Storage Management Technical Specification, Part 2 Common Profiles

Version 1.3.0, Rev 6

This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestion for revision should be directed to the Technical Council Managing Director at tcmd@snia.org.

SNIA Technical Position

21 April, 2009

Revision History

Revison 1

Date

4 January, 2007

SCRs Incorporated and other changes

Job Control Subprofile.

- Added explicit description of required properties for CIM_InstMethodCall (CORE-SMIS-SCR00021)

- Added Indication for job start (CORE-SMIS-SCR00022)

Proxy System Management subprofile

- Added this new profile. (CORE-SMIS-SCR00024)

IP Interface profile

- Added this new profile (CORE-SMIS-SCR00025)

Ethernet Port Profile

- Added this new profile (CORE-SMIS-SCR00025)

Comments

Editorial notes displayed.

Revison 2

Date

14 April 2007

SCRs Incorporated and other changes

Common Recipes Clause

- Removed empty Common Recipes Clause (SMIS-120-Errata-SCR0003)

Registry of Profiles and Subprofiles

- Removed clause containing registry of profiles and subprofiles (SMIS-120-Errata-SCR0004)

Device Credentials

- Replaced Device Credentials figure with version from 1.0 (SMIS-120-Errata-SCR0005)

SAS Target Port profile

- Removed extra space in name of SAS Target Port profile (SMIS-120-Errata-SCR00016)

- Fixed typo in class name in SAS Target Ports profile (SMIS-120-Errata-SCR00018)

Server & PRP Profiles

- Added properties to Product and moved to PRP (SMIS-120-Errata-SCR00022)

- Moved SoftwareIdentity, ElementSoftwareIdentity, Product, and roductSoftwareComponent from Server to Profile Registration Profile (SMIS-120-Errata-SCR00047)

Object Manager Adapter

- Clarified that CIM-XML adapter and the object manager are mandatory (SMIS-12-Errata-SCR00023)

Initiator and Target Ports profiles

- Updated all initiator and target port profiles to be consistent with the generic initiator and target port profiles (SMIS-12-Errata-SCR00024)

Multiple Computer System

- Added IsSpare usage information (SMIS-120-Errata-SCR00028)

Software Package

- Added clause explaining that Software Package is merged into Software subprofile

Base Server profile

- Added a Base Server Profile (SMIS-130-Draft-SCR00006) (4-0-1)

Cascading profile (CORE-SMIS-SCR-00027) (7-0-0)

- Changed versions to 1.3.0

- Added back the figure (was lost from 1.1.0 version) to the Resource Allocation/Deallocation section

- Added USAGE tags to a bunch of classes

- Added the ComputerSystem class for the Leaf System

- Added definitions (conditional) of StorageVolume and LogicalDisk

- Added a bunch of CONDITIONS and applied them

Media Access Device Profile

- Added a Media Access Device Profile (SMIS-130-Draft-SCR00007) (4-0-0)

Proxy Server Management Profile

- Added this profile (SMIS-130-Draft-SCR00008) (5-0-1)

Software Installation profile

- Deleted SMI-S Software Installation profile (SMIS-130-Draft-SCR00004) (6-0-0)

Software Inventory profile

- Added a specialized DMTF Software Inventory profile (SMIS-130-Draft-SCR00005) (6-0-0)

Storage Device Enclosure Subprofile

- Deleted the Storage Device Enclosure profile (CORE-SMIS-SCR-00026) (3-0-0)

Storage Enclosure Subprofile

- Added the new Storage Enclosure profile (CORE-SMIS-SCR-00026) (3-0-0)

Comments

Only minor editorial work for this revision.

Revison 3

Date

19 June 2007

SCRs Incorporated and other changes

Access Points Profile

- Access Points class table fixes (SMIS-12-Errata-SCR00048) (8-0-1)

Cascading Profile

- Incorporate enhancments from 1.2.0 (SMIS-12-Errata-SCR00053) (7-0-1)

Base Server Profile

- Promote Base Server profile to Experimental (SMIS-130-Draft-SCR00006) (2-0-1)

Software Inventory Profile

- Promote Software Inventory profile to Experimental (SMIS-130-Draft-SCR00005) (2-0-1)

Server Profile

- Incorporate fixes from 1.2.0 (SMIS-120-Errata-SCR00047) (9-0-1)

- Incorporate fixes from 1.2.0 (SMIS-120-Errata-SCR00060) (14-0-0)

  - Clarify use of component profiles in Server Profile

Profile Registration Profile

- Incorporate fixes from 1.2.0 (SMIS-120-Errata-SCR00047) (9-0-1)

Indication Profile

- Incorporate fixes from 1.2.0 errata (SMIS-120-Errata-SCR00051) (8-1-1)

- Add sections to clarify alert vs lifecycle indications (TSG-SMIS-SCR00220) (11-0-0)

Object Manager Adapter

- Incorporate fixes from 1.2.0 (SMIS-120-Errata-SCR00047) (9-0-1)

Proxy Server System Management Profile

- Update & promote to Experimental the Proxy System Management subprofile based on numerous ballot comments (SMIS-130-Draft-SCR00012) (4-0-0)

Comments

Editorial notes displayed.

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

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

Revision 4

Date

20 July 2007

SCRs Incorporated and other changes

Job Control Profile

- Fixed enumeration of OperationalStatus in the OperationalStatus to Job State Mapping (CORE-SMIS-SCR-00030) (4-0-0)

Storage HBA Profile

- **Promote Storage HBA profile to Experimental** (SMIS-130-Draft-SCR00013) (3-0-0)

Comments

Editorial notes displayed, but the DRAFT material is not.

Revision 5

Date

14 November 2007

SCRs Incorporated and other changes

Clause 9: iSCSI Target Ports Subprofile (SMIS-120-Errata-SCR00074)

- Made Capabilities dependent on the existence of the Service - fix for bug 3156

Comments

Editorial notes and DRAFT material are not displayed.

Revision 6

Date

14 January 2009

SCRs Incorporated and other changes

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

Removed test CARDINALITY (SMIS-130-Errata-SCR00001)

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

Indication Profile Content Review Changes, including changes to Indication Profile and Namespaces Diagram and Indication Profile Instance Diagram (SMIS-130-Errata-SCR00006)

Invalid version numbers in supported profiles tables replaced with valid numbers (SMIS-130-Errata-SCR00017)

Removed “Determine the Other Profile Supported by a Profile” recipe from Profile Registration Profile (SMIS-130-Errata-SCR00020)

Added missing value for CIM protocol version in XML for Server profile (SMIS-130-Errata-SCR00021)

In FC Target Ports, added missing VALUE attribute to SCSIProtocolEndpoint and ATAProtocolEndpoint, also replaced single HostedAccessPoint into separate definitions for SCSI/ATA and added VALIDATION_PROPERTY. Added VALIDATION_PROPERTY to each reference of DeviceSAPImplementation (SMIS-130-Errata-SCR00022)

Updated FC (and Generic) Initiator Ports (SMIS-130-Errata-SCR00023)

Updated the SMI-S version numbers for these profiles: Location, SB Initiator Ports, SB Target Ports, Security Resource Ownership, Software, Software Inventory and Software Repository (SMIS-130-Errata-SCR00024)

Promoted Profile Registration Profile from Experimental to Stable (SMIS-130-Errata-SCR00033)

Updated FC Initiator Ports and Generic Initiator Port profiles (SMIS-130-Errata-SCR00041)

Comments

Editorial notes and DRAFT material are not displayed.

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

INTENDED AUDIENCE

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

DISCLAIMER

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

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

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

CHANGES TO THE SPECIFICATION

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

- Major Revision: A major revision of the specification represents a substantial change to the underlying scope or architecture of the SMI-S API. A major revision results in an increase in the version number of the version identifier (e.g., from version 1.x.x to version 2.x x). There is no assurance of interoperability or backward compatibility between releases with different version numbers.
- Minor Revision: A minor revision of the specification represents a technical change to existing content or an adjustment to the scope of the SMI-S API. A minor revision results in an increase in the release number of the specification's identifier (e.g., from x.1.x to x.2.x). Minor revisions with the same version number preserve interoperability and backward compatibility.
- Update: An update to the specification is limited to minor corrections or clarifications of existing specification content. An update will result in an increase in the third component of the release identifier (e.g., from x.x.1 to x.x.2). Updates with the same version and minor release levels preserve interoperability and backward compatibility.

TYPOGRAPHICAL CONVENTIONS

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

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

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

Experimental Maturity Level

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

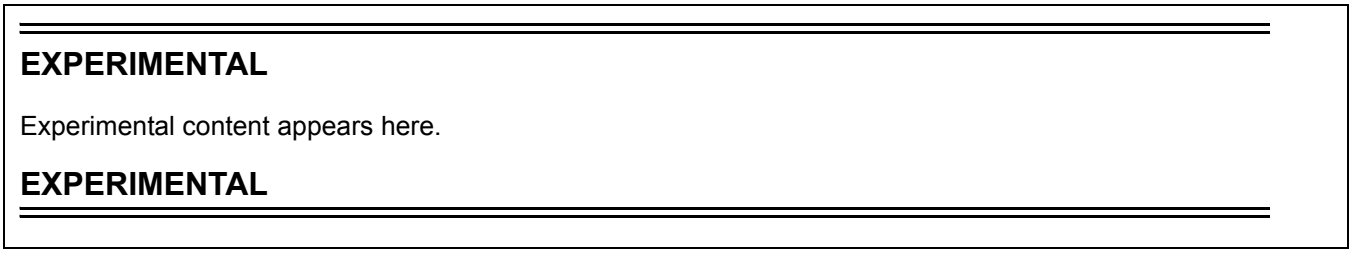

Figure 1 - Experimental Maturity Level Tag

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.

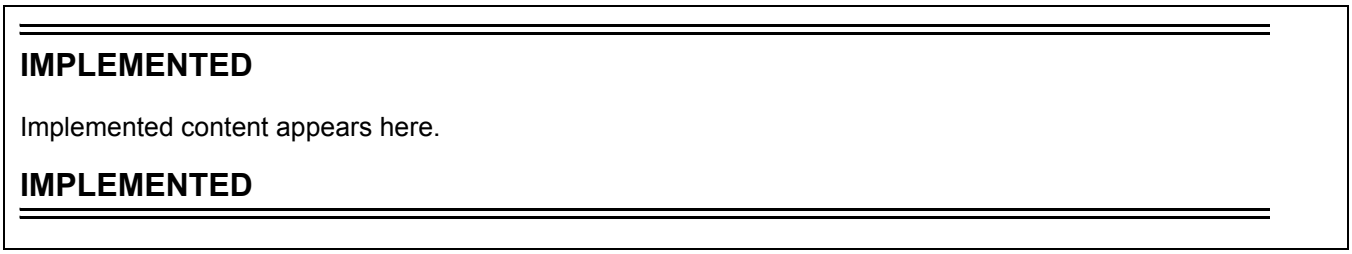

Figure 2 - Implemented Maturity Level Tag

Stable Maturity Level

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

STABLE

Stable content appears here.

STABLE

Figure 3 - Stable Maturity Level Tag

Finalized Maturity Level

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

Deprecated Material

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

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

DEPRECATED

Content that has been deprecated appears here.

DEPRECATED

Figure 4 - Deprecated Tag

USAGE

Contents

List of Tables

List of Figures

Foreword

Storage Management Technical Specification is published in several ports. *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6* defines profiles that are used by profiles in the 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 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 “specialized” to assure SNIA requirements are met.

Parts of this Standard

This standard is subdivided in the following parts:

- *Storage Management Technical Specification, Overview, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 3 Block Devices, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 4 File Systems, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 5 Fabric, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 6 Host Elements, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 7 Media Libraries, 1.3.0 Rev 6*

SNIA Web Site

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

SNIA Address

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

Acknowledgments

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

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

Clause 1: Scope

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

Clause 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 1.1
http://www.dmtf.org/standards/documents/WBEM/DSP200.html

DMTF DSP1001, Management Profile Specification Usage Guide
http://www.dmtf.org/standards/published_documents/DSP1001.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 DSP1009:2006, Sensors Profile 1.0.0
http://www.dmtf.org/standards/published_documents/DSP1009.pdf

DMTF DSP1013:2006, Fan Profile 1.0.0
http://www.dmtf.org/standards/published_documents/DSP1013.pdf

DMTF DSP1015:2006, Power Supply Profile 1.0.0
http://www.dmtf.org/standards/published_documents/DSP1015.pdf

DMTF DSP1011:2006, Physical Asset Profile 1.0.0
http://www.dmtf.org/standards/published_documents/DSP1011.pdf

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

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/

Clause 3: Terms and definitions

3.1 General

For the purposes of this document, the terms and definitions given in *Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6* 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.

Clause 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 1 Common Architecture, 1.3.0 Rev 6* Clause 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 1 Common Architecture, 1.3.0 Rev 6* Clause 10: 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 (above), 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	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 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.

Table 1: Profile Components (Sheet 2 of 2)

Profile Element	Goal
Recipes and Client Considerations	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.
CIM Server Requirements	A list of requirements on the CIM Server necessary to support the profile and its subprofiles.
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).
Dependencies on Other Standards	A table listing other standards on which this profile and its subprofiles are dependent.

Clause 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. In general, any CIM intrinsic read methods on profile or subprofile models are implied. However, CIM intrinsic write methods (Create/Delete/Modify) should not be assumed unless explicitly listed in the profile or subprofile definition with a well defined semantic.

For a profile or subprofile, 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.

CIM Operations and their parameters are taken directly from the *CIM Operations Over HTTP* specification. It is assumed that these methods are being called on the CIM Client API. 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

Clause 6: Generic Target Ports Profile

6.1 Synopsis

Profile name: Generic Target Ports

Version: 1.0.0

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 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 AT A ProtocolEndpoint) represent command protocols in use on the port.

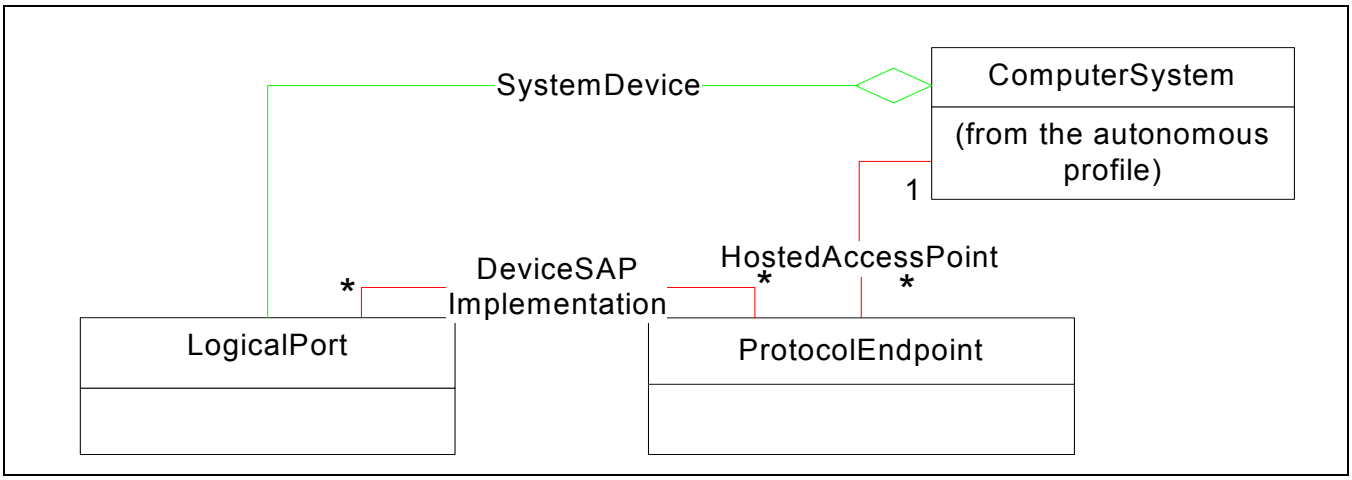

Figure 5 - Generic Target Port Classes

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.

