),
                                                   "CIM_DiskDrive",
                                                   "Dependent"
                                                  );
      }

      // There should be only one asociation found for each extent.
      if (0 != $disk_media.length) {
         &AddIfNotAlreadyAdded($disk_media[0], $nodes[], #added, #error_code);
```

```
         #new_added |= #added;
         &AddLinkIfNotAlreadyAdded($mediapresent_associations[0], $links[],
                                   #added, #error_code);
         #new_added |= #added;
       }

       } if $node != null.
    } if $node ISA .

} // for.


} // AddToGraphFromLayerLocalDiskDrive.
```

49.6.1.8 Logical Disk Layers

```
// Logical Disk Partitioning piece of the Logical Device Composition Recipe

// It uses the Disk Partition Subprofile.


// Given a CIM_GenericDiskPartition, recursively traverse the CIM_BasedOn
// associations finding other CIM_GenericDiskPartitions on which
// this partition is based
// and adding the partitions and associations to the found_partitions
// and found_partition_associations as you go.  Follow CIM_BasedOn associations
// to the underlying CIM_StorageExtents.

sub RecursivelyAddPartitions(
                  IN CIM_GenericDiskPartition $found_partition,
                  IN/OUT CIM_GenericDiskPartition[] $found_partitions[],
                  IN/OUT CIM_BasedOn[] $found_partition_associations[],
                  OUT boolean #new_added
                            );

sub AddToGraphFromLayerDiskPartitioning(IN/OUT CIM_LogicalElement $nodes[],
                                        IN/OUT CIM_LogicalElement $links[],
                                        OUT #new_added,
                                        OUT #error_code) {

// CIM_GenericDiskPartition        $found_partitions[];
// CIM_LogicalDiskBasedOnPartition $found_partition_associations[];
```

```
// CIM_StorageExtent              $found_extents[];
// CIM_BasedOn                    $found_extent_associations[];

boolean $added = false;

#new_added = false;

for #j in $nodes[] {

    // In the Disk Partitioning Profile
    // start with a LogicalDisk object, as it is defined
    // as that on which storage applications (volume managers or
    // filesystems) may be placed.
    // The LogicalDisk object has DeviceID and Name attributes
    // that should be set to OS device names like
    // (/dev/sda1 on Linux or C: on Windows)

    if ($nodes[#j] ISA CIM_LogicalDisk) {

    &GetProviderInstanceOf($nodes[#j], $node, #error_code);
    if (#error_code) { return; }

    if ($node != null) {
       // One would then follow the LogicalDiskBasedOn Partition
       // association to a GenericDiskPartition object.


      $found_partitions[] = Associators($node.getObjectPath(),
                                        “CIM_LogicalDiskBasedOnPartition”,
                                        “CIM_GenericDiskPartition”,
                                        “Dependent”,
                                        “Antecedent”
                                       );

      $found_partition_associations[] = References($node.getObjectPath(),
                                                   “CIM_GenericDiskPartition”,
                                                   “Dependent”
                                                  );

      // To found paritions, add all recursive BasedOn associations to
      // and their partitions.
      for (#i=0; #i<$found_partitions[].length; #i++) {
         &RecursivelyAddPartitions($found_partitions[#i],
                                   $found_partitions[],
                                   $found_partition_associations[]);
      }
```

```
 // Now add all parititons and associations found so far.
 for (#i=0; #i<$found_partitions[].length; #i++) {
     &AddIfNotAlreadyAdded($found_partitions[#i], $nodes[],
                           #added, #error_code);
     #new_added |= #added;
     &AddLinkIfNotAlreadyAdded($found_partition_associations[#i],
                               $links[], #added, #error_code);
     #new_added |= #added;
 }

// Now follow the BasedOn associations from partitions
// to extents.

for #k in $found_partitions[] {

   // look for a BasedOn association that
   // leads to a StorageExtent.

   $found_extents[] = Associators($found_partitions[#k].getObjectPath(),
                                  “CIM_BasedOn”,
                                  “CIM_StorageExtent”,
                                  “Dependent”,
                                  “Antecedent”
                                 );

   $found_extent_associations[] = References($node.getObjectPath(),
                                             “CIM_StorageExtent”,
                                             “Dependent”
                                            );

   if (  ($found_extents[0] != null) &&
         ($found_extent_associations[0] != null) &&
      ) {
       &AddLinkIfNotAlreadyAdded($found_extent_associations[0], $links[],
                                 #added, #error_code);
       #new_added |= #added;
       &AddIfNotAlreadyAdded($found_extents[0], $nodes[],
                             #added, #error_code);
       #new_added |= #added;
   }

}  // For over partitions.

// The DeviceID field of those StorageExents that are
// StorageVolumes should be correlatable
// to a StorageVolume object maintained by the Array profile.
// (see Host Discovered Resources profile).
```

```
    } // if $null != $node.

    } // if $node ISA.

}  // For over nodes.

} // AddToGraphFromLayerDiskPartitioning.



sub RecursivelyAddPartitions(
                      IN CIM_GenericDiskPartition $found_partition,
                      IN/OUT CIM_GenericDiskPartition[] $found_partitions[],
                      IN/OUT CIM_BasedOn[] $found_partition_associations[],
                      OUT boolean #new_added
                           ){

      // CIM_GenericDiskPartition $new_found_partitions[];
      // CIM_BasedOn $new_found_associations;

      $new_found_partitions[] = Associators($found_partition.getObjectPath(),
                                       “CIM_BasedOn”,
                                       “CIM_GenericDiskPartition”,
                                       “Dependent”,
                                       “Antecedent”
                                      );

      $new_found_associations[] = References($node.getObjectPath(),
                                          “CIM_GenericDiskPartition”,
                                          “Dependent”
                                         );

      for #i in $new_found_associations[] {
          $found_partition_associations[].add($new_found_associations[#i]);
          #new_added = true;
      }

      for #i in $new_found_partitions[] {
          $found_partitions[].add($new_found_partitions[#i]);
          &RecursivelyAddPartitions($new_found_partitions[#i],
                                    $found_partitions[],
                                    $found_partition_associations[]);
          #new_added = true;
      }
```

```
}  // RecursivelyAddPartitions.
```

49.6.1.9 Multipath Layer

```
// Multipath layer piece of the Logical Device Composition Recipe

// It uses the SCSI Multipath Management Subprofile

sub AddToGraphFromLayerMultipath(IN/OUT CIM_LogicalElement $nodes[],
                                 IN/OUT CIM_Dependency $links[],
                                 OUT boolean #new_added,
                                 OUT int    #error_code) {
boolean #added = false;
#new_added = false;

for #j in $nodes[] {

   // CIM_SCSIInitiatorTargetLogicalUnitPath $scsi_paths[];
   // CIM_SCSIProtocolEndpoint $initiator_endpoint;
   // CIM_SCSIProtocolEndpoint $target_endpoint;

   #i = 0;
   if ($nodes[#j] ISA CIM_LogicalDisk) {

          &GetProviderInstanceOf($nodes[#j], $node, #error_code);
          if (#error_code) { return; }

          if ($node != null) {

          // Find all CIM_SCSIInitiatorTargetLogicalUnitPath
          // with $node as the LogicalUnit reference.
          $scsi_paths[] = References($node.getObjectPath(),
                                     “CIM_SCSIProtocolEndpoint”,  // ResultClass
                                     “LogicalUnit”                // Role
                                    );

          for (#i=0; #i<$scsi_paths.length; #i++) {
              &AddLinkIfNotAlreadyAdded($scsi_paths[#i], $links[],
                                        #added, #error_code);
              #new_added |= #added;
              $initiator_endpoint = $scsi_paths[#i].Initiator;
              $target_endpoint = $scsi_paths[#i].Target;
              &AddIfNotAlreadyAdded($initiator_endpoint, $nodes[],
                                    #added, #error_code);
```

```
                #new_added |= #added;
                &AddIfNotAlreadyAdded($target_endpoint, $nodes[],
                                      #added, #error_code);
                #new_added |= #added;
            }    // for.

            } // if $node != null.

        } // if $node ISA.

    }  // For over nodes.

    } // AddToGraphFromLayerMultipath.
```

49.6.1.10 Virtualizer Layer

```
    // Virtualizer layer piece of the Logical Device Composition Recipe

    // It is based on the Storage Virtualizer Profile,
    // which includes the Target Port Subprofile,
    // the Block Services Package, and the
    // Initiator Port Subprofile.  It stitches StorageVolumes
    // it finds up to their ingress ports, across the layers
    // of virtualization, and out their egress ports.

    // For simplicity, this subroutine assumes there is no multipathing
    // of LogicalDevices across multiple ingress ports.

    // Given a CIM_StoragePool, recursively traverse the
    // CIM_AllocatedFromStoragePool
    // associations finding other CIM_StoragePools on which this pool is based
    // and adding the pools and associations to the found_pools
    // and found_allocated_associations as you go.  This method is implemented
    // in Volume Manager Layer subroutine of this recipe.
    sub RecursivelyAddPools(
              IN CIM_StoragePool$found_pool,
              IN/OUT CIM_StoragePool $found_pools[],
              IN/OUT CIM_AllocatedFromStoragePools $found_allocated_associations[],
              OUT boolean #new_added
                          );

    sub AddToGraphFromLayerStorageVirtualizer(IN/OUT CIM_LogicalElement $nodes[],
                                              IN/OUT CIM_Dependency $links[],
                                              OUT    boolean #new_added,
                                              int    #error_code) {
```

```
// CIM_SCSIProtocolController      $found_protocol_controllers[];
// CIM_ProtocolControllerForUnit   $found_for_unit_associations[];
// CIM_ProtocolEndpoint            $found_protocol_endpoints[];
// CIM_DeviceSAPImplementation     $found_sap_associations[];
// CIM_LogicalPort                 $found_ports[];
// CIM_SAPAvailableForElement      $found_available_associations[];
// CIM_StorageVolume               $found_storage_volumes[];
// CIM_StoragePool                 $found_storage_pools[];
// CIM_AllocatedFromStoragePool    $found_allocated_associations[];
// CIM_StorageExtent               $found_component_disks[];
// CIM_ConcreteComponent           $found_component_associations[];

boolean #added = false;

#new_added = false;

for #i in $nodes[] {

   if ($nodes[#i] ISA CIM_LogicalDevice) {

    &GetProviderInstanceOf($nodes[#i], $node, #error_code);
    if (#error_code) { return;}

    if ($node != null) {

      // First, work up the path to include the network port
      // for stitching in the network topology.

      // Follow an ProtocolControllerForUnit to a SCSIProtocolController.
      $found_protocol_controllers[] = Associators(
                              $node.getObjectPath(),
                              “CIM_SCSIProtocolControllerForUnit”,
                              “CIM_SCSIProtocolController”,
                              “Dependent”,
                              “Antecedent”
                                    );

      $found_for_unit_associations[] = References(
                              $node.getObjectPath(),
                              “CIM_SCSIProtocolController”,
                              “Dependent”
                                              );

      &AddIfNotAlreadyAdded($found_protocol_controllers[0],
                            $nodes[], #added, #error_code);
      #new_added |= #added;
```

```
&AddLinkIfNotAlreadyAdded($found_for_unit_associations[0],
                          $links[], #added, #error_code);
#new_added |= #added;

// Follow an SAPAvailableForElement to a SCSIProtocolEndpoint.
$found_protocol_endpoints[] = Associators(
                        $found_protocol_controllers[0].getObjectPath(),
                        “CIM_SAPAvailableForElement”,
                        “CIM_ProtocolEndpoint”,
                        “ManagedElement”,
                        “AvailableSAP”
                                        );

$found_available_associations[] = References(
                        $found_protocol_controllers[].getObjectPath(),
                        “CIM_ProtocolEndpoint”,
                        “ManagedElement”
                                          );

&AddIfNotAlreadyAdded($found_protocol_endpoints[0],
                      $nodes[], #added, #error_code);
#new_added |= #added;
&AddLinkIfNotAlreadyAdded($found_available_associations[0],
                          $links[], #added, #error_code);
#new_added |= #added;

// Follow the DeviceSAPImplementation to a LogicalPort.
$found_ports[] = Associators(
                      $found_protocol_endpoints[0].getObjectPath(),
                      “CIM_DeviceSAPImplementation”,
                      “CIM_LogicalPort”,
                      “Dependent”,
                      “Antecedent”
                            );

$found_sap_associations[] = References(
                    $found_protocol_endpoints.getObjectPath(),
                    “CIM_LogicalPort”,
                    “Dependent”
                                       );
&AddIfNotAlreadyAdded($found_ports[0],
                      $nodes[], #added, #error_code);
#new_added |= #added;
&AddLinkIfNotAlreadyAdded($found_sap_associations[0],
                          $links[], #added, #error_code);
#new_added |= #added;
```

```
// Now, starting from our StorageVolume node, work down the path
// through the virtualization layer and out the other side
// to the LogicalPorts.

// Follow the AllocatedFromStoragePool to a StoragePool.
$found_storage_pools[] = Associators(
                         $node.getObjectPath(),
                         “CIM_AllocatedFromStoragePool”,
                         “CIM_StoragePool”,
                         “Dependent”,
                         “Antecedent”
                               );

$found_allocated_associations[] = References($node.getObjectPath(),
                                             “CIM_StoragePool”,
                                             “Dependent”
                                            );

// Recursively add other StoragePools by following additional
// AllocatedFromStoragePool associations.
for #j in $found_storage_pools[] {
       &RecursivelyAddPools($found_storage_pools[#j],
                            $found_storage_pools[],
                            $found_allocated_associations[],
                            #added
                           );
       #new_added |= #added;
} // for #j.

for #j in $found_storage_pools[] {

    // Add the pools and allocated associations.
    &AddIfNotAlreadyAdded($found_storage_pools[#j],
                          $nodes[], #added, #error_code);
    #new_added |= #added;
    &AddLinkIfNotAlreadyAdded($found_allocated_associations[#j],
                              $links[], #added, #error_code);
    #new_added |= #added;

    // Follow the ConcreteComponent associations to StorageExtents.
    $found_component_disks[] = Associators(
                       $found_storage_pools[#j].getObjectPath(),
                       “CIM_ConcreteComponent”,
                       “CIM_StorageExtent”,
                       “Dependent”,
                       “Antecedent”
                                       );
```

```
$found_component_associations[] = References(
                  $found_storage_pools[#j].getObjectPath(),
                  “CIM_StorageExtent”,
                  “Dependent”
                                          );

// Now, work down each component_disk using the
// Initiator Port Subprofile.

for #k in $found_component_disks[] {

   // Add the disks and component associations.
   &AddIfNotAlreadyAdded($found_component_disks[#k],
                         $nodes[], #added, #error_code);
   #new_added |= #added;
   &AddLinkIfNotAlreadyAdded($found_component_associations[#k],
                             $links[], #added, #error_code);
   #new_added |= #added;

   // Find all CIM_SCSIInitiatorTargetLogicalUnitPath
   // with $found_component_disks[#k] as the LogicalUnit.
   $scsi_paths[] = References(
                        $found_component_disks[#k].getObjectPath(),
                        “CIM_SCSIProtocolEndpoint”,  // ResultClass
                        “LogicalUnit”                // Role
                       );

   // Backward compatibility note: SMI-S 1.0 used an
   // SAPAvailableForElement association to get the the
   // SCSIProtocolEndpoint here.  This recipe has been written
   // to the SMI-S 1.1 model, which uses the trinary association
   // SCSIInitiatorTargetLogicalUnitPath.

   for (#ii=0; #ii<$scsi_paths.length; #ii++) {
       &AddLinkIfNotAlreadyAdded($scsi_paths[#ii],
                                 $links[], #added, #error_code);
       #new_added |= #added;
       $initiator_endpoint = $scsi_paths[#ii].Initiator;
       $target_endpoint = $scsi_paths[#ii].Target;
       &AddIfNotAlreadyAdded($initiator_endpoint,
                             $nodes[], #added, #error_code);
       #new_added |= #added;
       &AddIfNotAlreadyAdded($target_endpoint,
                             $nodes[], #added, #error_code);
       #new_added |= #added;

       // Follow the DeviceSAPImplementation assocation
```

```
                    // to the LogicalPort object.
                    $found_ports[] = Associators(
                                $initiator_endpoint.getObjectPath(),
                                “CIM_DeviceSAPImplementation”,
                                “CIM_LogicalPort”,
                                “Dependent”,
                                “Antecedent”
                                            );

                    $found_sap_associations[] = References(
                                $initiator_endpoints.getObjectPath(),
                                “CIM_LogicalPort”,
                                “Dependent”
                                                    );

                    // Add the ports and sap associations.
                    &AddIfNotAlreadyAdded($found_ports[0],
                                        $nodes[], #added, #error_code);
                    #new_added |= #added;
                    &AddLinkIfNotAlreadyAdded($found_sap_associations[0],
                                            $links[], #added, #error_code);
                    #new_added |= #added;

                }    // for #ii.

            } // for #k.

        } // for #j.
      } // if $node != null.
    } // if $node ISA.
   } // for #i.

} // AddToGraphFromLayerStorageVirtualizer.
```

49.6.1.11 Volume Manager Layer

```
// Volume Manager layer piece of the Logical Device Composition Recipe

// It uses the Volume Management Profile.


// Given a CIM_StoragePool, recursively traverse the
// CIM_AllocatedFromStoragePool
// associations finding other CIM_StoragePools on which this pool is based
// and adding the pools and associations to the found_pools
```

```
// and found_allocated_associations as you go.

sub RecursivelyAddPools(
           IN CIM_StoragePool$found_pool,
           IN/OUT CIM_StoragePool $found_pools[],
           IN/OUT CIM_AllocatedFromStoragePools $found_allocated_associations[],
           OUT boolean #new_added
                       );

// Given a CIM_LogicalDisk, recursively traverse the CIM_BasedOn
// associations finding other CIM_LogicalDisks on which this
// LogicalDisk is based
// and adding the disks and associations to the found_disks
// and found_basedon_associations as you go.

sub RecursivelyAddDisks(
           IN CIM_LogicalDisk $found_disk,
           IN/OUT CIM_LogicalDisk $found_disks[],
           IN/OUT CIM_AllocatedFromStoragePools $found_basedon_associations[],
           OUT boolean #new_added
                       );

// We want the CIM_StoragePools to be part of the
// composition topology if they exist.

sub AddToGraphFromLayerVolumeManager(IN/OUT CIM_LogicalElement $nodes[],
                                     IN/OUT CIM_Dependency     $links[],
                                     OUT    boolean #new_added,
                                     OUT    int #error_code) {

#added = false;

for #j in $nodes[] {

   // CIM_StoragePool $found_storage_pools[];
   // CIM_AllocatedFromStoragePool $found_allocated_associations[];

   if ($nodes[#j] ISA CIM_LogicalDisk) {

   &GetProviderInstanceOf($nodes[#j], $node, #error_code);
   if (#error_code) { return;}

   if (($node != null)) {

      // This first method looks for cases where volume groups
      // have been created as StoragePools.
```

```
// Follow the CIM_AllocatedFromStoragePool association
// to a CIM_StoragePool.
$found_storage_pools[] = Associators($node.getObjectPath(),
                                     “CIM_AllocatedFromStoragePool”,
                                     “CIM_StoragePool”,
                                     “Dependent”,
                                     “Antecedent”
                                    );

$found_allocated_associations[] = References($node.getObjectPath(),
                                             “CIM_StoragePool”,
                                             “Dependent”
                                            );

// Then, recursively follow any CIM_AllocatedFromStoragePool
// associations to other CIM_StoragePools, adding associations
// and strorage pools as you go.
for (#i=0; #i<$found_storage_pools[].length, #i++) {
    &RecursivelyAddPools( $found_storage_pools[#i],
                          $found_storage_pools[],
                          $found_allocated_associations[],
                          #added
                         );
    #new_added |= #added;
}

for #k in $found_allocated_associations[] {
    &AddLinkIfNotAlreadyAdded($found_allocated_association[#k], $links[],
                              #added, #error_code);
    #new_added |= #added;
}
for #k in $found_storage_pools[] {
    &AddIfNotAlreadyAdded($found_pool_storage_pools[#k],
                          $nodes[], #added, #error_code);
    #new_added |= #added;
}

// Now find the component disks of the storage pools.
// CIM_LogicalDisk[] $found_component_disks[];
for #k in $found_storage_pools[] {
   $found_component_disks[] = Associators(
                              $found_storage_pools[#k].getObjectPath(),
                              “CIM_ConcreteComponent”,
                              “CIM_CIMLogicalDisk”,
                              “Dependent”,
                              “Antecedent”
                                        );
```

```
   $found_component_associations[] = References(
                           $found_storage_pools[#k].getObjectPath(),
                           “CIM_LogicalDisk”,
                           “Dependent”
                                                );


}

for (#i=0; i < $found_component_disks[].length; #i++) {
      &AddLinkIfNotAlreadyAdded($found_component_associations[#i],
                                $links[],
                                #added,
                                #error_code);
      #new_added |= #added;
}

// If this implementation does not use volume groups,
// look for the BasedOn associations to find the disks.

// CIM_LogicalDisk[] $found_logical_disks[];
// CIM_BasedOn[] $found_basedon_associations[];

$found_logical_disks[] = Associators($node.getObjectPath(),
                                     “CIM_BasedOn”,
                                     “CIM_LogicalDisk”,
                                     “Dependent”,
                                     “Antecedent”
                                      );

$found_basedon_associations[] = References($node.getObjectPath(),
                                           “CIM_LogicalDisk”,
                                           “Dependent”
                                          );

// Add these disks to the component_disks.
for (#i=0; #i<$found_basedon_associations[].length; #i++) {
   &AddLinkIfNotAlreadyAdded($found_basedon_associations[#i], $links[].
                             #added, #error_code);
   #new_added |= #added;
   &AddIfNotAlreadyAdded($found_logical_disks[#i],
                         $found_component_disks[],
                         #added, #error_code);
   #new_added |= #added;

}
```

```
      // Follow all BasedOn associations to find more component disks
      // recursively.
      // CIM_LogicalDisk $recusive_disks[];
      // CIM_BasedOn     $recursive_basedon_associations[];
      for (#i=0; #i<$found_component_disks[].length; #i++) {
         &RecursivelyAddDisks($found_component_disks[#i],
                              $recursive_disks[],
                              $recursive_basedon_associations[],
                              #added);
      }

      // Now add the recursive disks and associations to the
      // $nodes[] and $links[] arrays.
      for (#i=0; #i<$recursive_disks[].length; #i++) {
         &AddLinkIfNotAlreadyAdded($recursive_basedon_associations[#i],
                                   $links[], #added, #error_code);
         #new_added |= #added;
         &AddIfNotAlreadyAdded($recursive_disks[#i],
                               $nodes[],
                               #added, #error_code);
         #new_added |= #added;
      }


   } // if $node != null.

   } // if $node ISA.

}  // For over nodes.

} // AddToGraphFromLayerVolumeManager.


sub RecursivelyAddPools(
          IN CIM_StoragePool $found_pool,
          IN/OUT CIM_StoragePool $found_pools[],
          IN/OUT CIM_AllocatedFromStoragePools $found_allocated_associations[],
          OUT boolean #new_added
                      ) {

      // CIM_StoragePool $new_found_pools;
      // CIM_AllocatedFromStoragePool $new_found_associations;

      #new_added = false;

      $new_found_pools[] = Associators($found_pool.getObjectPath(),
                                       “CIM_AllocatedFromStoragePool”,
```

```
                                        “CIM_StoragePool”,
                                        “Dependent”,
                                        “Antecedent”
                                       );

      $new_found_associations[] = References($node.getObjectPath(),
                                             “CIM_StoragePool”,
                                             “Dependent”
                                            );
      for #i in $new_found_associations[] {
          $found_allocated_associations[].add($new_found_associations[#i]);
          #new_added = true;
      }
      for #i in $new_found_pools[] {
          $found_pools[].add($new_found_pools[#i]);
          &RecursivelyAddPools($new_found_pools[#i], $found_pools[],
                               $found_allocated_associations[], #new_added);
          #new_added = true;
      }


}  // RecursivelyAddPools.


sub RecursivelyAddDisks( IN CIM_LogicalDisk $found_disk,
                         IN/OUT CIM_LogicalDisk $found_disks[],
                         IN/OUT CIM_BasedOn $found_basedon_associations[],
                         OUT boolean #new_added
                       ) {

      // CIM_StoragePool $new_found_disks[];
      // CIM_AllocatedFromStoragePool $new_found_associations;

      #new_added = false;

      $new_found_disks[] = Associators($found_disk.getObjectPath(),
                                       “CIM_BasedOn”,
                                       “CIM_LogicalDisk”,
                                       “Dependent”,
                                       “Antecedent”
                                      );

      $new_found_associations[] = References($node.getObjectPath(),
                                             “CIM_LogicalDisk”,
                                             “Dependent”
                                            );
      for #i in $new_found_associations[] {
```

```
            $found_basedon_associations[].add($new_found_associations[#i]);
            #new_added = true;
        }
        for $new_found_disk in $new_found_disks[] {
            $found_disks[].add($new_found_disks[#i]);
            &RecursivelyAddDisks($new_found_disks[#i], $found_disks[],
                                 $found_basedon_associations[], #new_added);
            #new_added = true;
        }


}  // RecursivelyAddDisks.
```

49.6.1.12 Storage Library

```
// Storage Library layer piece of the Logical Device Composition Recipe

// This is based on the
// Storage Library Profile, which can include the Target Port Subprofile.
// It connects LogicalDevices left by the SCSI initiator
// side to TapeDrives and their LogicalPorts on the array side
// to allow network and logical device topologies to be correlated.

sub AddToGraphFromLayerStorageLibrary(IN/OUT CIM_LogicalElement $nodes[],
                                      IN/OUT CIM_Dependency $links[],
                                      OUT    boolean #new_added,
                                      OUT    int     #error_code) {

   // CIM_SCSIProtocolController      $found_protocol_controllers[];
   // CIM_ProtocolControllerForUnit   $found_for_unit_associations[];
   // CIM_ProtocolEndpoint            $found_protocol_endpoints[];
   // CIM_DeviceSAPImplementation     $found_sap_associations[];
   // CIM_LogicalPort                 $found_ports[];
   // CIM_SAPAvailableForElement      $found_available_associations[];

   boolean #added = false;

   #new_added = false;

   for #i in $nodes[] {

      if ($nodes[#i] ISA CIM_LogicalDevice) {

       // This should correlate by OtherIdentifyingInfo and should find the
       // corresponding CIM_TapeDrive object instance in this profile.
```

```
&GetProviderInstanceOf($nodes[#i], $node, #error_code);
if (#error_code) { return;}

if ($node != null) {

  // Work up the path to include the network ports
  // for stitching in the network topology.

  // Follow an ProtocolControllerForUnit to a SCSIProtocolController.
  $found_protocol_controllers[] = Associators(
                          $node.getObjectPath(),
                          “CIM_SCSIProtocolControllerForUnit”,
                          “CIM_SCSIProtocolController”,
                          “Dependent”,
                          “Antecedent”
                                );

  $found_for_unit_associations[] = References(
                                       $node.getObjectPath(),
                                       “CIM_SCSIProtocolController”,
                                       “Dependent”
                                             );

  // Each LogicalDevice may be handled by multiple controllers.
  for #j in $found_protocol_controllers[] {

     &AddIfNotAlreadyAdded($found_protocol_controllers[#j],
                          $nodes[], #added, #error_code);

     #new_added |= #added;
     &AddLinkIfNotAlreadyAdded($found_for_unit_associations[#j],
                              $links[], #added, #error_code);
     #new_added |= #added;

     // Follow an SAPAvailableForElement to a SCSIProtocolEndpoint.
     $found_protocol_endpoints[] = Associators(
                $found_protocol_controllers[#j].getObjectPath(),
                “CIM_SAPAvailableForElement”,
                “CIM_ProtocolEndpoint”,
                “ManagedElement”,
                “AvailableSAP”
                                       );

     $found_available_associations[] = References(
                $found_protocol_controllers[#j].getObjectPath(),
                “CIM_ProtocolEndpoint”,
                “ManagedElement”
```

```
                                                    );

            // Each controller may multipath through multiple ports.
            for #k in $found_protocol_endpoints[] {

               &AddIfNotAlreadyAdded($found_protocol_endpoints[#k],
                                     $nodes[], #added, #error_code);
               #new_added |= #added;
               &AddLinkIfNotAlreadyAdded($found_available_associations[#k],
                                         $links[], #added, #error_code);
               #new_added |= #added;

               // Follow the DeviceSAPImplementation to a LogicalPort.
               // This is a 1:1 relationship.
               $found_ports[] = Associators(
                                $found_protocol_endpoints[#k].getObjectPath(),
                                “CIM_DeviceSAPImplementation”,
                                “CIM_LogicalPort”,
                                “Dependent”,
                                “Antecedent”
                                           );

               $found_sap_associations[] = References(
                           $found_protocol_endpoints.getObjectPath(),
                           “CIM_LogicalPort”,
                           “Dependent”
                                                 );
               &AddIfNotAlreadyAdded($found_ports[0],
                                     $nodes[], #added, #error_code);
               #new_added |= #added;
               &AddLinkIfNotAlreadyAdded($found_sap_associations[0],
                                         $links[], #added, #error_code);
               #new_added |= #added;

            } // for #k.

         } // for #j.

      } // for #i.