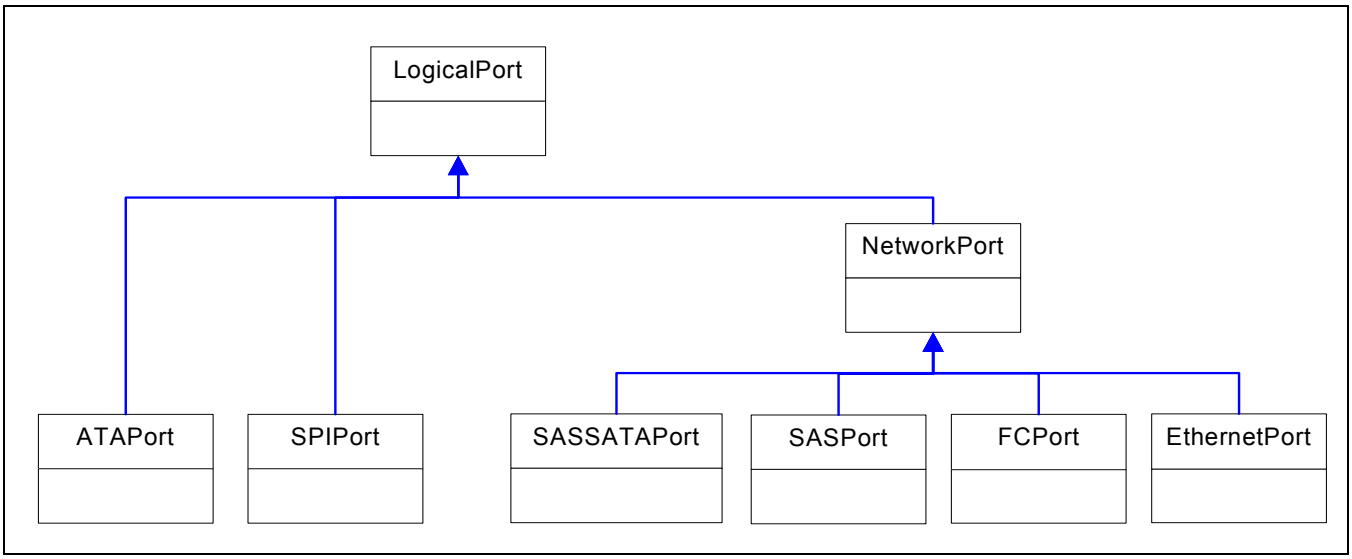

Figure 6 - LogicalPort Class Hierarchy

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 external systems. It

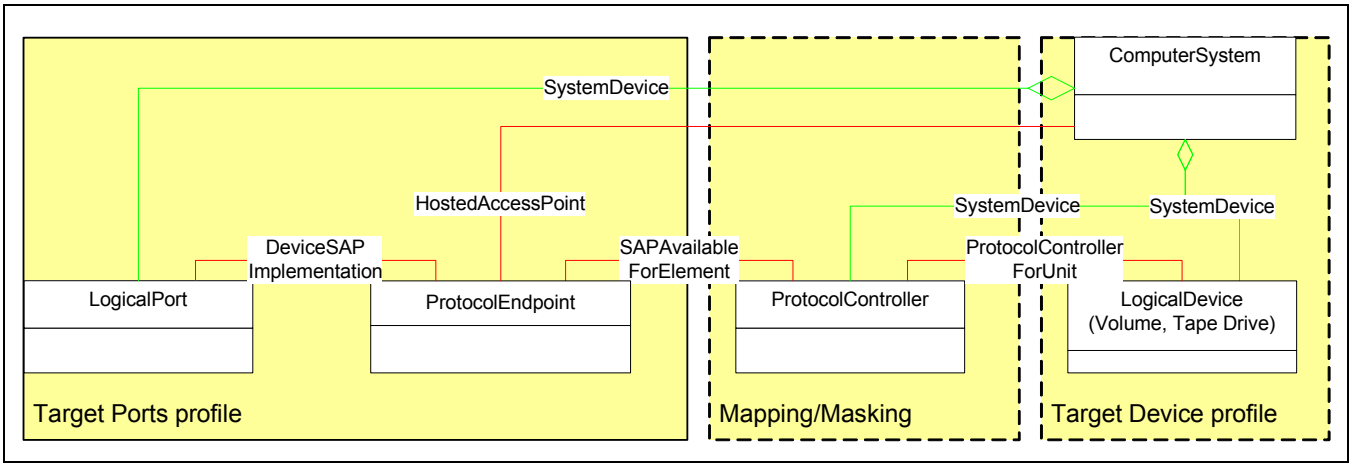

Figure 7 - Generic Target with LUN Masking

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”

Table 2 - Modeling of Common Storage Devices in CIM

SCSI Device Type	Inquiry Device Type	LogicalDevice subclass
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_ATAProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for ATA.
6.6.2 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.
6.6.3 CIM_HostedAccessPoint (ATA)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
6.6.4 CIM_HostedAccessPoint (SCSI)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.

Table 3 - CIM Elements for Generic Target Ports

Element Name	Requirement	Description
6.6.5 CIM_LogicalPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
6.6.6 CIM_ProtocolEndpoint	Mandatory	Represents a protocol (command set) associated to a port.
6.6.7 CIM_SCSIProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for SCSI.
6.6.8 CIM_SystemDevice	Mandatory	Associates ComputerSystem to LogicalPort.

6.6.1 CIM_ATAProtocolEndpoint

Specialization of ProtocolEndpoint for ATA.

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

Table 4 describes class CIM_ATAProtocolEndpoint.

Table 4 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target).
OtherTypeDescription	Mandatory	Shall be the string 'ATA'.
ConnectionType	Mandatory	Shall be 2|3 (PATA or SATA).

6.6.2 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

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

Table 5 describes class CIM_DeviceSAPImplementation.

Table 5 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : limit to targets
Antecedent	Mandatory	Validation Property : limit to targets

6.6.3 CIM_HostedAccessPoint (ATA)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 6 describes class CIM_HostedAccessPoint (ATA).

Table 6 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ATA)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem
Dependent	Mandatory	Reference to ATAProtocolEndpointValidation Property : limit to targets

6.6.4 CIM_HostedAccessPoint (SCSI)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 7 describes class CIM_HostedAccessPoint (SCSI).

Table 7 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (SCSI)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Reference to SCSIProtocolEndpointValidation Property : limit to targets

6.6.5 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 8 describes class CIM_LogicalPort.

Table 8 - SMI Referenced Properties/Methods for CIM_LogicalPort

Properties	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.

6.6.6 CIM_ProtocolEndpoint

Represents a protocol (command set) associated to a port.

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

Table 9 describes class CIM_ProtocolEndpoint.

Table 9 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint

Properties	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', 'SB', or 'iSCSI'. Initiator port specialized profiles specify the appropriate subset.

6.6.7 CIM_SCSIProtocolEndpoint

Specialization of ProtocolEndpoint for SCSI.

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

Table 10 describes class CIM_SCSIProtocolEndpoint.

Table 10 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

6.6.8 CIM_SystemDevice

Associates ComputerSystem to LogicalPort.

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

Table 11 describes class CIM_SystemDevice.

Table 11 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 7: Parallel SCSI (SPI) Target Ports Profile

7.1 Synopsis

Profile Name: SPI Target Ports

Version: 1.2.0

Organization: SNIA

CIM Schema Version: TBD

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

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.

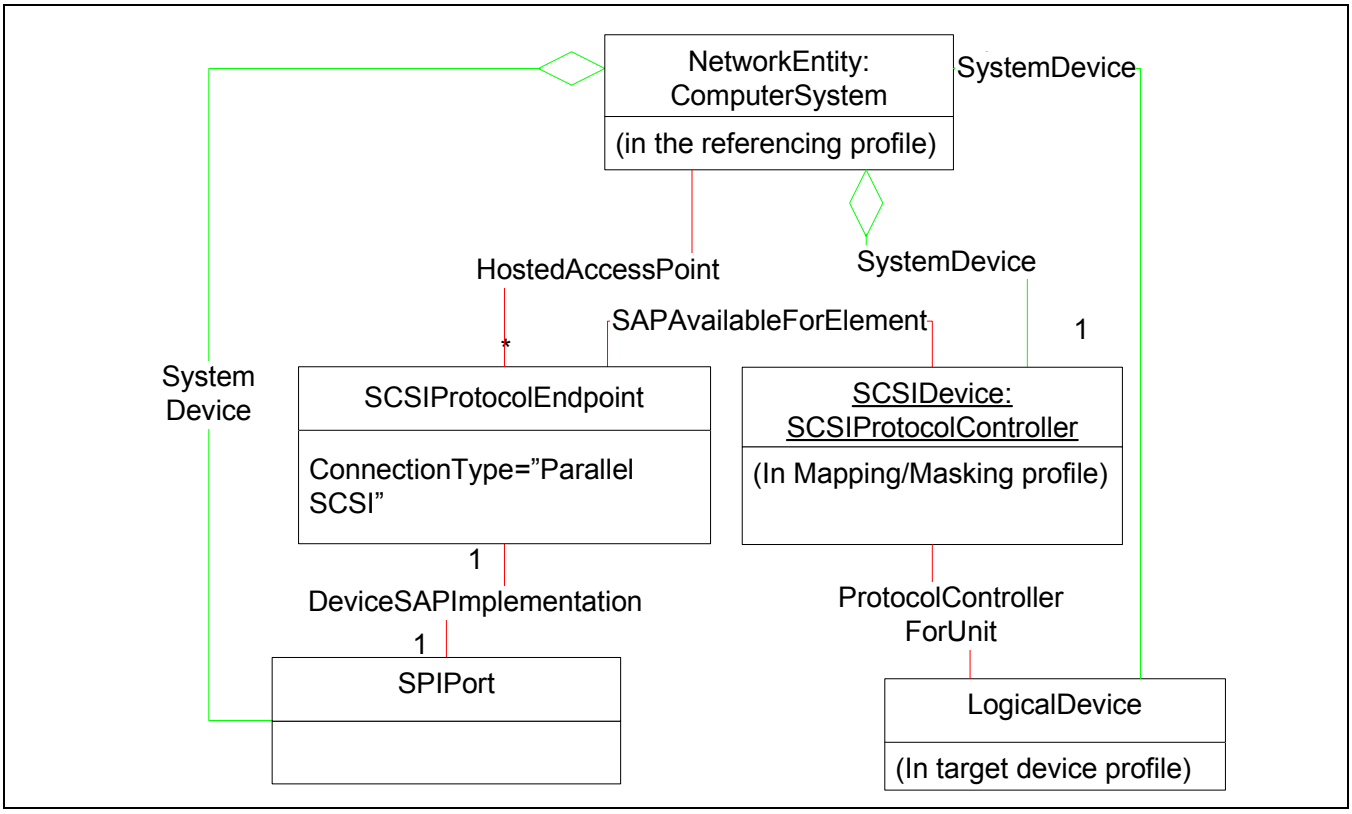

Figure 8 - SPI Target Port Instance Diagram

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 12 - 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 13 describes the CIM elements for SPI Target Ports.

Table 13 - CIM Elements for SPI Target Ports

Element Name	Requirement	Description
7.6.1 CIM_ATAProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for ATA.
7.6.2 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.
7.6.3 CIM_HostedAccessPoint (ATA)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
7.6.4 CIM_HostedAccessPoint (SCSI)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
7.6.5 CIM_SCSIProtocolEndpoint	Mandatory	Represents a protocol (command set) associated to a port.
7.6.6 CIM_SCSIProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for SCSI.
7.6.7 CIM_SPIPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
7.6.8 CIM_SystemDevice	Mandatory	Associates ComputerSystem to LogicalPort.

7.6.1 CIM_ATAProtocolEndpoint

Specialization of ProtocolEndpoint for ATA.

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 14 describes class CIM_ATAProtocolEndpoint.

Table 14 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target).
OtherTypeDescription	Mandatory	Shall be the string 'ATA'.
ConnectionType	Mandatory	Shall be 2|3 (PATA or SATA).

7.6.2 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

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

Table 15 describes class CIM_DeviceSAPImplementation.

Table 15 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : limit to targets
Antecedent	Mandatory	Validation Property : limit to targets

7.6.3 CIM_HostedAccessPoint (ATA)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 16 describes class CIM_HostedAccessPoint (ATA).

Table 16 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ATA)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem
Dependent	Mandatory	Reference to ATAProtocolEndpointValidation Property : limit to targets

7.6.4 CIM_HostedAccessPoint (SCSI)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 17 describes class CIM_HostedAccessPoint (SCSI).

Table 17 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (SCSI)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Reference to SCSIProtocolEndpointValidation Property : limit to targets

7.6.5 CIM_SCSIProtocolEndpoint

Represents a protocol (command set) associated to a port.

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

Table 18 describes class CIM_SCSIProtocolEndpoint.

Table 18 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	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', 'SB', or 'iSCSI'. Initiator port specialized profiles specify the appropriate subset.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

7.6.6 CIM_SCSIProtocolEndpoint

Specialization of ProtocolEndpoint for SCSI.

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

Requirement: Optional

Table 19 describes class CIM_SCSIProtocolEndpoint.

Table 19 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

7.6.7 CIM_SPIPort

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 20 describes class CIM_SPIPort.

Table 20 - SMI Referenced Properties/Methods for CIM_SPIPort

Properties	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	Shall be 101 (SCSI Parallel).

7.6.8 CIM_SystemDevice

Associates ComputerSystem to LogicalPort.

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

Table 21 describes class CIM_SystemDevice.

Table 21 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

STABLE

Clause 8: FC Target Ports Profile

8.1 Synopsis

Profile name: FC Target Ports

Version: 1.2.0

Organization: SNIA

CIM schema version: 2.9.0

Central Class: CIM_FCPortt

Scoping Class: a CIM_System in a separate 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.

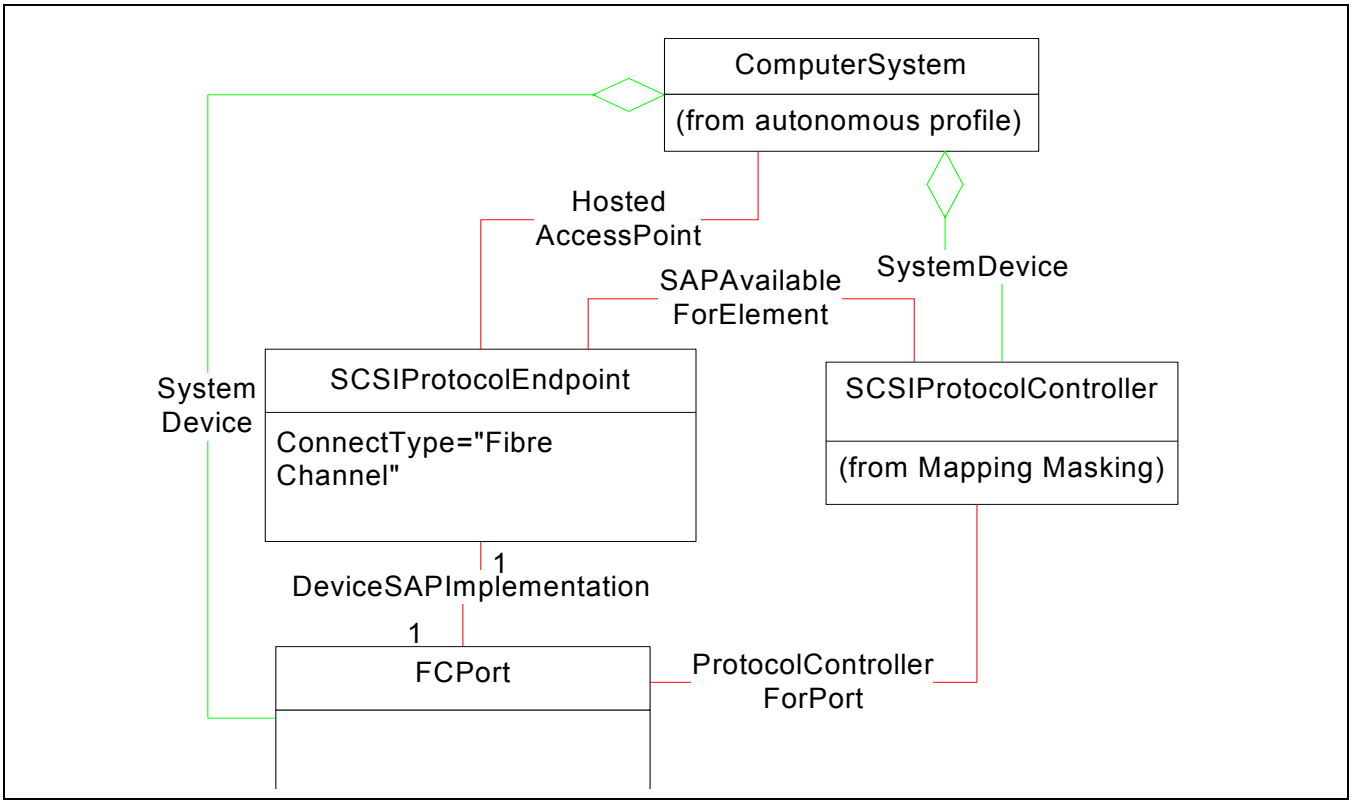

Figure 9 - FC Target Port Instance Diagram

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

Figure 22 descibes FCPort OperationalStatus.

Table 22 - 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 23 describes the CIM elements for FC Target Ports.

Table 23 - CIM Elements for FC Target Ports

Element Name	Requirement	Description
8.9.1 CIM_ATAProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for ATA.
8.9.2 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.
8.9.3 CIM_FCPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
8.9.4 CIM_HostedAccessPoint (ATA)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
8.9.5 CIM_HostedAccessPoint (SCSI)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
8.9.6 CIM_ProtocolControllerForPort	Conditional	Conditional requirement: Support for the Masking and Mapping profile.Only required if the instrumentation claims compatibility with 1.0
8.9.7 CIM_SCSIProtocolEndpoint	Mandatory	Represents a protocol (command set) associated to a port.
8.9.8 CIM_SCSIProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for SCSI.
8.9.9 CIM_SystemDevice	Mandatory	Associates ComputerSystem to LogicalPort.
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

Table 23 - CIM Elements for FC Target Ports

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

8.9.1 CIM_ATAProtocolEndpoint

Specialization of ProtocolEndpoint for ATA.

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

Table 24 describes class CIM_ATAProtocolEndpoint.

Table 24 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target).
OtherTypeDescription	Mandatory	Shall be the string 'ATA'.
ConnectionType	Mandatory	Shall be 2|3 (PATA or SATA).

8.9.2 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

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

Requirement: Mandatory

Table 25 describes class CIM_DeviceSAPImplementation.

Table 25 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : limit to targets
Antecedent	Mandatory	Validation Property : limit to targets

8.9.3 CIM_FCPort

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 26 describes class CIM_FCPort.

Table 26 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	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.
PermanentAddress	Mandatory	Port WWN. Shall be 16 unseparated uppercase hex digits.
SupportedCOS	Optional	
ActiveCOS	Optional	
SupportedFC4Types	Optional	
ActiveFC4Types	Optional	

8.9.4 CIM_HostedAccessPoint (ATA)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 27 describes class CIM_HostedAccessPoint (ATA).

Table 27 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ATA)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem
Dependent	Mandatory	Reference to ATAProtocolEndpointValidation Property : limit to targets

8.9.5 CIM_HostedAccessPoint (SCSI)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 28 describes class CIM_HostedAccessPoint (SCSI).

Table 28 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (SCSI)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Reference to SCSIProtocolEndpointValidation Property : limit to targets

8.9.6 CIM_ProtocolControllerForPort

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for the Masking and Mapping profile.

Table 29 describes class CIM_ProtocolControllerForPort.

Table 29 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForPort

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

8.9.7 CIM_SCSIProtocolEndpoint

Represents a protocol (command set) associated to a port.

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

Table 30 describes class CIM_SCSIProtocolEndpoint.

Table 30 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	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', 'SB', or 'iSCSI'. Initiator port specialized profiles specify the appropriate subset.
ConnectionType	Mandatory	Shall be 2 (Fibre Channel)

8.9.8 CIM_SCSIProtocolEndpoint

Specialization of ProtocolEndpoint for SCSI.

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

Table 31 describes class CIM_SCSIProtocolEndpoint.

Table 31 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

8.9.9 CIM_SystemDevice

Associates ComputerSystem to LogicalPort.

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 32 describes class CIM_SystemDevice.

Table 32 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

STABLE

STABLE

Clause 9: iSCSI Target Ports Subprofile

9.1 Synopsis

Profile name: iSCSI Target Ports

Version: 1.2.0

Organization: SNIA

CIM schema version: 2.11.0

Central Class: CIM_EthernetPortt

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. Figure 10 uses the UML instance naming notation (InstanceName:ClassName) with the iSCSI-style names before the CIM names. Figure 33 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. iSCSI does have a specific naming requirement for SCSIProtocolController that is described in Table 33.

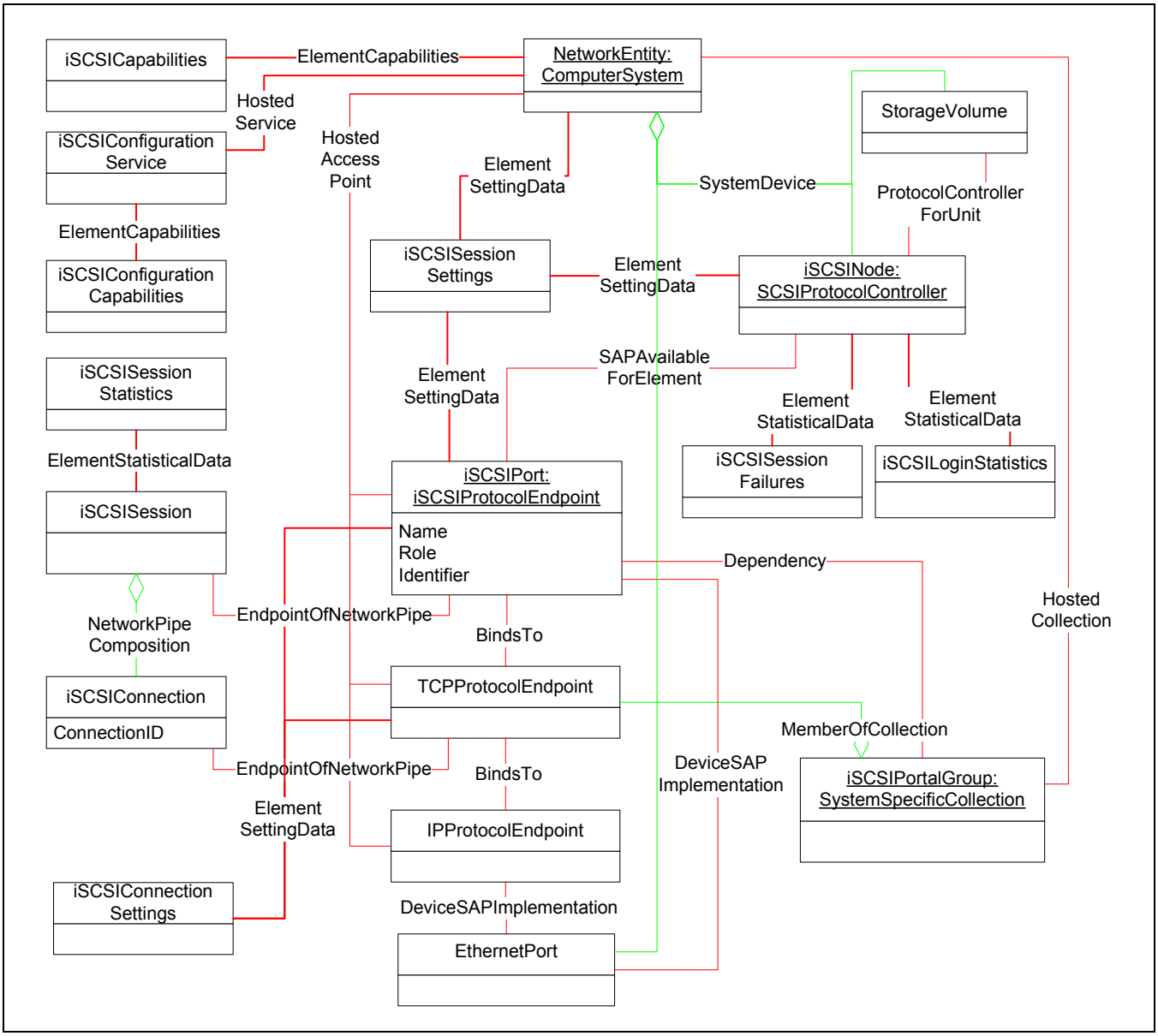

Figure 10 - iSCSI Target Ports Subprofile Instance Diagram

Table 33 - 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 an 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, an 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.
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 an 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.

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 an 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 sub-profiles.

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 an 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 sub-profile 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.

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 1 Common Architecture, 1.3.0 Rev 6* 7.8. 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 1 Common Architecture, 1.3.0 Rev 6* 7.8. ConnectionType shall be set to “iSCSI”.

9.4 Health and Fault Management

Table 34 defines the SMI-S-defined meanings of the OperationalStatus property for EthernetPort used in the SB Target Port Profile.

Table 34 - 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 an 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”})

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. 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.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. 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 35 describes the CIM elements for iSCSI Target Ports.

Table 35 - 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)	Mandatory	

Table 35 - CIM Elements for iSCSI Target Ports

Element Name	Requirement	Description
9.8.9 CIM_ElementSettingData (iSCSIConnectionSettings to iSCSIProtocolEndpoint)	Mandatory	
9.8.10 CIM_ElementSettingData (iSCSISessionSettings to SCSIProtocolController)	Mandatory	
9.8.11 CIM_ElementSettingData (iSCSISessionSettings to System)	Mandatory	
9.8.12 CIM_ElementSettingData (iSCSISessionSettings to iSCSIProtocolEndpoint)	Mandatory	
9.8.13 CIM_ElementStatisticalData (iSCSILoginStatistics to SCSIProtocolController)	Mandatory	
9.8.14 CIM_ElementStatisticalData (iSCSISessionFailures to SCSIProtocolController)	Mandatory	
9.8.15 CIM_ElementStatisticalData (iSCSISessionStatistics to iSCSISession)	Mandatory	
9.8.16 CIM_EndpointOfNetworkPipe (iSCSIConnection to TCPProtocolEndpoint)	Mandatory	
9.8.17 CIM_EndpointOfNetworkPipe (iSCSISession to iSCSIProtocolEndpoint)	Mandatory	
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	

Table 35 - CIM Elements for iSCSI Target Ports

Element Name	Requirement	Description
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	
9.8.38 CIM_iSCSILoginStatistics	Optional	
9.8.39 CIM_iSCSIProtocolEndpoint	Mandatory	
9.8.40 CIM_iSCSISession	Mandatory	
9.8.41 CIM_iSCSISessionFailures	Optional	
9.8.42 CIM_iSCSISessionSettings	Mandatory	
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

Table 35 - CIM Elements for iSCSI Target Ports

Element Name	Requirement	Description
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
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_iSCSISessionSettings	Mandatory	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 36 describes class CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint).

Table 36 - SMI Referenced Properties/Methods for CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)

Properties	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 37 describes class CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint).

Table 37 - SMI Referenced Properties/Methods for CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

9.8.3 CIM_ConcreteDependency

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

Table 38 describes class CIM_ConcreteDependency.

Table 38 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	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 39 describes class CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint).

Table 39 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)

Properties	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 40 describes class CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint).

Table 40 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)

Properties	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 41 describes class CIM_ElementCapabilities (iSCSIConfigurationCapabilities to System).

Table 41 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (iSCSIConfiguration-Capabilities to System)

Properties	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 42 describes class CIM_ElementCapabilities (iSCSIConfigurationCapabilities to iSCSIConfigurationService).

Table 42 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (iSCSIConfiguration-Capabilities to iSCSIConfigurationService)

Properties	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: Mandatory

Table 43 describes class CIM_ElementSettingData (iSCSIConnectionSettings to TCPProtocolEndpoint).

Table 43 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSIConnection-Settings to TCPProtocolEndpoint)

Properties	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: Mandatory

Table 44 describes class CIM_ElementSettingData (iSCSIConnectionSettings to iSCSIProtocolEndpoint).

Table 44 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSIConnection-Settings to iSCSIProtocolEndpoint)

Properties	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: Mandatory

Table 45 describes class CIM_ElementSettingData (iSCSISessionSettings to SCSIProtocolController).

Table 45 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSISessionSettings to SCSIProtocolController)

Properties	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: Mandatory

Table 46 describes class CIM_ElementSettingData (iSCSISessionSettings to System).

Table 46 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSISessionSettings to System)

Properties	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: Mandatory

Table 47 describes class CIM_ElementSettingData (iSCSISessionSettings to iSCSIProtocolEndpoint).

Table 47 - SMI Referenced Properties/Methods for CIM_ElementSettingData (iSCSISessionSettings to iSCSIProtocolEndpoint)

Properties	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 48 describes class CIM_ElementStatisticalData (iSCSILoginStatistics to SCSIProtocolController).

Table 48 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (iSCSILoginStatistics to SCSIProtocolController)

Properties	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: Mandatory

Table 49 describes class CIM_ElementStatisticalData (iSCSISessionFailures to SCSIProtocolController).

Table 49 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (iSCSISessionFailures to SCSIProtocolController)

Properties	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: Mandatory

Table 50 describes class CIM_ElementStatisticalData (iSCSISessionStatistics to iSCSISession).

Table 50 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (iSCSISessionStatistics to iSCSISession)

Properties	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: Mandatory

Table 51 describes class CIM_EndpointOfNetworkPipe (iSCSIConnection to TCPProtocolEndpoint).

Table 51 - SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (iSCSIConnection to TCPProtocolEndpoint)

Properties	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: Mandatory

Table 52 describes class CIM_EndpointOfNetworkPipe (iSCSISession to iSCSIProtocolEndpoint).

Table 52 - SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (iSCSISession to iSCSIProtocolEndpoint)

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

9.8.18 CIM_EthernetPort

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

Table 53 describes class CIM_EthernetPort.

Table 53 - SMI Referenced Properties/Methods for CIM_EthernetPort

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

9.8.19 CIM_HostedAccessPoint (System to IPProtocolEndpoint)

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

Table 54 describes class CIM_HostedAccessPoint (System to IPProtocolEndpoint).

Table 54 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to IPProtocolEndpoint)

Properties	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 55 describes class CIM_HostedAccessPoint (System to TCPProtocolEndpoint).

Table 55 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to TCPProtocolEndpoint)

Properties	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 56 describes class CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint).

Table 56 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

9.8.22 CIM_HostedCollection

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

Table 57 describes class CIM_HostedCollection.

Table 57 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

9.8.23 CIM_HostedService

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

Table 58 describes class CIM_HostedService.

Table 58 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

9.8.24 CIM_IPProtocolEndpoint

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

Table 59 describes class CIM_IPProtocolEndpoint.

Table 59 - SMI Referenced Properties/Methods for CIM_IPProtocolEndpoint

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	

Table 59 - SMI Referenced Properties/Methods for CIM_IPProtocolEndpoint

Properties	Requirement	Description & Notes
Name	Mandatory	
IPv4Address	Optional	
IPv6Address	Optional	
ProtocolIFType	Mandatory	

9.8.25 CIM_MemberOfCollection

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

Table 60 describes class CIM_MemberOfCollection.

Table 60 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	
Collection	Mandatory	

9.8.26 CIM_NetworkPipeComposition

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

Table 61 describes class CIM_NetworkPipeComposition.

Table 61 - SMI Referenced Properties/Methods for CIM_NetworkPipeComposition

Properties	Requirement	Description & Notes
PartComponent	Mandatory	
GroupComponent	Mandatory	

9.8.27 CIM_SAPAvailableForElement

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 62 describes class CIM_SAPAvailableForElement.

Table 62 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	
AvailableSAP	Mandatory	

9.8.28 CIM_SCSIProtocolController

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

Table 63 describes class CIM_SCSIProtocolController.

Table 63 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController

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

9.8.29 CIM_SystemDevice (System to EthernetPort)

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

Table 64 describes class CIM_SystemDevice (System to EthernetPort).

Table 64 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to EthernetPort)

Properties	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 65 describes class CIM_SystemDevice (System to SCSIProtocolController).

Table 65 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIProtocol-Controller)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

9.8.31 CIM_SystemSpecificCollection

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

Table 66 describes class CIM_SystemSpecificCollection.

Table 66 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	

9.8.32 CIM_TCPProtocolEndpoint

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 67 describes class CIM_TCPProtocolEndpoint.

Table 67 - SMI Referenced Properties/Methods for CIM_TCPProtocolEndpoint

Properties	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 68 describes class CIM_iSCSICapabilities.

Table 68 - SMI Referenced Properties/Methods for CIM_iSCSICapabilities

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	
MinimumSpecificationVersion Supported	Mandatory	
MaximumSpecificationVersio nSupported	Mandatory	
AuthenticationMethodsSuppo rted	Mandatory	

9.8.34 CIM_iSCSIConfigurationCapabilities

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

Table 69 describes class CIM_iSCSIConfigurationCapabilities.

Table 69 - SMI Referenced Properties/Methods for CIM_iSCSIConfigurationCapabilities

Properties	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 70 describes class CIM_iSCSIConfigurationService.

Table 70 - SMI Referenced Properties/Methods for CIM_iSCSIConfigurationService

Properties	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 71 describes class CIM_iSCSIConnection.

Table 71 - SMI Referenced Properties/Methods for CIM_iSCSIConnection

Properties	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 72 describes class CIM_iSCSIConnectionSettings.

Table 72 - SMI Referenced Properties/Methods for CIM_iSCSIConnectionSettings

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	
MaxReceiveDataSegmentLength	Mandatory	
PrimaryHeaderDigestMethod	Mandatory	
OtherPrimaryHeaderDigestMethod	Optional	

Table 72 - SMI Referenced Properties/Methods for CIM_iSCSIConnectionSettings

Properties	Requirement	Description & Notes
PrimaryDataDigestMethod	Mandatory	
OtherPrimaryDataDigestMethod	Optional	
SecondaryHeaderDigestMethod	Mandatory	
OtherSecondaryHeaderDigestMethod	Optional	
SecondaryDataDigestMethod	Mandatory	
OtherSecondaryDataDigestMethod	Optional	
RequestingMarkersOnReceive	Mandatory	
PrimaryAuthenticationMethod	Mandatory	
SecondaryAuthenticationMethod	Mandatory	

9.8.38 CIM_iSCSILoginStatistics

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

Table 73 describes class CIM_iSCSILoginStatistics.

Table 73 - SMI Referenced Properties/Methods for CIM_iSCSILoginStatistics

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	
LoginFailures	Optional	
LastLoginFailureTime	Optional	
LastLoginFailureType	Optional	
OtherLastLoginFailureType	Optional	
LastLoginFailureRemoteNodeName	Optional	
LastLoginFailureRemoteAddressType	Optional	

Table 73 - SMI Referenced Properties/Methods for CIM_iSCSILoginStatistics

Properties	Requirement	Description & Notes
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 74 describes class CIM_iSCSIProtocolEndpoint.

Table 74 - SMI Referenced Properties/Methods for CIM_iSCSIProtocolEndpoint

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

Table 75 describes class CIM_iSCSISession.

Table 75 - SMI Referenced Properties/Methods for CIM_iSCSISession

Properties	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 76 describes class CIM_iSCSISessionFailures.

Table 76 - SMI Referenced Properties/Methods for CIM_iSCSISessionFailures

Properties	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: Mandatory

Table 77 describes class CIM_iSCSISessionSettings.

Table 77 - SMI Referenced Properties/Methods for CIM_iSCSISessionSettings

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	
MaxConnectionsPerSession	Mandatory	
InitialR2TPreference	Mandatory	
ImmediateDataPreference	Mandatory	
MaxOutstandingR2T	Mandatory	
MaxUnsolicitedFirstDataBurstLength	Mandatory	
MaxDataBurstLength	Mandatory	
DataSequenceInOrderPreference	Mandatory	

Table 77 - SMI Referenced Properties/Methods for CIM_iSCSISessionSettings

Properties	Requirement	Description & Notes
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 78 describes class CIM_iSCSISessionStatistics.

Table 78 - SMI Referenced Properties/Methods for CIM_iSCSISessionStatistics

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

STABLE

EXPERIMENTAL

Clause 10: Serial Attached SCSI (SAS) Target Port Subprofile

10.1 Synopsis

Profile name: SAS Target Ports

Version: 1.2.0

Organization: SNIA

CIM schema version: 2.11.0

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.