   } // AddToGraphFromLayerStorageLibrary.
```

EXPERIMENTAL

50 Indications Profile

50.1 Synopsis

Profile Name: Indications (Component Profile)

Version: 1.6.0

Organization: SNIA

CIM Schema Version: 2.25.0

Related Profiles for Indications: Not defined in this standard.

Central Class: CIM_IndicationService

Scoping Class: CIM_System

Scoping algorithm: HostedService

The SNIA Indications profile is a specialization of the DMTF Indication Profile.

The scoping system is the system that hosts a WBEM server with one or more indication services.

NOTE The current version of the DMTF profile allows only one indication service per indication system; the limitation may be raised in a future version of the profile.

50.2 Description

The SNIA Indications profile is a specialization of the DMTF Indications profile. It adds in the SNIA elements and constraints. This specialization is for profiles that wish to conform to both the DMTF Indications and SNIA Indications profiles. The current SMI-S Indication and Experimental Indication profile are not being changed by the addition of the SNIA Indications profile.

The intent of the SNIA Indications Profile is to support profiles that want to conform to both the DMTF and SNIA Indications support. For example, a DMTF autonomous profile that wishes to reference SMI-S component profiles could reference the SNIA Indications profile to identify support for both the DMTF and SNIA Profiles for Indications. Similarly, a SNIA autonomous profile that references DMTF component profiles may reference the SNIA Indications profile as a means of identifying indication support for both DMTF and SNIA Indications.

A SNIA autonomous profile that wishes to support the DMTF Indications would list support for either the SNIA Indications, the SNIA Indication or the SNIA Experimental Indication profile. If the implementation incorporates a DMTF component profile that requires indications, then the implementation should support the SNIA Indications Profile. If the implementation does not incorporate any DMTF component profile that requires indications, then the implementation may reference either the SNIA Indication or SNIA Experimental Indication profile.

50.3 Implementation

Figure 77 illustrates the elements from the DMTF Indications Profile.

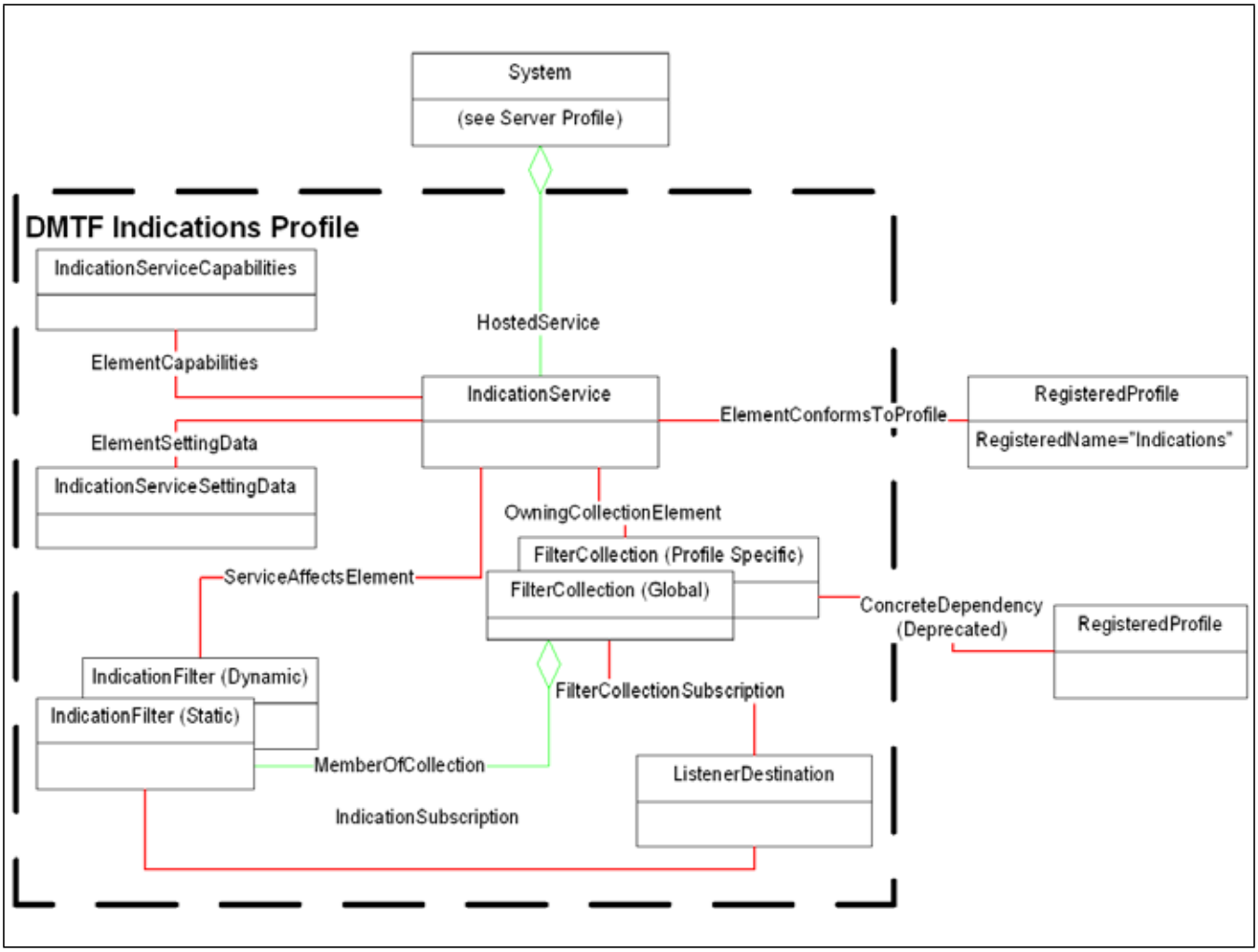

Figure 77 - Elements of the DMTF Indications Profile

To understand Figure 77 refer to the DMTF Indications Profile version 1.2.1. It is provided here as the context for adding the SNIA elements and constraints.

The Central Class of the DMTF Indications profile is the IndicationService. It represents the current settings of the indication functions of the WBEM Server. It has an associated IndicationServiceCapabilites that indicate what can be changed relative to the IndicationService. It also has an associated IndicationServiceSettingData that represents the default settings for the service. Every time the services is started up, it will adopt the settings as defined in the IndicationServiceSettingData.

Some classes defined by the DMTF profile are also used by the SNIA extensions, but are not the necessarily the same usages. The classes of special note are:

- IndicationFilters
  - DMTF defines Static and Dynamic Indication Filters. SNIA defines “Pre-defined” and “Client Defined” IndicationFilters. For the purposes of implementing this profile a “Client Defined” IndicationFilter is equivalent to a Dynamic IndicationFilter.

  - A “Pre-defined” IndicationFilter is similar to the DMTF Static IndicationFilter. However, the SNIA Pre-defined IndicationFilter is a declaration that the implementation of the owning profile provides support for producing the indication.

- FilterCollections
  - DMTF only defines Static FilterCollections (GlobalFilterCollections and ProfileSpecificFilterCollections). Static FilterCollections have a defined “coverage” that is defined by the referencing profile.
  - SNIA defines “Predefined” and “Client defined” FilterCollections. A SNIA “Predefined” FilterCollection is the collection of all Predefined IndicationFilters supported by the referencing profile implementation. This shall include all mandatory IndicationFilters of the profile and may include any optional or conditional IndicationFilters of the profile. It may also contain vendor specific Predefined IndicationFitlers.
    Client defined FilterCollections are collections of IndicationFilters (Predefined or Client defined) as defined by a client application. Client defined FilterCollections allow a client application to subscribe to only those indications that are of interest to the particular application.

It is important to note that a profile implementation that is supporting SNIA Indications profile needs to consider implementing both the DMTF Static FilterCollections and the SNIA Predefined FilterCollections.

The rest of this profile defines the SNIA extensions to this model and how those extensions relate to the elements in the DMTF profile.

50.3.1 SNIA Extensions to the DMTF Indications Profile

Figure 78 illustrates the extensions to the DMTF Indications Profile.

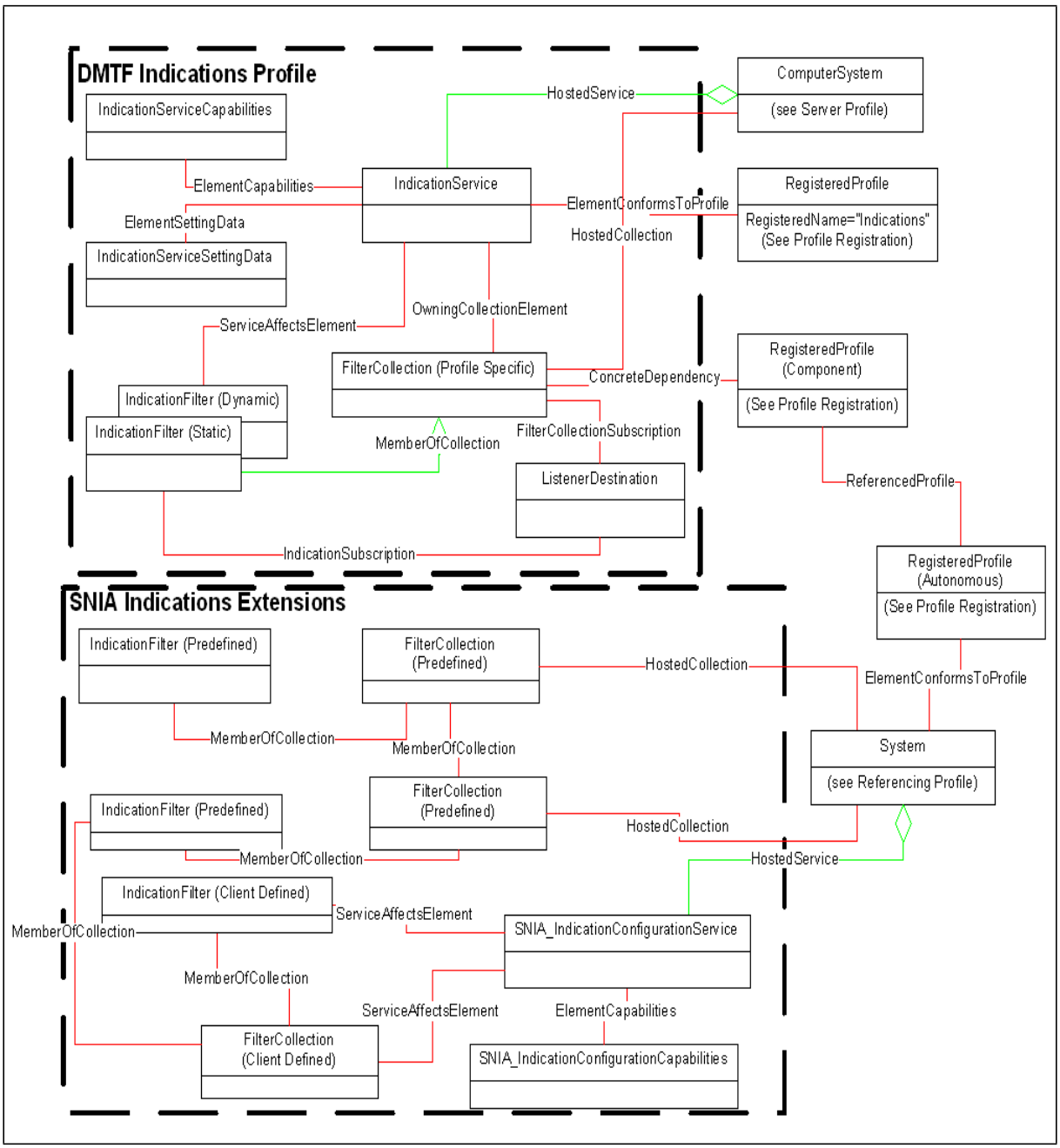

Figure 78 - The SNIA Extensions to the DMTF Indications Profile

The first thing to notice in Figure 78 is the System defined in the DMTF Indications profile, when implemented in an SMI-S environment refers to the System in the Server Profile that hosts the ObjectManager of the CIM Server. This is not clear in the DMTF Profile, because DMTF does not have a “Server Profile.”

The RegisteredProfile with RegisteredName=”Indications” would have RegisteredOrganization=”DMTF” if the SNIA extensions are not implemented and RegisteredOrganization=”SNIA” if the SNIA extensions are implemented.

The DMTF Indications profile provides for the IndicationService, which represents the indication service of the infrastructure (e.g., the CIMOM). The SNIA_IndicationConfigurationServices represents the indications support provided by a particular profile implementation (e.g., an Array implementation). Both the IndicationService and the IndicationConfigurationService have associated Capabilities that reflect the capabilities of the infrastructure and the specific profile implementation, respectively.

Both the DMTF Indication Profile and the SNIA extensions support the notion of FilterCollections. However, the formation of the collections are slightly different. The DMTF Indications Profile defines two collections; Global FilterCollections which would be implemented by the infrastructure and Profile Specific FilterCollections, which would be implemented by profiles. The SNIA FilterCollections define one FilterCollection per Profile for all predefined IndicationFilters supported by an implementation (e.g., an Array implementation). In addition, these FilterCollections are organized into a hierarchy. The top FilterCollection would be for predefined IndicationFilters for the autonomous profile implementation (e.g., an Array or Fabric Profile implementation). The second level of the hierarchy would be FilterCollections for any implementations of component profiles.

The SNIA extensions also provide support for client defined FilterCollections. These are FilterCollections explicitly created by a client application to collect IndicationFilters that the client application wants to subscribe to. Client defined FilterCollections may collect any predefined IndicationFilters, as well as client defined IndicationFilters.

In addition, the SNIA extensions call for FilterCollections to be hosted by some system. For the SNIA predefined FilterCollections the HostedCollection would be to the System of the autonomous profile (e.g., Array System or Fabric AdminDomain). But the SNIA extensions also call for a HostedCollection from the DMTF FilterCollections to the same system the IndicationService is hosted on.

Also note that SNIA predefined FilterCollections may contain SNIA_IndicationFilterTemplates (see section 50.3.7).

One last point on the SNIA extensions. SNIA defines predefined and client defined IndicationFilters. DMTF defines static and dynamic IndicationFilters. Both sets are defined in this profile to distinguish application of association requirements. Generally speaking a predefined IndicationFilter is also a static IndicationFilter and a client defined IndicationFilter is a dynamic IndicationFilter. However, the SNIA predefined IndicationFilter has the added semantics of representing a filter that has a provider to support generating the indication called for.

50.3.2CIM_IndicationServiceCapabilities Extensions

DMTF defines this class (and the related CIM_ElementCapabiities) as conditional on support for allowing updating of the IndicationService. The CIM_IndicationServiceCapabilities shall always be populated, even if all of FilterCreationEnabledIsSettable, DeliveryRetryAttemptsIsSettable, DeliveryRetryIntervalIsSettable, SubscriptionRemovalActionIsSettable and SubscriptionRemovalTimeIntervalIsSettable are “false”.

50.3.3 AlertIndication Extensions

The SNIA extensions to the DMTF profile override or add the following properties:

- Overridden Properties
  - AlertingManagedElement (overridden)
  - AlertType (overridden)
  - PerceivedSeverity (overridden)
  - SystemName (overridden)
  - CorrelatedIndications(overridden)

  - MessageArguments(overridden)
- Added Properties
  - SystemCreationClassName (added)
  - ProviderName (added)
  - EventID (added)
  - Description (added)

50.3.3.1 AlertingManagedElement encoding in AlertIndication Instances

When encoding the mandatory property “AlertingManagedElement” of an AlertIndication the SNIA extensions define following rules:

- If the element in question is modeled by the profile implementation, then the format for this property should be as a Typed WBEM URI as defined in DSP0207.
- If the element in question is not modeled by the profile implementation, then the encoding for this property should be as meaningful to clients as possible

The DMTF Indications profile only supports the first rule. If the AlertingManagedElement is not modeled, then DMTF calls for the property to be NULL.

50.3.3.2 AlertType

The SNIA Indications profile defines an definitive list of values for this property. The DMTF Indications profile refers to DSP0228.

50.3.3.3 PerceivedSeverity

The SNIA Indications profile defines the valid range of values supported for this property. The DMTF Profile does not, but does not preclude any of the values supported by the SNIA Indications profile.

50.3.3.4 SystemName

The SNIA Indications profile defines the property as:

> The scoping System's Name for the Provider generating this Indication.
>
> The SystemName would typically be the name of the system that generates the indication

The DMTF Indications profile defines the property as:

> Should be the value of the Name property of the scoping system of the managed element that is the AlertingManagedElement.

These definitions are not consider in conflict.

EXPERIMENTAL

50.3.3.5 CorrelatedIndications

The SNIA Indications profile considers this property to be Experimental. The DMTF Indications profile does not. But the property is optional in both profiles.

EXPERIMENTAL

50.3.3.6 MessageArguments

This property is Mandatory for both the DMTF and SNIA Indications profile. It is optional in the SNIA Indication Profile. If a standard message has no arguments this property should be coded as an empty array.

50.3.3.7 Additional AlertIndication properties

The SNIA Indications Profile requires the following additional properties not defined in the DMTF Indications Profile definition of CIM_AlertIndication:

- SystemCreationClassName
- ProviderName

In addition, the SNIA Indications profile defines the following properties as optional:

- EventID
- Description

50.3.4 CIM_IndicationSubscription

The SNIA Indications Profile defines this class as Mandatory. The DMTF Indications Profile defines this as conditional on support for IndicationFilters, which are optional. The SNIA Indications Profile also defines IndicationFiltters to be optional, but expects either predefined or client defined filters to exist, so support for the IndicationSubscription has been defined as Mandatory.

The DMTF Indications Profile introduces the possibility of support for CIM_FilterCollectionSubscription to the exclusion of CIM_IndicaitonSubscription. The SNIA Indications Profile does not currently have this capability (for backward compatibility).

In addition, the SNIA Indications profile extends the IndicationSubscription definition to include the following properties:

- LastIndicationIdentifier
- LastIndicationProductionDateTime

50.3.5 CIM_ListenerDestinations

The DMTF Indications Profile models indication handlers with CIM_ListenerDestinations. The SNIA Indications Profile uses this for both CIM-XML and WS-Management, but also supports the CIM_ListenerDestinationCIMXML for backward compatibility with previous releases of the standard.

50.3.6 Handling of Indication Storms

The SNIA Indications Profile extends the DMTF Profile to talk about bellwether events and indications. To contain the impact of indication storms an implementation can employ additional techniques:

- Use of Bellwether events (if they are defined by the profile)

50.3.6.0.1 Use of Bellwether Events

There are many state changes in the model for a device or application that results in changes in many CIM instances. For example, the addition of a device or application representation to a CIMOM should result in creation indications for every single member instance of that device or application. The activation of a ZoneSet from one of the member Switches in a fabric should result to indication listeners on another Switch's namespace creation indications for every instance of the new ZoneSet.

The worse case risk is that several of this type of situation may occur simultaneously and result in network storms and the sudden saturation of the LAN. Additionally, the use of computing resources of

the device or application producing the indication or client receiving the indications may be unacceptably high.

Indications provide the most value when they are used by a client as a mechanism to pick a significant or small number of changes in CIMOMs of interest. In order to capture a wide variety of changes, any of which may be pertinent to the client application, the client is likely to create many indication subscriptions and keep them all active simultaneously. This approach is not problematic because the number of management related changes to any device or application in the network is usually very small.

As mentioned previously, there are several potential situations where an excessive number of indications can be produced, thereby potentially overloading the network, originating CIMOM, and receiving client's resources. There is no need to occur such a risk because it is likely that the client is not going to be interested in all things at all times. The interest of the client in instance changes usually follows the needs of the current users of that client application.

Bellwether indications are used by SMI-S designers and individual implementation to signal many instance changes with one event. A client can assume that some previously defined graph of associated CIM instances are affected when it receives a bellwether indication. It can then choose, if warranted, to fetch all or some of these instances. This design prevent the previously mentioned adverse side effects.

Some rules being considered are:

- When a device or application is added to a namespace and there are indication subscription that cover some or all of the graph of instances added by side effect of the addition, then only a create indication is produced for the top level object for the device or application, like ComputerSystem, provided that there is an indication subscription for changes in the top-level object. Similarly, if a device or application is deleted in the same situation, then only a delete indication will be produced.
- Bellwether indication are mandatory if they exist in SMI-S and will be easily identified as being bellwether events.
    - The classes associated to the bellwether indication will be part of the definition of the indication. The client can assume that instances of these classes will have been affected and can choose to harvest that data. The implementation is not required to produce instances of every class listed as per the requirements defined elsewhere in SMI-S.
- SMI-S Designer's are encouraged to define bellwether indications, which can be of any class of indication, for major state changes of a model. In the previous examples, the device creation could be a life cycle indication where changes in ZoneSet change may be best communicated by an Alert Indication.

50.3.6.0.2 Bellwether Indications for ComputerSystem

It is important to not overload a SMI-S client when device or applications are added or removed from CIM Object Managers. The addition or removal of the representation of a device or application is attributed to the creation or deletion of a top-level computer system instance. This overloading would arise from a SMI-S Agent sending creation or deletion indications to every indication destination for all component or dependent instances to the top-level computer system. For this profile, when a top-level computer system instance is created in the model, the SMI-S agent shall not produce indications for indication subscriptions, on indications that do not reference the top-level computer system, that would otherwise receive InstCreation indications. Likewise, for this profile, when a top-level computer system is deleted from the model, the SMI-S agent shall not produce indications for indication subscriptions, on indications that do not reference the top-level computer system, that would otherwise receive InstDeletion indications.

50.3.7 Semi-Fixed Client Specific Indication Filters

The SNIA Indications profile extends the DMTF Indications Profile with support for semi-fixed IndicationFilters via a SNIA class definition for an IndicationFilterTemplate.

Semi-fixed Client specific IndicationFilters extend the support for indications in the following classes:

- SNIA_IndicationFilterTemplate

  This class mirrors the CIM_IndicationFilter, but the Query property supports a query with the string 'SUBSTITUTION_STRING' included in a CQL query. This is a template that may be used by a client application to create an instance of CIM_IndicationFilter with a client application supplied string in place of the string 'SUBSTITUTION_STRING'.

- SNIA_IndicationConfigurationCapabilities (and the SupportedFeatures property)

  The SupportedFeatures property of SNIA_IndicationConfigurationCapabilities includes the value ‘7’ (“Semi-fixed IndicationFilters”) that indicates semi-fixed IndicationFilters are supported.

When an implementation sets this enumeration, the implementation shall support creation (CreateInstance) of IndicationFilters that follow a pattern that includes substitution strings as defined in the SNIA_IndicationFilterTemplate Query property. The client is allowed to replace the substitution string with any simple expression (including constants).

Semi-fixed IndicationFilterTemplates are documented in this standard with the substitution string identified with the string ‘SUBSTITUTION_STRING’ in the Query property. For example, the following Query values would indicate a Semi-fixed IndicationFilterTemplate:

```
SELECT * FROM CIM_InstDeletion
WHERE SourceInstance ISA CIM_StorageSynchronized AND
OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH(‘SUBSTITUTION_STRING’)
```

or

```
SELECT * FROM CIM_InstModification
WHERE SourceInstance ISA CIM_StorageVolume AND
SourceInstance.OperationalStatus = ‘SUBSTITUTION_STRING’
```

When a semi-fixed IndicationFilterTemplate is defined in this standard, the description column for the CIM Element for the Indication will identify valid substitutions or will reference an implementation section that identifies the valid values.

50.3.7.1 Naming Conventions for IndicationFilterTemplates and IndicationFilters

The naming convention for IndicationFilters is consistent with conventions defined for in the DMTF Indications profile. The SNIA Indications Profile extends this to cover IndicationFilterTemplates and introduces an OrgID of “SNIA”.

Both IndicationFilters and IndicationFilterTemplates have a Name property. The value of the Name property shall be formatted as defined by the following ABNF rule:

```
OrgID ":" RegisteredName ":" UniqueID
```

Where OrgID identify the business entity owning the referencing profile. OrgID shall include a copyrighted, trademarked, or otherwise unique name that is owned by that business entity or that is a registered ID assigned to that business entity by a recognized global authority. In addition, to ensure uniqueness, OrgID shall not contain a colon (:).

For referencing profiles owned by the SNIA, OrgID shall match "SNIA" for IndicationFilterTemplates defined by the standard. For vendor unique IndicationFilterTemplates, the OrgID should be a unique name for the vendor. For client defined IndicationFilters that are based on IndicationFilterTemplates, the OrgID should identify the client (application) organization.

The RegisteredName shall be the registered name of the referencing profile, as defined by the value of its CIM_RegisteredProfile.RegisteredName property.

The UniqueID shall uniquely identify the instance within the referencing profile.

50.3.8 Filter Collections

Both the DMTF and SNIA Indications profiles support FilterCollections. However, their formation is different. That is, the two profiles define different collections of indications. The DMTF only defines “static” (e.g., pre-defined) FilterCollections. The SNIA defines support for client-defined (e.g., Dynamic) FilterCollections, as well as pre-defined FilterCollections. Furthermore, the SNIA pre-defined FilterCollections are different from the DMTF static FilterCollections.

The DMTF defines the following static filter collections:

- Global filter collections [See CIM_FilterCollection (GlobalFilterCollection)]

  These are “coverage” filter collections (collections with no members). At least one of the following global filter collections shall be implemented:

  - CollectionName=”DMTF:Indications:GlobalProfileSpecifiedAlertIndicationFilterCollection”

    If any alert indications specified in referencing profiles or in this profile are implemented, the implementation may expose a GlobalFilterCollection instance in the Interop namespace that covers all alert indications defined in profiles. In implementations where it is not possible to determine whether alert indications specified in referencing profiles are implemented, the instance may be exposed if the delivery of alert indications is implemented in general.

  - CollectionName=”DMTF:Indications:GlobalProfileSpecifiedLifecycleIndicationFilterCollection”

    If any lifecycle indications specified in referencing profiles or in this profile are implemented, the implementation may expose a GlobalFilterCollection instance in the Interop namespace that covers all lifecycle indications defined in profiles. In implementations where it is not possible to determine whether lifecycle indications specified in referencing profiles are implemented, the instance may be exposed if the delivery of lifecycle indications is implemented in general.

  - CollectionName=”DMTF:Indications:GlobalProfileSpecifiedIndicationFilterCollection”

    If any indications specified in referencing profiles or in this profile are implemented, the implementation may expose a GlobalFilterCollection instance in the Interop namespace that covers all indications defined in profiles. In implementations where it is not possible to determine whether indications specified in referencing profiles are implemented, the instance may be exposed if the delivery of indications is implemented in general.

  - CollectionName=”DMTF:Indications:GlobalLifecycleIndicationFilterCollection”

    If the implementation supports the delivery of lifecycle indications, the implementation shall expose a GlobalFilterCollection instance in the Interop namespace that covers all lifecycle indications defined in profiles.

- Optional profile specific filter collections [See CIM_FilterCollection (ProfileSpecificFilterCollection)]

  There are two profile specific filter collections for each profile:

  - CollectionName=”<OrgID>:<Profile RegisteredName>:ProfileSpecifiedAlertIndicationFilterCollection”

    If and only if a referencing profile defines alert indications, the implementation may expose a ProfileSpecificFilterCollection instance in the Interop namespace that covers all alert indications defined in that profile. The members of a profile-specific filter collection covering all alert indications defined in a referencing profile shall be all indication-specific indication filters covering the alert indications defined in that referencing profile. This definition in effect defines the defined coverage as all alert indications defined in the referencing profile.

- CollectionName="<OrgID>:<Profile RegisteredName>:ProfileSpecifiedLifecycleIndicationFilterCollection"

  If and only if a referencing profile defines lifecycle indications, the implementation may expose a ProfileSpecificFilterCollection instance in the Interop namespace that covers all lifecycle indications defined in that profile. The members of a profile-specific filter collection covering all lifecycle indications defined in a referencing profile shall be all indication-specific indication filters covering the lifecycle indications defined in that referencing profile or in this profile. This definition in effect defines the defined coverage as all lifecycle indications defined in the referencing profile.

These filter collections cover indications defined by the profile (as opposed to indications implemented by an specific profile implementation).

The SNIA Indications Profile extends the DMTF Indications Profile to supports two types of filter collections:

- Predefined Filter Collections - The predefined filter collections augments the Indication Profile support for predefined indication filters and indication filter templates by collecting them into a structure of collections (one per profile) that are hosted on the top level system of the autonomous profile. This provides a convenient means of finding an implementations support for predefined filters.

  The collection name for the pre-defined Array filter collection would be "SNIA:Array:Predefined". The collection would include all pre-defined indication filters supported by the implementation of the Array (including vendor specific indication filters). It would **not** include optional indication filters defined by the SMI-S Array profile that are not implemented by the implementation.

- Client Defined Filter Collections - The client defined filter collections are collections that are defined by applications of the standard. Client defined filter collections may include predefined and/or client defined indication filters and allow the application to collect a set of related indication filters to which the application wishes to subscribe.

  The collection name for the Acme application client defined filter collection might be "Acme:AcmeAlerts" or "Acme:Array:AcmeAlerts". In the first case, "AcmeAlerts would need to be unique in the context of the OrgID "Acme". In the second case, "Array:AcmeAlerts" would be Acme unique alerts for Array profiles.

50.3.8.1 Predefined Filter Collections

Predefined filter collections are an optional feature of the standard. Support would be indicated via the IndicationConfigurationCapabilities (see section 50.3.10) in the SupportedFeatures property. If the SupportedFeatures array includes the value '5' it means the implementation supports predefined filter collections.

Figure 79 illustrates the classes associated with predefined FilterCollection support.

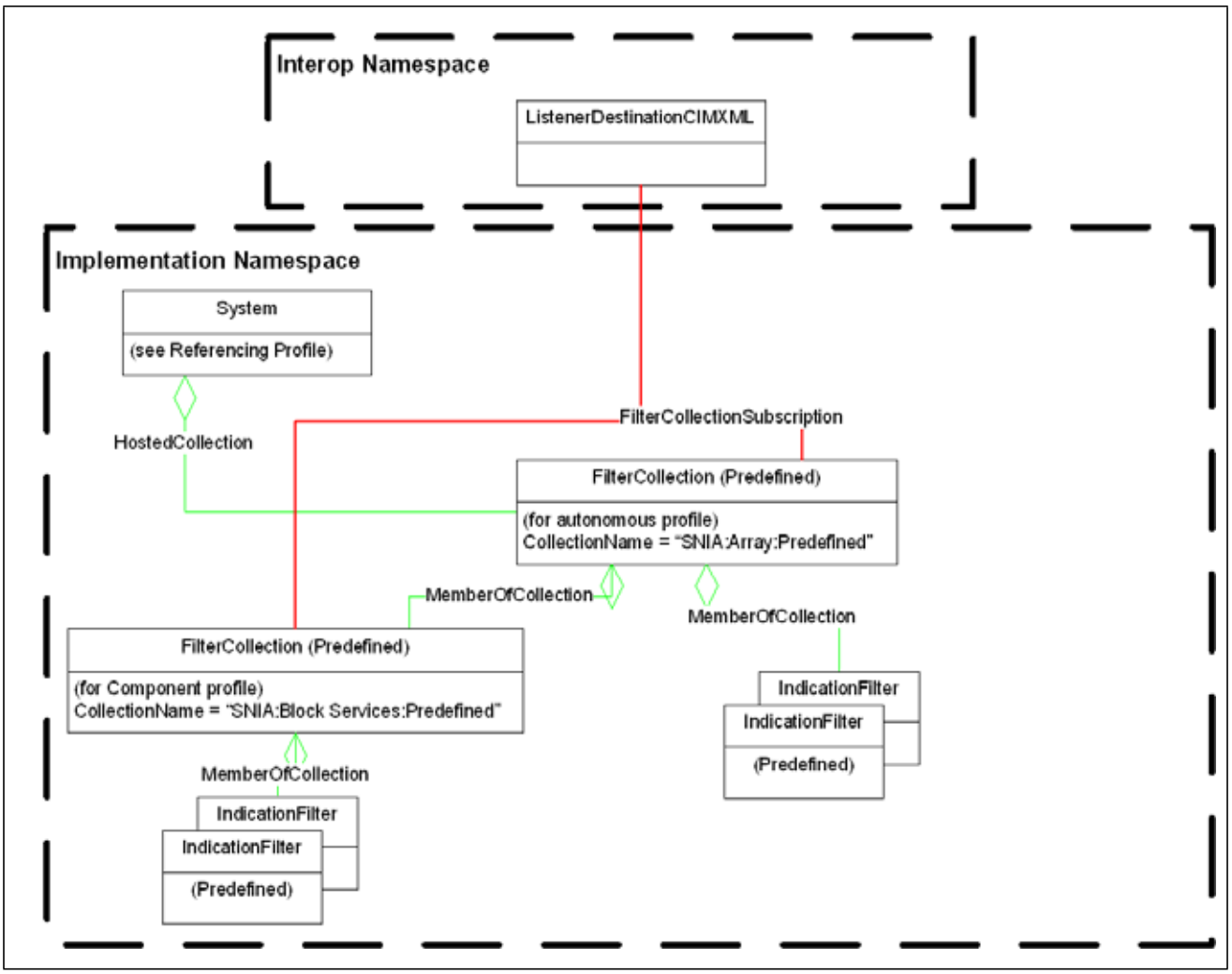

Figure 79 - Predefined Filter Collections

Support for predefined filter collections includes instantiations of the following classes and associations to the model:

- FilterCollection (Predefined)

  A predefined FilterCollection collects a set of predefined IndicationFilters. The primary purpose of a predefined FilterCollection is for an implementation to declare the IndicationFilters that it supports. Minimally this should include all IndicationFilters that are defined as mandatory for the profile. However, it may also include optional, conditional or vendor extension IndicationFilters supported by the implementation.

  One predefined FilterCollection is defined for each profile supported by the implementation. The FilterCollections are organized in a 2 level hierarchy. The top most FilterCollection is the FilterCollection for the autonomous profile. The name of the FilterCollection (CollectionName property) is of the form "SNIA:<profile name>:Predefined". In Figure 79 the autonomous profile is an Array profile. In addition to collecting predefined IndicationFilters of the autonomous profile, the top level FilterCollection would also collect FilterCollections for each of the component profiles supported by the in implementation. In Figure 79 the component profile FilterCollection shown is for the Block Services Package (CollectionName = "SNIA:Block Services:Predefined").

- HostedCollection

  This associates each filter collection with the top level system for which the collection applies (e.g., the top level system of the autonomous profile).

- FilterCollectionSubscription

  This associates (subscribes) a ListenerDestination (Handler) to a FilterCollection (collection). All indication filters in the collection will be reported to the ListenerDestination (without the need for individual subscriptions on the indication filters).

  Note: A FilterCollectionSubscription will not return any indications for any IndicationFilterTemplates in a pre-defined FilterCollection. Templates are only used for creation of client defined IndicationFilters.

  A client may subscribe to the collection of indications rather than subscribing to each individual filter in the collection. However, a client might prefer to form its own collection of filters that it wants to subscribe to (see section 50.3.8.2).

- MemberOfCollection

  This associates predefined IndicationFilters and predefined component FilterCollections to the FilterCollections in which they belong. MemberOfCollection associations are static and established by the implementation.

50.3.8.2 Client Defined Filter Collections

Client defined filter collections are an optional feature of the standard. Support would be indicated via the IndicationConfigurationCapabilities (see section 50.3.10) in the SupportedFeatures property. If the SupportedFeatures array includes the value ‘6’ it means the implementation supports client defined filter collections.

Figure 80 illustrates the classes associated with client defined FilterCollection support.

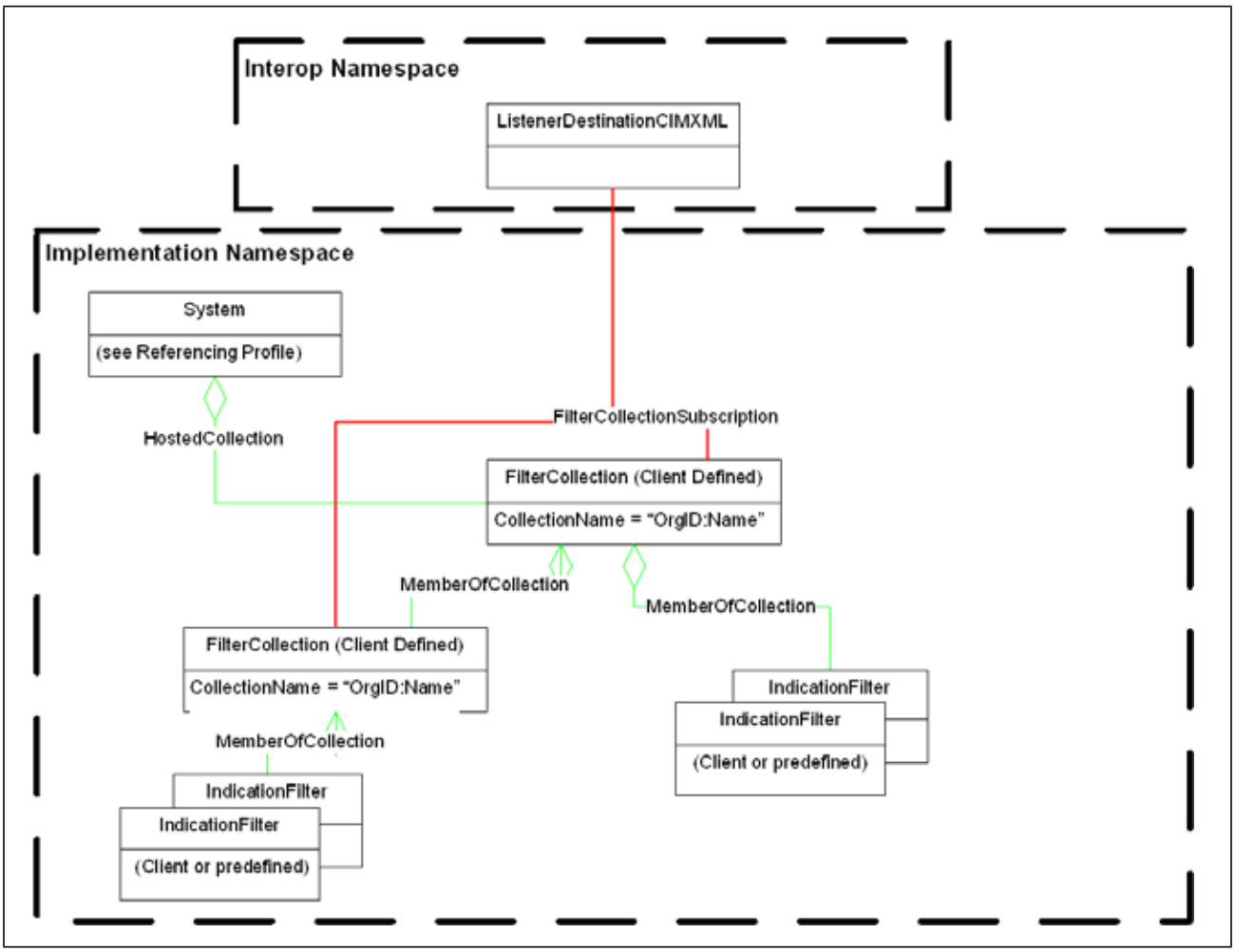

Figure 80 - Client Defined Filter Collections

Support for client defined filter collections includes support of the following classes and associations to the model:

- FilterCollection (Client Defined)

  A client defined FilterCollection collects a set of IndicationFilters (or other FilterCollections). The primary purpose of a client defined FilterCollection is to allow a client to establish a set of indication filters in which it wishes to subscribe to as a group. The indication filters collected may be either client defined or predefined. They may include any IndicationFilters supported by the implementation.

  Unlike predefined FilterCollections client defined FilterCollections may be organized for the convenience of the client application. The FilterCollections may be organized in a hierarchy of any number of levels and a any one FilterCollection need not correspond to a profile.

  The name of the FilterCollection (CollectionName property) is of the form "<OrgID:<Unique Name>. In Figure 80 a top level collection is defined with a lower level FilterCollection. The <OrgID> component should be a company indicator (e.g., stock ticker). The <Unique Name> part of the name should uniquely identify the collection within that company. It may be desirable to make the <Unique Name> part of the CollectionName a compound construction (e.g., <Product Name:Name within Product>). But all that this standard dictates is that the <OrgID> cannot be "SNIA".

A set of methods for creating and maintaining client defined FilterCollections are provided by the IndicationConfigurationService (see section 50.3.10).

- HostedCollection

  This associates each filter collection with the top level system for which the collection applies (e.g., the top level system of the autonomous profile).

- FilterCollectionSubscription

  This associates (subscribes) a ListenerDestination (Handler) to a FilterCollection (collection). All indication filters in the collection will be reported to the ListenerDestination (without the need for individual subscriptions on the indication filters.

  A client may subscribe to the collection of indications rather than subscribing to each individual filter in the collection.

- MemberOfCollection

  This associates IndicationFilters (predefined or client defined) and other FilterCollections (client defined or predefined) to higher level FilterCollections. MemberOfCollection is established using methods of the IndicationConfigurationService (see section 50.3.10).

50.3.9 DMTF and SNIA IndicationFilters

The DMTF Indications Profile defines three types of IndicationFilters (DynamicIndicationFilter, GlobalIndicationFilter and IndicationSpecificIndicationFilter) and two abstract IndicationFilters from which the three types are derived (IndicationFilter and StaticIndicationFilter). In addition, the DMTF Indications Profile, itself, defines two concrete IndicationFilters (SubscriptionRemovalIndication and ListenerDestinationRemovalIndication).The SNIA defines two additional types of IndicationFilters (pre-defined and Client Defined). The derivation relationship among these class definitions are shown in Figure 81.

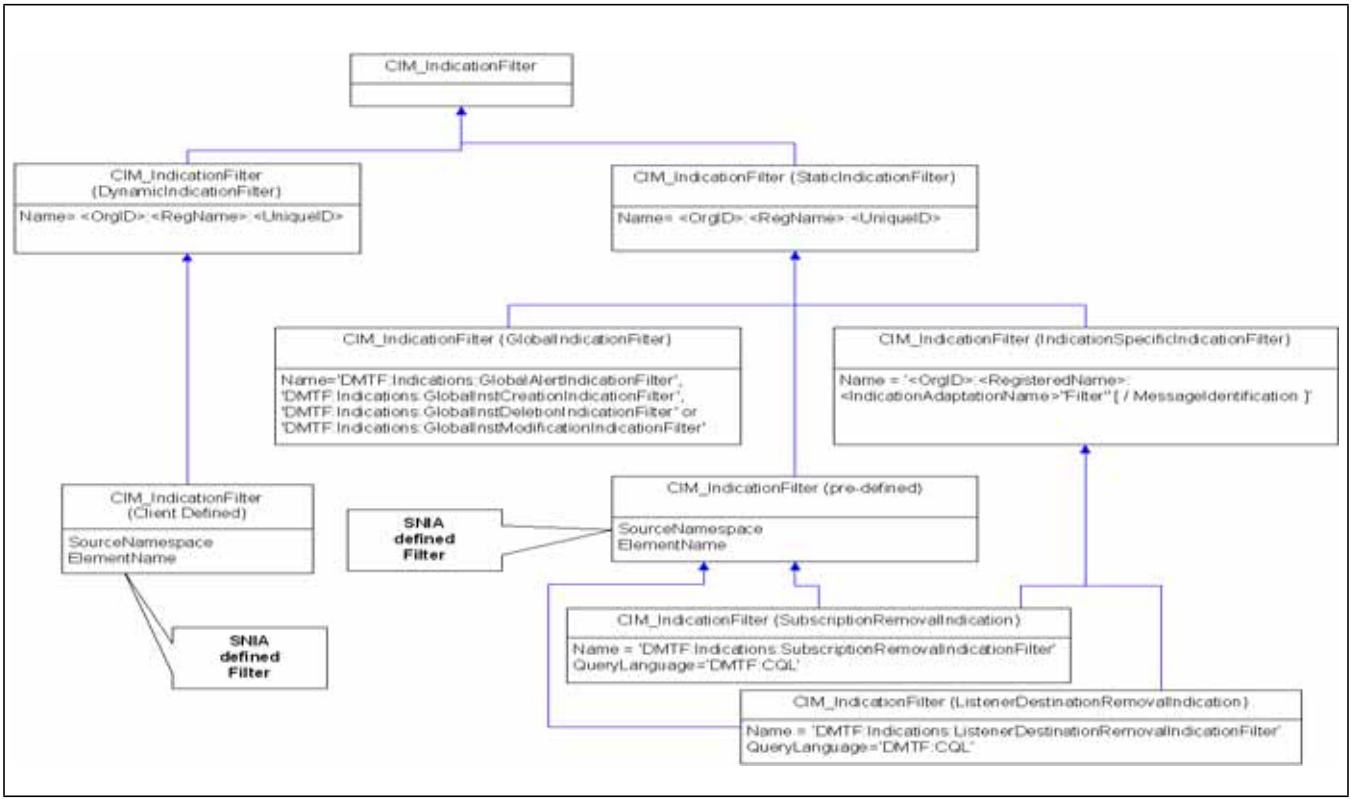

Figure 81 - Derivation Relationships among IndicationFilters

The SNIA IndicationFilter (Client Defined) is derived from the DMTF IndicationFilter (DynamicIndicationFilter). SNIA adds a couple of properties (ElementName and SourceNamespace). The SourceNamespace is deprecated, but provided by in the client defined IndicationFilter for backward compatibility.

The DMTF IndicationFilter (GlobalIndicationFilter), DMTF IndicationFilter (IndicationSpecificIndicationFilter) and the SNIA IndicationFilter (pre-defined) are all derived from the DMTF IndicationFilter (StaticIndicationFilter).

The DMTF Indications Profile defines two filters that are specific to the Indications profile. They are the SubscriptionRemovalIndication and ListenerDestinationRemovalIndication filters. They are derived from both the DMTF IndicationFilter (IndicationSpecificIndicationFilter) and the SNIA IndicationFilter (pre-defined). The requirements of these two do not conflict in as much as the Name property conforms to both and the SNIA IndicationFilter adds properties. The QueryLanguage is coded as 'DMTF:CQL', since this does not cause a backward incompatibility for existing SNIA indications.

50.3.9.1 DMTF and SNIA encoding of QueryLanguage

DMTF and SNIA encode QueryLanguage differently. All DMTF defined IndicationFilters have a QueryLanguage of 'DMTF:CQL'. SNIA has several encodings of QueryLanguage, one of which is 'DMTF:CQL'. The SNIA encodings for QueryLanguage are 'DMTF:CQL', 'WQL' or 'SMI-S V1.0'. The 'WQL' and 'SMI-S V1.0' encodings are deprecated.

50.3.10 Indication Configuration Services

The SNIA Indications Profile extends the DMTF Indications Profile to cover implementation specific service methods (and capabilities) and related classes. These define the indications support for a specific implementation of an autonomous profile (e.g., Array or Fabric).

Figure 82 illustrates the classes associated with support for methods for configuring and testing indications support.

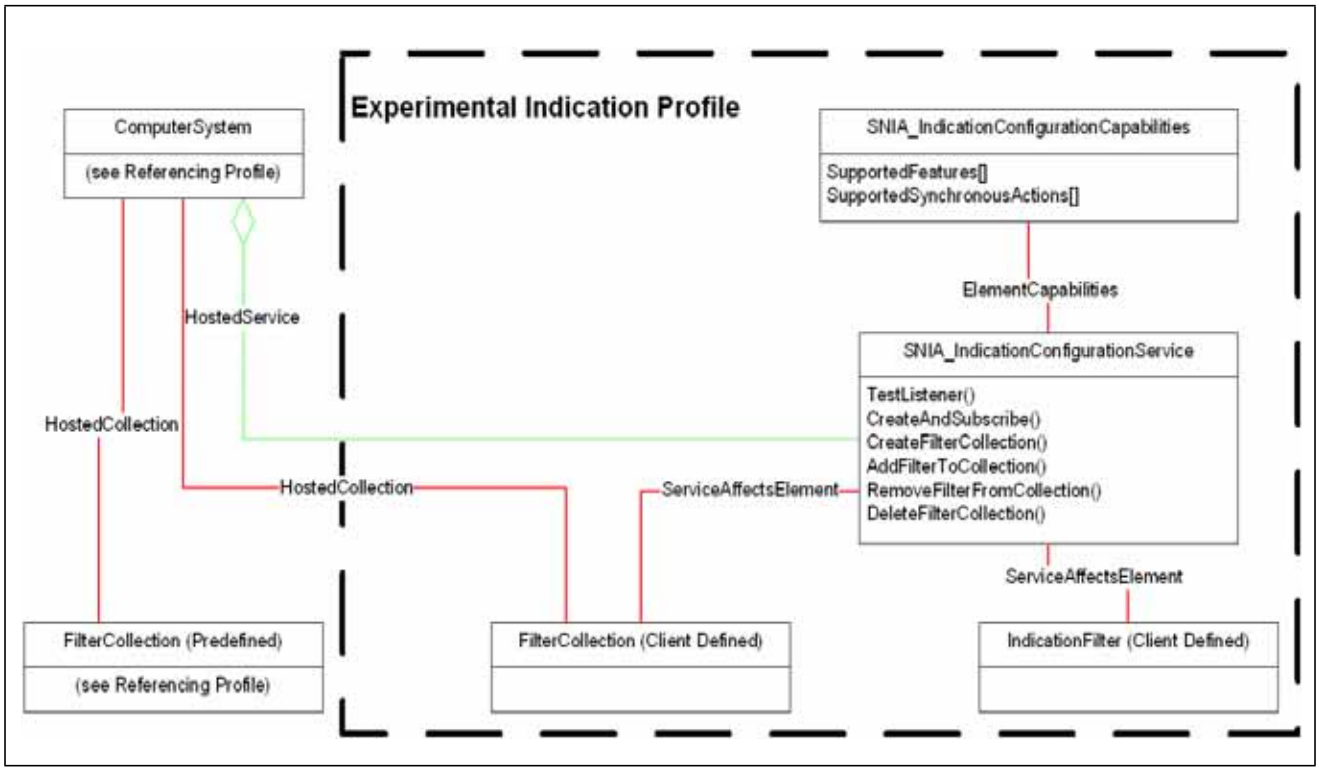

Figure 82 - Indication Configuration Service Classes

Support for the Indication Configuration Service add the following classes and associations to the model:

- SNIA_IndicationConfigurationService

  This service includes methods for testing a listener, handling the creation and subscription to indications as one extrinsic method (rather than a set of CreateInstances) and methods for managing client defined filter collections. The SNIA_IndicationConfigurationService is a service that is specific to a particular profile implementation and the classes managed are instantiated in the implementation namespace. The SNIA_IndicationConfigurationService shall be instantiated when the Experimental Indication Profile is supported, but support for the individual methods are conditional (See the SNIA_IndicationConfigurationCapabilities).

  Note that predefined FilterCollections are not managed by the methods of the indication configuration services. They are, by their nature, static and maintained by the implementation.

- HostedService

  This associates the SNIA_IndicationConfigurationService to the system of the autonomous profile (referencing profile) for which the service applies. For example, for the Array Profile, the indication configuration service would be hosted on the top level computer system for the Array. An autonomous profile shall have exactly one indication configuration service and it shall be hosted on the top level

system. It may not be hosted on non-top-level systems (e.g., component computer systems defined in the Multiple Computer System Profile).

- SNIA_IndicationConfigurationCapabilities

  There is one instance of SNIA_IndicationConfigurationCapabilities for an SNIA_IndicationConfigurationService. These capabilities define the extrinsic methods supported by the implementation and a set of SupportedFeatures. The possible SupportedSynchronousActions values and their definitions are as follows:

  - '2' (None) - None of the IndicatonConfigurationService methods are supported.
  - '3' (Test Listener) - The TestListener method is supported.
  - '4' ("Create and Subscribe) - The CreateAndSubscribe method is supported.
  - '5' (Filter Collection Methods) - The methods for managing client defined FilterCollections are supported.

  The possible SupportedFeatures values and their definitions are as follows:

  - '2' (None) - None of the optional features are supported. Specifically, FilterCollections (either predefined or client defined) are not supported. Filters (either predefined or client defined) are not supported. And semi-fixed Indication Filters are not supported.

    Note: In SMI-S, 'none' is only valid for profiles that don't support indications. Any profile that supports indications shall support either or both predefined or client defined indication filters.

  - '3' (Predefined Filters) - The implementation has populated a set of predefined IndicationFilters for indications that it supports. These should include those specified by SMI-S, but may include vendor specific IndicationFilters that the implementation supports.
  - '4' (Client Defined Filters) - The implementation supports client defined IndicationFilters through CreateInstance (and possibly through the CreateAndSubscribe method). If SupportedFeatures includes '4', but SupportedSynchronousActions does not include '4', it means that only CreateInstance is supported for creation of client defined filters.
  - '5' (Predefined Filter Collections) - The implementation has collected its predefined IndicationFilters into FilterCollections. Each predefined FilterCollection corresponds to an SMI-S profile (autonomous or component) and is structured as a two level hierarchy. The top FilterCollection contains the predefined IndicationFilters of the autonomous profile and FilterCollections for the component profiles.
  - '6' (Client Defined Filter Collections) - The implementation supports client defined FilterCollections through CreateInstance (and possibly through the CreateAndSubscribe method). If SupportedFeatures includes '6', but SupportedSynchronousActions does not include '4', it means that only CreateInstance is supported for creation of client defined filter collections.
  - '7' (Semi-fixed Indication Filters) - The implementation supports semi-fixed IndicationFilters, which means that it can accept client definitions of filters that fit the pattern defined by the semi-fixed IndicationFilter.

- ElementCapabilities

  This associates the SNIA_IndicationConfigurationService instance to its SNIA_IndicationConfigurationCapabilities instance.

- ServiceAffectsElement

  This associates the IndicationConfigurationService to client defined FilterCollections it manages.

50.3.11Health and Fault Management Consideration

EXPERIMENTAL

50.3.11.1 Indication Correlation

There are cases where many indications are produced in response to a single event. In fact, the indications themselves are correctly viewed as presenting an aspect or view of the event itself and not as a comprehensive representation of the event. AlertIndications provide a means of notification that is direct to the point than life cycle indications, even though the production of life cycle indications are also important. The subtleties of the effect of the event are better communicated through life cycle indications.

A given event, like a network port communication failure, can itself be reported as an AlertIndication. It is also important to communicate the change in status of the port itself through life cycle indications. It is probable that the network port communication failure will cause some function of the device which contains the point to also fail or become degraded. The impact of the failure (or significant state or status change) is of great interest to management clients as it assist in the triage of the error and potentially can also assist HFM aware clients to contain the failure, fence off the failing component, or even prevent a more serious failure of the system in which the component participates, like the failure of business function (like closing the book at quarter end or dropping transactions at Christmas time).

SMI-S provides the mechanism where storage management can be affected without requiring a priori knowledge of the device or application being managed. In this world, the overall system or service component that is most able to assess and report the impact of the failure (or significant state or status change) is the managed device or application itself. Indication correlation provides the mechanism that can be used to asynchronously report the changes brought about by the event.

The mechanism requires that a single indication be the first reporter of the event. This first reporter may be an AlertIndication or a life cycle indication. This indication should report the state or status change caused by the event in the simplest and most direct manner. All other indications that report state or status change and are associated directly to the first reporter indications should correlated to the first reported indication. Indication correlation shall be done by the implementation through reporting the IndicationIdentifier of the correlated and previously produced indications in the CorrelatedIndications array. The elements in the CorrelatedIndications may be in any order. The linkage of indication thusly correlated is like a one-way linked list. The beginning of the correlation link is indicated by the nullness of the CorrelatedIndications property.

Indication correlation shall be accomplished in the path of cause and effect or scoping relationships. If indication B is to correlated to indication A, then the model change reported by B is caused by or is a side-effect of the model change reported by A. Indication correlation shall not be accomplished by sorting the indications to be correlated by PerceivedSeverity. That being said, Indication correlation should not be used to report secondary events, themselves caused by the primary event, and side-effects of the secondary event.

Indication correlation provides important information about the onset of the condition and its immediate impact that may not be retrievable when the client can react. The spread of the effects of the event within a device or application can certainly be faster than maximum speed of the management network.

Indication correlation shall be accomplished through scoping relationships, like the part to group component or dependent to antecedent relationships, or across direct cause and effect relationships for peer components. For example, given that a network port communication failure within a given device causes changes to the status of port, the scoping computer system, the port communications statistics, the status of the network pipe, and the overall communication statistics of the device, then indication correlation shall not report correlation of the network port communication failure to the changes in the

overall communications statistics of the device. This requirement is necessary to limit the potentially lengthy correlation and impose undue burden on the implementation without value to the client.

EXPERIMENTAL

50.4 Methods

50.4.1 Extrinsic Methods of the Profile

EXPERIMENTAL

50.4.1.1 TestListener Method

The TestListener method allows a client to test that the ListenerDestination is actually reachable from the implementation. It also provides information, in error cases, on why the test failed (e.g., destination not resolvable, Port not reachable, certificate errors, etc.).

```
uint32 IndicationConfigurationService.TestListener(
   [IN] CIM_ListenerDestination REF Destination );
```

The TestListener method takes as input a reference to a CIM_ListenerDestination class.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the indication was sent. The implementation will send an AlertIndication with the standard message MP22 (Listener Destination Test).

1 - The method is not supported (e.g., the provider does not support the function)

4 - Failed. If the return code is 4, it means that the provider was not able to send the indication.

5 - Invalid parameter. The protocol specified is not a recognized protocol or the destination was not in a valid URI format.

50.4.1.2 CreateAndSubscribe Method

The CreateAndSubscribe method allows a client to create IndicationFilters, ListenerDestinations and Subscriptions in a single extrinsic call. If the instance supplied as input already exist (e.g., an IndicationFilter, FilterCollection and ListenerDestination), then the only element created is the subscription.

```
uint32 IndicationConfigurationService.CreateAndSubscribe(
   [IN, EmbeddedInstance(“CIM_IndicationFilter")]
       string IndicationFilter,
   [IN, EmbeddedInstance(“CIM_FilterCollection")]
       string FilterCollection,
   [IN, EmbeddedInstance(“CIM_ListenerDestination")]
       string ListenerDestination,
   [IN, EmbeddedInstance(“CIM_AbstractIndicationSubscription")]
       string SubscriptionData,
   [OUT] CIM_AbstractIndicationSubscription REF Subscription);
```

The CreateAndSubscribe method takes as input a set of 4 embedded Instances provided by the client application:

- CIM_IndicationFilter - This would be filled in if the desired output is an instance of CIM_IndicationSubscription. The method will use properties of the embedded instance to determine if the

IndicationFilter exists. If it does not exist and IndicationConfigurationService.SupportedFeatures includes ‘4’, then the method will create the instance.

- CIM_FilterCollection - This would be filled in if the desired output is an instance of CIM_FilterCollectionSubscription. The method will use properties of the embedded instance to determine if the FilterCollection exists. If it does not exist and IndicationConfigurationService.SupportedFeatures includes ‘6’, the method will create the instance (but it will be an empty collection).
- CIM_ListenerDestination - The method will use properties of the embedded instance to determine if the ListenerDestination exists. If it does not exist, the method will create the instance.
- CIM_AbstractIndicationSubscription - As an input embedded instance, the references should be NULL. The other properties of the instance will be used to establish the properties of the subscription created.

The CreateAndSubscribe method output (Subscription) is a reference to the created subscription which can be either an IndicationSubscription or a FilterCollectionSubscription.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the subscription (and any related instances) has been created.

1 - Not Supported. The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘4’.

4 - Failed. If the return code is 4, it means that the provider was not able to create the subscription.

5 - Invalid Parameter. The implementation does not recognize the value of a parameter.

50.4.1.3 CreateFilterCollection

The CreateFilterCollection method allows a client to create a client specific collection of IndicationFilters (and/or other filter collections) for the purpose of subscribing to all collected indications with one subscription to the FilterCollection.

```
uint32 IndicationConfigurationService.CreateFilterCollection(
[IN] string FilterCollectionName,
[IN] CIM_ManagedElement REF Members[],
[OUT] CIM_FilterCollection REF FilterCollection );
```

The CreateFilterCollection method takes as input the name of the collection and a list of members to be added to the collection:

- FilterCollectionName - FilterCollectionName takes the form ‘OrgID:CollectionID’ where OrgID is the client vendor and product is part of CollectionID. Predefined collections have an OrgID of ‘SNIA’, so a client defined filter collection shall not use the prefix ‘SNIA’.
- Members[] - This is a list of references to instances of either CIM_IndicationFilter or CIM_FilterCollection. A Member may be an IndicationFilter (a ManagedElement) or another FilterCollection (a Collection).

The CreateFilterCollection method output (FilterCollection) is a reference to the created FilterCollection.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the FilterCollection has been created and the specified members have been added to the collection.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to create the FilterCollection.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, the FilterCollectionName has a prefix of ‘SNIA’ or one of the Members is not a CIM_IndicationFilter or CIM_FilterCollection.

50.4.1.4 AddFilterToCollection

The AddFilterToCollection method allows a client to add members to an existing client defined FilterCollection.