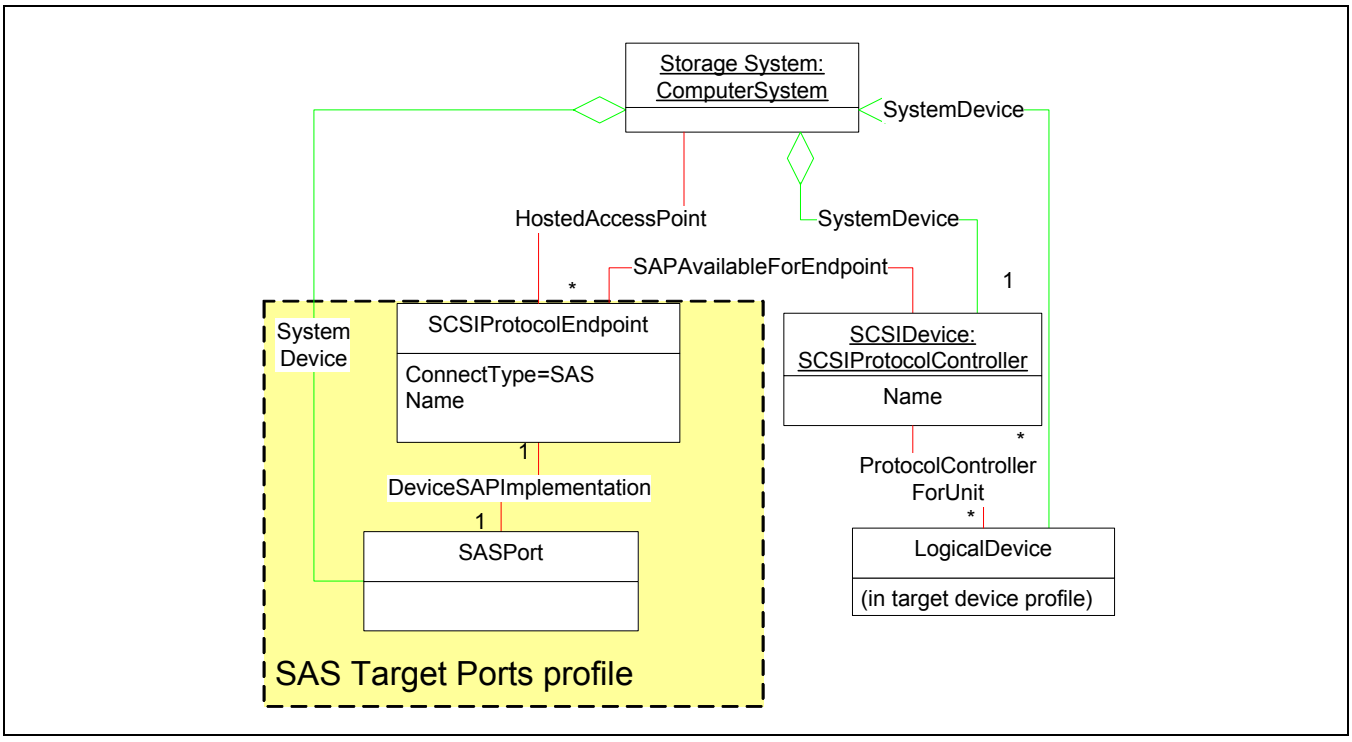

Figure 11 - Serial Attached SCSI (SAS) Target Port Instance Diagram

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

10.2.1 Health and Fault Management

Table 79 describes SASPort OperationalStatus.

Table 79 - 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 80 describes the CIM elements for SAS Target Ports.

Table 80 - CIM Elements for SAS Target Ports

Element Name	Requirement	Description
10.5.1 CIM_ATAProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for ATA.
10.5.2 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.

Table 80 - CIM Elements for SAS Target Ports

Element Name	Requirement	Description
10.5.3 CIM_HostedAccessPoint (ATA)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
10.5.4 CIM_HostedAccessPoint (SCSI)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
10.5.5 CIM_SASPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
10.5.6 CIM_SCSIProtocolEndpoint	Mandatory	Represents a protocol (command set) associated to a port.
10.5.7 CIM_SCSIProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for SCSI.
10.5.8 CIM_SystemDevice	Mandatory	Associates ComputerSystem to LogicalPort.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SASPort	Mandatory	Create SASPort
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SASPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Modify SASPort
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SASPort	Mandatory	Delete SASPort

10.5.1 CIM_ATAProtocolEndpoint

Specialization of ProtocolEndpoint for ATA.

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

Table 81 describes class CIM_ATAProtocolEndpoint.

Table 81 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target).
OtherTypeDescription	Mandatory	Shall be the string 'ATA'.
ConnectionType	Mandatory	Shall be 2|3 (PATA or SATA).

10.5.2 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 82 describes class CIM_DeviceSAPImplementation.

Table 82 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : limit to targets
Antecedent	Mandatory	Validation Property : limit to targets

10.5.3 CIM_HostedAccessPoint (ATA)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 83 describes class CIM_HostedAccessPoint (ATA).

Table 83 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ATA)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem
Dependent	Mandatory	Reference to ATAProtocolEndpointValidation Property : limit to targets

10.5.4 CIM_HostedAccessPoint (SCSI)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 84 describes class CIM_HostedAccessPoint (SCSI).

Table 84 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (SCSI)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Reference to SCSIProtocolEndpointValidation Property : limit to targets

10.5.5 CIM_SASPort

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 85 describes class CIM_SASPort.

Table 85 - SMI Referenced Properties/Methods for CIM_SASPort

Properties	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	Shall be 94 (SAS).
PermanentAddress	Mandatory	SAS Address. Shall be 16 un-separated upper case hex digits.

10.5.6 CIM_SCSIProtocolEndpoint

Represents a protocol (command set) associated to a port.

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

Table 86 describes class CIM_SCSIProtocolEndpoint.

Table 86 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

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

Table 86 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
OtherTypeDescription	Mandatory	Shall be the string 'SCSI', 'ATA', 'SB', or 'iSCSI'. Initiator port specialized profiles specify the appropriate subset.
ConnectionType	Mandatory	Shall be 8 (SAS).

10.5.7 CIM_SCSIProtocolEndpoint

Specialization of ProtocolEndpoint for SCSI.

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

Table 87 describes class CIM_SCSIProtocolEndpoint.

Table 87 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

10.5.8 CIM_SystemDevice

Associates ComputerSystem to LogicalPort.

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

Table 88 describes class CIM_SystemDevice.

Table 88 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 11: Serial ATA (SATA) Target Ports Profile

11.1 Synopsis

Profile name: SATA Target Ports

Version: 1.2.0

Organization: SNIA

CIM schema version: 2.11.0

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.

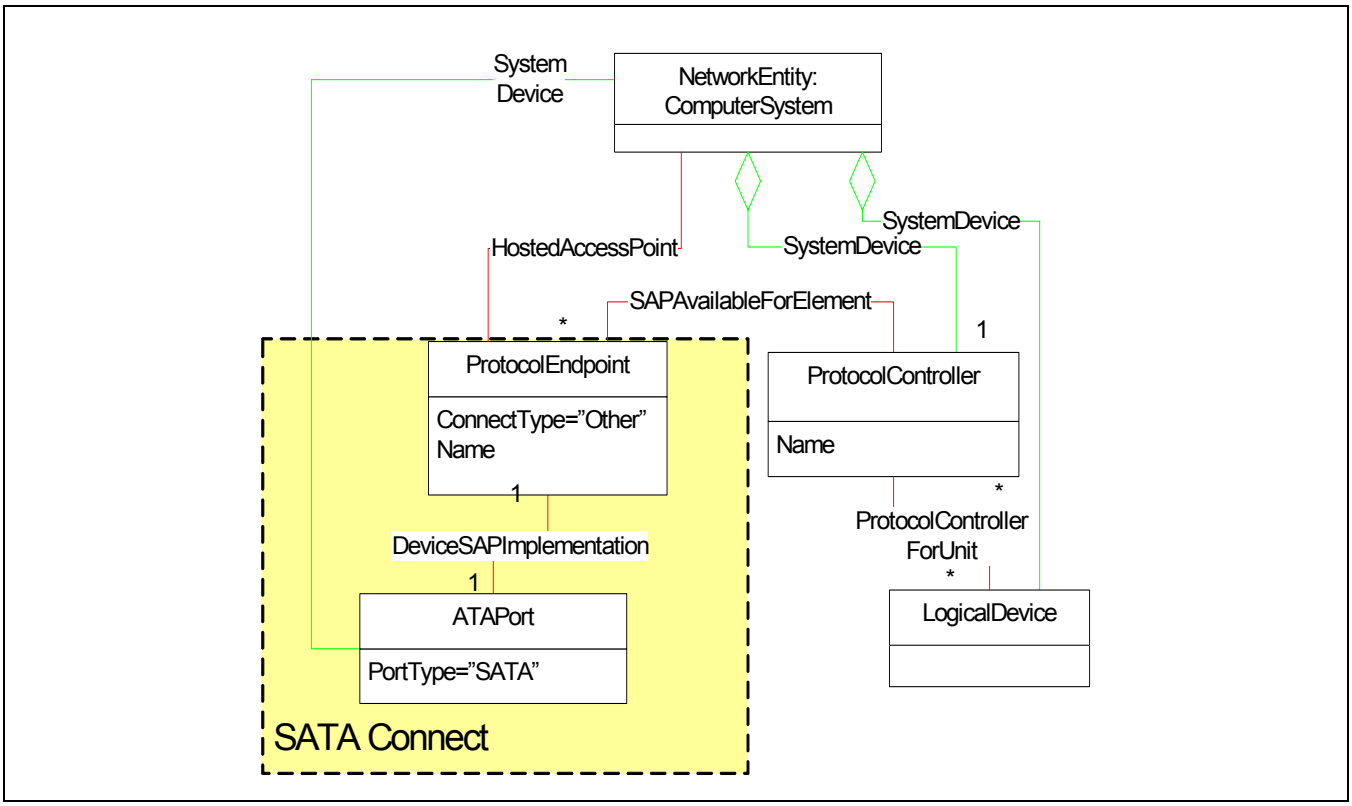

Figure 12 - SATA Target Port Instance Diagram

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 89 describes **ATAPort OperationalStatus.**

Table 89 - 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 90 describes the CIM elements for SATA Target Ports.

Table 90 - 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 a protocol (command set) associated to a port.
11.5.3 CIM_ATAProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for ATA.
11.5.4 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.
11.5.5 CIM_HostedAccessPoint (ATA)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
11.5.6 CIM_HostedAccessPoint (SCSI)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
11.5.7 CIM_SCSIProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for SCSI.

Table 90 - CIM Elements for SATA Target Ports

Element Name	Requirement	Description
11.5.8 CIM_SystemDevice	Mandatory	Associates ComputerSystem to LogicalPort.
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.

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

Table 91 describes class CIM_ATAPort.

Table 91 - SMI Referenced Properties/Methods for CIM_ATAPort

Properties	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	Shall be 92|93 (SATA or SATA2) .

11.5.2 CIM_ATAProtocolEndpoint

Represents a protocol (command set) associated to a port.

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

Table 92 describes class CIM_ATAProtocolEndpoint.

Table 92 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	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', 'SB', or 'iSCSI'. Initiator port specialized profiles specify the appropriate subset.
ConnectionType	Mandatory	Shall be 3 (SATA).

11.5.3 CIM_ATAProtocolEndpoint

Specialization of ProtocolEndpoint for ATA.

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

Table 93 describes class CIM_ATAProtocolEndpoint.

Table 93 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target).
OtherTypeDescription	Mandatory	Shall be the string 'ATA'.
ConnectionType	Mandatory	Shall be 2|3 (PATA or SATA).

11.5.4 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

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	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : limit to targets
Antecedent	Mandatory	Validation Property : limit to targets

11.5.5 CIM_HostedAccessPoint (ATA)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 95 describes class CIM_HostedAccessPoint (ATA).

Table 95 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ATA)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem
Dependent	Mandatory	Reference to ATAProtocolEndpointValidation Property : limit to targets

11.5.6 CIM_HostedAccessPoint (SCSI)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 96 describes class CIM_HostedAccessPoint (SCSI).

Table 96 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (SCSI)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Reference to SCSIProtocolEndpointValidation Property : limit to targets

11.5.7 CIM_SCSIProtocolEndpoint

Specialization of ProtocolEndpoint for SCSI.

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 97 describes class CIM_SCSIProtocolEndpoint.

Table 97 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

11.5.8 CIM_SystemDevice

Associates ComputerSystem to LogicalPort.

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

Table 98 describes class CIM_SystemDevice.

Table 98 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 12: SB Target Port Profile

12.1 Synopsis

Profile Name: SB Target Port

Version: 1.2.0

Organization: SNIA

CIM schema version: 2.13.0

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.

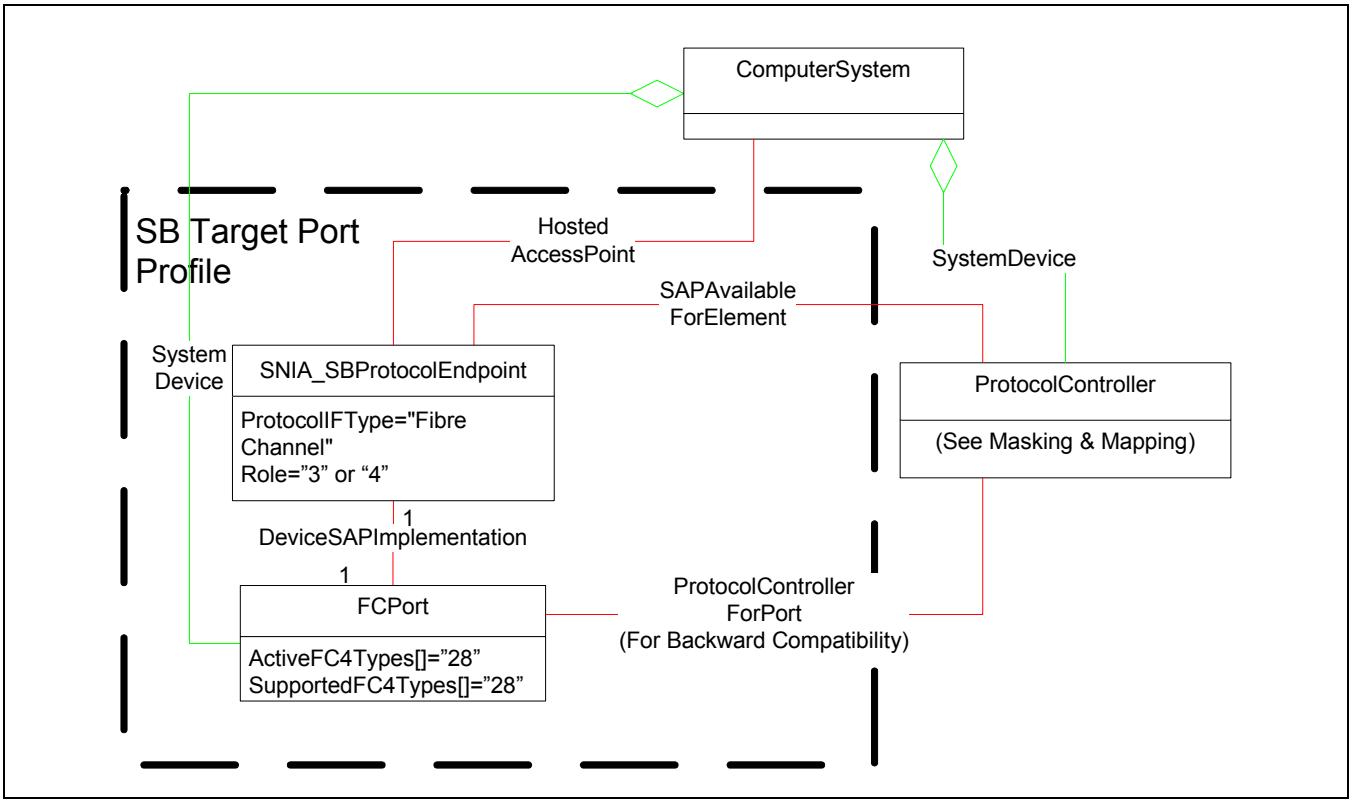

Figure 13 - SB Target Port Instance Diagram

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

Table 99 - FCPort OperationalStatus

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

Table 99 - FCPort OperationalStatus

OperationalStatus	Description
InService	Port is in self test
Unknown	

12.5 Cascading Considerations

None

12.6 Supported Profiles, Subprofiles, and Packages

Profile Name: SB Target Ports

Version: 1.3.0

Organization: SNIA

CIM Schema Version: 2.13

Table 100 describes the related profiles for SB Target Ports.

Table 100 - Related Profiles for SB Target Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.3.0	Mandatory	

12.7 Methods of the Profile

12.7.1 Extrinsic Methods of the Profile

None

12.7.2 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

12.8 Client Considerations and Recipes

None

12.9 CIM Elements

Table 101 describes the CIM elements for SB Target Ports.

Table 101 - CIM Elements for SB Target Ports

Element Name	Requirement	Description
12.9.1 CIM_ATAProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for ATA.
12.9.2 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.
12.9.3 CIM_FCPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
12.9.4 CIM_HostedAccessPoint (ATA)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
12.9.5 CIM_HostedAccessPoint (SCSI)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
12.9.6 CIM_SCSIProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for SCSI.
12.9.7 CIM_SystemDevice	Mandatory	Associates ComputerSystem to LogicalPort.
12.9.8 SNIA_SBProtocolEndpoint	Mandatory	Represents a protocol (command set) associated to a port.
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_InstDeletion WHERE SourceInstance ISA CIM_FCPort	Mandatory	Delete FCPort

12.9.1 CIM_ATAProtocolEndpoint

Specialization of ProtocolEndpoint for ATA.

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

Table 102 describes class CIM_ATAProtocolEndpoint.

Table 102 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target).
OtherTypeDescription	Mandatory	Shall be the string 'ATA'.
ConnectionType	Mandatory	Shall be 2|3 (PATA or SATA).

12.9.2 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

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

Table 103 describes class CIM_DeviceSAPImplementation.

Table 103 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : limit to targets
Antecedent	Mandatory	Validation Property : limit to targets

12.9.3 CIM_FCPort

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 104 describes class CIM_FCPort.

Table 104 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	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.
PermanentAddress	Mandatory	Port WWN. Shall be 16 unseparated uppercase hex digits.
SupportedCOS	Optional	
ActiveCOS	Optional	
SupportedFC4Types	Mandatory	For SB Target Ports this array shall contain 28 (FC-SB-2 Control Unit).
ActiveFC4Types	Mandatory	For SB Target Ports this array should contain 28 (FC-SB-2 Control Unit).
Caption	Optional	Not Specified in this version of the Profile
Description	Optional	Not Specified in this version of the Profile
ElementName	Optional	Not Specified in this version of the Profile
InstallDate	Optional	Not Specified in this version of the Profile.
Name	Optional	Not Specified in this version of the Profile.
StatusDescriptions	Optional	Not Specified in this version of the Profile.
HealthState	Optional	Not Specified in this version of the Profile.
EnabledState	Optional	Not Specified in this version of the Profile.
OtherEnabledState	Optional	Not Specified in this version of the Profile.
RequestedState	Optional	Not Specified in this version of the Profile.
EnabledDefault	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	Optional	Not Specified in this version of the Profile.
OtherIdentifyingInfo	Optional	Not Specified in this version of the Profile.
IdentifyingDescriptions	Optional	Not Specified in this version of the Profile.
AdditionalAvailability	Optional	Not Specified in this version of the Profile.