```
uint32 IndicationConfigurationService.AddFilterToCollection(
   [IN] CIM_ManagedElement REF Members[],
   [IN] CIM_FilterCollection REF FilterCollection );
```

The AddFilterToCollection method takes as input

- Members[] - This is a list of references to instances of either CIM_IndicationFilter or CIM_FilterCollection to be added to the FilterCollection.
- FilterCollection - This is a reference to the (Client Defined) FilterCollection to which the new members are to be added.

The AddFilterToCollection method output is simply success or failure.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the Members have been added to the FilterCollection.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to add the members to the FilterCollection and none of the additions were done.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, one of the Members is not a CIM_IndicationFilter or CIM_FilterCollection.

50.4.1.5 RemoveFilterFromCollection

The RemoveFilterFromCollection method allows a client to remove filters from client defined collections.

```
uint32 IndicationConfigurationService.RemoveFilterFromCollection(
   [IN] CIM_ManagedElement REF Members[],
   [IN] CIM_FilterCollection REF FilterCollection );
```

The RemoveFilterFromCollection method takes as input

- Members[] - This is a list of references to instances of either CIM_IndicationFilter or CIM_FilterCollection to be removed to the FilterCollection.
- FilterCollection - This is a reference to the (Client Defined) FilterCollection to which the members are to be removed.

The RemoveFilterFromCollection method output is simply success or failure.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the Members have been removed to the FilterCollection.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to remove the members to the FilterCollection and none of the removals were done.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, one of the Members is not a CIM_IndicationFilter or CIM_FilterCollection.

50.4.1.6 DeleteFilterCollection

The DeleteFilterCollection method allows a client to remove a client defined FilterCollection.

```
uint32 IndicationConfigurationService.DeleteFilterCollection(
   [IN, Required] CIM_FilterCollection REF FilterCollection.
   [IN] boolean RemoveMembers );
```

The DeleteFilterCollection method takes as input

- FilterCollection - This is a reference to the (Client Defined) FilterCollection to which the members are to be removed.
- RemoveMembers - This boolean, when “true” means the method should imply removal of existing members of the collection. When “false”, the method will fail if members exist in the collection.

The DeleteFilterCollection method output (FilterCollection) is simply success or failure.

The return codes that may be produced by this method are:

0 - If a 0 return code is returned, it means the FilterCollection has been deleted.

1 - The method is not supported (e.g., the provider does not support the function). That is, IndicationConfigurationService.SupportedSynchronousActions does not include ‘5’.

4 - Failed. If the return code is 4, it means that the provider was not able to delete the FilterCollection. For example, this method will return this error if RemoveMembers is ‘false’ and there are members in the FilterCollection.

5 - Invalid parameter. A parameter is not recognized as a valid value. For example, the FilterCollectionName has a prefix of ‘SNIA’. That is, this method will return this error if the application attempts to delete a predefined FilterCollection.

EXPERIMENTAL

50.4.2 Intrinsic Methods of the Profile

The SNIA Indications Profile extensions to the DMTF profile are mostly populated by providers and is accessible to clients using basic read and association traversal. However, there are three constructs that may be created by Clients. These are the FilterCollection, MemberOfCollection and the FilterCollectionSubscription. In addition to being able to create them, client may delete them (except “pre-defined” filters which cannot be deleted), and a client may modify any IndicationFilter that was client created.

NOTE The IndicationConfigurationService provides extrinsic methods for creating and managing client defined FilterCollections. The intrinsic methods for creating and maintaining FilterCollections is provided for clients that have a preference to use of intrinsic methods.

EXPERIMENTAL

50.4.2.1 FilterCollection

The CreateInstance and DeleteInstance operations are supported for client defined FilterCollections.

CreateInstance

```
<instanceName>CreateInstance (
        [IN] <instance> NewInstance
 )
```

On CreateInstance on FilterCollections, the implementation should allow NULL to be specified for key properties (the InstanceID). If key properties are passed in on the CreateInstance, the implementation may ignore the keys.

If the client supplies an CollectionName, the implementation shall persist the property for later use by the client.

If successful, the return value defines the object path of the new CIM Instance relative to the target Namespace (i.e., the Model Path).

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_ALREADY_EXISTS (the CIM Instance already exists), CIM_ERR_FAILED (some other unspecified error occurred).

NOTE If a client attempts to create an FilterCollection that already exists (has the same InstanceID), but other properties are different, then the request will fail. If the Client attempts to create an FilterCollection that has identical properties to an existing FilterCollection instance, it will succeed and CreateInstance need not treat the instance as a separate instance.

DeleteInstance

```
void DeleteInstance (
         [IN] <instanceName> InstanceName
)
```

The InstanceName input parameter defines the name (model path) of the Instance to be deleted.

If successful, the specified FilterCollection Instance shall have been removed.

The deletion of a FilterCollection instance will cause the automatic deletion of any associated MemberOfCollection and FilterCollectionSubscription instances.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED (the filter collection is a pre-defined filter collection), CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class does not exist in the specified namespace), CIM_ERR_NOT_FOUND (the CIM Class does exist, but the requested CIM Instance does not exist in the specified namespace), CIM_ERR_FAILED (some other unspecified error occurred).

50.4.2.2 MemberOfCollection

The CreateInstance and DeleteInstance operations are supported on MemberOfCollection for associating client defined FilterCollections and their members.

CreateInstance

```
<instanceName>CreateInstance (
        [IN] <instance> NewInstance
 )
```

An implementation should populate all fields of references (scheme, hostname, port number, namespace, key) in object path for the instance being created.

The host name portion shall be set to a valid, client-resolvable host name (i.e., DNS) or IPv4 or IPv6 address. Internal names (e.g., /etc/hosts) are not valid. If host name, then must be FQDN (not a short name). This implies the provider shall be configured as a valid DNS client to use host names and cannot rely on an administratively defined name.

If successful, the return value defines the object path of the new CIM Instance relative to the target Namespace (i.e., the Model Path).

Note that for CreateInstance of a MemberOfCollection requires that the FilterCollection instance and the member instances (FilterCollection or IndicationFilter) exist.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED (the FilterCollection referenced is a pre-defined filter collection), CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_ALREADY_EXISTS (the CIM Instance already exists), CIM_ERR_FAILED (some other unspecified error occurred).

DeleteInstance

```
void DeleteInstance (
         [IN] <instanceName> InstanceName
)
```

The InstanceName input parameter defines the name (model path) of the Instance to be deleted.

If successful, the specified MemberOfCollection Instance shall have been removed.

Deletion of a MemberOfCollection will not cause the deletion of any corresponding FilterCollection or IndicationFilter instances.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED (the FilterCollection referenced is a pre-defined filter collection), CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class does not exist in the specified namespace), CIM_ERR_NOT_FOUND (the CIM Class does exist, but the requested CIM Instance does not exist in the specified namespace), CIM_ERR_FAILED (some other unspecified error occurred).

EXPERIMENTAL

50.4.2.3 FilterCollectionSubscription

The CreateInstance and DeleteInstance operations are supported on FilterCollectionSubscription for associating client defined FilterCollections with ListenerDestinations.

CreateInstance

```
<instanceName>CreateInstance (
        [IN] <instance> NewInstance
 )
```

An implementation should populate all fields of references (scheme, hostname, port number, namespace, key) in object path for the instance being created.

The host name portion shall be set to a valid, client-resolvable host name (i.e., DNS) or IPv4 or IPv6 address. Internal names (e.g., /etc/hosts) are not valid. If host name, then must be FQDN (not a short name). This implies the provider shall be configured as a valid DNS client to use host names and cannot rely on an administratively defined name.

If successful, the return value defines the object path of the new CIM Instance relative to the target Namespace (i.e., the Model Path).

Note that for CreateInstance of an FilterCollectionSubscription requires that the ListenerDestinationCIMXML instance and the FilterCollection exist.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class of which this is to be a new Instance does not exist), CIM_ERR_ALREADY_EXISTS (the CIM Instance already exists), CIM_ERR_FAILED (some other unspecified error occurred).

DeleteInstance

```
void DeleteInstance (
         [IN] <instanceName> InstanceName
)
```

The InstanceName input parameter defines the name (model path) of the Instance to be deleted.

If successful, the specified FilterCollectionSubscription Instance shall have been removed.

Deletion of a FilterCollectionSubscription will not cause the deletion of any corresponding ListenerDestinationCIMXML or FilterCollection instances.

If unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error in is given in parentheses.

CIM_ERR_ACCESS_DENIED, CIM_ERR_NOT_SUPPORTED, CIM_ERR_INVALID_NAMESPACE, CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters), CIM_ERR_INVALID_CLASS (the CIM Class does not exist in the specified namespace), CIM_ERR_NOT_FOUND (the CIM Class does exist, but the requested CIM Instance does not exist in the specified namespace), CIM_ERR_FAILED (some other unspecified error occurred).

50.5 Use Cases

The use cases in the following sections illustrate some of the features (and particularly the methods) of this profile.

50.5.1 Testing a Listener Destination

Table 493 identifies the elements of the use case to test whether a listener destination can receive indications from a CIM Server.

Table 493 - Test that a Listener Destination if Functioning Properly

Use Case Element	Description
Summary	Given an application that listens for indications, test whether or not a CIM_Server can successfully communicate with the application.
Basic Course of Events	1. Start the application that is listening for indications 2. Tell the CIM_Server to send an indication to the listener application 3. Verify the CIM_Server sent the indication 4. Verify that the listening application received the indication
Alternative Paths	None
Exception Paths	None
Triggers	Installing a new or upgraded application that listens for indications.
Assumptions	None
Preconditions	The CIM Server is operational and supports a profile with support for the TestListener function.
Postconditions	The listener application receives the indication and displays it (or logs it).

The use cases in the following sections illustrate some of the features (and particularly the methods) of this profile.

50.5.2 Testing a Listener Destination

Table 493 identifies the elements of the use case to test whether a listener destination can receive indications from a CIM Server.

Table 494 - Test that a Listener Destination if Functioning Properly

Use Case Element	Description
Summary	Given an application that listens for indications, test whether or not a CIM_Server can successfully communicate with the application.
Basic Course of Events	1. Start the application that is listening for indications 2. Tell the CIM_Server to send an indication to the listener application 3. Verify the CIM_Server sent the indication 4. Verify that the listening application received the indication
Alternative Paths	None
Exception Paths	None
Triggers	Installing a new or upgraded application that listens for indications.
Assumptions	None
Preconditions	The CIM Server is operational and supports a profile with support for the TestListener function.
Postconditions	The listener application receives the indication and displays it (or logs it).

50.5.3 Discovering predefined IndicationFilters of an implementation

Table 495 identifies the elements of the use case to discover predefined indication filters supported by a profile implementation.

Table 495 - Discovery of Predefined IndicationFilters

Use Case Element	Description
Summary	Given an implementation of an autonomous Profile and its top level ComputerSystem, determine any predefined IndicationFilters it has.
Basic Course of Events	1. Determine if the implementation has IndicationConfigurationCapabilities 2. If it does, verify that it supports Predefined Indications and/or Predefined FilterCollections 2a. If Predefined FilterCollections are supported, then look for the Top Level FilterCollection and determine the predefined IndicationFilters supported 2b. If FilterCollections are not supported (or the IndicationConfigurationService is not supported), then simply enumerate CIM_IndicationFilter in the namespace of the top level ComputerSystem
Alternative Paths	None
Exception Paths	None
Triggers	The administrator (or application) wants to inspect filters that are declared to be supported by an implementation.
Assumptions	None
Preconditions	The top level system of the profile has been discovered from profile registration and ElementConformsToProfile.
Postconditions	A list of predefined IndicationFilters (possibly by Profile) is produced.

50.5.4 Creating a subscription to a predefined IndicationFilter

Table 496 identifies the elements of the use case to create a subscription to a predefined indication filter.

Table 496 - Create a subscription to a predefined indication filter

Use Case Element	Description
Summary	Given a ListenerDestination and a predefined indication filter, subscribe to the filter
Basic Course of Events	1. See if the implementation supports the CreateAndSubscribe method 2. Retrieve the predefined IndicationFilter 2a. If CreateAndSubscribe is supported, copy properties of the predefined indication filter into and embedded instance and invoke CreateAndSubscribe passing the Destination of the Listener 2b. If CreateAndSubscribe is not supported, Create the ListenerDestination and Create the IndicationSubscription
Alternative Paths	None
Exception Paths	None
Triggers	Set up a listener to get indications for an indication in the SMI-S Specification.
Assumptions	None
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The subscription is recorded in the CIM Server.

50.5.5 Creating a client defined indication and subscription

Table 497 identifies the elements of the use case to create an indication filter and subscribe to it.

Table 497 - Create an IndicationFilter and subscribe to it

Use Case Element	Description
Summary	Given a top level system of an autonomous profile and a URI for an indication listener create a client defined indication and subscribe to it.
Basic Course of Events	1. Get the IndicationConfigurationService if it exists 1b. If not try to do a CreateInstance for the IndicationFilter (if it succeeds, then continue) 2. If the service exists, get the IndicationConfigurationCapabilities to find out if the implementation supports client defined IndicationFilters 3. If the capability exists, then do a CreateAndSubscribe for the indication 3b. Do a CreateInstance on the ListenerDestination and another CreateInstance on the IndicationSubscription
Alternative Paths	1. Create a FilterCollection and put the client defined indication filter in that collection
Exception Paths	None
Triggers	The administrator wants to listen for a specific indication of his/her choosing.
Assumptions	The implementation supports client defined IndicationFilters
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The IndicationFilter is created and a subscription to it is recorded in the CIM Server.

50.5.6 Creating a semi-fixed indication filter

Table 498 identifies the elements of the use case to create a semi-fixed indication filter..

Table 498 - Creation of a semi-fixed Indication filters

Use Case Element	Description
Summary	A client application wants to create an indication filter that has application specific information to include in the filter.
Basic Course of Events	1. Determine if the implementation supports semi-fixed IndicationFilters 2. Find the semi-fixed IndicationFilter 2a. Look in the 'SNIA' FilterCollections if they exist 2b. Enumerate predefined indication filters if not 3. Do a CreateAndSubscribe for the IndicationFilter substituting the application specific information in the query 3a. Do a CreateInstance on the filter and the subscription if CreateAndSubscribe is not supported.
Alternative Paths	1. If none of the new functions are supported try to create the filter using a CreateInstance
Exception Paths	None
Triggers	The application wants to restrict the number of indications it receives by adding application specific information to the filter query.
Assumptions	The implementation supports semi-fixed IndicationFilters
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The IndicationFilter is created and a subscription to it is recorded in the CIM Server.

50.5.7 Creating a FilterCollection

Table 499 identifies the elements of the use case to create a client defined Filter Collection.

Table 499 - Creation of a client defined FilterCollection

Use Case Element	Description
Summary	Given a top level system of an autonomous profile and a URI for an indication listener and a list of IndicationFilter that the administrator wants to listen for create a client defined FilterCollection and subscribe to it.
Basic Course of Events	1. Determine if the implementation supports client defined FilterCollections 1a. If not quit, you have no options. 2. Determine if the implementation supports FilterCollection Methods 2a. If not go to 3a 3. Do a CreateFilterCollection to create the FilterCollection 3a. Do a CreateInstance on the FilterCollection and a bunch of CreateInstances for the MemberOfCollection associations to each of the IndicationFilters. And then do a CreateInstance on FilterCollectionSubscription.
Alternative Paths	None
Exception Paths	None
Triggers	The client was subscriptions to IndicationFilters in a client specific list of filters.
Assumptions	None
Preconditions	The top level system of the profile and a listener destination for the application to get the indications.
Postconditions	The FilterCollection is created and a subscription to it is recorded in the CIM Server.

50.6 CIM Elements

Table 500 describes the CIM elements for Indications.

Table 500 - CIM Elements for Indications

Element Name	Requirement	Description
50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription)	Optional	CIM_AbstractIndicationSubscription (AbstractSubscription) is an Abstract class that holds properties common to CIM_IndicationSubscription and CIM_FilterCollectionSubscription.
50.6.2 CIM_AlertIndication (AlertIndication)	Optional	This is a specialization of the CIM_AlertIndication class in the DMTF Indications Profile.
50.6.3 CIM_ConcreteDependency (ProfileOfFilterCollection)	Mandatory	Deprecated. CIM_ConcreteDependency (ProfileOfFilterCollection) models the relationship between the a filter collection, and the referencing profile and the profile registration of that referencing profile.
50.6.4 CIM_ElementCapabilities (CapabiliesOfIndicationService)	Mandatory	Associates an IndicationServiceCapabilities to its IndicationService.
50.6.5 CIM_ElementCapabilities (Indication Config Service to Capabilities)	Mandatory	Experimental. This associates the IndicationConfigurationService to the IndicationConfigurationCapabilities.

Table 500 - CIM Elements for Indications

Element Name	Requirement	Description
50.6.6 CIM_ElementConformsToProfile (ElementConformsToProfile)	Mandatory	The CIM_ElementConformsToProfile (ElementConformsToProfile) models the relationship between an indication service and a scoping managed element.
50.6.7 CIM_ElementSettingData (InitialSettingsOfIndicationService)	Conditional	Conditional requirement: Support for instances of CIM_IndicationServiceSettingData. CIM_ElementSettingData models the relationship between an indication service and its initial settings.
50.6.8 CIM_FilterCollection (Client Defined)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections). This is a client defined collection of IndicationFilters to which a client may subscribe.
50.6.9 CIM_FilterCollection (GlobalFilterCollection)	Mandatory	Global filter collections address the needs of clients requiring notifications about large sets of events.
50.6.10 CIM_FilterCollection (Indications Predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is a collection of predefined IndicationFilters to which a client may subscribe.
50.6.11 CIM_FilterCollection (Predefined)	Optional	Experimental. This is an abstract class that would be specialized for individual profiles, for profiles that support predefined indication collections.
50.6.12 CIM_FilterCollection (ProfileSpecificFilterCollection)	Optional	The CIM_FilterCollection (ProfileSpecificFilterCollection) models profile-specific filter collections.
50.6.13 CIM_FilterCollection (StaticFilterCollection)	Optional	The Static filter collection is an ABSTRACT class (for common properties).
50.6.14 CIM_FilterCollectionSubscription (CollectionSubscription)	Optional	This associates the FilterCollection to the ListenerDestination that wants to receive the indications defined in the collection.
50.6.15 CIM_HostedCollection (Hosted Client Filter Collection)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections). This associates a client defined FilterCollection to the system in the referencing profile.
50.6.16 CIM_HostedCollection (Hosted Global FilterCollection or a Profile Specific FilterCollection)	Conditional	Experimental. Conditional requirement: Support for instances of CIM_FilterCollection (GlobalFilterCollection) or CIM_FilterCollection (ProfileSpecificFilterCollection). This associates a Global FilterCollection or a Profile Specific FilterCollection to the system in the referencing profile.
50.6.17 CIM_HostedCollection (Hosted Predefined Filter Collection)	Optional	Experimental. This associates a Predefined FilterCollection to the system in the referencing profile.
50.6.18 CIM_HostedCollection (System to predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).
50.6.19 CIM_HostedService (HostedIndicationService)	Mandatory	This associates the IndicationService to its hosting system.

Table 500 - CIM Elements for Indications

Element Name	Requirement	Description
50.6.20 CIM_HostedService (Indication Config Service to System)	Mandatory	Experimental. This associates the IndicationConfigurationService to the System in the referencing profile.
50.6.21 CIM_IndicationFilter (DynamicIndicationFilter)	Conditional	Conditional requirement: The IndicationService instance has the value TRUE is set for FilterCreationEnabled. Dynamic IndicationFilter (Filters that are created).
50.6.22 CIM_IndicationFilter (GlobalIndicationFilter)	Mandatory	This models a global indication filter for all indications.
50.6.23 CIM_IndicationFilter (IndicationSpecificIndicationFilter)	Optional	Indication-specific indication filters are a specialization of static indication filters. They model indication-specific indication filters for indications defined in referencing profiles or in this profile.
50.6.24 CIM_IndicationFilter (ListenerDestinationRemovalIndication)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a CIM_ListenerDestination instance.
50.6.25 CIM_IndicationFilter (StaticIndicationFilter)	Optional	Static indication filters are provided by an implementation. Their lifecycle and coverage is controlled solely by the implementation, and clients are not able to create or delete static indication filters.
50.6.26 CIM_IndicationFilter (SubscriptionRemovalIndication)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a CIM_IndicationSubscription or a CIM_FilterCollectionSubscription instance.
50.6.27 CIM_IndicationFilter (client defined)	Optional	This is for 'client defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the value and format of an indication filter string.
50.6.28 CIM_IndicationFilter (pre-defined)	Optional	This is for 'pre-defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the value and format of an indication filter string.
50.6.29 CIM_IndicationService (IndicationService)	Mandatory	An indication service is a component within an implementation that is responsible for delivering indications to listeners.
50.6.30 CIM_IndicationServiceCapabilities (IndicationServiceCapabilities)	Mandatory	CIM_IndicationServiceCapabilities is an optional element that represents the capabilities of the CIM_IndicationService instance.
50.6.31 CIM_IndicationServiceSettingData (IndicationServiceInitialSettings)	Optional	CIM_IndicationServiceSettingData is used to represent the initial configuration of the CIM_IndicationService instance.
50.6.32 CIM_IndicationSubscription (FilterSubscription)	Optional	This association defines a subscription to a specific IndicationFilter instance by a specific indication handler (as represented by a ListenerDestination instance).
50.6.33 CIM_InstCreation	Optional	CIM_InstCreation is an indication of the creation of a CIM instance. It would be generated when an instance of the SourceInstance class is created (either explicitly or implicitly).
50.6.34 CIM_InstDeletion	Optional	CIM_InstDeletion is an indication of the Deletion of a CIM instance. It would be generated when an instance of the SourceInstance class is deleted from the model (either explicitly or implicitly).
50.6.35 CIM_InstIndication (LifecycleIndication)	Optional	The CIM_InstIndication (LifecycleIndication) models lifecycle indications of CIM instances.

Table 500 - CIM Elements for Indications

Element Name	Requirement	Description
50.6.36 CIM_InstModification	Optional	CIM_InstModification is an indication of the modification or change to a CIM instance. It would be generated when an instance of the SourceInstance class is modified or changed (either explicitly or implicitly).
50.6.37 CIM_ListenerDestination (ListenerDestination)	Mandatory	CIM_ListenerDestination (ListenerDestination) models listener destinations.
50.6.38 CIM_ListenerDestinationCIMXML (Indication Handler)	Mandatory	Deprecated. A CIM_ListenerDestinationCIMXML describes the destination for CIM Export Messages to be delivered via CIM-XML.
50.6.39 CIM_MemberOfCollection (Client Defined Filter Collection to Filters)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections). This associates a client defined FilterCollection to the Filters in the collection.
50.6.40 CIM_MemberOfCollection (FilterCollectionInFilterCollection)	Optional	CIM_MemberOfCollection models the relationship between a filter collection and its contained other filter collections.
50.6.41 CIM_MemberOfCollection (IndicationFilterInFilterCollection)	Optional	CIM_MemberOfCollection models the relationship between a filter collection and its contained indication filters.
50.6.42 CIM_MemberOfCollection (Predefined Filter Collection to Indications Filters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Indications predefined FilterCollection to the predefined Filters supported by the implementation.
50.6.43 CIM_OwningCollectionElement (IndicationServiceOfFilterCollection)	Mandatory	CIM_OwningCollectionElement (IndicationServiceOfFilterCollection) models the relationship with between a filter collection and the indication service that owns the filter collection.
50.6.44 CIM_ServiceAffectsElement (IndicationServiceOfIndicationFilter)	Mandatory	CIM_ServiceAffectsElement is used to associate instances of CIM_IndicationFilter with an instance of CIM_IndicationService.
50.6.45 CIM_ServiceAffectsElement (IndicationServiceOfListenerDestination)	Mandatory	CIM_ServiceAffectsElement (IndicationServiceOfListenerDestination) models the relationship between indication services and the listener destinations they manage.
50.6.46 SNIA_IndicationConfigurationCapabilities (IndicationConfigurationCapabilities)	Mandatory	Experimental. This is the capabilities of the implementation of indications.
50.6.47 SNIA_IndicationConfigurationService (IndicationConfigurationService)	Mandatory	Experimental. This is the indication services of the implementation.
50.6.48 SNIA_IndicationFilterTemplate (semi-fixed)	Conditional	Experimental. Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='7' (Semi-fixed Indication Filters). This is a template for 'semi-fixed' IndicationFilter instances.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_AbstractIndicationSubscription	Optional	CQL -

Table 500 - CIM Elements for Indications

Element Name	Requirement	Description
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ListenerDestination	Optional	CQL -
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="SNIA" and MessageID="MP22"	Conditional	Conditional requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedSynchronousActions='3' (TestListener). CQL -This indication is sent in response to the TestListener method invocation. See *Storage Management Technical Specification, Part 2 Common Architecture, 1.6.1 Rev 5* 8.4.1.22 Message: Listener Destination Test.

50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription)

CIM_AbstractIndicationSubscription (AbstractSubscription) is an Abstract class that holds properties common to CIM_IndicationSubscription and CIM_FilterCollectionSubscription.

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Optional

Table 501 describes class CIM_AbstractIndicationSubscription (AbstractSubscription).

Table 501 - SMI Referenced Properties/Methods for CIM_AbstractIndicationSubscription (AbstractSubscription)

Properties	Flags	Requirement	Description & Notes
OnFatalErrorPolicy		Mandatory	The value of the OnFatalErrorPolicy property shall indicate the behavior that the implementation exposes with respect to represented subscriptions in case of failures that imply that some aspect of indication generation processing or indication delivery is no longer functioning and indications may be lost. This shall be the default behavior.
OtherOnFatalErrorPolicy		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.OnFatalErrorPolicy = 1.Value shall be non-Null if the value of the OnFatalErrorPolicy property is 1 (Other).
FailureTriggerTimeInterval		Mandatory	Value shall be the minimum delay before the policy indicated by the value of the OnFatalErrorPolicy property is applied.
SubscriptionState		Mandatory	See CIM schema definition.
OtherSubscriptionState		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.SubscriptionState = 1.Value shall be non-Null if the value of the SubscriptionState property is 1 (Other).
RepeatNotificationPolicy		Mandatory	The value of the RepeatNotificationPolicy property shall indicate the policy that the implementation applies with respect to the avoidance of repeated indication delivery of repeated indications. The possible values are 2 (None), 3 (Suppress) or 4 (Delay).

Table 501 - SMI Referenced Properties/Methods for CIM_AbstractIndicationSubscription (AbstractSubscription)

Properties	Flags	Requirement	Description & Notes
RepeatNotificationInterval		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 3. or Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. If the implementation applies the SuppressRepeatNotificationPolicy feature for the represented subscription, as indicated by the value 3 (Suppress) for the RepeatNotification property, the value of the RepeatNotificationInterval property shall be the length of the time interval in seconds that the implementation waits after initial delivery of a number of repeated indications as indicated by the value of the RepeatNotificationCount property before delivering the next repeated indication. If the implementation applies the DelayRepeatNotificationPolicy feature for the represented subscription, as indicated by the value 4 (Delay) for the RepeatNotification property, the value of the RepeatNotificationInterval property shall be the length of the monitoring time interval in seconds during which the implementation monitors the indication gate referenced by the subscription for a number of additional repeated indications. Furthermore, only if during that monitoring interval at least the number of repeated indications as indicated by the value of the RepeatNotificationCount accrue, delivers only the first indication as a substitute for all the repeated indications accrued during the monitoring time interval.
RepeatNotificationGap		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4.The value of the RepeatNotificationGap property shall be the length of the delay time interval in seconds that the implementation waits after delivering the first of a number of repeated indications that accrued during the monitoring time interval, before starting another monitoring time interval with respect to implementations of the DelayRepeatNotificationPolicy feature.
RepeatNotificationCount		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 3. or Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. If the implementation applies the SuppressRepeatNotificationPolicy feature for the represented subscription, as indicated by the value 3 (Suppress) for the RepeatNotification property, the value of the RepeatNotificationCount property shall be the number of repeated indications that the implementation delivers before suppressing the delivery of further repeated indications within the time interval exposed by the value of the RepeatNotificationInterval property. If the implementation applies the DelayRepeatNotificationPolicy feature for the represented subscription, as indicated by the value 4 (Delay) for the RepeatNotification property, the value of the RepeatNotificationCount property shall be the number of repeated indications that the implementation is required to monitor and delay during the monitoring time interval exposed by the value of the RepeatNotificationInterval property. Only if during that monitoring time interval the number of accrued repeated indications reaches that number, the implementation shall deliver the first of repeated indication as a substitute for the accrued repeated indications. In other words, the quotient of the values of the RepeatNotificationCount and the RepeatNotificationInterval properties expresses a rate of repeated indications that must have been reached or exceeded during the monitoring time interval before one indication is delivered at the end of the monitoring time interval.
Filter		Mandatory	Key: Value shall reference the IndicationFilter instance or the StaticFilterCollection instance.
Handler		Mandatory	Key: Value shall reference the ListenerDestination instance.

50.6.2 CIM_AlertIndication (AlertIndication)

This is a specialization of the CIM_AlertIndication class in the DMTF Indications Profile.

CIM_AlertIndication is subclassed from CIM_ProcessIndication. The class definition specializes the CIM_AlertIndication definition in the Indications profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Requirement: Optional

Table 502 describes class CIM_AlertIndication (AlertIndication).

Table 502 - SMI Referenced Properties/Methods for CIM_AlertIndication (AlertIndication)

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Mandatory	See CIM schema definition.
IndicationTime	N	Mandatory	See CIM schema definition.
IndicationFilterName		Mandatory	Experimental. The value of the IndicationFilterName property shall contain the name of the indication emitter that the indication passed before being delivered to the listeners subscribed to that indication emitter. For indication filters, the name is exposed by the value of the Name property in representing IndicationFilter instances. For filter collections, the name is exposed by the value of the CollectionName property in representing FilterCollection instances.
SequenceContext		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceContext property shall contain the sequence context portion of the sequence identifier; see the CIM schema description for a recommended format and structure. The value of the SequenceContext property shall be identical for all indications delivered to a particular listener while the representing listener destination is defined within the implementation. This definition is required to remain effective over restarts of the implementation environment.
SequenceNumber		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceNumber property shall contain the sequence number portion of the sequence identifier. The value of the SequenceNumber property shall be unique for each indication delivered to a particular listener while the listener destination is defined within the implementation; see the CIM schema description for required value constraints. This definition is required to remain effective over restarts of the implementation environment.
AlertingManagedElement (overridden)		Mandatory	The identifying information of the entity for which this Indication is generated. If the element in question is modeled by the profile implementation, then the format for this property should be as a Typed WBEM URI as defined in DSP0207.
AlertingElementFormat		Mandatory	Value shall match 2 (CIMObjectPath)'.
AlertType (overridden)		Mandatory	This shall be 1|2|3|4|5|6|7|8 ('Other' | 'Communications Alert' | 'Quality of Service Alert' | 'Processing Error' | 'Device Alert' | 'Environmental Alert' | 'Model Change' | 'Security Alert') **SMI-S defines the valid range of values in this property.**
PerceivedSeverity (overridden)		Mandatory	This shall be 0|1|2|3|4|5|6|7 ('Unknown', 'Other' | 'Information' | 'Degraded/ Warning' | 'Minor' | 'Major' | 'Critical' | 'Fatal/NonRecoverable') **SMI-S defines the valid range of values supported for this property.**
ProbableCause		Mandatory	See CIM schema definition.

Table 502 - SMI Referenced Properties/Methods for CIM_AlertIndication (AlertIndication)

Properties	Flags	Requirement	Description & Notes
SystemName (overridden)		Mandatory	The scoping System's Name for the Provider generating this Indication. The SystemName would typically be the name of the system that generates the indication. **The SMI-S definition is slightly different from the DMTF definition.**
CorrelatedIndications (overridden)		Optional	Experimental. IndicationIdentifiers whose notifications are correlated with this one **SMI-S defines this property as Experimental.**
OtherAlertType		Conditional	Conditional requirement: Support for the AlertType = 1.See CIM schema definition.
OtherSeverity		Conditional	Conditional requirement: Support for the Severity = 1.If PerceivedSeverity matches 1 (Other), this property is mandatory.
ProbableCauseDescription		Conditional	Conditional requirement: Support for the ProbableCause = 1.See CIM schema definition.
OwningEntity		Mandatory	The requirements of DSP0228 apply. Note that DSP0228 requires the value of the OwningEntity property in CIM_AlertIndication instances conveying a message from a message registry to be set to the content of the OWNING_ENTITY element from the message definition in the message registry.
MessageID		Mandatory	The requirements of DSP0228 apply. Note that DSP0228 requires the value of the MessageID property in CIM_AlertIndication instances conveying a message from a message registry to be set to the concatenation of the PREFIX and SEQUENCE_NUMBER attribute values from the message definition in the message registry (i.e. no further padding or adjustment of these values takes place).
MessageArguments (overridden)	N	Mandatory	An array of strings that contain the dynamic content of the message. **This was defined as Optional in previous versions of SMI-S.** **SMI-S allows this property to be NULL.**
Message		Optional	The Message property may contain the formatted message from the registry.
SystemCreationClassName (added)		Mandatory	
ProviderName (added)		Mandatory	
EventID (added)		Optional	
Description (added)		Optional	A free form text description.

50.6.3 CIM_ConcreteDependency (ProfileOfFilterCollection)

Deprecated. Each StaticFilterCollection instance representing a filter collection defined in a referencing profile shall be associated through a ProfileOfFilterCollection instance with the ProfileRegistration instance representing the implemented version of the referencing profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 503 describes class CIM_ConcreteDependency (ProfileOfFilterCollection).

Table 503 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (ProfileOfFilterCollection)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Value shall reference the ProfileRegistration instance.
Dependent		Mandatory	Value shall reference the StaticFilterCollection instance.

50.6.4 CIM_ElementCapabilities (CapabiliesOfIndicationService)

SMI-S defines this class as Mandatory. DMTF allows the IndicationServiceCapabilities and this association to be absent if the service cannot be modified. The class definition specializes the CIM_ElementCapabilities definition in the Indications profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 504 describes class CIM_ElementCapabilities (CapabiliesOfIndicationService).

Table 504 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (CapabiliesOfIndicationService)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Value shall reference a single IndicationService instance.
Capabilities		Mandatory	Value shall reference the IndicationServiceCapabilities instance representing the capabilities of the indication service identified by the value of ManagedElement.

50.6.5 CIM_ElementCapabilities (Indication Config Service to Capabilities)

Experimental. This associates the IndicationConfigurationService to the IndicationConfigurationCapabilities.

Requirement: Mandatory

Table 505 describes class CIM_ElementCapabilities (Indication Config Service to Capabilities).

Table 505 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Indication Config Service to Capabilities)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The indication capabilities instance associated with the indication configuration service.
ManagedElement		Mandatory	The indication configuration service.

50.6.6 CIM_ElementConformsToProfile (ElementConformsToProfile)

Each IndicationService instance representing an indication service shall be associated through an ElementConformsToProfile instance with exactly one IndicationProfileRepresentation instance representing the implemented version of this profile providing the indication service.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 506 describes class CIM_ElementConformsToProfile (ElementConformsToProfile).

Table 506 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (ElementConformsTo-Profile)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The CIM_IndicationService instance that represents the available Indication Service of the implementation..
ConformantStandard		Mandatory	RegisteredProfile instance describing the DMTF Indications profile.

50.6.7 CIM_ElementSettingData (InitialSettingsOfIndicationService)

CIM_ElementSettingData is used to associate an instance of CIM_IndicationServiceSettingData with an instance of CIM_IndicationService.

This should be implemented if the IndicationServiceSettingData is instantiated.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for instances of CIM_IndicationServiceSettingData.

Table 507 describes class CIM_ElementSettingData (InitialSettingsOfIndicationService).

Table 507 - SMI Referenced Properties/Methods for CIM_ElementSettingData (InitialSettingsOfIndication-Service)

Properties	Flags	Requirement	Description & Notes
IsDefault		Mandatory	Value shall be 1 (Is Default).
IsNext		Mandatory	Value shall be 1 (Is Next).
ManagedElement		Mandatory	Value shall reference an IndicationService instance.
SettingData		Mandatory	Value shall reference the IndicationServiceInitialSettings instance that represents the initial settings of the indication service identified by the value of ManagedElement.

50.6.8 CIM_FilterCollection (Client Defined)

Experimental. This is a client defined collection of IndicationFilters to which a client may subscribe. An implementation would indicate support for client defined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported='6' (Client Defined Filter Collections).

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections).

Table 508 describes class CIM_FilterCollection (Client Defined).

Table 508 - SMI Referenced Properties/Methods for CIM_FilterCollection (Client Defined)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be constructed using the following algorithm: <OrgID>:<CollectionID> where OrgID and CollectionID are separated by a colon ':'. OrgID should identify the client business entity that created the collection. OrgID should include a copyrighted, trademarked, or otherwise unique name that is owned by that client business entity or that is a registered ID assigned to that business entity by a recognized global authority.

50.6.9 CIM_FilterCollection (GlobalFilterCollection)

Global filter collections address the needs of clients requiring notifications about large sets of events. Global filter collections are a specialization of static filter collections. The defined coverage of global filter collections covers large sets of indications, such as

All alert indications

All alert indications specified in profiles

All lifecycle indications

All lifecycle indications specified in profiles.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 509 describes class CIM_FilterCollection (GlobalFilterCollection).

Table 509 - SMI Referenced Properties/Methods for CIM_FilterCollection (GlobalFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	This shall be "DMTF:Indications:GlobalProfileSpecifiedAlertIndicationFilterCollection", "DMTF:Indications:GlobalProfileSpecifiedLifecycleIndicationFilterCollection", "DMTF:Indications:GlobalProfileSpecifiedIndicationFilterCollection" or "DMTF:Indications:GlobalLifecycleIndicationFilterCollection".

50.6.10CIM_FilterCollection (Indications Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A SNIA Indications implementation shall indicate support for predefined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '5' (Predefined Filter Collections).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 510 describes class CIM_FilterCollection (Indications Predefined FilterCollection).

Table 510 - SMI Referenced Properties/Methods for CIM_FilterCollection (Indications Predefined FilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Indications:Predefined'.

50.6.11CIM_FilterCollection (Predefined)

Experimental. This is an abstract class that would be specialized for individual profiles, for profiles that support predefined indication collections. For example, see *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.10 CIM_FilterCollection (Indications Predefined FilterCollection).

Requirement: Optional

Table 511 describes class CIM_FilterCollection (Predefined).

Table 511 - SMI Referenced Properties/Methods for CIM_FilterCollection (Predefined)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be constructed using the following algorithm: <OrgID>:<Profile Name>'Predefined' where OrgID, the Profile Name and 'Predefined' are separated by a colon ':'.

50.6.12CIM_FilterCollection (ProfileSpecificFilterCollection)

The CIM_FilterCollection (ProfileSpecificFilterCollection) models profile-specific filter collections. Profile-specific filter collection address the needs of clients requiring notifications about events reported by the indications specified in a particular profile. Profile specific filter collections are static filter collections. The defined coverage of a profile-specific filter collection covers all indications of a particular type (that is, all alert indications or all lifecycle indications) defined in a profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 512 describes class CIM_FilterCollection (ProfileSpecificFilterCollection).

Table 512 - SMI Referenced Properties/Methods for CIM_FilterCollection (ProfileSpecificFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The CollectionName property shall be of the form: OrgID ":" RegisteredName ":" "ProfileSpecified" Type "IndicationFilterCollection" Where Type shall be "Alert" in case the represented profile-specific filter collection covers all alert indications, and shall be "Lifecycle" in case the represented profile-specific filter collection covers all lifecycle indications.

50.6.13CIM_FilterCollection (StaticFilterCollection)

The Static filter collection is an ABSTRACT class (for common properties).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 513 describes class CIM_FilterCollection (StaticFilterCollection).

Table 513 - SMI Referenced Properties/Methods for CIM_FilterCollection (StaticFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key: See CIM schema description.
CollectionName		Mandatory	The value of the CollectionName property shall be formatted as follows: OrgID ":" RegisteredName ":" UniqueID.

50.6.14CIM_FilterCollectionSubscription (CollectionSubscription)

SNIA defines this class as Experimental. The class definition specializes the CIM_FilterCollectionSubscription definition in the Indications profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Optional

Table 514 describes class CIM_FilterCollectionSubscription (CollectionSubscription).

Table 514 - SMI Referenced Properties/Methods for CIM_FilterCollectionSubscription (CollectionSubscription)

Properties	Flags	Requirement	Description & Notes
OnFatalErrorPolicy		Mandatory	See the OnFatalErrorPolicy definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
OtherOnFatalErrorPolicy		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.OnFatalErrorPolicy = 1. See the OtherOnFatalErrorPolicy definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
FailureTriggerTimeInterval		Mandatory	See the FailureTriggerTimeInterval definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
SubscriptionState		Mandatory	See the SubscriptionState definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
OtherSubscriptionState		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.SubscriptionState = 1. See the OtherSubscriptionState definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
RepeatNotificationPolicy		Mandatory	See the RepeatNotificationPolicy definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