Table 104 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	Requirement	Description & Notes
LocationIndicator	Optional	Not Specified in this version of the Profile.
Speed	Optional	Not Specified in this version of the Profile
MaxSpeed	Optional	Not Specified in this version of the Profile
RequestedSpeed	Optional	Not Specified in this version of the Profile
OtherPortType	Optional	Not Specified in this version of the Profile.
PortNumber	Optional	Not Specified in this version of the Profile
OtherLinkTechnology	Optional	Not Specified in this version of the Profile.
NetworkAddresses	Optional	Not Specified in this version of the Profile.
FullDuplex	Optional	Not Specified in this version of the Profile.
AutoSense	Optional	Not Specified in this version of the Profile.
SupportedMaximumTransmissionUnit	Optional	Not Specified in this version of the Profile.
ActiveMaximumTransmissionUnit	Optional	Not Specified in this version of the Profile.

12.9.4 CIM_HostedAccessPoint (ATA)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 105 describes class CIM_HostedAccessPoint (ATA).

Table 105 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ATA)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem
Dependent	Mandatory	Reference to ATAProtocolEndpointValidation Property : limit to targets

12.9.5 CIM_HostedAccessPoint (SCSI)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 106 describes class CIM_HostedAccessPoint (SCSI).

Table 106 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (SCSI)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Reference to SCSIProtocolEndpointValidation Property : limit to targets

12.9.6 CIM_SCSIProtocolEndpoint

Specialization of ProtocolEndpoint for SCSI.

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

Table 107 describes class CIM_SCSIProtocolEndpoint.

Table 107 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

12.9.7 CIM_SystemDevice

Associates ComputerSystem to LogicalPort.

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

Table 108 describes class CIM_SystemDevice.

Table 108 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

12.9.8 SNIA_SBProtocolEndpoint

Represents a protocol (command set) associated to a port.

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

Table 109 describes class SNIA_SBProtocolEndpoint.

Table 109 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ProtocolIFType	Mandatory	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be the string 'SB'.
ConnectionType	Mandatory	Shall be 2 (Fibre Channel)
Caption	Optional	Not Specified in this version of the Profile.
ElementName	Optional	Not Specified in this version of the Profile.
InstallDate	Optional	Not Specified in this version of the Profile.
StatusDescriptions	Optional	Not Specified in this version of the Profile.
HealthState	Optional	Not Specified in this version of the Profile.
EnabledDefault	Optional	Not Specified in this version of the Profile.
BroadcastResetSupported	Optional	Not Specified in this version of the Profile.

EXPERIMENTAL

EXPERIMENTAL

Clause 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., ISA, EISA, 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/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.

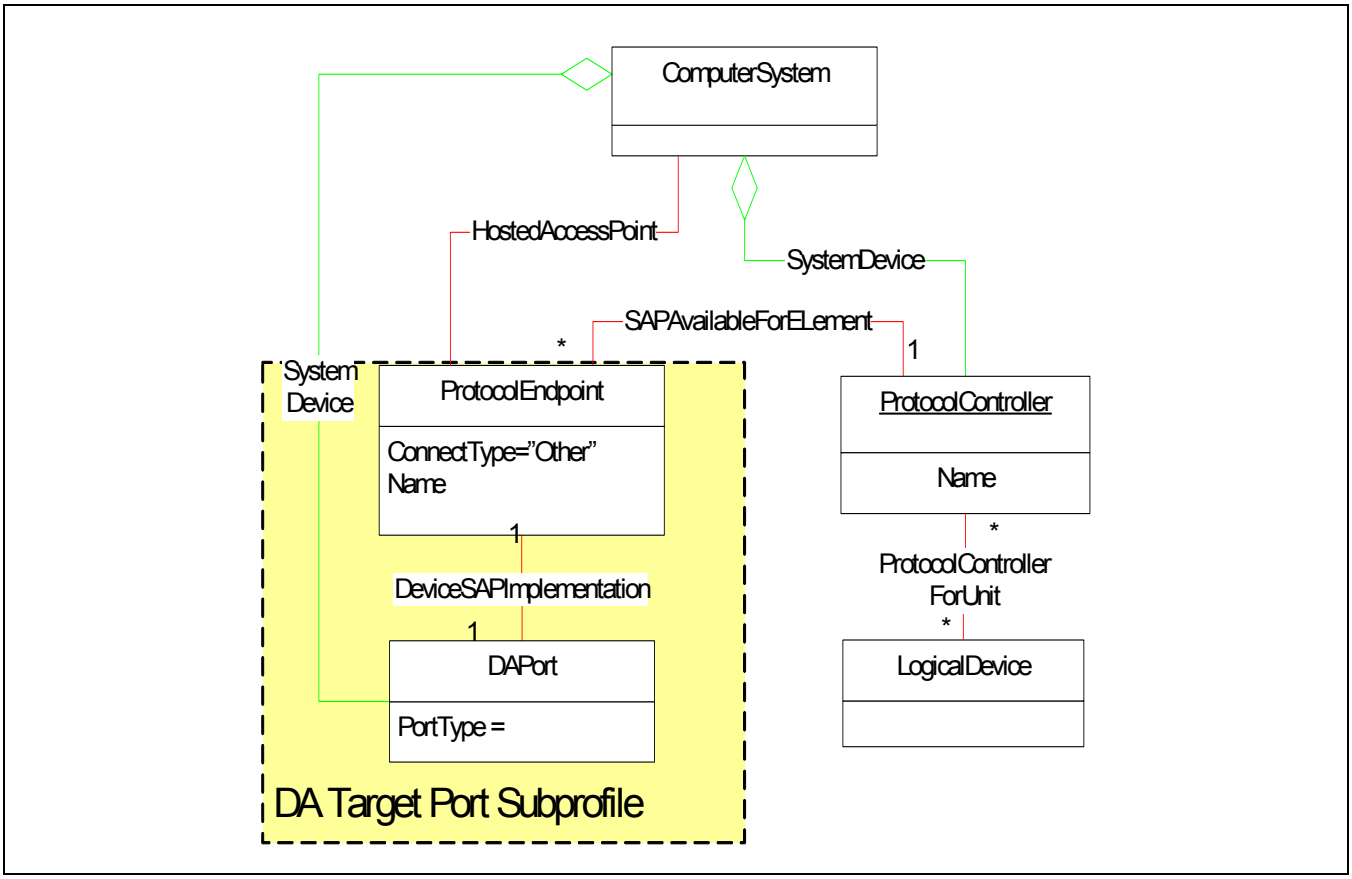

Figure 14 - DA Port Instance Diagram

The DAPort class is connected to the ProtocolEndpoint and optionally to a PhysicalPackage. The DAPort also contains a port type attribute to identify the interconnect technology.

13.2 Health and Fault Management

Table 110 describes DAPort OperationalStatus.

Table 110 - DAPort OperationalStatus

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

13.3 Supported Profiles and Packages

None

13.4 Extrinsic Methods

None

13.5 Client Considerations and Recipes

13.6 CIM Elements

Table 111 describes the CIM elements for DA Target Ports.

Table 111 - CIM Elements for DA Target Ports

Element Name	Requirement	Description
13.6.1 CIM_ATAProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for ATA.
13.6.2 CIM_DAPort	Mandatory	Represents the logical aspects of the physical port and may have multiple associated protocols.
13.6.3 CIM_DeviceSAPImplementation	Mandatory	Associates front-end LogicalPort and target ProtocolEndpoint.
13.6.4 CIM_HostedAccessPoint (ATA)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
13.6.5 CIM_HostedAccessPoint (SCSI)	Mandatory	Associates ComputerSystem to ProtocolEndpoint.
13.6.6 CIM_ProtocolEndpoint	Mandatory	Represents a protocol (command set) associated to a port.
13.6.7 CIM_SCSIProtocolEndpoint	Optional	Specialization of ProtocolEndpoint for SCSI.
13.6.8 CIM_SystemDevice	Mandatory	Associates ComputerSystem to LogicalPort.

13.6.1 CIM_ATAProtocolEndpoint

Specialization of ProtocolEndpoint for ATA.

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

Table 112 describes class CIM_ATAProtocolEndpoint.

Table 112 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target).
OtherTypeDescription	Mandatory	Shall be the string 'ATA'.
ConnectionType	Mandatory	Shall be 2|3 (PATA or SATA).

13.6.2 CIM_DAPort

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 113 describes class CIM_DAPort.

Table 113 - SMI Referenced Properties/Methods for CIM_DAPort

Properties	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	Set to the type of port this DAPort emulates.

13.6.3 CIM_DeviceSAPImplementation

Associates front-end LogicalPort and target ProtocolEndpoint.

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 114 describes class CIM_DeviceSAPImplementation.

Table 114 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : limit to targets
Antecedent	Mandatory	Validation Property : limit to targets

13.6.4 CIM_HostedAccessPoint (ATA)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 115 describes class CIM_HostedAccessPoint (ATA).

Table 115 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ATA)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem
Dependent	Mandatory	Reference to ATAProtocolEndpointValidation Property : limit to targets

13.6.5 CIM_HostedAccessPoint (SCSI)

Associates ComputerSystem to ProtocolEndpoint. Limit to targets (Role = 3).

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

Table 116 describes class CIM_HostedAccessPoint (SCSI).

Table 116 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (SCSI)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Reference to SCSIProtocolEndpointValidation Property : limit to targets

13.6.6 CIM_ProtocolEndpoint

Represents a protocol (command set) associated to a port.

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

Table 117 describes class CIM_ProtocolEndpoint.

Table 117 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint

Properties	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', 'SB', or 'iSCSI'. Initiator port specialized profiles specify the appropriate subset.

13.6.7 CIM_SCSIProtocolEndpoint

Specialization of ProtocolEndpoint for SCSI.

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

Table 118 describes class CIM_SCSIProtocolEndpoint.

Table 118 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Requirement	Description & Notes
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).

13.6.8 CIM_SystemDevice

Associates ComputerSystem to LogicalPort.

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

Requirement: Mandatory

Table 119 describes class CIM_SystemDevice.

Table 119 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 14: Generic Initiator Ports Profile

14.1 Synopsis

Profile name: Generic Initiator Ports

Version: 1.0.0

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 Clause 6: Generic Target Ports Profile)

14.3 Implementation

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

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.

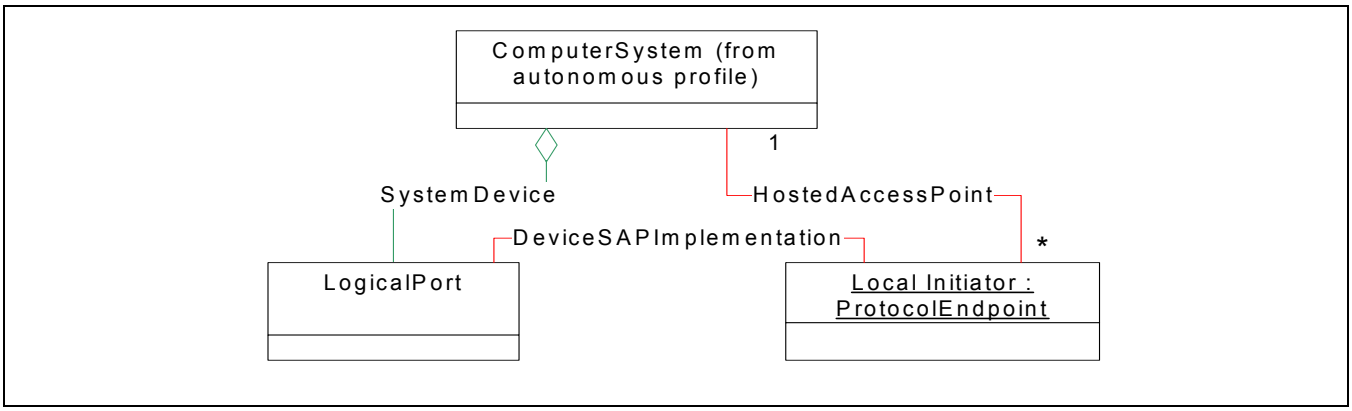

Figure 15 - Generic Initiator Port Model

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.

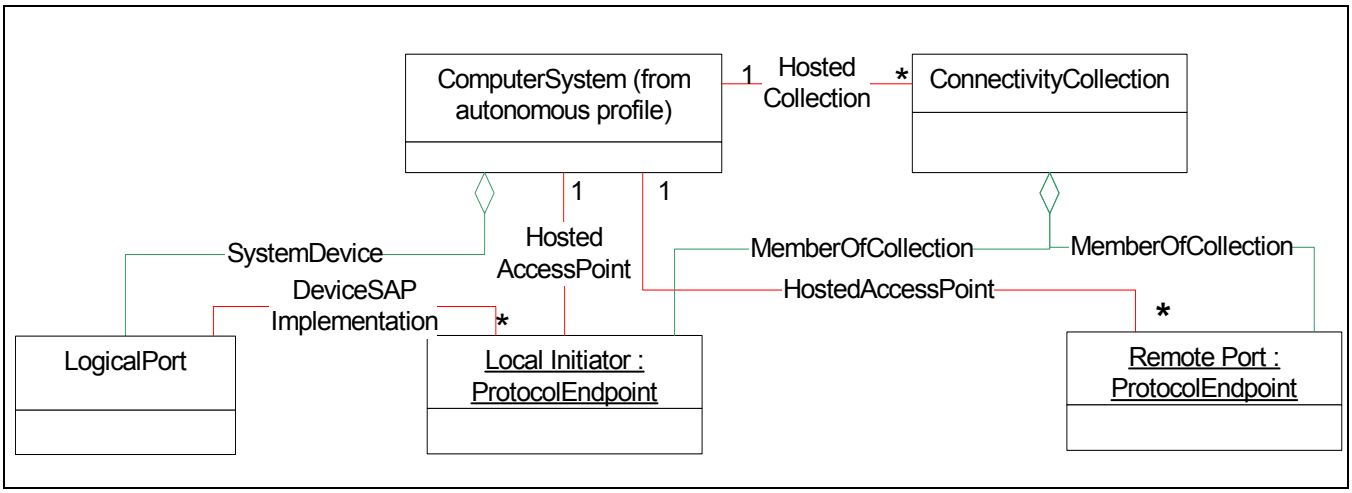

Figure 16 - Optional Connectivity Collection Model

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.

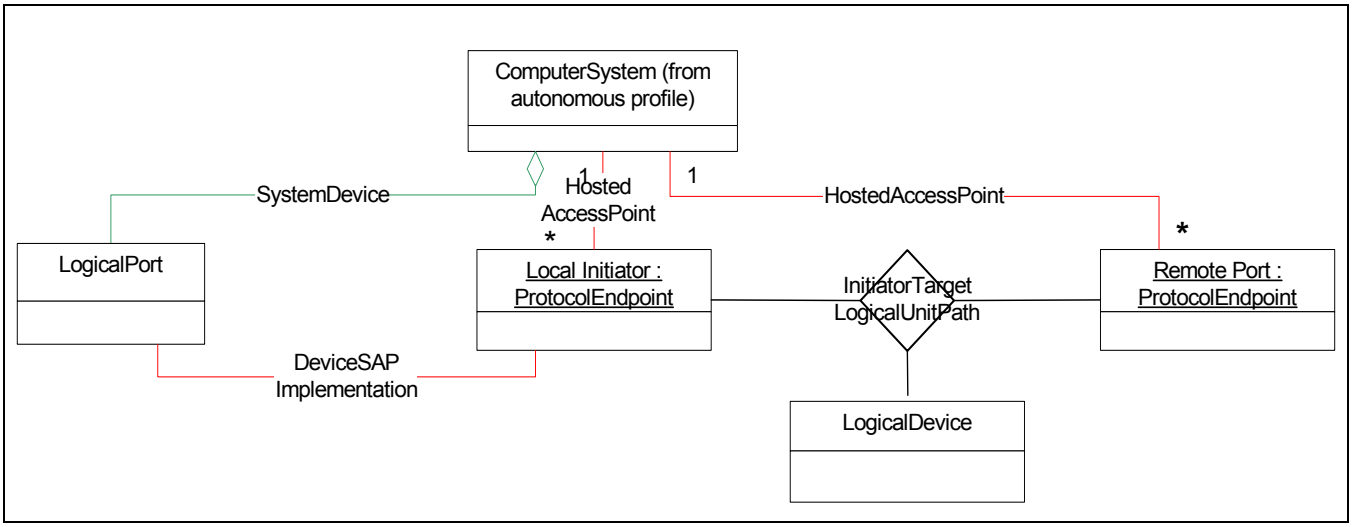

Figure 17 - Optional Full-Path Model

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

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 are SCSI or ATA. This association references LogicalDisk 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 LogicalDisk 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.