RepeatNotificationInterval		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 3. or Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. See the RepeatNotificationInterval definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
RepeatNotificationGap		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. See the RepeatNotificationGap definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
RepeatNotificationCount		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 3. or Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. See the RepeatNotificationCount definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
Filter (overridden)		Mandatory	Reference to the FilterCollection.
Handler (overridden)		Mandatory	Reference to the ListenerDestination.

50.6.15CIM_HostedCollection (Hosted Client Filter Collection)

Experimental. This associates a client defined FilterCollection to the system in the referencing profile.

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections).

Table 515 describes class CIM_HostedCollection (Hosted Client Filter Collection).

Table 515 - SMI Referenced Properties/Methods for CIM_HostedCollection (Hosted Client Filter Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the FilterCollection.
Antecedent		Mandatory	Reference to the conforming System that hosts the collection.

50.6.16CIM_HostedCollection (Hosted Global FilterCollection or a Profile Specific FilterCollection)

Experimental. This associates a Global FilterCollection or a Profile Specific FilterCollection to the system in the referencing profile.

Requirement: Support for instances of CIM_FilterCollection (GlobalFilterCollection) or CIM_FilterCollection (ProfileSpecificFilterCollection).

Table 516 describes class CIM_HostedCollection (Hosted Global FilterCollection or a Profile Specific FilterCollection).

Table 516 - SMI Referenced Properties/Methods for CIM_HostedCollection (Hosted Global FilterCollection or a Profile Specific FilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the Global FilterCollection or a Profile Specific FilterCollection.
Antecedent		Mandatory	Reference to the System that hosts the collection.

50.6.17CIM_HostedCollection (Hosted Predefined Filter Collection)

Experimental. This associates a Predefined FilterCollection to the system in the referencing profile.

Requirement: Optional

Table 517 describes class CIM_HostedCollection (Hosted Predefined Filter Collection).

Table 517 - SMI Referenced Properties/Methods for CIM_HostedCollection (Hosted Predefined Filter Collection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the Predefined FilterCollection.
Antecedent		Mandatory	Reference to the conforming System that hosts the collection.

50.6.18CIM_HostedCollection (System to predefined FilterCollection)

Experimental.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 518 describes class CIM_HostedCollection (System to predefined FilterCollection).

Table 518 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined FilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for the Indications implementation.
Antecedent		Mandatory	Reference to the System of the referencing profile.

50.6.19CIM_HostedService (HostedIndicationService)

CIM_HostedService is used to relate the CIM_IndicationService instance to its scoping CIM_System instance.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 519 describes class CIM_HostedService (HostedIndicationService).

Table 519 - SMI Referenced Properties/Methods for CIM_HostedService (HostedIndicationService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Value shall reference the CIM_System instance representing the indication system hosting the indication service identified by the value of Dependent.
Dependent		Mandatory	Value shall reference an CIM_IndicationService instance representing an indication service.

50.6.20CIM_HostedService (Indication Config Service to System)

Experimental. This associates the IndicationConfigurationService to the System in the referencing profile.

Requirement: Mandatory

Table 520 describes class CIM_HostedService (Indication Config Service to System).

Table 520 - SMI Referenced Properties/Methods for CIM_HostedService (Indication Config Service to System)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting system.
Dependent		Mandatory	The Indication configuration service hosted on the system.

50.6.21CIM_IndicationFilter (DynamicIndicationFilter)

Indication filters shall be represented by CIM_IndicationFilter instances. If a particular indication filter is represented, it shall be represented by exactly one IndicationFilter instance in the Interop namespace. In addition, it may be represented by other IndicationFilter instances in application namespaces; these instances shall have the same key properties as the one in the Interop namespace.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: The IndicationService instance has the value TRUE is set for FilterCreationEnabled.

Table 521 describes class CIM_IndicationFilter (DynamicIndicationFilter).

Table 521 - SMI Referenced Properties/Methods for CIM_IndicationFilter (DynamicIndicationFilter)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system.
CreationClassName		Mandatory	Shall be populated by the WBEM server with the name of the class of which this is an instance.
SystemName		Mandatory	Shall be populated by the WBEM server with the name of the scoping system.
Name		Mandatory	The Name property shall be formatted as follows: <OrgID> ':' <RegisteredName> ':' <uniqueID>. For referencing profiles owned by DMTF, the value shall be formatted as follows:'DMTF:' <RegisteredName> ':' <uniqueID>.
Query		Mandatory	The value of the Query property be a properly formed query statement conformant to the requirements of the query language identified by the value of the QueryLanguage property that states the coverage of the indication filter.
QueryLanguage		Mandatory	The value of the QueryLanguage property shall identify the query language in which the query statement exposed by the value of the Query property is expressed.
SourceNamespaces		Mandatory	A non-Null value of this property is required for IndicationFilter instances in the Interop namespace; for IndicationFilter instances in other namespaces it is optional. If not Null, the value of the SourceNamespaces[ ] array property shall contain the names of local namespaces that are considered as potential indication origin namespaces during indication filtering. The value shall not be an empty array. It is not required that the local namespaces identified by elements of value of the SourceNamespaces[ ] array property exist. If a non-existing local namespace is identified, no indications can originate out of that non-existing namespace; consequently, that element does not have an effect on indication filtering. However, if the identified namespace is added to the implementation at a later point in time, per the requirements of indications originating out of that namespace are to be considered for indication filtering from then on. The value elements of the SourceNamespaces[ ] array property shall be formatted using the format that the implementation uses for value of the Name property in instances of the CIM_Namespace class that represent namespaces.
IndividualSubscriptionSupported		Mandatory	The value of the IndividualSubscriptionSupported property shall be True if the FilterSubscription feature is implemented, and is supported for the represented indication filter; otherwise, the value shall be False.

50.6.22CIM_IndicationFilter (GlobalIndicationFilter)

The CIM_IndicationFilter (GlobalIndicationFilter) would be implemented if the WBEM server supports the delivery of indications. Global indication filters are a specialization of static indication filters.

Created By: Static

Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 522 describes class CIM_IndicationFilter (GlobalIndicationFilter).

Table 522 - SMI Referenced Properties/Methods for CIM_IndicationFilter (GlobalIndicationFilter)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'DMTF:Indications:GlobalAlertIndicationFilter', 'DMTF:Indications:GlobalInstCreationIndicationFilter', 'DMTF:Indications:GlobalInstDeletionIndicationFilter' or 'DMTF:Indications:GlobalInstModificationIndicationFilter'.
Query		Mandatory	This shall be 'SELECT * FROM CIM_AlertIndication', 'Select * from CIM_InstCreation', 'Select * from CIM_InstDeletion' or 'Select * from CIM_InstModification'.
QueryLanguage		Mandatory	See the QueryLanguage definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SourceNamespaces		Mandatory	See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
IndividualSubscriptionSupported		Mandatory	See the IndividualSubscriptionSupported definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

50.6.23CIM_IndicationFilter (IndicationSpecificIndicationFilter)

The CIM_IndicationFilter (IndicationSpecificIndicationFilter) would be implemented if indications defined in a referencing profile or in this profile are implemented.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 523 describes class CIM_IndicationFilter (IndicationSpecificIndicationFilter).

Table 523 - SMI Referenced Properties/Methods for CIM_IndicationFilter (IndicationSpecificIndicationFilter)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	OVERRIDE: The value of the Name property shall be formatted as defined by the following ABNF rule: OrgID ":" RegisteredName ":" IndicationAdaptationName "Filter" ["/" MessageIdentification] IndicationAdaptationName shall be the name of the indication adaptation defined in the profile. The MessageIdentification suffix only applies for the representation of indication-specific indication filters covering alert indications modeled by an adaptation based on the AlertIndication adaptation. In this case for each alert indication defined by an alert message reference in the profile, a specific IndicationSpecificIndicationFilter instance is defined, where MessageIdentification shall be set to the OwningEntity concatenated with the MessageID.
Query		Mandatory	OVERRIDE: The value of the Query property shall be the event definition query statement of the indication adaptation defined in the referencing profile or in this profile.
QueryLanguage		Mandatory	See the QueryLanguage definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SourceNamespaces		Mandatory	See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
IndividualSubscriptionSupported		Mandatory	See the IndividualSubscriptionSupported definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

50.6.24CIM_IndicationFilter (ListenerDestinationRemovalIndication)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a CIM_ListenerDestination instance. This would typically occur as a result of an invocation of DeleteInstance operation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 524 describes class CIM_IndicationFilter (ListenerDestinationRemovalIndication).

Table 524 - SMI Referenced Properties/Methods for CIM_IndicationFilter (ListenerDestinationRemovalIndication)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'DMTF:Indications:ListenerDestinationRemovalIndicationFilter'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ListenerDestination.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.28 CIM_IndicationFilter (pre-defined).

50.6.25CIM_IndicationFilter (StaticIndicationFilter)

Static indication filters are uniquely identified by means of a naming convention that involves the name of the organization defining the profile, the name of this profile and a string that is required to be unique within the implementation of this profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 525 describes class CIM_IndicationFilter (StaticIndicationFilter).

Table 525 - SMI Referenced Properties/Methods for CIM_IndicationFilter (StaticIndicationFilter)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system.
CreationClassName		Mandatory	Shall be populated by the WBEM server with the name of the class of which this is an instance.
SystemName		Mandatory	Shall be populated by the WBEM server with the name of the scoping system.

Table 525 - SMI Referenced Properties/Methods for CIM_IndicationFilter (StaticIndicationFilter)

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	The Name property shall be formatted as follows: <OrgID> ':' <RegisteredName> ':' <uniqueID>. For referencing profiles owned by DMTF, the value shall be formatted as follows:'DMTF:' <RegisteredName> ':' <uniqueID>.
Query		Mandatory	The value of the Query property be a properly formed query statement conformant to the requirements of the query language identified by the value of the QueryLanguage property that states the coverage of the indication filter.
QueryLanguage		Mandatory	The value of the QueryLanguage property shall identify the query language in which the query statement exposed by the value of the Query property is expressed. In referencing profiles owned by DMTF, the value shall be "DMTF:CQL", thereby requiring CQL as the query language.
SourceNamespaces		Mandatory	A non-Null value of this property is required for IndicationFilter instances in the Interop namespace; for IndicationFilter instances in other namespaces it is optional. If not Null, the value of the SourceNamespaces[ ] array property shall contain the names of local namespaces that are considered as potential indication origin namespaces during indication filtering. The value shall not be an empty array. It is not required that the local namespaces identified by elements of value of the SourceNamespaces[ ] array property exist. If a non-existing local namespace is identified, no indications can originate out of that non-existing namespace; consequently, that element does not have an effect on indication filtering. However, if the identified namespace is added to the implementation at a later point in time, per the requirements of indications originating out of that namespace are to be considered for indication filtering from then on. The value elements of the SourceNamespaces[ ] array property shall be formatted using the format that the implementation uses for value of the Name property in instances of the CIM_Namespace class that represent namespaces.
IndividualSubscriptionSupported		Mandatory	The value of the IndividualSubscriptionSupported property shall be True if the FilterSubscription feature is implemented, and is supported for the represented indication filter; otherwise, the value shall be False.

50.6.26CIM_IndicationFilter (SubscriptionRemovalIndication)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a CIM_IndicationSubscription or a CIM_FilterCollectionSubscription instance. This would typically occur as a result of an invocation of DeleteInstance operation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 526 describes class CIM_IndicationFilter (SubscriptionRemovalIndication).

Table 526 - SMI Referenced Properties/Methods for CIM_IndicationFilter (SubscriptionRemovalIndication)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'DMTF:Indications:SubscriptionRemovalIndicationFilter'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_AbstractIndicationSubscription.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.28 CIM_IndicationFilter (pre-defined).

50.6.27CIM_IndicationFilter (client defined)

CIM_IndicationFilter (Client Defined) is a specialization of CIM_IndicationFilter (DynamicIndicationFilter). CIM_IndicationFilter instances that are 'client defined' are IndicationFilters that are be created by a client using CreateInstance. If a profile implementation can support client defined IndicationFilters, the implementation would support 'client defined' IndicationFilter instances. The implementation shall support 'client defined' filters that are defined by SMI-S profile as mandatory, but may also support additional filters supported by the implementation.

CIM_IndicationFilter is subclassed from CIM_ManagedElement.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Optional

Table 527 describes class CIM_IndicationFilter (client defined).

Table 527 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.21 CIM_IndicationFilter (DynamicIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.21 CIM_IndicationFilter (DynamicIndicationFilter).

Table 527 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

Properties	Flags	Requirement	Description & Notes
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.21 CIM_IndicationFilter (DynamicIndicationFilter).
Name		Mandatory	See the Name definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.21 CIM_IndicationFilter (DynamicIndicationFilter).
SourceNamespace	N	Optional	Deprecated. ADD: For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Mandatory	See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.21 CIM_IndicationFilter (DynamicIndicationFilter).
Query		Mandatory	See the Query definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.21 CIM_IndicationFilter (DynamicIndicationFilter).
QueryLanguage		Mandatory	OVERRIDE: For SNIA profiles, this shall be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName		Optional	ADD: A Client Defined user friendly string that identifies the Indication Filter.
IndividualSubscriptionSupported		Mandatory	Experimental. See the IndividualSubscriptionSupported definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.21 CIM_IndicationFilter (DynamicIndicationFilter).

50.6.28CIM_IndicationFilter (pre-defined)

CIM_IndicationFilter (pre-defined) is a specialization of CIM_IndicationFilter (StaticIndicationFilter). CIM_IndicationFilter instances that are 'pre-defined' are IndicationFilters that are populated automatically by the profile provider. If a profile implementation cannot support client defined IndicationFilters, the implementation can populate its model with 'pre-defined' IndicationFilter instances. 'Pre-defined' filters shall include those that are required by the profile, but may also contain additional filters supported by the implementation.

CIM_IndicationFilter is subclassed from CIM_ManagedElement.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 528 describes class CIM_IndicationFilter (pre-defined).

Table 528 - SMI Referenced Properties/Methods for CIM_IndicationFilter (pre-defined)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	See the Name definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SourceNamespace	N	Optional	Deprecated. ADD: For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Mandatory	See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	See the Query definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
QueryLanguage		Mandatory	OVERRIDE: For SNIA profiles, this shall be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	ADD: SMI-S does not specify this property for pre-defined IndicationFilters.
IndividualSubscriptionSupported		Mandatory	Experimental. See the IndividualSubscriptionSupported definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

50.6.29CIM_IndicationService (IndicationService)

Within an implementation there shall be exactly one indication service. That indication service shall be represented by an IndicationService instance in the Interop namespace.

Several of the properties of this class are marked as modifiable (Flag="M"). To determine if these properties may be modified see CIM_IndicationServiceCapabilities.

Created By: Static

Modified By: ModifyInstance

Deleted By: Static

Requirement: Mandatory

Table 529 describes class CIM_IndicationService (IndicationService).

Table 529 - SMI Referenced Properties/Methods for CIM_IndicationService (IndicationService)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
Name		Mandatory	Key.
FilterCreationEnabled	M	Mandatory	A value of False indicates that the Dynamic Indicatoin Filters feature is not supported, a value of True indicates that the feature is supported.
DeliveryRetryAttempts	M	Mandatory	The number of retry attempts after the initial attempt.
DeliveryRetryInterval	M	Mandatory	The minimal time interval in seconds that the implementation waits before delivering an indication to a particular listener destination after a previous delivery failure.
SubscriptionRemovalAction	M	Mandatory	The removal action for subscriptions after two failed indication deliveries where the time interval between the failed deliveries, without any intermediate successful indication delivery, exceeds the timeout reflected by the value of the SubscriptionRemovalTimeInterval property.
SubscriptionRemovalTimeInterval	M	Mandatory	The minimum time interval that implementations shall wait after two failed indication deliveries without any intermediate successful indication delivery, before performing the activity designated by the value of the SubscriptionRemovalAction property.

50.6.30CIM_IndicationServiceCapabilities (IndicationServiceCapabilities)

SNIA defines this as Mandatory. DMTF defines this class as Optional. The class definition specializes the CIM_IndicationServiceCapabilities definition in the Indications profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 530 describes class CIM_IndicationServiceCapabilities (IndicationServiceCapabilities).

Table 530 - SMI Referenced Properties/Methods for CIM_IndicationServiceCapabilities (IndicationService-Capabilities)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
FilterCreationEnabledIsSettable		Mandatory	Value shall indicate whether the implementation supports modification of the FilterCreationEnabled property of the associated IndicationService instance.
DeliveryRetryAttemptsIsSettable		Mandatory	Value shall indicate whether the implementation supports modification of the DeliveryRetryAttempts property of the associated IndicationService instance.

Table 530 - SMI Referenced Properties/Methods for CIM_IndicationServiceCapabilities (IndicationService-Capabilities)

Properties	Flags	Requirement	Description & Notes
DeliveryRetryIntervalIsSettable		Mandatory	Value shall indicate whether the implementation supports modification of the DeliveryRetryInterval property of the associated IndicationService instance.
SubscriptionRemovalActionIsSettable		Mandatory	Value shall indicate whether the implementation supports modification of the SubscriptionRemovalAction property of the associated IndicationService instance.
SubscriptionRemovalTimeIntervalIsSettable		Mandatory	Value shall indicate whether the implementation supports modification of the SubscriptionRemovalTimeInterval property of the associated IndicationService instance.
MaxListenerDestinations		Mandatory	Value shall indicate the maximum number of listener destinations.
MaxActiveSubscriptions		Mandatory	Value shall indicate the maximum number of active subscriptions.
SubscriptionsPersisted		Mandatory	Value shall indicate whether subscriptions are persisted across restarts of the indication service.

50.6.31CIM_IndicationServiceSettingData (IndicationServiceInitialSettings)

CIM_IndicationServiceSettingData models initial settings for indication services. The initial settings of an indication service are the settings that apply at the point in time when the WBEM server hosting the indication service initially starts up the indication service.

This class would be implemented if the initial settings of the indication service deviate from the default settings for properties specified by the DMTF Indications Profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 531 describes class CIM_IndicationServiceSettingData (IndicationServiceInitialSettings).

Table 531 - SMI Referenced Properties/Methods for CIM_IndicationServiceSettingData (IndicationServiceInitialSettings)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
FilterCreationEnabled		Mandatory	Value shall be the default value for the FilterCreationEnabled property in the associated IndicationService instance.
DeliveryRetryAttempts		Mandatory	Value shall be the default value for the DeliveryRetryAttempts property in the associated IndicationService instance.
DeliveryRetryInterval		Mandatory	Value shall be the default value for the DeliveryRetryInterval property in the associated IndicationService instance.
SubscriptionRemovalAction		Mandatory	Value shall be the default value for the SubscriptionRemovalAction property in the associated IndicationService instance.
SubscriptionRemovalTimeInterval		Mandatory	Value shall be the default value for the SubscriptionRemovalTimeInterval property in the associated IndicationService instance.

50.6.32CIM_IndicationSubscription (FilterSubscription)

This association defines a subscription to a specific IndicationFilter instance by a specific indication handler (as represented by a ListenerDestinationCIMXML or CIM_ListenerDestinationWSManagement instance)

SMI-S defines this class as Mandatory (DMTF defines it as conditional on the existence of an IndicationFilter associated to the IndicationService)

A CIM_IndicationSubscription is subclassed from CIM_AbstractIndicationSubscription. The class definition specializes the CIM_IndicationSubscription definition in the Indications profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Optional

Table 532 describes class CIM_IndicationSubscription (FilterSubscription).

Table 532 - SMI Referenced Properties/Methods for CIM_IndicationSubscription (FilterSubscription)

Properties	Flags	Requirement	Description & Notes
OnFatalErrorPolicy		Mandatory	See the OnFatalErrorPolicy definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
OtherOnFatalErrorPolicy		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.OnFatalErrorPolicy = 1. See the OtherOnFatalErrorPolicy definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
FailureTriggerTimeInterval		Mandatory	See the FailureTriggerTimeInterval definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
SubscriptionState		Mandatory	See the SubscriptionState definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
OtherSubscriptionState		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.SubscriptionState = 1. See the OtherSubscriptionState definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
RepeatNotificationPolicy		Mandatory	See the RepeatNotificationPolicy definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
RepeatNotificationInterval		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 3. or Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. See the RepeatNotificationInterval definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
RepeatNotificationGap		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. See the RepeatNotificationGap definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).

Table 532 - SMI Referenced Properties/Methods for CIM_IndicationSubscription (FilterSubscription)

Properties	Flags	Requirement	Description & Notes
RepeatNotificationCount		Conditional	Conditional requirement: Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 3. or Support for the CIM_IndicationSubscription.RepeatNotificationPolicy = 4. See the RepeatNotificationCount definition in section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 50.6.1 CIM_AbstractIndicationSubscription (AbstractSubscription).
LastIndicationIdentifier (added)		Optional	Experimental. The IndicationIdentifier of the last indication produced for this subscription regardless if that indication were delivered **SMI-S defines this property for backward compatibility (DMTF does not define a use of this property).**
LastIndicationProductionDateTime (added)		Optional	Experimental. The date and time of the production of the last indication produced for this subscription regardless if that indication were delivered **SMI-S defines this property for backward compatibility (DMTF does not define a use of this property).**
Filter (overridden)		Mandatory	
Handler (overridden)		Mandatory	

50.6.33CIM_InstCreation

CIM_InstCreation notifies a handler when a new instance (of a class defined in the Filter QueryString) is created.

CIM_InstCreation is subclassed from CIM_InstIndication.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 533 describes class CIM_InstCreation.

Table 533 - SMI Referenced Properties/Methods for CIM_InstCreation

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Mandatory	See CIM schema definition.
IndicationTime		Mandatory	See CIM schema definition.
IndicationFilterName		Mandatory	The value of the IndicationFilterName property shall contain the name of the indication emitter that the indication passed before being delivered to the listeners subscribed to that indication emitter. For indication filters, the name is exposed by the value of the Name property in representing IndicationFilter instances. For filter collections, the name is exposed by the value of the CollectionName property in representing FilterCollection instances.
SequenceContext		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceContext property shall contain the sequence context portion of the sequence identifier; see the CIM schema description for a recommended format and structure. The value of the SequenceContext property shall be identical for all indications delivered to a particular listener while the representing listener destination is defined within the implementation. This definition is required to remain effective over restarts of the implementation environment.

Table 533 - SMI Referenced Properties/Methods for CIM_InstCreation

Properties	Flags	Requirement	Description & Notes
SequenceNumber		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceNumber property shall contain the sequence number portion of the sequence identifier. The value of the SequenceNumber property shall be unique for each indication delivered to a particular listener while the listener destination is defined within the implementation; see the CIM schema description for required value constraints. This definition is required to remain effective over restarts of the implementation environment.
SourceInstance		Mandatory	The value of the SourceInstance property shall be an embedded instance of the class selected in the query statement defining the event. The embedded instance shall be a copy of the instance for that the lifecycle indication is reported. If the query statement specifies a specific selection of properties (other than '*'), then the set of properties contained in the embedded instance shall be limited to those selected; otherwise, the embedded instance shall at least contain values for each of the properties required by the related adaptation of the selected class in the same referencing profile.
SourceInstanceModelPath		Mandatory	The value of the SourceInstanceModelPath property shall refer to the same instance that is copied as an embedded instance through the value of the SourceInstance property.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.

50.6.34CIM_InstDeletion

CIM_InstDeletion notifies a handler when a new instance (of a class defined in the Filter QueryString) is deleted.

CIM_InstDeletion is subclassed from CIM_InstIndication.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 534 describes class CIM_InstDeletion.

Table 534 - SMI Referenced Properties/Methods for CIM_InstDeletion

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Mandatory	See CIM schema definition.
IndicationTime		Mandatory	See CIM schema definition.
IndicationFilterName		Mandatory	The value of the IndicationFilterName property shall contain the name of the indication emitter that the indication passed before being delivered to the listeners subscribed to that indication emitter. For indication filters, the name is exposed by the value of the Name property in representing IndicationFilter instances. For filter collections, the name is exposed by the value of the CollectionName property in representing FilterCollection instances.

Table 534 - SMI Referenced Properties/Methods for CIM_InstDeletion

Properties	Flags	Requirement	Description & Notes
SequenceContext		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceContext property shall contain the sequence context portion of the sequence identifier; see the CIM schema description for a recommended format and structure. The value of the SequenceContext property shall be identical for all indications delivered to a particular listener while the representing listener destination is defined within the implementation. This definition is required to remain effective over restarts of the implementation environment.
SequenceNumber		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceNumber property shall contain the sequence number portion of the sequence identifier. The value of the SequenceNumber property shall be unique for each indication delivered to a particular listener while the listener destination is defined within the implementation; see the CIM schema description for required value constraints. This definition is required to remain effective over restarts of the implementation environment.
SourceInstance		Mandatory	The value of the SourceInstance property shall be an embedded instance of the class selected in the query statement defining the event. The embedded instance shall be a copy of the instance for that the lifecycle indication is reported. If the query statement specifies a specific selection of properties (other than '*'), then the set of properties contained in the embedded instance shall be limited to those selected; otherwise, the embedded instance shall at least contain values for each of the properties required by the related adaptation of the selected class in the same referencing profile.
SourceInstanceModelPath		Mandatory	The value of the SourceInstanceModelPath property shall refer to the same instance that is copied as an embedded instance through the value of the SourceInstance property.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.

50.6.35CIM_InstIndication (LifecycleIndication)

Requirement: Optional

Table 535 describes class CIM_InstIndication (LifecycleIndication).

Table 535 - SMI Referenced Properties/Methods for CIM_InstIndication (LifecycleIndication)

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Mandatory	See CIM schema definition.
IndicationTime	N	Mandatory	See CIM schema definition.
IndicationFilterName		Mandatory	Experimental. The value of the IndicationFilterName property shall contain the name of the indication emitter that the indication passed before being delivered to the listeners subscribed to that indication emitter. For indication filters, the name is exposed by the value of the Name property in representing IndicationFilter instances. For filter collections, the name is exposed by the value of the CollectionName property in representing FilterCollection instances.

Table 535 - SMI Referenced Properties/Methods for CIM_InstIndication (LifecycleIndication)

Properties	Flags	Requirement	Description & Notes
SequenceContext		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceContext property shall contain the sequence context portion of the sequence identifier; see the CIM schema description for a recommended format and structure. The value of the SequenceContext property shall be identical for all indications delivered to a particular listener while the representing listener destination is defined within the implementation. This definition is required to remain effective over restarts of the implementation environment.
SequenceNumber		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceNumber property shall contain the sequence number portion of the sequence identifier. The value of the SequenceNumber property shall be unique for each indication delivered to a particular listener while the listener destination is defined within the implementation; see the CIM schema description for required value constraints. This definition is required to remain effective over restarts of the implementation environment.
SourceInstance		Mandatory	The value of the SourceInstance property shall be an embedded instance of the class selected in the query statement defining the event. The embedded instance shall be a copy of the instance for that the lifecycle indication is reported. If the query statement specifies a specific selection of properties (other than '*'), then the set of properties contained in the embedded instance shall be limited to those selected; otherwise, the embedded instance shall at least contain values for each of the properties required by the related adaptation of the selected class in the same referencing profile.
SourceInstanceModelPath		Mandatory	The value of the SourceInstanceModelPath property shall refer to the same instance that is copied as an embedded instance through the value of the SourceInstance property.

50.6.36CIM_InstModification

CIM_InstModification notifies a handler when a new instance (of a class defined in the Filter QueryString) is modified or changed. To avoid undue effort on Providers, the select list (in the query filter) for this indication should only call for properties that are needed.

CIM_InstModification is subclassed from CIM_InstIndication.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 536 describes class CIM_InstModification.

Table 536 - SMI Referenced Properties/Methods for CIM_InstModification

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Mandatory	See CIM schema definition.
IndicationTime		Mandatory	See CIM schema definition.

Table 536 - SMI Referenced Properties/Methods for CIM_InstModification

Properties	Flags	Requirement	Description & Notes
IndicationFilterName		Mandatory	The value of the IndicationFilterName property shall contain the name of the indication emitter that the indication passed before being delivered to the listeners subscribed to that indication emitter. For indication filters, the name is exposed by the value of the Name property in representing IndicationFilter instances. For filter collections, the name is exposed by the value of the CollectionName property in representing FilterCollection instances.
SequenceContext		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceContext property shall contain the sequence context portion of the sequence identifier; see the CIM schema description for a recommended format and structure. The value of the SequenceContext property shall be identical for all indications delivered to a particular listener while the representing listener destination is defined within the implementation. This definition is required to remain effective over restarts of the implementation environment.
SequenceNumber		Conditional	Conditional requirement: The CIM_IndicationService.DeliveryRetryAttempts is greater than 0. The value of the SequenceNumber property shall contain the sequence number portion of the sequence identifier. The value of the SequenceNumber property shall be unique for each indication delivered to a particular listener while the listener destination is defined within the implementation; see the CIM schema description for required value constraints. This definition is required to remain effective over restarts of the implementation environment.
SourceInstance		Mandatory	The value of the SourceInstance property shall be an embedded instance of the class selected in the query statement defining the event. The embedded instance shall be a copy of the instance for that the lifecycle indication is reported. If the query statement specifies a specific selection of properties (other than '*'), then the set of properties contained in the embedded instance shall be limited to those selected; otherwise, the embedded instance shall at least contain values for each of the properties required by the related adaptation of the selected class in the same referencing profile.
SourceInstanceModelPath		Mandatory	The value of the SourceInstanceModelPath property shall refer to the same instance that is copied as an embedded instance through the value of the SourceInstance property.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.
PreviousInstance		Optional	A copy of the 'previous' instance whose change generated the Indication. PreviousInstance contains 'older' values of an instance's properties (as compared to SourceInstance), selected by the IndicationFilter's Query.

50.6.37CIM_ListenerDestination (ListenerDestination)

CIM_ListenerDestination instances shall be represented by ListenerDestination instances. If a particular listener destination is represented, it shall be represented by exactly one ListenerDestination instance in the Interop namespace. In addition, it may be represented by other ListenerDestination instances in application namespaces; these instances shall have the same key properties as the one in the Interop namespace.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Mandatory

Table 537 describes class CIM_ListenerDestination (ListenerDestination).

Table 537 - SMI Referenced Properties/Methods for CIM_ListenerDestination (ListenerDestination)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
Name		Mandatory	Key.
PersistenceType	N	Mandatory	The value of the PersistenceType property shall describe the durability of the represented listener destination. The property values shall be constrained to 3 (Transient), 2 (Permanent), and NULL.
ElementName		Mandatory	See CIM schema description.
Destination	N	Mandatory	The value of the Destination property shall identify the listener represented by the listener destination. A value of NULL for the Destination property indicates a free listener destination.
Protocol		Mandatory	See CIM schema description.

50.6.38CIM_ListenerDestinationCIMXML (Indication Handler)

Deprecated.

SMI-S defines the specific subclasses of CIM_ListenerDestination supported by SMI-S (DMTF only defines CIM_ListenerDestination)