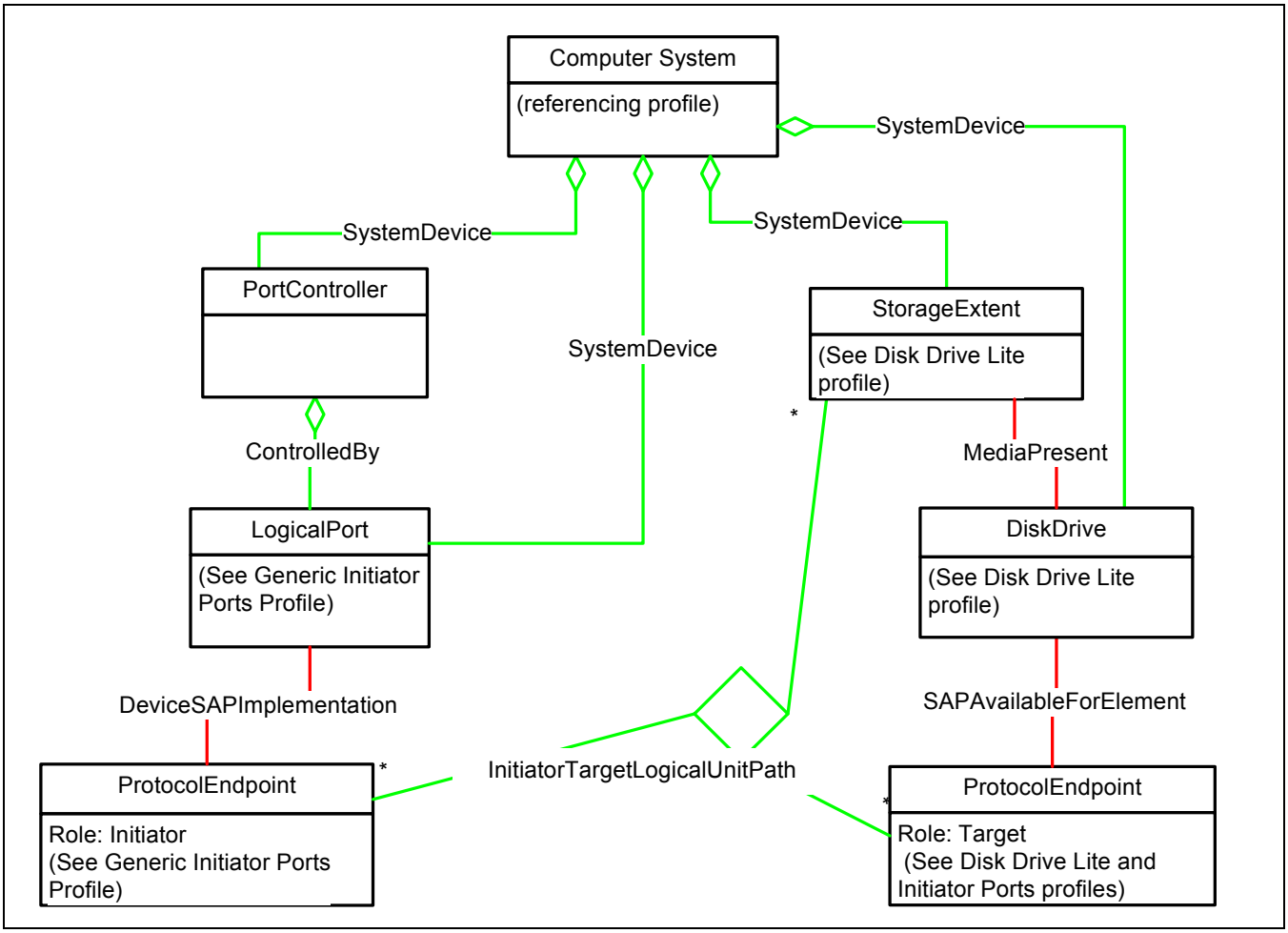

Figure 18 - HBA and Disk Model

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.

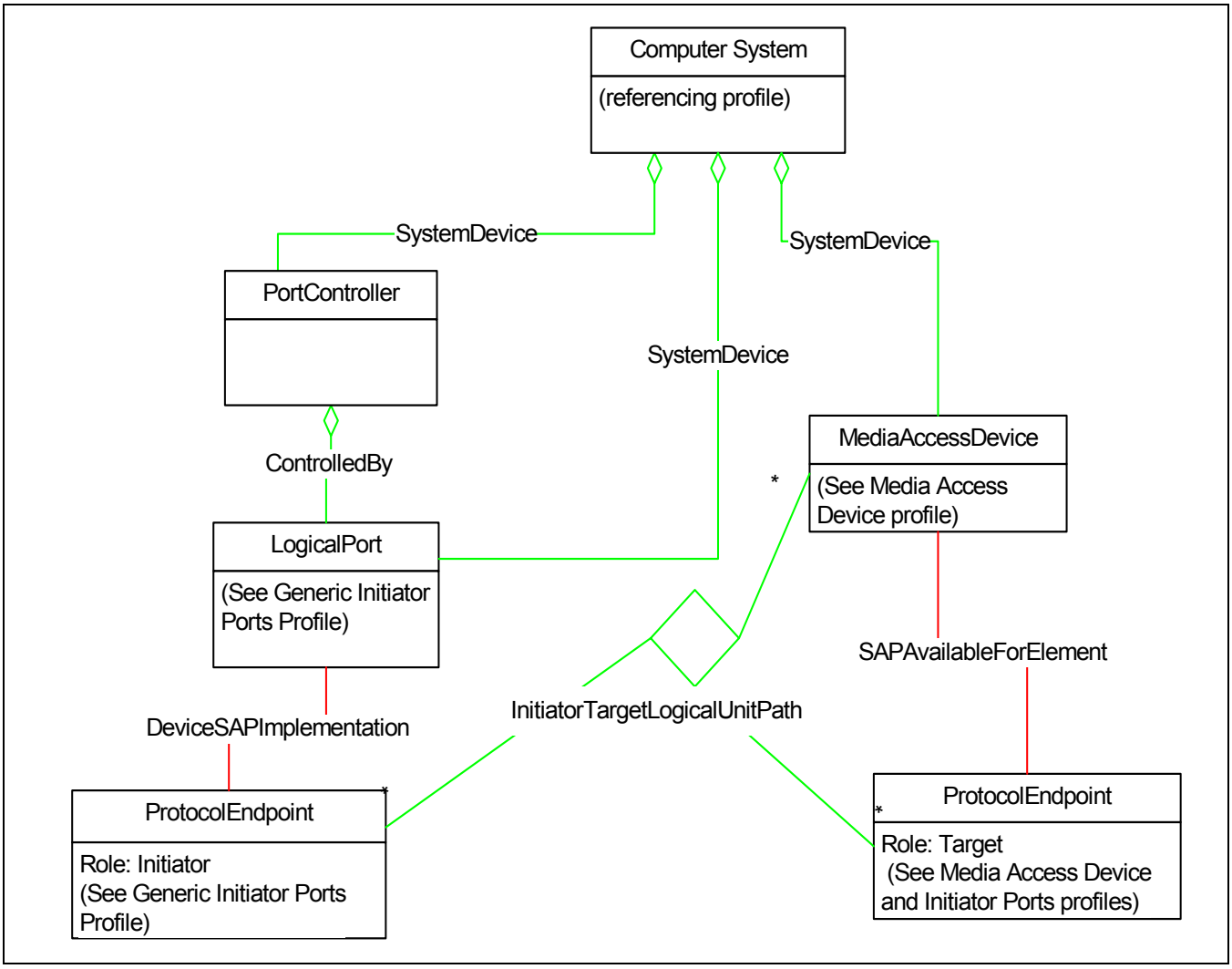

Figure 19 - HBA and Tape or Optical Devices

14.3.2 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 Detailed Use Cases and Recipes

The optional remote element models are not modifiable by clients. Implementation support for these options may be determined by looking for instances of MemberOfCollection or InitiatorTargetLogicalUnitPath associations referencing initiator ProtocolEndpoints,

14.6 CIM Elements

Table 120 describes the CIM elements for Generic Initiator Ports.

Table 120 - 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_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
14.6.4 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
14.6.5 CIM_HostedCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates the ConnectivityCollection to the hosting System.
14.6.6 CIM_LogicalPort	Mandatory	
14.6.7 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates ProtocolEndpoints to the ConnectivityCollection

Table 120 - CIM Elements for Generic Initiator Ports

Element Name	Requirement	Description
14.6.8 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
14.6.9 CIM_SystemDevice (Non-port devices)	Optional	Associates system to ports and optional logical unit LogicalDevices

14.6.1 CIM_ConnectivityCollection

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

Table 121 describes class CIM_ConnectivityCollection.

Table 121 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	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 122 describes class CIM_DeviceSAPImplementation.

Table 122 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : restrict to initiators
Antecedent	Mandatory	Validation Property : restrict to back-end ports (initiators)

14.6.3 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

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	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Initiators

14.6.4 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

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	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Targets

14.6.5 CIM_HostedCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 125 describes class CIM_HostedCollection.

Table 125 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

14.6.6 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 126 describes class CIM_LogicalPort.

Table 126 - SMI Referenced Properties/Methods for CIM_LogicalPort

Properties	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	Mandatory	Initiator port specialized profiles specify the appropriate subset of values.

14.6.7 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 127 describes class CIM_MemberOfCollection.

Table 127 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	The reference to the ProtocolEndpoint
Collection	Mandatory	The reference to the ConnectivityCollection

14.6.8 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

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

Table 128 describes class CIM_SystemDevice (Initiator Ports).

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

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem
PartComponent	Mandatory	Reference to LogicalPort.Validation Property : Restrict to back-end or unrestricted ports.

14.6.9 CIM_SystemDevice (Non-port devices)

Associates system to ports and optional logical unit LogicalDevices

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

Table 129 describes class CIM_SystemDevice (Non-port devices).

Table 129 - SMI Referenced Properties/Methods for CIM_SystemDevice (Non-port devices)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	Reference to non-port devices.

EXPERIMENTAL

EXPERIMENTAL

Clause 15: Parallel SCSI (SPI) Initiator Ports Profile

15.1 Synopsis

Profile Name: SPI Initiator Ports

Version: 1.2.0

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 profiles 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 20.

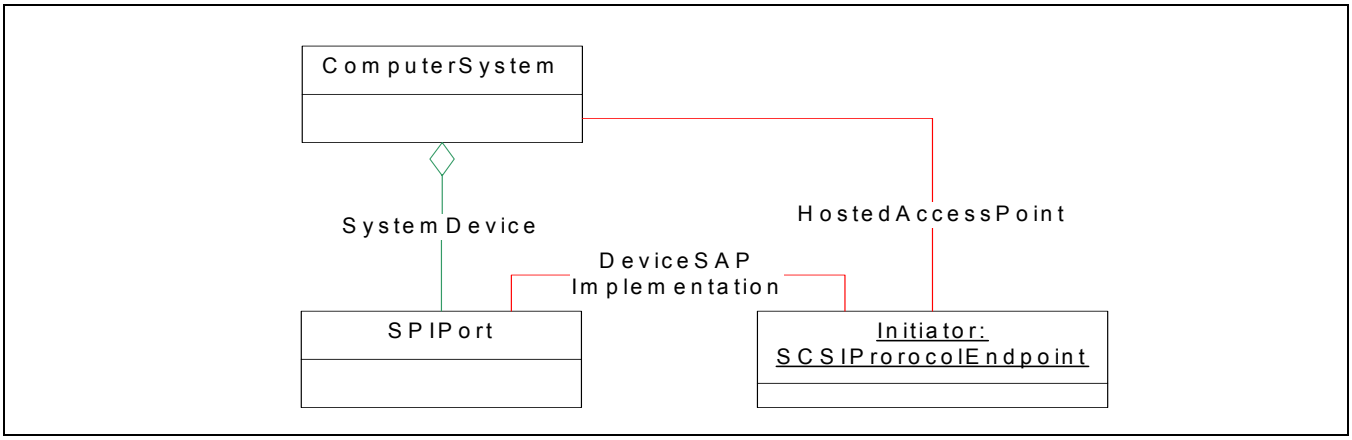

Figure 20 - SPI Initiator Port Instance Diagram

15.3.1 Health and Fault Management Considerations

Table 130 summarizes the Health and Fault Management issues that are unique to this profile.

Table 130 - 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 131 describes the CIM elements for SPI Initiator Ports.

Table 131 - CIM Elements for SPI Initiator Ports

Element Name	Requirement	Description
15.6.1 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
15.6.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator LogicalPort and ProtocolEndpoint
15.6.3 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
15.6.4 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
15.6.5 CIM_HostedCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates the ConnectivityCollection to the hosting System.
15.6.6 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates ProtocolEndpoints to the ConnectivityCollection
15.6.7 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
15.6.8 CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)	Mandatory	
15.6.9 CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)	Optional	Models remote ports - target devices and possibly other initiators.
15.6.10 CIM_SPIPort	Mandatory	
15.6.11 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
15.6.12 CIM_SystemDevice (Non-port devices)	Optional	Associates system to ports and optional logical unit LogicalDevices

15.6.1 CIM_ConnectivityCollection

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

Table 132 describes class CIM_ConnectivityCollection.

Table 132 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	

15.6.2 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint

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

Table 133 describes class CIM_DeviceSAPImplementation.

Table 133 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : restrict to initiators
Antecedent	Mandatory	Validation Property : restrict to back-end ports (initiators)

15.6.3 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

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

Table 134 describes class CIM_HostedAccessPoint (Initiator).

Table 134 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Initiators

15.6.4 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

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

Table 135 describes class CIM_HostedAccessPoint (Target).

Table 135 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Targets

15.6.5 CIM_HostedCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 136 describes class CIM_HostedCollection.

Table 136 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

15.6.6 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 137 describes class CIM_MemberOfCollection.

Table 137 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	The reference to the ProtocolEndpoint
Collection	Mandatory	The reference to the ConnectivityCollection

15.6.7 CIM_SCSIInitiatorTargetLogicalUnitPath

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Optional

Table 138 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 138 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Requirement	Description & Notes
LogicalUnit	Mandatory	
Target	Mandatory	
Initiator	Mandatory	

15.6.8 CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)

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

Table 139 describes class CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint).

Table 139 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI)
Role	Mandatory	Shall be 2 (Initiator) or 4 (Both Initiator and Target)

15.6.9 CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)

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

Table 140 describes class CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint).

Table 140 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ProtocolIFType	Mandatory	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be 'SCSI'.
ConnectionType	Mandatory	Shall be 3 (Parallel SCSI).
Role	Mandatory	Should be set appropriately by the instrumentation. If not know, use 0 (Unknown).

15.6.10 CIM_SPIPort

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 141 describes class CIM_SPIPort.

Table 141 - SMI Referenced Properties/Methods for CIM_SPIPort

Properties	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	Mandatory	Initiator port specialized profiles specify the appropriate subset of values.

15.6.11 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

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

Table 142 describes class CIM_SystemDevice (Initiator Ports).

Table 142 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem
PartComponent	Mandatory	Reference to LogicalPort.Validation Property : Restrict to back-end or unrestricted ports.

15.6.12 CIM_SystemDevice (Non-port devices)

Associates system to ports and optional logical unit LogicalDevices

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

Table 143 describes class CIM_SystemDevice (Non-port devices).

Table 143 - SMI Referenced Properties/Methods for CIM_SystemDevice (Non-port devices)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	Reference to non-port devices.

EXPERIMENTAL

EXPERIMENTAL

Clause 16: iSCSI Initiator Port Profile

16.1 Synopsis

Profile Name: iSCSI Initiator Ports

Version: 1.2.0

Organization: SNIA

CIM Schema Version: TBD

Table 144 describes the related profiles for iSCSI Initiator Ports.

Table 144 - Related Profiles for iSCSI Initiator Ports

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.3.0	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 21 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.

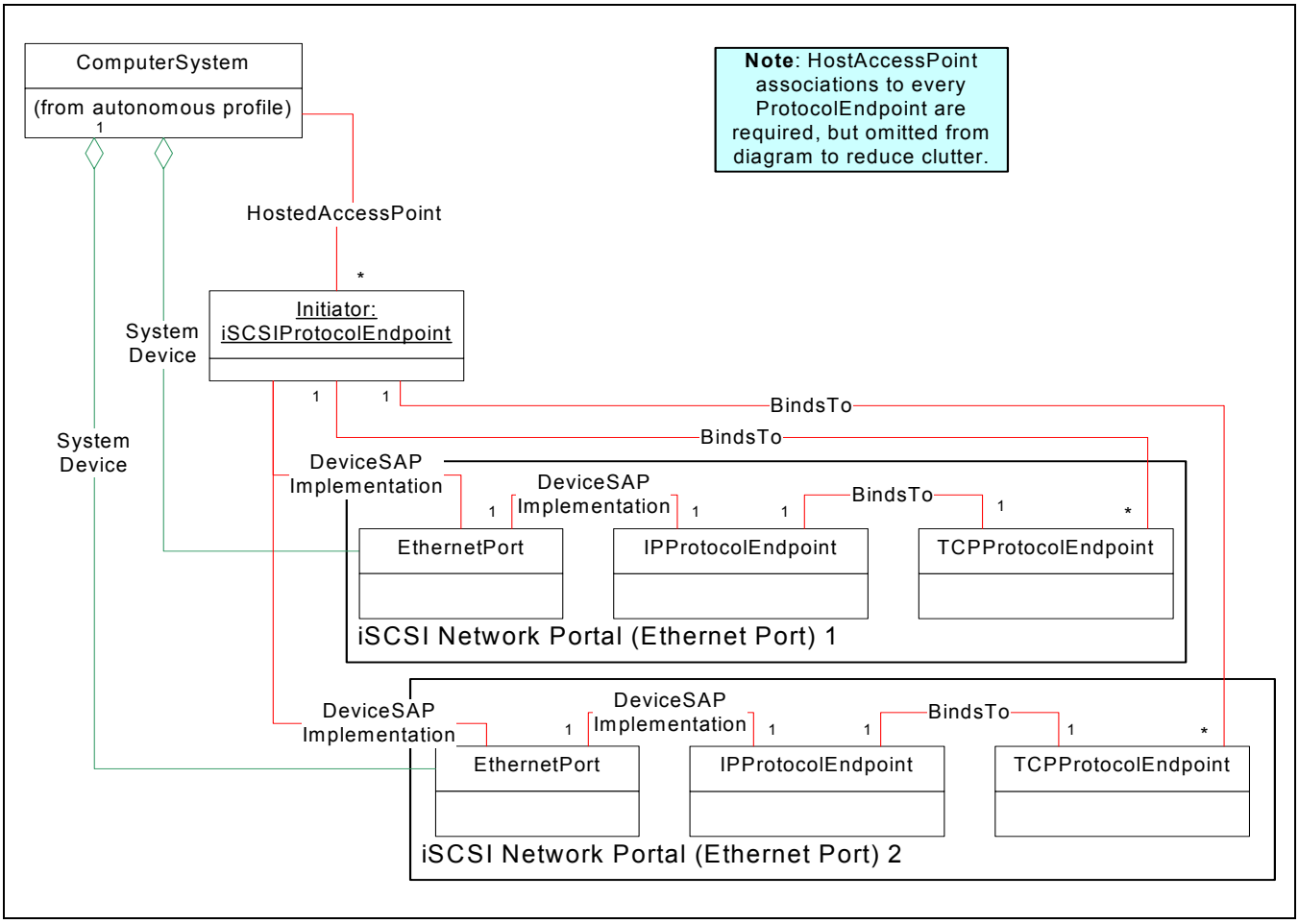

Figure 21 - iSCSI Initiator Port Instance Diagram

16.3.1 Health and Fault Management Considerations

Table 145 describes **EthernetPort OperationalStatus.**

Table 145 - 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 144 describes the CIM elements for iSCSI Initiator Ports.

Table 146 - CIM Elements for iSCSI Initiator Ports

Element Name	Requirement	Description
16.6.1 CIM_BindsTo	Mandatory	
16.6.2 CIM_DeviceSAPImplementation (IPProtocolEndpoint to EthernetPort)	Mandatory	
16.6.3 CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthenetPort)	Mandatory	
16.6.4 CIM_EthernetPort	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	Mandatory	
16.6.9 CIM_LogicalDevice	Optional	

Table 146 - 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	Mandatory	
16.6.13 CIM_iSCSIProtocolEndpoint	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

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

Table 147 describes class CIM_BindsTo.

Table 147 - SMI Referenced Properties/Methods for CIM_BindsTo

Properties	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 148 describes class CIM_DeviceSAPImplementation (IPProtocolEndpoint to EthernetPort).

Table 148 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (IPProtocolEndpoint to EthernetPort)

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

16.6.3 CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthenetPort)

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

Table 149 describes class CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthenetPort).

Table 149 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (iSSIProtocolEndpoint to EthenetPort)

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

16.6.4 CIM_EthernetPort

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

Table 150 describes class CIM_EthernetPort.

Table 150 - SMI Referenced Properties/Methods for CIM_EthernetPort

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

Table 150 - SMI Referenced Properties/Methods for CIM_EthernetPort

Properties	Requirement	Description & Notes
OperationalStatus	Mandatory	
PermanentAddress	Mandatory	

16.6.5 CIM_HostedAccessPoint (System to IPProtocolEndpoint)

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

Table 151 describes class CIM_HostedAccessPoint (System to IPProtocolEndpoint).

Table 151 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to IPProtocolEndpoint)

Properties	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 152 describes class CIM_HostedAccessPoint (System to TCPProtocolEndpoint).

Table 152 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to TCPProtocolEndpoint)

Properties	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 153 describes class CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint).

Table 153 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (System to iSCSIProtocolEndpoint)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

16.6.8 CIM_IPProtocolEndpoint

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

Table 154 describes class CIM_IPProtocolEndpoint.

Table 154 - SMI Referenced Properties/Methods for CIM_IPProtocolEndpoint

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

16.6.9 CIM_LogicalDevice

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

Table 155 describes class CIM_LogicalDevice.

Table 155 - SMI Referenced Properties/Methods for CIM_LogicalDevice

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

16.6.10 CIM_SystemDevice (System to EthernetPort)

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

Table 156 describes class CIM_SystemDevice (System to EthernetPort).

Table 156 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to EthernetPort)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

16.6.11 CIM_SystemDevice (System to LogicalDevice)

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

Table 157 describes class CIM_SystemDevice (System to LogicalDevice).

Table 157 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to LogicalDevice)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

16.6.12 CIM_TCPProtocolEndpoint

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

Table 158 describes class CIM_TCPProtocolEndpoint.

Table 158 - SMI Referenced Properties/Methods for CIM_TCPProtocolEndpoint

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

16.6.13 CIM_iSCSIProtocolEndpoint

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

Table 159 describes class CIM_iSCSIProtocolEndpoint.

Table 159 - SMI Referenced Properties/Methods for CIM_iSCSIProtocolEndpoint

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

STABLE

Clause 17: Fibre Channel Initiator Port Profile

17.1 Synopsis

Profile Name: FC Initiator Ports

Version: 1.3.0

Organization: SNIA

CIM Schema Version: 2.9.0

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

Specializes: Generic Initiator Ports Profile

Central Class: CIM_FCPort

Scoping Class: a CIM_System in a referencing autonomous profile

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

17.2 Description

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

17.3 Implementation

Figure 22 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 Clause 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.

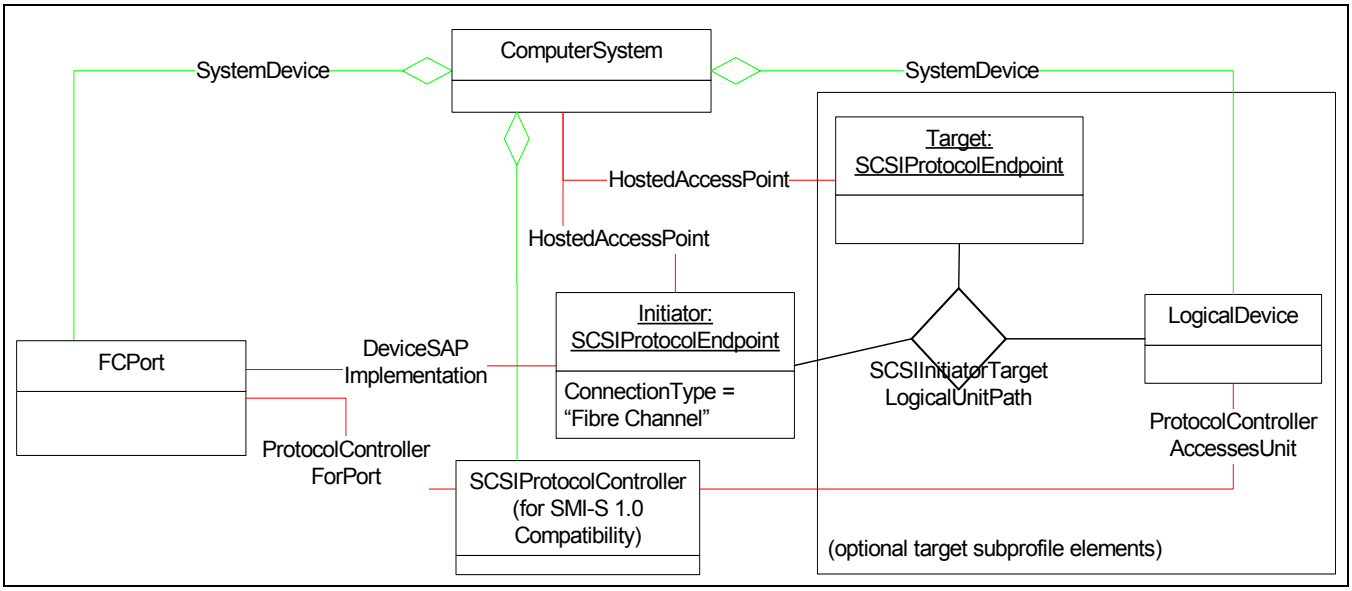

Figure 22 - Fibre Channel Initiator Instance Diagram

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.

17.3.3 Health and Fault Management Considerations

Table 160 summarized the Health and Fault Management considerations specific to this profile.

Table 160 - 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 Detailed Use Cases and Recipes

None

17.6 CIM Elements

Table 161 describes the CIM elements for FC Initiator Ports.

Table 161 - 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 LogicalPort and ProtocolEndpoint
17.6.3 CIM_FCPort	Mandatory	
17.6.4 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
17.6.5 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
17.6.6 CIM_HostedCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates the ConnectivityCollection to the hosting System.

Table 161 - CIM Elements for FC Initiator Ports

Element Name	Requirement	Description
17.6.7 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates ProtocolEndpoints to the ConnectivityCollection
17.6.8 CIM_ProtocolControllerForPort	Optional	
17.6.9 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
17.6.10 CIM_SCSIProtocolController	Optional	Represents a SCSI logical unit inventory.
17.6.11 CIM_SCSIProtocolEndpoint (Initiator)	Mandatory	
17.6.12 CIM_SCSIProtocolEndpoint (Target)	Optional	Models remote ports - target devices and possibly other initiators.
17.6.13 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
17.6.14 CIM_SystemDevice (Non-port devices)	Optional	Associates system to ports and optional logical unit LogicalDevices
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

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

Table 162 describes class CIM_ConnectivityCollection.

Table 162 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	

17.6.2 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint

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

Table 163 describes class CIM_DeviceSAPImplementation.

Table 163 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : restrict to initiators
Antecedent	Mandatory	Validation Property : restrict to back-end ports (initiators)

17.6.3 CIM_FCPort

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 164 describes class CIM_FCPort.

Table 164 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	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.

Table 164 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	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).
ElementName	Mandatory	Port Symbolic Name
Speed	Mandatory	Shall be 0 or 1062500000 or 2125000000 or 4250000000 or 10518750000 or 12750000000.
MaxSpeed	Mandatory	Port Supported Speed (for example, from HBA API).
PortNumber	Optional	
PermanentAddress	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	Optional	For Fibre Channel end device ports, the Fibre Channel ID. Shall be 16 un-separated upper case hex digits.
SupportedCOS	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	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	Optional	
ActiveFC4Types	Optional	
LinkTechnology	Mandatory	
SupportedMaximumTransmissionUnit	Mandatory	
ActiveMaximumTransmissionUnit	Optional	

17.6.4 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

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

Table 165 describes class CIM_HostedAccessPoint (Initiator).

Table 165 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Initiators

17.6.5 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

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

Table 166 describes class CIM_HostedAccessPoint (Target).

Table 166 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Targets

17.6.6 CIM_HostedCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 167 describes class CIM_HostedCollection.

Table 167 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

17.6.7 CIM_MemberOfCollection

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

Requirement: Support for ConnectivityCollections.

Table 168 describes class CIM_MemberOfCollection.

Table 168 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	The reference to the ProtocolEndpoint
Collection	Mandatory	The reference to the ConnectivityCollection

17.6.8 CIM_ProtocolControllerForPort

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

Table 169 describes class CIM_ProtocolControllerForPort.

Table 169 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForPort

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : restrict to initiator ports

17.6.9 CIM_SCSIInitiatorTargetLogicalUnitPath

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

Table 170 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 170 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Requirement	Description & Notes
LogicalUnit	Mandatory	
Initiator	Mandatory	
Target	Mandatory	

17.6.10 CIM_SCSIProtocolController

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

Table 171 describes class CIM_SCSIProtocolController.

Table 171 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController

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

17.6.11 CIM_SCSIProtocolEndpoint (Initiator)

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

Table 172 describes class CIM_SCSIProtocolEndpoint (Initiator).

Table 172 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator)

Properties	Requirement	Description & Notes
SystemCreationClassName	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) or 4 (Both Initiator and Target)

17.6.12 CIM_SCSIProtocolEndpoint (Target)

Models remote ports - target devices and possibly other initiators.

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

Table 173 describes class CIM_SCSIProtocolEndpoint (Target).

Table 173 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target)

Properties	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)

17.6.13 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

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

Table 174 describes class CIM_SystemDevice (Initiator Ports).

Table 174 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem
PartComponent	Mandatory	Reference to LogicalPort.Validation Property : Restrict to back-end or unrestricted ports.

17.6.14 CIM_SystemDevice (Non-port devices)

Associates system to ports and optional logical unit LogicalDevices

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

Table 175 describes class CIM_SystemDevice (Non-port devices).

Table 175 - SMI Referenced Properties/Methods for CIM_SystemDevice (Non-port devices)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	Reference to non-port devices.

STABLE

EXPERIMENTAL

Clause 18: SAS Initiator Ports Profile

18.1 Synopsis

Profile Name: SAS Initiator Ports

Version: 1.2.0

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 to parallel SCSI ports. A typical instance diagram is provided in Figure 23.

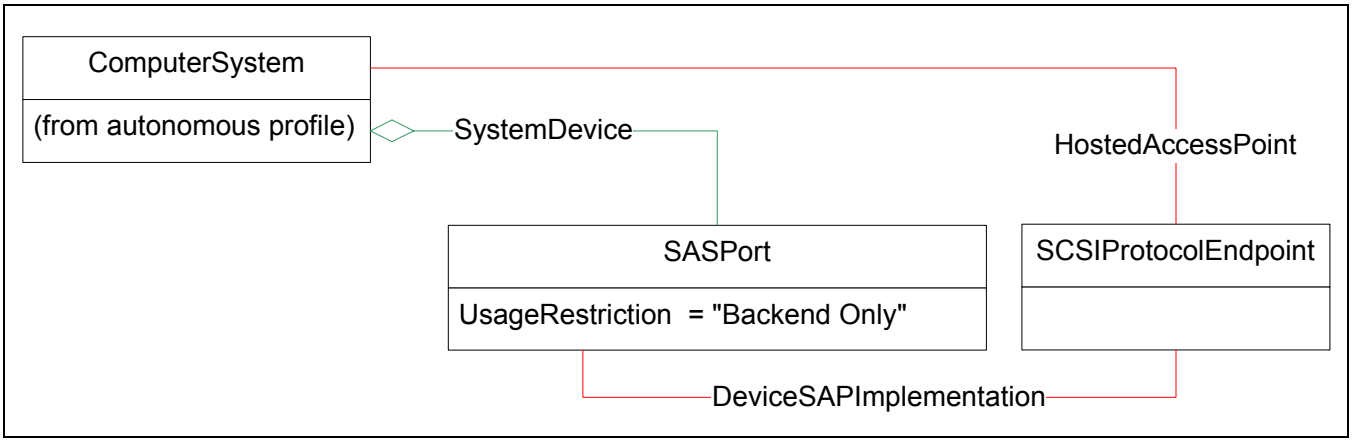

Figure 23 - SAS Initiator Port Model

18.2.1 Health and Fault Management Considerations

Table 176 summarizes the Health and Fault Management issues that are unique to this profile.

Table 176 - 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 177 describes the CIM elements for SAS Initiator Ports.

Table 177 - CIM Elements for SAS Initiator Ports

Element Name	Requirement	Description
18.5.1 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
18.5.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator LogicalPort and ProtocolEndpoint
18.5.3 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
18.5.4 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
18.5.5 CIM_HostedCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates the ConnectivityCollection to the hosting System.
18.5.6 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates ProtocolEndpoints to the ConnectivityCollection
18.5.7 CIM_SASPort	Mandatory	

Table 177 - CIM Elements for SAS Initiator Ports

Element Name	Requirement	Description
18.5.8 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
18.5.9 CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)	Mandatory	ProtocolEndpoints associated to initiator ports.
18.5.10 CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)	Optional	Models remote ports - target devices and possibly other initiators.
18.5.11 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
18.5.12 CIM_SystemDevice (Non-port devices)	Optional	Associates system to ports and optional logical unit LogicalDevices

18.5.1 CIM_ConnectivityCollection

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

Table 178 describes class CIM_ConnectivityCollection.

Table 178 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	

18.5.2 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint

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

Table 179 describes class CIM_DeviceSAPImplementation.

Table 179 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : restrict to initiators
Antecedent	Mandatory	Validation Property : restrict to back-end ports (initiators)

18.5.3 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

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

Table 180 describes class CIM_HostedAccessPoint (Initiator).

Table 180 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Initiators

18.5.4 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

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

Table 181 describes class CIM_HostedAccessPoint (Target).

Table 181 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Targets

18.5.5 CIM_HostedCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 182 describes class CIM_HostedCollection.

Table 182 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

18.5.6 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 183 describes class CIM_MemberOfCollection.

Table 183 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	The reference to the ProtocolEndpoint
Collection	Mandatory	The reference to the ConnectivityCollection

18.5.7 CIM_SASPort

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 184 describes class CIM_SASPort.

Table 184 - SMI Referenced Properties/Methods for CIM_SASPort

Properties	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.

Table 184 - SMI Referenced Properties/Methods for CIM_SASPort

Properties	Requirement	Description & Notes
PortType	Mandatory	Initiator port specialized profiles specify the appropriate subset of values.
PermanentAddress	Mandatory	SAS Address. Shall be 16 un-separated upper case hex digits.

18.5.8 CIM_SCSIInitiatorTargetLogicalUnitPath

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

Table 185 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 185 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Requirement	Description & Notes
LogicalUnit	Mandatory	
Target	Mandatory	
Initiator	Mandatory	

18.5.9 CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)

ProtocolEndpoints associated to initiator ports.

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

Table 186 describes class CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint).

Table 186 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ConnectionType	Mandatory	Shall be 8 (SAS)
ProtocolIFType	Mandatory	Shall be 1 (Other).

Table 186 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)

Properties	Requirement	Description & Notes
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
Role	Mandatory	Shall be 2 (Initiator) or 4 (Both Initiator and Target)

18.5.10 CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)

Models remote ports - target devices and possibly other initiators.

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

Table 187 describes class CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint).

Table 187 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)

Properties	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	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 8 (SAS)

18.5.11 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

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

Table 188 describes class CIM_SystemDevice (Initiator Ports).

Table 188 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem
PartComponent	Mandatory	Reference to LogicalPort.Validation Property : Restrict to back-end or unrestricted ports.

18.5.12 CIM_SystemDevice (Non-port devices)

Associates system to ports and optional logical unit LogicalDevices

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

Table 189 describes class CIM_SystemDevice (Non-port devices).

Table 189 - SMI Referenced Properties/Methods for CIM_SystemDevice (Non-port devices)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	Reference to non-port devices.

EXPERIMENTAL

EXPERIMENTAL

Clause 19: ATA Initiator Ports Profile

19.1 Synopsis

Profile Name: ATA Initiator Ports

Version: 1.2.0

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 24 shows a class diagram for this profile.