CIM_ListenerDestinationCIMXML is subclassed from CIM_ListenerDestination.

Created By: CreateInstance
Modified By: Static
Deleted By: DeleteInstance
Requirement: Mandatory

Table 538 describes class CIM_ListenerDestinationCIMXML (Indication Handler).

Table 538 - SMI Referenced Properties/Methods for CIM_ListenerDestinationCIMXML (Indication Handler)

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A client defined user friendly string that identifies the CIMXML Listener destination.
SystemCreationClassName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
SystemName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
CreationClassName		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.
Name		Mandatory	Shall be populated by the WBEM server with the class name of the scoping system. If the client supplies a value, the WBEM server shall ignore it.

Table 538 - SMI Referenced Properties/Methods for CIM_ListenerDestinationCIMXML (Indication Handler)

Properties	Flags	Requirement	Description & Notes
PersistenceType		Mandatory	For SMI-S, this shall be 2|3 ('permanent' | 'transient').
Destination		Mandatory	The destination URL to which CIM-XML Export Messages are to be delivered. The scheme prefix shall be consistent with the DMTF CIM-XML specifications.If a scheme prefix is not specified, the scheme \http:\'shallbeassumed.'
Protocol		Mandatory	For CIM-XML, this shall be '2' (CIM-XML).

50.6.39CIM_MemberOfCollection (Client Defined Filter Collection to Filters)

Experimental. This associates a client defined FilterCollection to the Filters in the collection.

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='6' (Client Defined Filter Collections).

Table 539 describes class CIM_MemberOfCollection (Client Defined Filter Collection to Filters).

Table 539 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Client Defined Filter Collection to Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Client Defined FilterCollection.
Member		Mandatory	Reference to an IndicationFilter or FilterCollection.

50.6.40CIM_MemberOfCollection (FilterCollectionInFilterCollection)

Each FilterCollection instance representing a filter collection that contains filter collections shall be associated through a CIM_MemberOfCollection (FilterCollectionInFilterCollection) instance with each of the FilterCollection instances representing a contained filter collection.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 540 describes class CIM_MemberOfCollection (FilterCollectionInFilterCollection).

Table 540 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (FilterCollectionInFilterCollection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Value shall reference a FilterCollection instance representing a filter collection.
Member		Mandatory	Value shall reference a FilterCollection instance representing a contained filter collection.

50.6.41CIM_MemberOfCollection (IndicationFilterInFilterCollection)

Each FilterCollection instance representing a filter collection that contains indication filters shall be associated through an CIM_MemberOfCollection (IndicationFilterInFilterCollection) instance with each of the IndicationFilter instances representing contained indication filters.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 541 describes class CIM_MemberOfCollection (IndicationFilterInFilterCollection).

Table 541 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (IndicationFilterInFilterCollection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Value shall reference a FilterCollection instance representing a filter collection.
Member		Mandatory	Value shall reference an IndicationFilter instance representing a contained indication filters.

50.6.42CIM_MemberOfCollection (Predefined Filter Collection to Indications Filters)

Experimental. This associates the Indications predefined FilterCollection to the predefined Filters supported by the implementation.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 542 describes class CIM_MemberOfCollection (Predefined Filter Collection to Indications Filters).

Table 542 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Indications Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Indications predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Indications implementation.

50.6.43CIM_OwningCollectionElement (IndicationServiceOfFilterCollection)

Each FilterCollection instance shall be associated with exactly one IndicationService instance through a CIM_OwningCollectionElement (IndicationServiceOfFilterCollection) instance.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 543 describes class CIM_OwningCollectionElement (IndicationServiceOfFilterCollection).

Table 543 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement (IndicationServiceOfFilterCollection)

Properties	Flags	Requirement	Description & Notes
OwningElement		Mandatory	Value shall reference the IndicationService instance representing the indication service owning the filter collection represented by the value of OwnedElement.
OwnedElement		Mandatory	Value shall reference a FilterCollection instance.

50.6.44CIM_ServiceAffectsElement (IndicationServiceOfIndicationFilter)

Each IndicationFilter instance representing an indication filter shall be associated through an CIM_ServiceAffectsElement (IndicationServiceOfIndicationFilter) instance with the IndicationService instance representing the indication service that manages the indication filter.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 544 describes class CIM_ServiceAffectsElement (IndicationServiceOfIndicationFilter).

Table 544 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (IndicationServiceOfIndicationFilter)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Value shall reference the IndicationService instance representing the indication service managing the indication filter represented by the value of AffectedElement.
AffectedElement		Mandatory	Value shall reference an IndicationFilter instance representing a managed indication filter.

50.6.45CIM_ServiceAffectsElement (IndicationServiceOfListenerDestination)

Each ListenerDestination instance representing a listener destination shall be associated through an IndicationServiceOfListenerDestination instance with the IndicationService instance representing the indication service that manages the listener destination.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 545 describes class CIM_ServiceAffectsElement (IndicationServiceOfListenerDestination).

Table 545 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (IndicationServiceOfListenerDestination)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Value shall reference the IndicationService instance representing the indication service managing the listener destination represented by the value of AffectedElement.
AffectedElement		Mandatory	Value shall reference an ListenerDestination instance representing a managed listener destination.

50.6.46SNIA_IndicationConfigurationCapabilities (IndicationConfigurationCapabilities)

Experimental. This is the capabilities of the implementation of indications.

Requirement: Mandatory

Table 546 describes class SNIA_IndicationConfigurationCapabilities (IndicationConfigurationCapabilities).

Table 546 - SMI Referenced Properties/Methods for SNIA_IndicationConfigurationCapabilities (IndicationConfigurationCapabilities)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Optional	This is a user friendly name of the capabilities instance.
SupportedFeatures		Mandatory	This may be any or all of the following values: '2' (none), '3' (Predefined Filters), '4' (Client Defined Filters), '5' (Predefined Filter Collections), '6' (Client Defined Filter Collections) or '7' (Semi-fixed Indication Filters).
SupportedSynchronousActions		Mandatory	This shall be '2' (none), '3' (Test Listener), '4' ("Create and Subscribe) or '5' (Filter Collection Methods).

50.6.47SNIA_IndicationConfigurationService (IndicationConfigurationService)

Experimental. This is the indication services of the implementation.

Requirement: Mandatory

Table 547 describes class SNIA_IndicationConfigurationService (IndicationConfigurationService).

Table 547 - SMI Referenced Properties/Methods for SNIA_IndicationConfigurationService (IndicationConfigurationService)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	
TestListener()		Optional	A method for testing if the listener can receive indications.

Table 547 - SMI Referenced Properties/Methods for SNIA_IndicationConfigurationService (IndicationConfigurationService)

Properties	Flags	Requirement	Description & Notes
CreateAndSubscribe()		Optional	A method for creating a Filter and subscribing to it.
CreateFilterCollection()		Optional	A method for creating a FilterCollection and adding initial members.
AddFilterToCollection()		Optional	A method for adding members to a client defined FilterCollection.
RemoveFilterFromCollection()		Optional	A method for removing members from a client defined FilterCollection.
DeleteFilterCollection()		Optional	A method for Deleting a FilterCollection.

50.6.48SNIA_IndicationFilterTemplate (semi-fixed)

Experimental. IndicationFilter instances that are 'semi-fixed' are IndicationFilters that are be created by a client using CreateInstance, but they follow a pattern defined by an IndicationFilterTemplate. If a profile implementation can support semi-fixed IndicationFilters, the implementation would support 'semi-fixed' IndicationFilterTemplate instances. The implementation shall support 'semi-fixed' filters that are defined by SMI-S profile as mandatory.

SNIA_IndicationFilterTemplate is subclassed from CIM_ManagedElement.

Created By: CreateInstance

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Required if SNIA_IndicationConfigurationCapabilities.SupportedFeatures='7' (Semi-fixed Indication Filters).

Table 548 describes class SNIA_IndicationFilterTemplate (semi-fixed).

Table 548 - SMI Referenced Properties/Methods for SNIA_IndicationFilterTemplate (semi-fixed)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This shall take the form OrgID ":" RegisteredName ":" UniqueID. For more details, see section *Storage Management Technical Specification, Part 3 Common Profiles, 1.6.1 Rev 5* 43.1.4.1 Naming Conventions for IndicationFilterTemplates and IndicationFilters.
SourceNamespace	N	Optional	Deprecated. For instances in the InteropNamespace, this shall be the namespace where the indications are to originate. For instances in the implementation namespace where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.
SourceNamespaces	N	Optional	Experimental. For instances in the InteropNamespace, this should be all the namespaces where the indications may originate. For instances in the implementation namespaces where the indications are to originate (e.g., the namespace of the profile that supports the filter), this may be NULL to indicate the Filter is registered in the Namespace where the indications originate.

Table 548 - SMI Referenced Properties/Methods for SNIA_IndicationFilterTemplate (semi-fixed)

Properties	Flags	Requirement	Description & Notes
Query		Mandatory	A string that specifies a template (in QueryLanguage terms with SUBSTITUTION_STRINGs) what IndicationFilters may be created from this template.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL' for CQL queries with substitution.
ElementName		Optional	A Client Defined user friendly string that identifies the Indication Filter.

EXPERIMENTAL

EXPERIMENTAL

Clause 51: FCoE Target Ports Profile

51.1 Synopsis

Profile Name: FCoE Target Ports (Component Profile)

Version: 1.6.1

Organization: SNIA

CIM Schema Version: 2.29.1

Table 549 describes the related profiles for FCoE Target Ports.

Table 549 - Related Profiles for FCoE Target Ports

Profile Name	Organization	Version	Requirement	Description
Indications	SNIA	1.6.0	Mandatory	

Central Class: CIM_FCPort

Scoping Class: a CIM_ComputerSystem in a referencing autonomous profile

51.2 Description

The FCoE Target Ports Profile is a component profile that models the behavior of the Fibre Channel over Ethernet (FCoE) functionality.

Figure 83 shows the topology of FCoE where the Fibre Channel protocol is carried over the Ethernet.

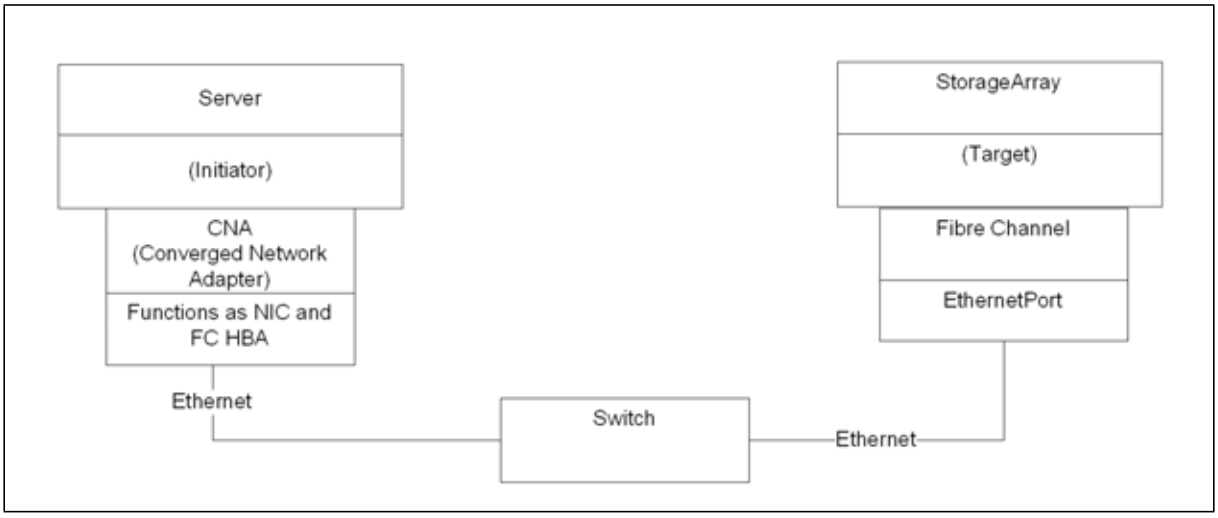

Figure 83 - FCoE Topology

51.3 Implementation

The FCPort is always associated 1-1 with a SCSIProtocolEndpoint instance. The EthernetPort is associated to 0 or more FCPorts. The EthernetPort used for FCoE protocol, shown in Figure 84, has a PortDiscriminator property with the value of "SNIA:FCoE".

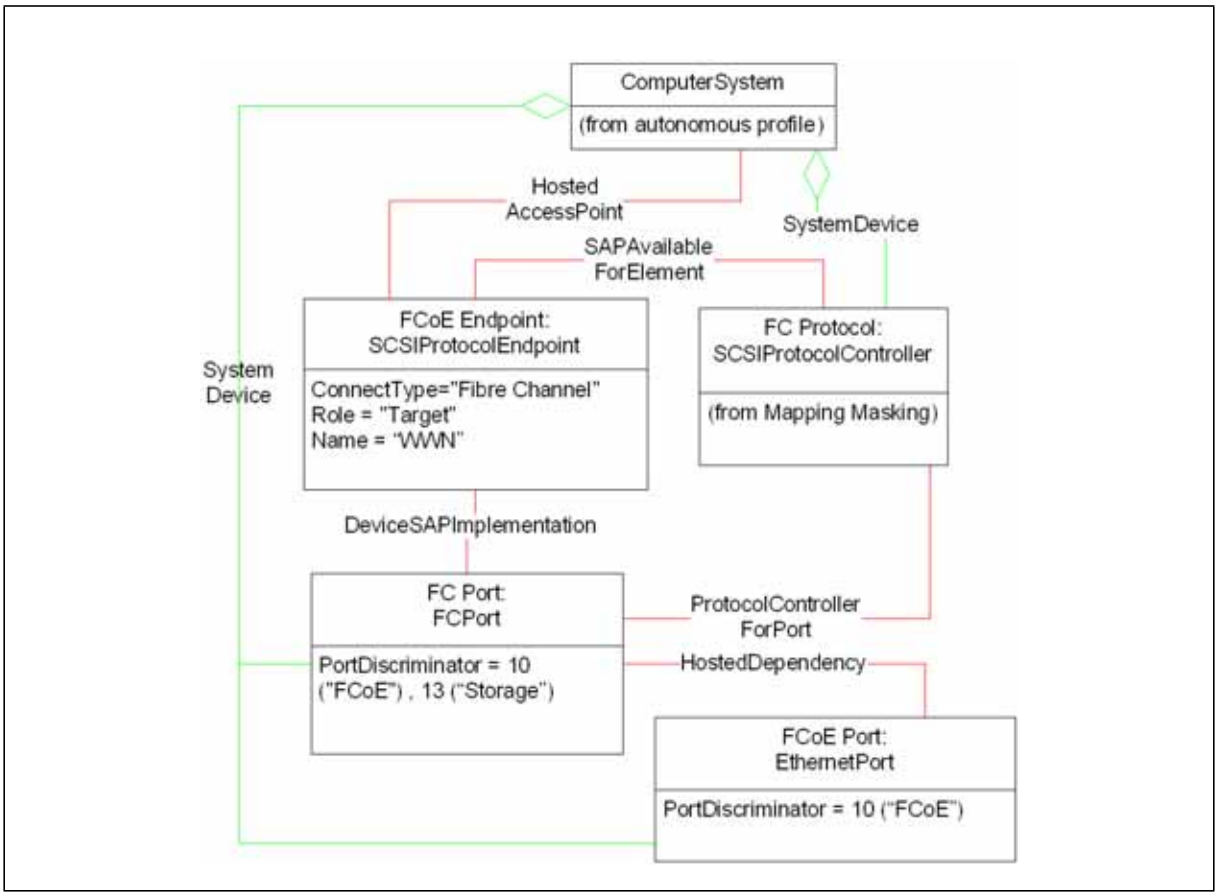

Figure 84 - EthernetPort used for FCoE

51.4 Durable Names and Correlatable IDs of the Subprofile

FCPort.PermanantAddress shall contain the port's Port WWN.

51.4.1 Health and Fault Management Consideration

Table 550 describes FCPort OperationalStatus.

Table 550 - FCPort OperationalStatus

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

51.4.2 Cascading Considerations

None

51.5 Methods

None

51.6 Use Cases

None

51.7 CIM Elements

Table 551 describes the CIM elements for FCoE Target Ports.

Table 551 - CIM Elements for FCoE Target Ports

Element Name	Requirement	Description
51.7.1 CIM_DeviceSAPImplementation	Mandatory	Associates FCPort and SCSIProtocolEndpoint.
51.7.2 CIM_EthernetPort	Mandatory	
51.7.3 CIM_FCPort (For FCoE)	Mandatory	FCPort specialized to handle FCoE.
51.7.4 CIM_HostedAccessPoint	Mandatory	Associates ComputerSystem to SCSIProtocolEndpoint.
51.7.5 CIM_HostedDependency (NetworkPort to FCPort)	Mandatory	Association between EthernetPort and FCPort.
51.7.6 CIM_LogicalPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
51.7.7 CIM_ProtocolControllerForPort	Conditional	Conditional requirement: Support for the Masking and Mapping profile. Only required if the instrumentation claims compatibility with 1.0.
51.7.8 CIM_SCSIProtocolEndpoint	Mandatory	Represents management characteristics related to the SCSI command set.
51.7.9 CIM_SystemDevice (Port)	Mandatory	Associates controller ComputerSystem to FCPort.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort	Mandatory	Create FCPort.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Deprecated WQL -Change to FCPort OperationalStatus.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::OperationalStatus <> PreviousInstance.CIM_FCPort::OperationalStatus	Mandatory	CQL -Change to FCPort OperationalStatus.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.Speed <> PreviousInstance.Speed	Optional	Deprecated WQL -Change to FCPort Speed.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::Speed <> PreviousInstance.CIM_FCPort::Speed	Optional	CQL -Change to FCPort Speed.

Table 551 - CIM Elements for FCoE Target Ports

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.NetworkAddresses <> PreviousInstance.NetworkAddresses	Optional	Deprecated WQL -Change to FCPort NetworkAddresses.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.CIM_FCPort::NetworkAddresses <> PreviousInstance.CIM_FCPort::NetworkAddresses	Optional	CQL -Change to FCPort NetworkAddresses.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort	Mandatory	Delete FCPort.

51.7.1 CIM_DeviceSAPImplementation

Associates FCPort and SCSIProtocolEndpoint. The class definition specializes the CIM_DeviceSAPImplementation definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 552 describes class CIM_DeviceSAPImplementation.

Table 552 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Flags	Requirement	Description & Notes
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.
Antecedent (overridden)		Mandatory	Reference to FCPort.

51.7.2 CIM_EthernetPort

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 553 describes class CIM_EthernetPort.

Table 553 - SMI Referenced Properties/Methods for CIM_EthernetPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	

Table 553 - SMI Referenced Properties/Methods for CIM_EthernetPort

Properties	Flags	Requirement	Description & Notes
LinkTechnology		Mandatory	Shall be 2 (Ethernet).
OperationalStatus		Mandatory	Shall be 0 (Unknown), 2 (OK), 6 (Error), 10 (Stopped), or 11 (In Service).
PermanentAddress	CD	Mandatory	The MAC Address. Shall be formatted as 12 un-separated upper case hex digits.
PortDiscriminator		Mandatory	Discriminates the supported context of this EthernetPort. Value: 10 ('FCoE').

51.7.3 CIM_FCPort (For FCoE)

FCPort specialized to handle FCoE.

Requirement: Mandatory

Table 554 describes class CIM_FCPort (For FCoE).

Table 554 - SMI Referenced Properties/Methods for CIM_FCPort (For FCoE)

Properties	Flags	Requirement	Description & Notes
PortType		Mandatory	Shall be 0|1|10|11|12|13|14|15|16|17|18 (Unknown or Other or N or NL or F/NL or Nx or E or F or FL or B or G).
PermanentAddress	CD	Mandatory	Port WWN. Shall be 16 unseparated uppercase hex digits.
SupportedCOS		Optional	
ActiveCOS		Optional	
SupportedFC4Types		Optional	
ActiveFC4Types		Optional	
Speed		Optional	Speed in bits per second. Shall be 0, 1062500000 (1GFC), 2125000000 (2GFC), 4250000000 (4GFC), 8500000000 (8GFC), 10518750000 (10GFC), 14025000000 (16GFC), 21037500000 (20GFC) or 28500000000 (32GFC).
MaxSpeed		Optional	Maximum Port Speed.
NetworkAddresses		Optional	For Fibre Channel end device ports, the Fibre Channel ID. Shall be 16 un-separated upper case hex digits.
PortDiscriminator		Mandatory	Discriminates the supported context of this FCPort. Value: 10, 13 ('FCoE', 'Storage').

51.7.4 CIM_HostedAccessPoint

Associates ComputerSystem to SCSIProtocolEndpoint. Limit to targets. The class definition specializes the CIM_HostedAccessPoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 555 describes class CIM_HostedAccessPoint.

Table 555 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Antecedent (overridden)		Mandatory	
Dependent (overridden)		Mandatory	Reference to SCSIProtocolEndpoint.

51.7.5 CIM_HostedDependency (NetworkPort to FCPort)

Association between EthernetPort and FCPort.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 556 describes class CIM_HostedDependency (NetworkPort to FCPort).

Table 556 - SMI Referenced Properties/Methods for CIM_HostedDependency (NetworkPort to FCPort)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to EthernetPort.
Dependent		Mandatory	Reference to FCPort.

51.7.6 CIM_LogicalPort

Represents the logical aspects of the physical port and may have multiple associated protocols.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 557 describes class CIM_LogicalPort.

Table 557 - SMI Referenced Properties/Methods for CIM_LogicalPort

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Mandatory	
UsageRestriction		Mandatory	Shall be 2 for ports restricted to Front-end only or 4 if the port is unrestricted.
PortType		Mandatory	VALUE and DESC should be set appropriately for each specialized target port profile.

51.7.7 CIM_ProtocolControllerForPort

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for the Masking and Mapping profile.

Table 558 describes class CIM_ProtocolControllerForPort.

Table 558 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForPort

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to SCSIProtocolController.
Dependent		Mandatory	Reference to FCPort.

51.7.8 CIM_SCSIProtocolEndpoint

Represents management characteristics related to the SCSI command set. The class definition specializes the CIM_ProtocolEndpoint definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 559 describes class CIM_SCSIProtocolEndpoint.

Table 559 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
OtherTypeDescription (overridden)		Mandatory	Shall be the string 'SCSI'.
ConnectionType (added)		Mandatory	Shall be 2 (Fibre Channel).
Role (added)		Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target).

51.7.9 CIM_SystemDevice (Port)

Associates controller ComputerSystem to FCPort. The class definition specializes the CIM_SystemDevice definition in the Generic Target Ports profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 560 describes class CIM_SystemDevice (Port).

Table 560 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port)

Properties	Flags	Requirement	Description & Notes
GroupComponent (overridden)		Mandatory	Reference to ComputerSystem in the referencing profile.
PartComponent (overridden)		Mandatory	Reference to FCPort.

EXPERIMENTAL

EXPERIMENTAL

Clause 52: Diagnostics Job Control Profile

52.1 Synopsis

Profile Name: Diagnostics Job Control (Component Profile)

Version: 1.0.0b

Organization: SNIA

CIM Schema Version: 2.33.0

Table 561 describes the related profiles for Diagnostics Job Control.

Table 561 - Related Profiles for Diagnostics Job Control

Profile Name	Organization	Version	Requirement	Description
Job Control	DMTF	1.0.0	Mandatory	Experimental. This profile is a specialization of DSP1103, version 1.0.0
Indications	DMTF	1.2.0	Mandatory	Experimental. See DSP1054, version 1.2.2

Central Class: CIM_ConcreteJob

Scoping Class: CIM_DiagnosticTest

The Diagnostics Job Control Profile specializes the DMTF Job Control Profile.

52.2 Description

Figure 85 presents the class schema for the Diagnostics Job Control Profile. For simplicity, the prefix CIM_ has been removed from the names of the classes.

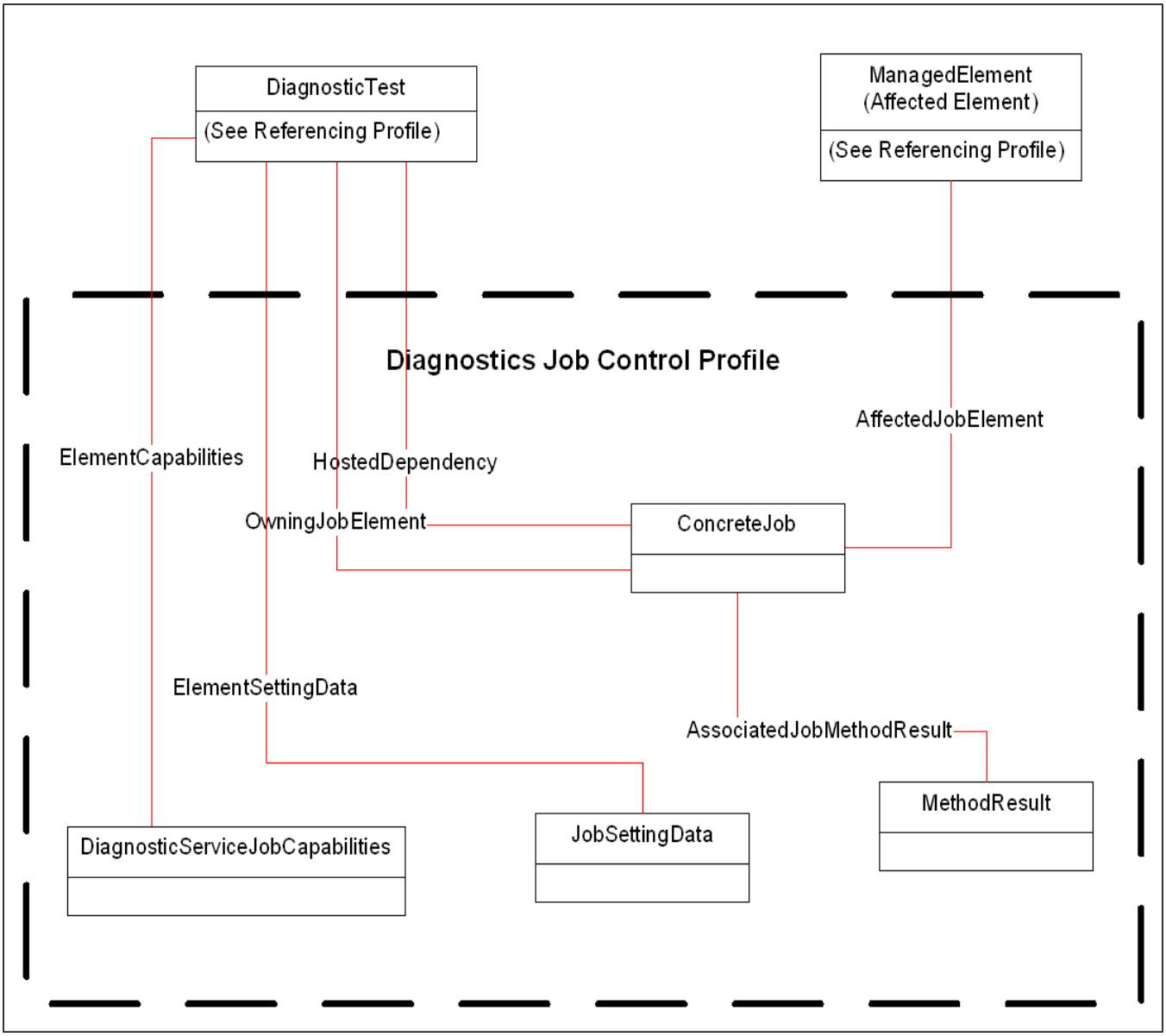

Figure 85 - Diagnostics Job Control Instance Diagram

- Added methods to ConcreteJob
  - Job Control is extended by adding ResumeWithInput and ResumeWithAction methods to ConcreteJob for interactive jobs

52.3 Implementation

52.3.1 CIM_ConcreteJob

This clause defines the properties of the CIM_ConcreteJob class. Each execution of a test will create an instance of CIM_ConcreteJob so that a client can track the progress and control the execution of the executing diagnostic.

52.3.1.1 CIM_ConcreteJob.JobState

The Diagnostic Job Control Profile extends the JobState enumerations defined by the DMTF Job Control profile to include the “12” (Query Pending) JobState. The Job would go into the JobState of Query Pending when it sends an AlertIndication to the client requesting input or action. It would leave the Query Pending state when it successfully received a ResumeWithInput or ResumeWithAction request, or the InteractiveTimeout expires.

On a successful ResumeWithInput or ResumeWithAction operation, the JobState would change to “4” (running). If the operation fails, then the job may wait for a client retry. If the job waits for a client retry, it would stay in the Query Pending state. If the client has exceeded the number of retries (see JobSettingData.ClientRetries) or the InteractiveTimeout expires the job may terminate (JobState = “8” (terminated)).

52.3.1.2 CIM_ConcreteJob.DeleteOnCompletion

The Diagnostics Job Control profile extends the definition of the Job Control profile by specifying that the default for this property shall be TRUE.

52.3.1.3 CIM_ConcreteJob.TimeBeforeRemoval

The Diagnostics Job Control extends the Job Control Profile to identify the time at which the CIM_ConcreteJob object can be deleted, using the calculated Completion Time. The algorithm is as follows:

If JobState=Completed OR Terminated OR Killed, then Completion Time=StartTime + ElapsedTime.

The CIM_ConcreteJob object may be deleted at Completion Time + TimeBeforeRemoval.

NOTE StartTime and CompleteTime are clock (time of day) times. ElapsedTime and TimeBeforeRemoval are time intervals.

52.3.1.4 CIM_ConcreteJob.PercentComplete

This property indicates the percentage of the job that has completed at the time that this value is requested. Implementation of this property is mandatory in order to provide progress indication to clients.

The value of this property should be kept current to be useful. Service implementations should update this property within one second of becoming aware of a progress change.

The PercentComplete property should always report the actual percent complete of how much testing was done. It shall be set to 100 percent only when the test is complete. It shall not be set to 100 percent if the test stops for any other reason (for example, the test stopped or was killed by user, the test exited due to a critical failure, or the test found an error and HaltOnError is TRUE) because the actual percent complete is not 100 percent.

52.3.1.5 CIM_ConcreteJob.InstanceID

CIM_ConcreteJob.InstanceID should be constructed using the following preferred algorithm:

<OrgID>:<LocalID>

where <OrgID> identifies the business entity (for example, ACME) and <LocalID> is a value that uniquely identifies each ConcreteJob instance that is launched on a system when a test is executed. See the MOF file description for further information.

The purpose for <LocalID> is to provide some form of uniqueness within the context of running separate diagnostic tests over a period of time for the domain of the test execution (whether just the local system or several remote systems). In practice, <LocalID> could be an incremented counter or a timestamp in combination with other test identifiers or factors.

A unique <LocalID> allows a user to easily retrieve test results from the diagnostic log for a specific test execution because the InstanceID values of CIM_ConcreteJob and the subclasses of CIM_DiagnosticRecord are closely related.

52.3.1.6 CIM_ConcreteJob.StartTime

The StartTime is mandatory. This is the time the job was actually started.

52.3.1.7 CIM_ConcreteJob.ElapsedTime

The ElapsedTime is mandatory. This property should be updated periodically so as to be useful as a 'heartbeat.'

52.3.1.8 CIM_ConcreteJob.RequestedState

The RequestedState is mandatory.

52.3.2 CIM_JobSettingData

This clause defines the properties of the CIM_JobSettingData class. An instance of this class is used to control the execution of a diagnostic test job. This class is mandatory. It shall represent the default JobSettingData for the DiagnosticTest. This default instance is identified by ElementSettingData with IsDefault="true" between the instance and the instance of CIM_DiagnosticTest.

A CIM_JobSettingData may be specified by a client as an embedded instance input to an invocation of the RunDiagnosticService method. This embedded instance is not instantiated as an instance of CIM_JobSettingData, but the class CIM_JobSettingData (Client) is described in the CIM Elements tables to define what the client may include in the embedded instance. In addition, the client should refer to the CIM_DiagnosticServiceJobCapabilities class to see what restrictions the implementation may put on the client provided CIM_JobSettingData embedded instance.

52.3.2.1 CIM_JobSettingData.InstanceID

CIM_JobSettingData.InstanceID should be constructed using the following preferred algorithm:

<OrgID>:<LocalID>

where <OrgID> identifies the business entity (for example, ACME) and <LocalID> is a value that uniquely identifies each JobSettingData instance that is instantiated on a system.

The purpose for <LocalID> is to provide some form of uniqueness within the context of different JobSettingData instances within the system. In practice, <LocalID> could be an incremented counter or a timestamp in combination with other test identifiers or factors.

52.3.2.2 CIM_JobSettingData.ElementName

This property shall be formatted as a free-form string of variable length.

52.3.2.3 CIM_JobSettingData.DeleteOnCompletion

This property indicates whether the job should be automatically deleted upon completion. The property is mandatory. If DeleteOnCompletion is set to TRUE, then the job will be deleted after the CIM_ConcreteJob.TimeBeforeRemoval time interval. If DeleteOnCompletion is set to FALSE, then the job must be deleted by an DeleteInstance operation.

NOTE If CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported is FALSE, then DeleteOnCompletion must always be TRUE.

52.3.2.4 CIM_JobSettingData.InteractiveTimeout

The InteractiveTimeout is required if CIM_DiagnosticTest.Characteristics contains the value of 3 (IsInteractive). If the test is not interactive (“3” not contained in Characteristics or the job never enters the QueryPending state), then this value is ignored.

The value is a datetime specification of a time interval to wait for input from a client before the Job leaves the QueryPending JobState (in favor of either terminating, continuing with defaults or reissuing the query). The default is 15 minutes (00000000001500.000000:000).

52.3.2.5 CIM_JobSettingData.TerminateOnTimeout

The TerminateOnTimeout is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). If the test is not interactive (“3” not contained in Characteristics or the job never enters the QueryPending state), then this value is ignored.

The value is a boolean that defines the behavior when the InteractiveTimeout is exceeded. If this value is TRUE the job will terminate when the InteractiveTimeout is exceeded. If FALSE the job will use DefaultInputValues.

52.3.2.6 CIM_JobSettingData.DefaultInputValues

This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). If the test is not interactive (“3” not contained in Characteristics or the job never enters the QueryPending state), then this value is ignored.

The value is an array of strings that are default values for inputs that will be requested from clients. Each array has a name as defined in the DefaultInputNames array. If this is a provider supplied (e.g., Default) JobSettingData, then these are the defaults the provider will use if no defaults were supplied on the invocation of the test that generated the job. If the client supplied a JobSettingData on the invocation of the RunDiagnosticService, then the default specified in the embedded JobSettingData will be used if none are supplied with a ResumeWithInput operation.

52.3.2.7 CIM_JobSettingData.DefaultInputNames

The DefaultInputNames is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). If the test is not interactive (“3” not contained in Characteristics or the job never enters the QueryPending state), then this value is ignored.

The value is an array of strings that define default names for client defined default inputs that will be requested from clients. The values in this array are the names of the values in the DefaultInputValues array. Both this array and the DefaultInputValues array are ordered, such that the Names in this array are for the values in the DefaultInputValues array.

52.3.2.8 CIM_JobSettingData.ClientRetries

The ClientRetries is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). If the test is not interactive (“3” not contained in Characteristics or the job never enters the QueryPending state), then this value is ignored.

The value is a numeric value indicating the number of retries a client may make on ResumeWithInput after a failed attempt at ResumeWithInput. Zero means that the client has no retries. For example, this would be set this to zero if the test will use default inputs if an initial attempt at ResumeWithInput fails.

52.3.2.9 CIM_JobSettingData.RunInSilentMode

This shall be set to FALSE if CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is FALSE. If CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is TRUE, then this property identifies the default value.” for the default JobSettingData. For the client defined (parameter) the property description is “This shall be set to FALSE if CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is FALSE.

If CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is TRUE, then this property identifies the value desired by the client."

52.3.3 CIM_DiagnosticServiceJobCapabilities

This clause defines the properties of the CIM_DiagnosticServiceJobCapabilities class. An instance of this class specifies the controls that a client may impose on the execution of a diagnostic test job. This class is optional, but should be populated for the convenience of clients. If it is not present, then the only controls allowed are those specified by the default CIM_JobSettingData.

52.3.3.1 CIM_DiagnosticServiceJobCapabilities.InstanceID

CIM_DiagnosticServiceJobCapabilities.InstanceID should be constructed using the following preferred algorithm:

<OrgID>:<LocalID>

where <OrgID> identifies the business entity (for example, ACME) and <LocalID> is a value that uniquely identifies each DiagnosticServiceJobCapabilities instance that is instantiated on a system.

The purpose for <LocalID> is to provide some form of uniqueness within the context of different DiagnosticServiceJobCapabilities instances within the system. In practice, since there would be only one CIM_DiagnosticServiceJobCapabilities for an instance of the CIM_DiagnosticTest.

52.3.3.2 CIM_DiagnosticServiceJobCapabilities.ElementName

The value shall be set to the DiagnosticTest ElementName.

52.3.3.3 CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported

The DeleteJobSupported is a Mandatory property. It is a boolean that identifies whether or not the implementation supports DeleteInstance on Jobs (and setting of CIM_ConcreteJob.DeleteOnCompletion to FALSE). If the value is FALSE, then the DeleteOnCompletion property shall be TRUE and cannot be set by the client application (in a client defined CIM_JobSettingData). If DeleteJobSupported is TRUE, then a client may set the DeleteOnCompletion to FALSE and use DeleteInstance to delete the job.

NOTE DeleteInstance is not allowed on a job that is a new, starting, running, suspended or in a query pending state, even if DeleteJobSupported is TRUE and DeleteOnCompletion is FALSE. If a JobState is new, starting, running, suspended or query pending, it may be Killed or Terminated using RequestedStateChange.

52.3.3.4 CIM_DiagnosticServiceJobCapabilities.RequestedStatesSupported

The RequestedStatesSupported is a Mandatory property. It is an array property that identifies the JobStates that may be requested by a client application using the CIM_ConcreteJob.RequestStateChange method. Support for the method is mandatory and the RequestedState parameter shall support "4" (terminate) and "5" (Kill). These shall be contained in the RequestedStatesSupported property. In addition, the RequestedState values of 2 (Start) and 3 (Suspend) may also be included. Suspend allows a client to suspend a job. Start allows a client to resume a suspended job.

52.3.3.5 CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax

The InteractiveTimeoutMax identifies the maximum timeout interval that the implementation will support (See CIM_JobSettingData.InteractiveTimeout). This property is conditional. This is only required if the CIM_DiagnosticTest.Characteristics property contains the value "3". The property is a datetime data type that specifies an interval time.

If a client sets the CIM_JobSettingData.InteractiveTimeout above the CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax, the RunDiagnosticService shall reset the value to be the CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax.

NOTE If a default for InteractiveTimeout is supported, then this would be specified in the Default CIM_JobSettingData.InteractiveTimeout property.

52.3.3.6 CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported

The DefaultValuesSupported identifies whether or not the implementation will supply default input values for an interactive test run. This property is conditional. This is only required if the CIM_DiagnosticTest.Characteristics property contains the value “3”. If the value is TRUE, then CIM_JobSettingData.DefaultInputValues and CIM_JobSettingData.DefaultInputNames should have values and the client application may set DefaultInputValues in a client defined CIM_JobSettingData. If the DefaultValuesSupported is set to FALSE, then the implementation requires inputs to be supplied by the client and if a client application supplies DefaultInputValues in the CIM_JobSettingData that it passes into the RunDiagnosticService method the inputs will be ignored.

52.3.3.7 CIM_DiagnosticServiceJobCapabilities.ClientRetriesMax

The ClientRetriesMax identifies the maximum number of client retries the implementation will support on an interactive test run before terminating or assuming default values**.** This property is conditional. This is only required if the CIM_DiagnosticTest.Characteristics property contains the value “3”. The property identifies the maximum value that the implementation will support for CIM_JobSettingData.ClientRetries.

52.3.3.8 CIM_DiagnosticServiceJobCapabiities.CleanupInterval

The CleanupInterval is an optional datetime property identifying the time interval an implementation shall wait before removing a job when DeleteOnCompletion is FALSE.

52.3.3.9 CIM_DiagnosticServiceJobCapabiities.SilentModeSupported

From the CIM Elements table entry for this property the current description says: If an interactive test will support running in "silent mode", then this shall be set to TRUE. If the test cannot support "silent mode", then this shall be set to FALSE. If the test is NOT interactive, then this should be NULL or set to TRUE.

52.3.4 DiagnosticServiceJobCapabilities, Default JobSettingData and the JobSetting Parameter

The default CIM_JobSettingData is mandatory. The CIM_DiagnosticServiceJobCapabilities and the JobSetting parameter of the RunDiagnosticService method are optional. If the CIM_DiagnosticServiceJobCapabilities is not implemented, then the client application cannot alter the default CIM_JobSettingData for the test and the JobSetting parameter should be NULL or set to the default CIM_JobSettingData. If the client application sets the JobSetting parameter to values that conflict with the default CIM_JobSettingData, the test will not fail, but the JobSetting parameter will be reset to the default values (the “effective” JobSetting) and a warning alert message will be issued. The effective JobSetting parameter values will also be logged in the DiagnosticLog.

If the CIM_DiagnosticServiceJobCapabilities is implemented, then the client application may specify values in the JobSetting parameter that conform to the capabilities identified. For example, the client application may specify an InteractiveTimeout that is equal or less than the InteractiveTimeoutMax. If the client application specifies a value that is in conflict with the options allowed by the DiagnosticServiceJobCapabilities for the test, then the conflicting value will be reset to one of two values: The value in the default JobSettingData or the maximum allowed by the DiagnosticServiceJobCapabilities. If the client application leaves the JobSetting parameter NULL, then the default JobSettingData will be used. In either case, if any value was changed, an alert message will be issued. Whether a value was changed or not, the effective Jobsetting used by the test execution will be logged in the DiagnosticLog.

Table 562 identifies the rules for valid combinations of capabilities and settings.

Table 562 - Execution Rules for Valid capabilities and settings

DiagnosticService JobCapabilities	Default JobSettingData	JobSetting	Execution Rule
Absent	Present	NULL	The default JobSettingData is used and the JobSetting is logged.
Absent	Present	Provided - No conflict	The JobSetting provided is used and the JobSetting is logged.
		- Conflicts	The default JobSettingData is used, a warning alert is issued and the JobSetting used is logged.
Present	Present	Null	The default JobSettingData is used and the JobSetting is logged.
Present	Present	Provided - No conflict	The JobSetting provided is used and the JobSetting is logged.
		- Conflicts	The JobSetting is modified to conform to the capabilities (e.g., reset to max values), a warning alert is issued and the JobSetting used is logged.

Table 563 identifies combinations of capabilities and settings that are not defined in this version of the profile

Table 563 - Capabilities and settings not specified by the Profile

DiagnosticService JobCapabilities	Default JobSettingData	JobSetting	Notes
Absent or Present	**Absent**		This is not specified by this version of the standard. The Default JobSettingData is Mandatory, so this configuration is not recognized by this profile.
Present	Present, but conflicting		This is not specified by this version of the standard. The Default JobSettingData should conform to the DiagnosticServiceJobCapabilities if it is present.

52.3.5 CIM_DiagnosticServiceJobCapabilities and CIM_JobSettingData properties

An instance of CIM_DiagnosticServiceJobCapabilities identifies the capabilities supported for a particular CIM_DiagnosticTest. This instance defines the behavior supported by jobs for that test. The JobSetting specified on the RunDiagnosticService further constrains the capabilities to a specific setting. If a client does not specify a JobSetting, the implementation uses the default CIM_JobSettingData for the service.

This section discusses the relationships between the DiagnosticServiceJobCapabilities and the CIM_JobSettingData (either the default or the JobSetting input to the service).

52.3.5.1 Job Deletion Options

Table 564 shows the valid relationships among the DeleteJobSupport and CleanupInterval properties of CIM_DiagnosticServiceJobCapabilities and the corresponding DeleteOnCompletion property of CIM_JobSettingData or the JobSetting parameter of RunDiagnosticService.

Table 564 - Job Deletion Options

Capability	JobSettingData	JobSetting	Execution Rule
DeleteJobSupported = true And **CleanupInterval = non-null value**	DeleteOnCompletion = true	DeleteOnCompletion = true	With DeleteJobSupported = true (supported) and CleanupInterval set to a non-null value, and the default DeleteOnCompletion and the JobSetting set to true, then the job will be deleted TimeBeforeRemoval after completion (or termination) of the job.
		DeleteOnCompletion = false	In this case the JobSetting.DeleteOnCompletion set to false, the client should delete the job, but if the client does not the provider may delete the job after the CleanupInterval.
	DeleteOnCompletion = false	DeleteOnCompletion = true	In this case, the job may be deleted after the TimeBeforeRemoval interval after completion (or termination) of the job.
		DeleteOnCompletion = false	In this case the JobSetting.DeleteOnCompletion set to false, the client should delete the job, but if the client does not the provider may delete the job after the CleanupInterval.
DeleteJobSupported = true And **CleanupInterval = NULL**	DeleteOnCompletion = true	DeleteOnCompletion = true	With DeleteJobSupported = true (supported) and CleanupInterval set to NULL, and the DeleteOnCompletion is true in both the default JobSettingData and the JobSetting parameter, then the job will be deleted TimeBeforeRemoval after completion (or termination) of the job.
		DeleteOnCompletion = false	In this case, a client DeleteInstance is required and the job will not be deleted if the client does not issue the DeleteInstance.
	DeleteOnCompletion = false	DeleteOnCompletion = true	In this case, the job may be deleted after the TimeBeforeRemoval interval after completion (or termination) of the job.
		DeleteOnCompletion = false	In this case the JobSetting.DeleteOnCompletion set to false, the client should delete the job, and if the client does not the provider may not delete the job.
DeleteJobSupported = false And CleanupInterval = NULL	DeleteOnCompletion = true	DeleteOnCompletion = true	With DeleteJobSupported = false (not supported), then the DeleteOnCompletion shall be set to true and the CleanupInterval capability should be set to NULL.
		DeleteOnCompletion = false	In this case, the DeleteOnCompletion will be reset to true and an alert indication (DIAG39) will indicate that the JobSetting was reset and the effective JobSetting will be logged.
DeleteJobSupported = false And CleanupInterval = non-null value	DeleteOnCompletion = true	DeleteOnCompletion = true	With DeleteJobSupported = false (not supported) and DeleteOnCompletion = true, the CleanupInterval will be ignored (even if it is set to a non-null value).
		DeleteOnCompletion = false	In this case, the DeleteOnCompletion will be reset to true and an alert indication (DIAG39) will indicate that the JobSetting was reset and the effective JobSetting will be logged.

Table 565 shows the combination of properties that are not defined in this profile.

Table 565 - Job Deletion combinations not specified by this Profile

Capability	JobSettingData	JobSetting	Execution Rule
DeleteJobSupported = false And CleanupInterval = NULL	**DeleteOnCompletion = false**		This is not specified by this version of the standard. The default JobSettingData should not be set to false when the DeleteJobSupport in the capabilities is false.
DeleteJobSupported = false And CleanupInterval = non-null value	DeleteOnCompletion = false		This is not specified by this version of the standard. With DeleteJobSupported = false (not supported) and DeleteOnCompletion = true, the CleanupInterval will be ignored (even if it is set to a non-null value).

52.3.5.2 Interactive Options

There are a number of properties in CIM_DiagnosticServiceJobCapabilities and CIM_JobSettingData or the JobSetting parameter of RunDiagnosticService that relate to interactive jobs. Table 566 shows the valid relationships among the InteractiveTimeoutMax, DefaultValuesSupported, ClientRetriesMax and SilentModeSupported properties of CIM_DiagnosticServiceJobCapabilities and the corresponding InteractiveTimeout, DefaultInputValues, DefaultInputNames, TerminateOnTimeout, ClientRetries and RunInSilentMode properties of CIM_JobSettingData or the JobSetting parameter of RunDiagnosticService.

Table 566 - Interactive Options

Capability	JobSettingData	JobSetting	Notes
InteractiveTimeoutMax	InteractiveTimeout <= InteractiveTimeoutMax	InteractiveTimeout <= InteractiveTimeoutMax	The Capability bounds how large the Setting may be.
		InteractiveTimeout > InteractiveTimeoutMax	If the JobSetting exceeds the maximum, the setting will be reset to the maximum. An Alert Indication (DIAG39) will be issued and the effective JobSetting will be logged.
DefaultValuesSupported = true	DefaultInputValues = non-null values DefaultInputNames = non-null values	DefaultInputValues = non-null values DefaultInputNames = non-null values	When the capability is true, then default settings for DefaultInputValues and DefaultInputNames should have values and the client application may set DefaultInputValues in a JobSetting input.
	DefaultInputValues = non-null values DefaultInputNames = non-null values	DefaultInputValues = NULL DefaultInputNames = NULL	If NULL is specified for JobSetting (or the individual JobSetting parameters), the default JobSettingData values will be used. An Alert Indication (DIAG39) will indicate that the JobSetting was reset to the defaults and the effective job setting will be logged.
DefaultValuesSupported = false	DefaultInputValues = NULL DefaultInputNames = NULL	DefaultInputValues = NULL DefaultInputNames = NULL	When the capability is false, then the implementation requires inputs to be supplied by the client.
		DefaultInputValues or DefaultInputNames with non-null values	If a client application supplies DefaultInputValues in the JobSetting input to the RunDiagnosticService method the inputs will be ignored, an Alert Indication (DIAG40) will be issued indicating the default values were not used and the effective JobSetting will be logged.

Table 566 - Interactive Options

Capability	JobSettingData	JobSetting	Notes
DefaultValuesSupported = true AND SilentModeSupported = true		DefaultInputValues AND DefaultInputNames with non-null values RunInSilentMode = true	The Job is run in silent mode using the default values supplied by the JobSetting parameter.
	DefaultInputValues AND DefaultInputNames with non-null values	DefaultInputValues = NULL DefaultInputNames = NULL RunInSilentMode = true	In this case, the default values in the default JobSettingData are used for running in silent mode. The client indicates that this is desired by setting the defaults in the JobSetting to NULL. An Alert Indication (DIAG39) will indicate that the JobSetting was reset to the defaults and the effective job setting will be logged. Alternatively, the client could repeat the defaults in the default JobSettingData.
ClientRetriesMax = n DefaultValuesSupported = false	TerminateOnTimeout = true	TerminateOnTimeout = true	The Job will terminate after n tries to solicit input.
		TerminateOnTimeout = false	The JobSetting will be reset to TerminateOnTimeout = true, since there is no other option without defaults. An Alert Indication (DIAG39) will indicate that the JobSetting was reset to the defaults and the effective job setting will be logged.
ClientRetriesMax = n DefaultValuesSupported = true		TerminateOnTimeout = false	The Job will use default values from JobSetting after n tries to solicit input. If Jobsetting defaults were not supplied, the job will use defaults in the default JobSettingData for the service.
ClientRetriesMax = n	ClientRetries = m where m <= n	ClientRetries = m where r <= n	The job will execute with “r” retries. The job will wait one InteractiveTimeout for each reply. After the timeout, it will re-issue the input request. After the r-th retry, the job will either terminate or run with defaults.
		ClientRetries = NULL	The job will execute with “m” retries. The job will wait one InteractiveTimeout for each reply. After the timeout, it will re-issue the input request. After the m-th retry, the job will either terminate or run with defaults. An Alert Indication (DIAG39) will indicate that the JobSetting was reset to the defaults and the effective job setting will be logged.
ClientRetriesMax = n	ClientRetries = m where m <= n	ClientRetries = r where r > n	The job will execute with n retries. The job will wait one InteractiveTimeout for each reply. After the timeout, it will re-issue the input request. After the n-th retry, the job will either terminate or run with defaults. An Alert Indication (DIAG39) will indicate that the JobSetting was reset to the max and the effective job setting will be logged.

Table 567 shows combinations of properties that are not specified by this profile.

Table 567 - Interactive Options Not Specified by Profile

Capability	JobSettingData	JobSetting	Notes
InteractiveTimeoutMax	InteractiveTimeout > InteractiveTimeoutMax		This is not specified by this version of the standard. The default JobSettingData should be consistent with the Capabilities for the test.

Table 567 - Interactive Options Not Specified by Profile

Capability	JobSettingData	JobSetting	Notes
DefaultValuesSupported = true	DefaultInputValues = NULL DefaultInputNames = NULL		This is not specified by this version of the standard. The default JobSettingData cannot have null values.
DefaultValuesSupported = false	DefaultInputValues or DefaultInputNames with non-null values		This is not specified by this version of the standard. The default JobSettingData must conform to the capabilities.
DefaultValuesSupported = true AND SilentModeSupported = true	DefaultInputValues = NULL DefaultInputNames = NULL RunInSilentMode = true or false	DefaultInputValues = NULL DefaultInputNames = NULL RunInSilentMode = true	This is not specified by this version of the standard. The test cannot run in silent mode without default values.
DefaultValuesSupported = false AND SilentModeSupported = true			This is not specified by this version of the standard. In order for the test to run in silent mode, default values must be supported and available
ClientRetriesMax = n DefaultValuesSupported = false	TerminateOnTimeout = false		This is not specified by this version of the standard. If TerminateOnTimeout is false, then DefaultValuesSupported shall be supported.
ClientRetriesMax = n	ClientRetries = m where m > n		This is not specified by this version of the standard. The default JobSettingData value for ClientRetries shall be equal or less than the stated capabilities.

52.4 Methods

52.4.1 CIM_ConcreteJob.RequestStateChange( ) Extrinsic Method

All CIM_DiagnosticService.RunDiagnosticService( ) calls will return a reference to a CIM_ConcreteJob instance, which represents the diagnostic execution. The CIM_ConcreteJob.RequestStateChange( ) method is invoked to control the diagnostic program execution. The input parameters specify the execution control to be performed (Suspend, Kill, Terminate) and a timeout period that specifies the maximum amount of time that the client expects the transition to the new state to take.

Before invoking this method, clients examine the appropriate capabilities to verify whether the execution control is supported. The RequestStateChange( ) method shall change the JobState value if the transition is successfully performed.

RequestStateChange( ) return values are specified in Table 568 and parameters are specified in Table 569. No standard messages are defined.

Table 568 - RequestStateChange( ) Method: Return Code Values

Value	Description
0	Completed with No Error
2	Unknown/Unspecified Error
3	Cannot complete within Timeout Period
4	Failed
5	Invalid Parameter

Table 568 - RequestStateChange( ) Method: Return Code Values

Value	Description
6	In Use
4096	Method parameters checked — transition started
4097	Invalid state transition
4098	Use of timeout parameter not supported
4099	Busy — indicates that the method cannot be invoked "at this time." It is not an error condition, but signals that the implementation is doing something else and cannot respond."
32768..65535	Vendor specific

Table 569 - RequestStateChange( ) Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	RequestedState	uint16	The requested state of a job, which may be one of the following values: Start (2), Suspend (3), Terminate (4) or Kill (5)
IN	TimeoutPeriod	datetime	A timeout period that specifies the maximum amount of time that the client expects the transition to the new state to take. The interval format shall be used to specify the TimeoutPeriod.

52.4.2 CIM_ConcreteJob.ResumeWithInput( ) Extrinsic Method

The CIM_ConcreteJob.ResumeWithInput( ) method is invoked to resume the diagnostic program execution when it has a JobState of 12 (Query Pending). The input parameters specify an array of strings that constitute the client supplied inputs to the diagnostic test. If the array is empty or NULL this means that the client application wants the job to use the DefaultInputValues as defined in the JobSettingData that is controlling the execution of the job.

NOTE The profile supports multiple invocations of ResumeWithInput on a single job. Each invocation is initiated by a DIAG34 Alert Indication (see the Diagnostics Standard Messages). This message includes a “List of Inputs” that are being requested at this time. The inputs that would be supplied by the client would those inputs.

Before invoking this method, clients examine the JobState to verify that it is 12 (Query Pending). The ResumeWithInput( ) method shall change the JobState value to 4 (Running) if the operation is successfully performed.

ResumeWithInput( ) return values are specified in Table 570 and parameters are specified in Table 571. No standard messages are defined.

Table 570 - ResumeWithInput( ) Method: Return Code Values

Value	Description
0	Completed with No Error
2	Unknown/Unspecified Error
3	The job has already timed out

Table 570 - ResumeWithInput( ) Method: Return Code Values

Value	Description
4	Failed
5	Invalid Parameter
6	JobState not QueryPending
32768..65535	Vendor specific

Table 571 - ResumeWithInput( ) Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	Inputs	String[]	The client inputs being requested by the job when its state changed to 12 (Query Pending)

The return codes supported for this method and their meaning are:

- 0 - Completed with No Error

  The ResumeWithInput was accepted and the job has resumed. The JobState has changed from “12” (Query Pending) to “4” (Running).

- 2 - Unknown/Unspecified Error

  The JobState was “12” (Query Pending) and the inputs were valid, but the request failed for other reasons.

- 3 - The job has already timed out

- 4 - Failed

  The JobState was “12” (Query Pending) and the inputs were valid, but the request failed for other reasons.

- 5 - Invalid Parameter

  One or more of the Inputs values is not valid. The JobState will depend on the effective JobSetting parameter used for the Job execution and the DiagnosticServiceJobCapabilities for the job.

  - If the ClientRetriesMax has not been exceeded, then JobState remains “12” (Query Pending) and the client has InteractiveTimeout to respond with another ResumeWithInput.
  - If the ClientRetriesMax has been exceeded, then the JobState will be in one of two states:
    - “8” - Terminated, if JobSetting.TerminateOnTimeout is TRUE
    - “4” - Running, if TerminateOnTimeout is FALSE. The job is running with default inputs.

- 6 - JobState not QueryPending

  The JobState for the job to be resumed was not “12” (Query Pending). The ResumeWithInput is not processed and the JobState is unaltered.

The only input to the RequestWithInput is the Inputs string array. The Inputs strings are values for the inputs requested in the DIAG34 Alert Indication. The DIAG34 message includes the names of the inputs desired by the job.

NOTE An interactive job may interleave DIAG34 (request for inputs) and DIAG35 (request for action) Alert Indications.

52.4.3 CIM_ConcreteJob.ResumeWithAction( ) Extrinsic Method

The CIM_ConcreteJob.ResumeWithAction( ) method is invoked to resume the diagnostic program execution when it has a JobState of 12 (Query Pending) and an action (rather than input) was requested. The pending query is a request to perform an action and the test merely needs to know when the action is completed.

NOTE The profile supports multiple invocations of ResumeWithAction on a single job. Each invocation is initiated by a DIAG35 Alert Indication (see the Diagnostics Standard Messages). This message includes an “Action String” that identifies the action being requested at this time.

Before invoking this method, clients examine the JobState to verify that it is 12 (Query Pending). The ResumeWithAction( ) method shall change the JobState value to 4 (Running) if the operation is successfully performed.

ResumeWithAction( ) return values are specified in Table 572. No standard messages are defined.

Table 572 - ResumeWithAction( ) Method: Return Code Values

Value	Description
0	Completed with No Error
2	Unknown/Unspecified Error
3	The job has already timed out
4	Failed
6	JobState not QueryPending
32768..65535	Vendor specific

There are no input parameters to ResumeWithAction.

The return codes supported for this method and their meaning are:

- 0 - Completed with No Error

  The ResumeWithAction was accepted and the job has resumed. The JobState has changed from “12” (Query Pending) to “4” (Running).

- 2 - Unknown/Unspecified Error

  The JobState was “12” (Query Pending), but the request failed for other reasons.

- 3 - The job has already timed out

- 4 - Failed

  The JobState was “12” (Query Pending), but the request failed for other reasons.

- 6 - JobState not QueryPending

  The JobState for the job to be resumed was not “12” (Query Pending). The ResumeWithAction is not processed and the JobState is unaltered. NOTE: One of the conditions in which this might occur is when

the job automatically detects that the action was taken (e.g., media inserted) and automatically continues processing (e.g., JobState = “4” (running)).

The ResumeWithAction method takes no input parameters and has no output parameters (other than the Return Code). It should be noted that the ResumeWithAction method is the appropriate response to the DIAG35 Alert Indication (after the action is taken).

NOTE An interactive job may interleave DIAG34 (request for inputs) and DIAG35 (request for action) Alert Indications.

52.4.4 (8.15) CIM_ElementCapabilities

Table 573 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 573, all operations in the default list in XREF shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 573 - Operations: CIM_ElementCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

52.4.5 CIM_ConcreteJob

Table 574 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 574, all operations in the default list in XREF shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 574 - Operations: CIM_ConcreteJob

Operation	Requirement	Messages
GetInstance	Mandatory	None
ModifyInstance	Optional	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
InvokeMethod	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None
DeleteInstance	Optional	None

52.4.5.1 CIM_ConcreteJob ModifyInstance

52.4.5.2 CIM_ConcreteJob DeleteInstance

If the CIM_ConcreteJob.DeleteOnCompletion value is FALSE, the instance can only be deleted with the intrinsic method DeleteInstance. However, the DeleteInstance is not supported until the Job has a JobState of “Completed”, “Terminated” or “Killed”.

NOTE The job may also be automatically deleted by the implementation after it has completed or terminated or killed for a period of time that exceeds the CleanupInterval. If the CleanupInterval has not been specified by the implementation, then it could only be deleted by the DeleteInstance.

52.4.6 CIM_OwningJobElement

Table 575 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 575 all operations in the default list in XREF shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 575 - Operations: CIM_OwningJobElement

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

52.4.7 CIM_AffectedJobElement

Table 576 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 576, all operations in the default list in XREF shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 576 - Operations: CIM_AffectedJobElement

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

52.4.8 CIM_JobSettingData

Table 577 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 577, all operations in the default list in XREF shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 577 - Operations: CIM_JobSettingData

Operation	Requirement	Messages
GetInstance	Mandatory	None

Table 577 - Operations: CIM_JobSettingData

Operation	Requirement	Messages
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

52.4.9 CIM_ElementSettingData

Table 578 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 578, all operations in the default list in XREF shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 578 - Operations: CIM_ElementSettingData

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

52.4.10CIM_DiagnosticServiceJobCapabilities

Table 579 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 579, all operations in the default list in XREF shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 579 - Operations: CIM_DiagnosticServiceJobCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None

Table 579 - Operations: CIM_DiagnosticServiceJobCapabilities

Operation	Requirement	Messages
ReferenceNames	Optional	None

52.4.10.1 CIM_DiagnosticServiceJobCapabilities.CreateGoalSettings()

The CIM_DiagnosticServiceJobCapabilities.CreateGoalSettings() method is invoked in the context of a specific CIM_DiagnosticServiceJobCapabilities instance.

This method is used to create a JobSettingData using the DiagnosticServiceJobCapabilities as a template. The purpose of this method is to create a JobSettingData based on the DiagnosticServiceJobCapabilities on which this method is invoked and has properties set in line with those DiagnosticServiceJobCapabilities.

CreateGoalSettings( ) return values are specified in Table 580 and parameters are specified in Table 581. No standard messages are defined.

Table 580 - CreateGoalSettings( ) Method: Return Code Values

Value	Description
0	Success
1	Not Supported
2	Unknown
3	Timeout
4	Failed
5	Invalid Parameter
6	Alternative Proposed
32768..65535	Vendor specific

Table 581 - CreateGoalSettings( ) Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	TemplateGoalSettings[]	String	An array of JobSettingData embedded instances that reflect what the client wants. This parameter may be NULL. If NULL, the method returns a setting that conforms to the Capabilities.
IN / OUT	SupportedGoalSettings[]	string	An array of JobSettingData embedded instances that are consistent with the DiagnosticServiceJobCapabilities and are closest matches to the input TemplateGoalSettings

The return codes supported for this method and their meaning are:

- 0 - Success

  The method executed successfully and the client gets back Success if the TemplateGoalSettings is exactly supportable (same as the SupportedGoalSettings).

- 1 - Not Supported

  The CreateGoalSettings method is not supported by the implementation.

- 2 - Unknown

  The method failed for unknown reasons.

- 3 - Timeout

  The method timed out. The client may try again.

- 4 - Failed

  The method is supported, but the implementation was not able to come up with a “best match” for the input TemplateGoalSettings.

- 5 - Invalid Parameter

  One or both of the input parameters have an invalid value or an invalid combination of named properties of a setting parameter

- 6 - Alternative Proposed

  The method executed successfully, but the client gets back a supported alternative. This alternative should be a best match, as defined by the implementation.

The input parameters for this method and their interpretation are:

- TemplateGoalSettings[]

  Normally a client would only supply one template JobSettingData. However, a client may supply multiple embedded instance of JobSettingData.

  If only one embedded instance is supplied, it should be consistent with the DiagnosticServiceJobCapabilities (e.g., InteractiveTimeout should be less than or equal the InteractiveTimeoutMax). The implementation will generate (in SupportedGoalSettings) a JobSettingData that either matches the TemplateGoalSetting or is a “best match” with the TemplateGoalSetting and is consistent with the DiagnosticServiceJobCapabilities.

  If multiple embedded instances of JobSettingData are supplied in the TemplateGoalSettings parameter, then the SupportedGoalSettings array (if not NULL) shall correspond to the entries in the TemplateGoalSettings parameter.

  If the input TemplateGoalSettings is NULL or the empty string, this method returns a default SettingData element that is supported by DiagnosticServiceJobCapabilities element (e.g., the default JobSettingData).

- SupportedGoalSettings[]

  If this parameter is NULL on input, the implementation shall not use this parameter in determining the output.

  If this parameter has values, then the implementation shall assume these are values from a previous invocation of the method and that the client is looking for a different (closer) result.

The output parameter for this method and its meaning are:

- SupportedGoalSettings[]

  If the method was successful (Return Code 0 or 6) this will contain embedded instances of JobSettingData that may be used as the JobSetting parameter of RunDiagnosticService.

  If the method was not successful (Return Codes 1, 2, 3, 4 or 5) this shall be NULL as an output.

52.5 Use Cases

Not defined.

52.6 CIM Elements

Table 582 describes the CIM elements for Diagnostics Job Control.

Table 582 - CIM Elements for Diagnostics Job Control

Element Name	Requirement	Description
52.6.1 CIM_AffectedJobElement	Optional	Association to link a job to a managed element.
52.6.2 CIM_ConcreteJob	Mandatory	**Used by the client to monitor and control the execution of a diagnostic test.**
52.6.3 CIM_DiagnosticServiceJobCapabilities (Job Capabilities)	Optional	Experimental. Publishes the diagnostic test's job capabilities.
52.6.4 CIM_ElementCapabilities (Job Capabilities)	Optional	Associates a diagnostic test with its Job capabilities.
52.6.5 CIM_ElementSettingData (Default JobSettingData)	Mandatory	Associates the job settings with the job used to run a diagnostic test.
52.6.6 CIM_JobSettingData (Client)	Optional	CIM_JobSettingData is used to override the defaults settings.
52.6.7 CIM_JobSettingData (Default)	Mandatory	Default settings published by the diagnostic tests.
52.6.8 CIM_OwningJobElement	Mandatory	**Associates a diagnostic test with its jobs.**
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG9"	Conditional	Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). CQL -The test continued execution using a default response because a query timeout occurred.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG12"	Optional	CQL -The test did not run because the job could not be started.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG19"	Mandatory	CQL -The test was Killed by the client.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG20"	Mandatory	CQL -The test was Terminated by the client.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG21"	Optional	CQL -The test was suspended by the client.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG34"	Optional	CQL -This is an alert indication to solicit input to a test from a client.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG35"	Optional	CQL -This is an alert indication to solicit user action from a client.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG36"	Optional	CQL -The test was Killed by the test (provider).
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG37"	Optional	CQL -The test was Terminated by the test (provider).

Table 582 - CIM Elements for Diagnostics Job Control

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG38"	Optional	CQL -The test was resumed by the client.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG39"	Conditional	Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). CQL -A JobSetting parameter was reset by the job.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG40"	Conditional	Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). CQL -A JobSetting parameter was reset by the job.

52.6.1 CIM_AffectedJobElement

CIM_AffectedJobElement is used to associate a **diagnostics** job with its affected managed elements (devices).

Created By: Extrinsic: RunServiceDiagnostic

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 583 describes class CIM_AffectedJobElement.

Table 583 - SMI Referenced Properties/Methods for CIM_AffectedJobElement

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	This property shall be a reference to an instance of CIM_ConcreteJob **that represents the diagnostic test.**

52.6.2 CIM_ConcreteJob

Each successful RunDiagnosticService( ) call will return a CIM_ConcreteJob instance. Each CIM_ConcreteJob instance represents a diagnostic execution.

Created By: Extrinsic: RunDiagnosticService

Modified By: Extrinsic: RequestedStateChange

Deleted By: DeleteInstance

Requirement: Mandatory

Table 584 describes class CIM_ConcreteJob.

Table 584 - SMI Referenced Properties/Methods for CIM_ConcreteJob

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key:**InstanceID should be constructed using the following preferred algorithm:** **\<OrgID**
Name		Mandatory	**The property will be formatted as a free-form string of variable length.**

Table 584 - SMI Referenced Properties/Methods for CIM_ConcreteJob

Properties	Flags	Requirement	Description & Notes
JobState		Mandatory	This is used to communicate job-specific state **for the Diagnostics Job.**The possible values are 2 (New), 3 (Starting), 4 (Running), 5 (Suspended), 6 (Shutting Down), 7 (Completed), 8 (Terminated), 9 (Killed), 10 (Exception) or **12 (Query Pending).**
StartTime		Mandatory	**None. The Diagnostics Job Control makes this Mandatory.**
ElapsedTime		Mandatory	This property should be updated periodically so as to be useful as a 'heartbeat.'.
DeleteOnCompletion		Mandatory	**The default value for this property is TRUE.** **Version 2 of DSP1002 had this property Optional. It is Mandatory in the Diagnostics Job Control Profile to be a compatible extension of the Job Control Profile. Also note that defaulting this to TRUE is also an extension to Job Control.**
ErrorDescription		Optional	Mandatory when ErrorCode value is non-null. **DSP1002 says "If ErrorCode is implemented, ErrorDescription should be filled in to explain the error.".**
RequestedState		Mandatory	None.
ResumeWithInput()		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive).
ResumeWithAction()		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive).

52.6.3 CIM_DiagnosticServiceJobCapabilities (Job Capabilities)

Experimental. CIM_DiagnosticServiceCapabilities publishes the diagnostic test's job capabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 585 describes class CIM_DiagnosticServiceJobCapabilities (Job Capabilities).

Table 585 - SMI Referenced Properties/Methods for CIM_DiagnosticServiceJobCapabilities (Job Capabilities)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key: InstanceID shall be unique and should be constructed using the following preferred algorithm: <OrgID>:<LocalID>.
ElementName		Mandatory	This property shall contain the value of the diagnostic test’s ElementName property.
DeleteJobSupported		Mandatory	This boolean property identifies whether or not the implementation supports DeleteInstance on Job (and setting of CIM_ConcreteJob.DeleteOnCompletion to FALSE).
RequestedStatesSupported		Mandatory	The RequestedStatesSupported may be 2 (Start), 3 (Suspend), 4 (Terminate) or 5 (Kill). Support for 4 (Terminate) and 5 (Kill) are mandatory. Support for 2 (Start), which is used to resume a suspended job, and 3 (Suspend) are optional.

Table 585 - SMI Referenced Properties/Methods for CIM_DiagnosticServiceJobCapabilities (Job Capabilities)

Properties	Flags	Requirement	Description & Notes
InteractiveTimeoutMax		Conditional	Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). This is a datetime property that identifies the maximum value for the JobSettingData.InteractiveTimeout.
DefaultValuesSupported		Conditional	Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). This boolean property identifies whether or not the implementation will supply default input values for an interactve test run. If the value is TRUE, then CIM_JobSettingData.DefaultInputValues and CIM_JobSettingData.DefaultInputNames should have values.
ClientRetriesMax		Conditional	Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). This property identifies the maximum number of retries the implementation will support on an interactive test run. That is, the maximum value supported for CIM_JobSettingData.ClientRetries.
CleanupInterval		Optional	A datetime property identifying the time interval an implementation shall wait before removing a job when DeleteOnCompletion is FALSE.
SilentModeSupported	N	Conditional	Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). If an interactive test will support running in "silent mode", then this shall be set to TRUE. If the test cannot support "silent mode", then this shall be set to FALSE. If the test is NOT interactive, then this should be NULL or set to TRUE.
CreateGoalSettings()		Optional	

52.6.4 CIM_ElementCapabilities (Job Capabilities)

CIM_ElementCapabilities associates a diagnostic test with its Job capabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 586 describes class CIM_ElementCapabilities (Job Capabilities).

Table 586 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Job Capabilities)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	This property shall be a reference to an instance of CIM_DiagnosticTest.
Capabilities		Mandatory	This property shall be a reference to an instance of CIM_DiagnosticServiceJobCapabilities.

52.6.5 CIM_ElementSettingData (Default JobSettingData)

CIM_ElementSettingData associates the job settings with the job used to run a diagnostic test.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 587 describes class CIM_ElementSettingData (Default JobSettingData).

Table 587 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Default JobSettingData)

Properties	Flags	Requirement	Description & Notes
IsDefault		Mandatory	This shall be TRUE. The CIM_JobSettingData referenced is the default CIM_JobSettingData **Based on the 8/31/2011 DIAG SIG call, this should always be set to TRUE. That is, the only setting data that should ever get instantiated is the Default SettingData. Client versions of JobSettingData are only passed as embedded instances and are never instantiated in the model (except as log information).**
ManagedElement		Mandatory	This property shall be a reference to an instance of CIM_DiagnosticTest.
SettingData		Mandatory	This property shall be a reference to an instance of CIM_JobSettingData.

52.6.6 CIM_JobSettingData (Client)

A client uses CIM_JobSettingData to override the defaults settings and run a diagnostic test using specific job settings. Such settings are passed as the JobSetting argument when the RunDiagnosticService( ) extrinsic method of CIM_DiagnosticTest is invoked.

Requirement: Optional

Table 588 describes class CIM_JobSettingData (Client).

Table 588 - SMI Referenced Properties/Methods for CIM_JobSettingData (Client)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
ElementName		Mandatory	This property shall be formatted as a free-form string of variable length.
DeleteOnCompletion		Conditional	Conditional requirement: Required if CIM_JobSettingData is supported. This property indicates whether the job should be automatically deleted upon completion. If DeleteOnCompletion is set to TRUE, then the job will be deleted after the CIM_ConcreteJob.TimeBeforeRemoval time interval. If DeleteOnCompletion is set to FALSE, then the job must be deleted by an DeleteInstance operation.
InteractiveTimeout		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). A datetime specification of a time interval to wait for input from a client before it leaves the QueryPending JobState (in favor or either terminating or continuing with defaults). The default is 15 minutes (00000000001500.000000:000).
TerminateOnTimeout		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). A boolean that defines the behavior when the InteractiveTimeout is exceeded. If this value is TRUE the job will terminate when the InteractiveTimeout is exceeded. If FALSE the job will use DefaultInputValues.
DefaultInputValues	N	Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). An array of strings that are client defined default values for inputs that will be requested from clients. Each array has a name as defined in the DefaultInputNames array.

Table 588 - SMI Referenced Properties/Methods for CIM_JobSettingData (Client)

Properties	Flags	Requirement	Description & Notes
DefaultInputNames	N	Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). An array of strings that define default names for client defined default inputs that will be requested from clients. The values in this array are the names of the values in the DefaultInputValues array.
ClientRetries	N	Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). A numeric value indicating the number of retries a client may make on ResumeWithInput after a failed attempt at ResumeWithInput. Zero means that the client has no retries. For example, the client would set this to zero to tell the test to use default inputs if an initial attempt at ResumeWithInput fails.
RunInSilentMode		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). This shall be set to FALSE if CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is FALSE. If CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is TRUE, then this property identifies the value desired by the client.

52.6.7 CIM_JobSettingData (Default)

diagnostic tests use CIM_JobSettingData to publish default settings using CIM_ElementSettingData where the IsDefault property has the value of TRUE.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 589 describes class CIM_JobSettingData (Default).

Table 589 - SMI Referenced Properties/Methods for CIM_JobSettingData (Default)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
ElementName		Mandatory	This property shall be formatted as a free-form string of variable length.
DeleteOnCompletion		Conditional	Conditional requirement: Required if CIM_JobSettingData is supported. This property indicates whether the job should be automatically deleted upon completion. If DeleteOnCompletion is set to TRUE, then the job will be deleted after the CIM_ConcreteJob.TimeBeforeRemoval time interval. If DeleteOnCompletion is set to FALSE, then the job must be deleted by an DeleteInstance operation.
InteractiveTimeout		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). A datetime specification of a time interval to wait for input from a client before it leaves the QueryPending JobState (in favor or either terminating or continuing with defaults). The default is 15 minutes (00000000001500.000000:000).

Table 589 - SMI Referenced Properties/Methods for CIM_JobSettingData (Default)

Properties	Flags	Requirement	Description & Notes
TerminateOnTimeout		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). A boolean that defines the behavior when the InteractiveTimeout is exceeded. If this value is TRUE the job will terminate when the InteractiveTimeout is exceeded. If FALSE the job will use DefaultInputValues.
DefaultInputValues	N	Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). An array of strings that define default values for inputs that will be requested from clients. Each array has a name as defined in the DefaultInputNames array.
DefaultInputNames	N	Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). An array of strings that define default names for inputs that will be requested from clients. The values in this array are the names of the values in the DefaultInputValues array.
ClientRetries		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). A numeric value indicating the number of retries a client may make on ResumeWithInput after a failed attempt at ResumeWithInput. Zero means that the client has no retries. For example, this would be set this to zero if the test will use default inputs if an initial attempt at ResumeWithInput fails.
RunInSilentMode		Conditional	Experimental. Conditional requirement: This is required if CIM_DiagnosticTest.Characteristics has the value of 3 (IsInteractive). This shall be set to FALSE if CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is FALSE. If CIM_DiagnosticServiceJobCapabilities.SilentModeSupported is TRUE, then this property identifies the default value.

52.6.8 CIM_OwningJobElement

CIM_OwningJobElement associate a diagnostic test with its jobs (jobs that are launched by this diagnostic).

Created By: Extrinsic: RunDiagnosticService

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 590 describes class CIM_OwningJobElement.

Table 590 - SMI Referenced Properties/Methods for CIM_OwningJobElement

Properties	Flags	Requirement	Description & Notes
OwningElement		Mandatory	**This property shall be a reference to an instance of CIM_DiagnosticTest.**

EXPERIMENTAL

Annex A (informative) SMI-S Information Model

This standard is based on DMTF's CIM schema, version 2.41. The DMTF schema is available in the machine-readable Managed Object Format (MOF) format. DMTF MOFs are simultaneously released both as an "Experimental" and a "Final" version of the schema. This provides developers with early access to experimental parts of the models. Both versions are available at

http://dmtf.org/standards/cim/cim_schema_v2410

Most SMI-S Profiles are primarily based on the DMTF Final MOFs. Content marked as "Experimental" or "Implemented" may be based on DMTF's Experimental MOFs. Some SMI-S Experimental Profiles may also use classes with a SNIA_ prefix; MOFs from these classes are available from SNIA.