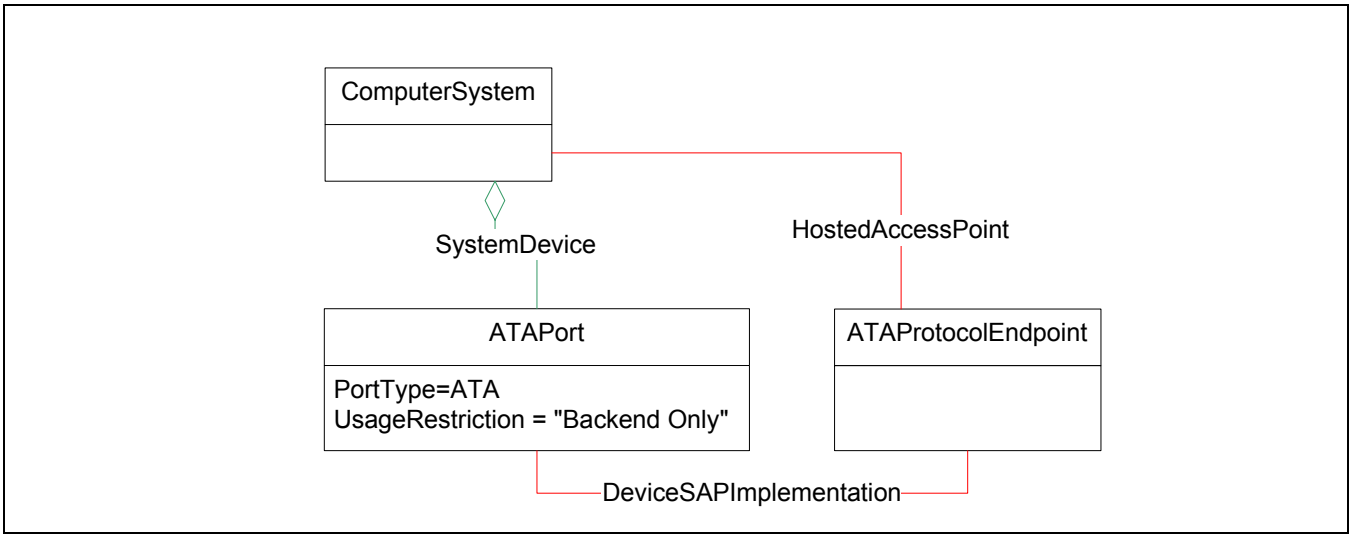

Figure 24 - ATA Initiator Port Class Diagram

19.3.1 Health and Fault Management Consideration

Table 190 summarizes the Health and Fault Management considerations that are specific to this profile.

Table 190 - 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 191 describes the CIM elements for ATA Initiator Ports.

Table 191 - CIM Elements for ATA Initiator Ports

Element Name	Requirement	Description
19.6.1 CIM_ATAPort	Mandatory	
19.6.2 CIM_ATAProtocolEndpoint (Initiator ProtocolEndpoint)	Mandatory	ProtocolEndpoints associated to initiator ports.
19.6.3 CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint)	Optional	Models remote ports - target devices and possibly other initiators.
19.6.4 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
19.6.5 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator LogicalPort and ProtocolEndpoint
19.6.6 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
19.6.7 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
19.6.8 CIM_HostedCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates the ConnectivityCollection to the hosting System.
19.6.9 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates ProtocolEndpoints to the ConnectivityCollection
19.6.10 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
19.6.11 CIM_SystemDevice (Non-port devices)	Optional	Associates system to ports and optional logical unit LogicalDevices

19.6.1 CIM_ATAPort

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 192 describes class CIM_ATAPort.

Table 192 - SMI Referenced Properties/Methods for CIM_ATAPort

Properties	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	Mandatory	Initiator port specialized profiles specify the appropriate subset of values.

19.6.2 CIM_ATAProtocolEndpoint (Initiator ProtocolEndpoint)

ProtocolEndpoints associated to initiator ports.

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

Table 193 describes class CIM_ATAProtocolEndpoint (Initiator ProtocolEndpoint).

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

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
Role	Mandatory	Shall be 2 (Initiator)
ProtocolIFType	Mandatory	Shall be 1 (Other)

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

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

19.6.3 CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint)

Models remote ports - target devices and possibly other initiators.

Created By: External
Modified By: External
Deleted By: External
Requirement: Optional

Table 194 describes class CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint).

Table 194 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint)

Properties	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	Shall be 1 (Other)
OtherTypeDescription	Mandatory	Shall be 'ATA'
ConnectionType	Mandatory	Shall be 2 (ATA for PATA ports) or 3 (SATA).

19.6.4 CIM_ConnectivityCollection

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

Table 195 describes class CIM_ConnectivityCollection.

Table 195 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	

19.6.5 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint

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	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : restrict to initiators
Antecedent	Mandatory	Validation Property : restrict to back-end ports (initiators)

19.6.6 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

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

Table 197 describes class CIM_HostedAccessPoint (Initiator).

Table 197 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Initiators

19.6.7 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

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

Table 198 describes class CIM_HostedAccessPoint (Target).

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

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Targets

19.6.8 CIM_HostedCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 199 describes class CIM_HostedCollection.

Table 199 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

19.6.9 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 200 describes class CIM_MemberOfCollection.

Table 200 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	The reference to the ProtocolEndpoint
Collection	Mandatory	The reference to the ConnectivityCollection

19.6.10 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 201 describes class CIM_SystemDevice (Initiator Ports).

Table 201 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem
PartComponent	Mandatory	Reference to LogicalPort.Validation Property : Restrict to back-end or unrestricted ports.

19.6.11 CIM_SystemDevice (Non-port devices)

Associates system to ports and optional logical unit LogicalDevices

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

Table 202 describes class CIM_SystemDevice (Non-port devices).

Table 202 - SMI Referenced Properties/Methods for CIM_SystemDevice (Non-port devices)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	Reference to non-port devices.

EXPERIMENTAL

EXPERIMENTAL

Clause 20: FC-SB-x Initiator Ports Profile

20.1 Synopsis

Profile Name: SB Initiator Ports

Version: 1.3.0

Organization: SNIA

CIM Schema Version: 2.13.0

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

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 25 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.

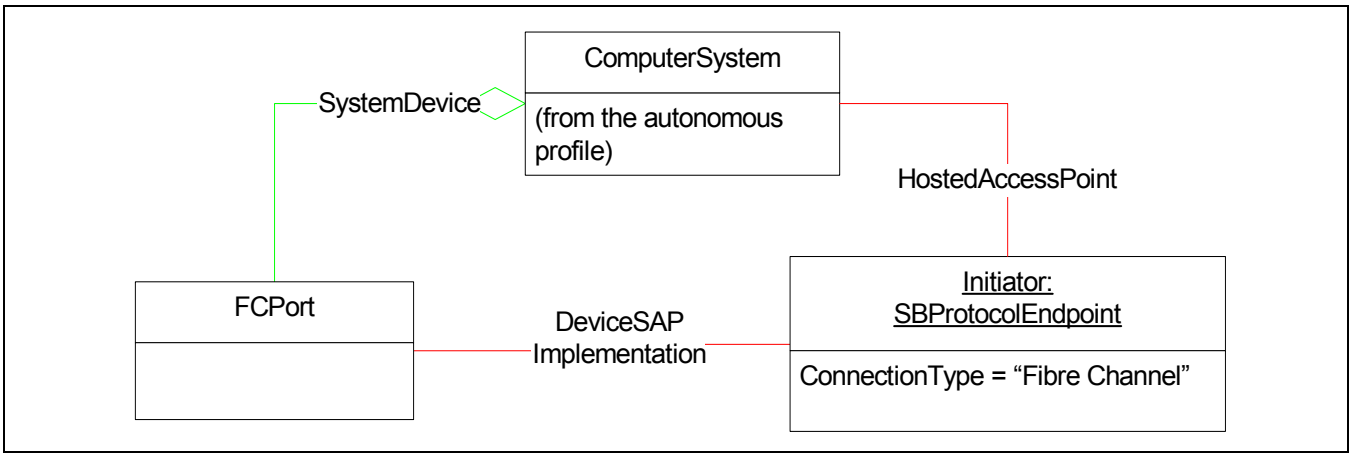

Figure 25 - Fibre Channel Initiator Instance Diagram

20.3.1 Health and Fault Management Considerations

Table 203 summarizes the Health and Fault Management considerations specific to this profile.

Table 203 - 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 204 describes the CIM elements for SB Initiator Ports.

Table 204 - CIM Elements for SB Initiator Ports

Element Name	Requirement	Description
20.6.1 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
20.6.2 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator LogicalPort and ProtocolEndpoint
20.6.3 CIM_FCPort	Mandatory	
20.6.4 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
20.6.5 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
20.6.6 CIM_HostedCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates the ConnectivityCollection to the hosting System.
20.6.7 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates ProtocolEndpoints to the ConnectivityCollection
20.6.8 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
20.6.9 CIM_SystemDevice (Non-port devices)	Optional	Associates system to ports and optional logical unit LogicalDevices
20.6.10 SNIA_SBInitiatorTargetLogicalUnitPath	Optional	
20.6.11 SNIA_SBProtocolEndpoint (Initiator ProtocolEndpoint)	Mandatory	
20.6.12 SNIA_SBProtocolEndpoint (Target or non-local ProtocolEndpoint)	Optional	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.OperationalStatus <> PreviousInstance.OperationalStatus	Optional	Deprecated WQL -Modify FCPort

Table 204 - CIM Elements for SB Initiator Ports

Element Name	Requirement	Description
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

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

Table 205 describes class CIM_ConnectivityCollection.

Table 205 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	

20.6.2 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint

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

Table 206 describes class CIM_DeviceSAPImplementation.

Table 206 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : restrict to initiators
Antecedent	Mandatory	Validation Property : restrict to back-end ports (initiators)

20.6.3 CIM_FCPort

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 207 describes class CIM_FCPort.

Table 207 - SMI Referenced Properties/Methods for CIM_FCPort

Properties	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	Mandatory	Initiator port specialized profiles specify the appropriate subset of values.
ElementName	Mandatory	Port Symbolic Name
Speed	Mandatory	
MaxSpeed	Mandatory	Port Supported Speed from HBA API.
PortNumber	Optional	
PermanentAddress	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.
NetworkAddresses	Optional	For Fibre Channel end device ports, the Fibre Channel ID
SupportedCOS	Optional	
ActiveCOS	Optional	
SupportedFC4Types	Optional	
ActiveFC4Types	Optional	
LinkTechnology	Mandatory	
SupportedMaximumTransmissionUnit	Mandatory	
ActiveMaximumTransmissionUnit	Optional	

20.6.4 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

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

Table 208 describes class CIM_HostedAccessPoint (Initiator).

Table 208 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Initiators

20.6.5 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

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

Table 209 describes class CIM_HostedAccessPoint (Target).

Table 209 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Targets

20.6.6 CIM_HostedCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 210 describes class CIM_HostedCollection.

Table 210 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

20.6.7 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 211 describes class CIM_MemberOfCollection.

Table 211 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	The reference to the ProtocolEndpoint
Collection	Mandatory	The reference to the ConnectivityCollection

20.6.8 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

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

Table 212 describes class CIM_SystemDevice (Initiator Ports).

Table 212 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem
PartComponent	Mandatory	Reference to LogicalPort.Validation Property : Restrict to back-end or unrestricted ports.

20.6.9 CIM_SystemDevice (Non-port devices)

Associates system to ports and optional logical unit LogicalDevices

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

Table 213 describes class CIM_SystemDevice (Non-port devices).

Table 213 - SMI Referenced Properties/Methods for CIM_SystemDevice (Non-port devices)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	Reference to non-port devices.

20.6.10 SNIA_SBInitiatorTargetLogicalUnitPath

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

Table 214 describes class SNIA_SBInitiatorTargetLogicalUnitPath.

Table 214 - SMI Referenced Properties/Methods for SNIA_SBInitiatorTargetLogicalUnitPath

Properties	Requirement	Description & Notes
UsePreferredPath	Optional	SB only - Boolean indicating whether preferred path processing is required
PreferredPath	Optional	SB only - boolean indicating whiether this is a preferred path
PathGroupState	Optional	SB only - One of 'Unknown, 'Path grouping not supported','Reset', 'Grouped', 'Ungrouped'
PathGroupMode	Optional	SB only - One of 'Unknown', 'None', 'Single path', 'Multipath' (SIngle path and multipath only valid if PathGroupState is grouped.
PathGroupID	Optional	SB only - String containing the ID from the OS, only valid if PathGroupState is Grouped
LogicalUnit	Mandatory	
Target	Mandatory	
Initiator	Mandatory	

20.6.11 SNIA_SBProtocolEndpoint (Initiator ProtocolEndpoint)

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

Table 215 describes class SNIA_SBProtocolEndpoint (Initiator ProtocolEndpoint).

Table 215 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint (Initiator ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ProtocolIFType	Mandatory	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be 'SB'
ConnectionType	Mandatory	Shall be 2 (Fibre Channel)
Role	Mandatory	Shall be 2 (Initiator) or 4 (Both Initiator and Target)

20.6.12 SNIA_SBProtocolEndpoint (Target or non-local ProtocolEndpoint)

Target or non-local ProtocolEndpoint

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

Table 216 describes class SNIA_SBProtocolEndpoint (Target or non-local ProtocolEndpoint).

Table 216 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint (Target or non-local ProtocolEndpoint)

Properties	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	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be 'SB'
ConnectionType	Mandatory	Shall be 2 (Fibre Channel)

EXPERIMENTAL

EXPERIMENTAL

Clause 21: SAS/SATA Initiator Port Profile

21.1 Synopsis

Profile Name: SAS/SATA Initiator Ports

Version: 1.2.0

Organization: SNIA

CIM Schema Version: 2.13.1

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

Specializes: Generic Initiator Port Profile

Central Class: CIM_SASSATAPort

Scoping Class: a CIM_System in a separate autonomous profile

The SAS/SATA Initiator Port Profile models the management of a port that initiates commands to both SAS and SATA devices.

21.2 Description

The SAS/SATA Initiator Port profile defines the model to ports that initiates commands to both SAS and SATA devices.

21.3 Implementation

A typical instance diagram is provided in Figure 26.

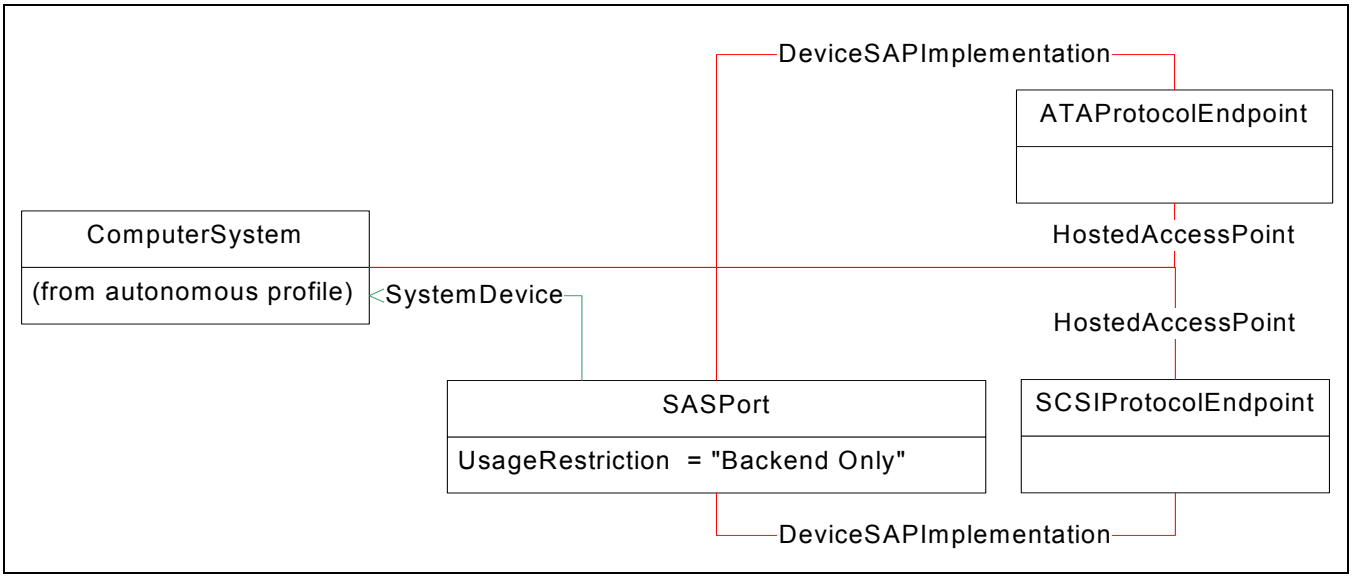

Figure 26 - SAS/SATA Initiator Port Instance Diagram

Remote ports may optionally be included - see Clause 6: Generic Target Ports Profile.

21.4 Health and Fault Management Considerations

Table 217 summarizes the Health and Fault Management issues that are unique to this profile.

Table 217 - SASSATAPort OperationalStatus

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

21.4.1 Health and Fault Management Considerations

Not defined in this standard.

21.4.2 Cascading Considerations

Not defined in this standard.

21.5 Methods

21.5.1 Extrinsic Methods of this Profile

None

21.5.2 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

21.6 Detailed Use Cases and Recipes

None

21.7 CIM Elements

Table 218 describes the CIM elements for SAS/SATA Initiator Ports.

Table 218 - CIM Elements for SAS/SATA Initiator Ports

Element Name	Requirement	Description
21.7.1 CIM_ATAProtocolEndpoint (Initiator ProtocolEndpoint)	Mandatory	Initiator ATA endpoints
21.7.2 CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint)	Optional	Remote ATA endpoints
21.7.3 CIM_ConnectivityCollection	Optional	Represents a collection of connected ProtocolEndpoints.
21.7.4 CIM_DeviceSAPImplementation	Mandatory	Connects Initiator LogicalPort and ProtocolEndpoint
21.7.5 CIM_HostedAccessPoint (Initiator)	Mandatory	Associates system to initiator protocol endpoints.
21.7.6 CIM_HostedAccessPoint (Target)	Optional	Associates system to optional remote protocol endpoints.
21.7.7 CIM_HostedCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates the ConnectivityCollection to the hosting System.
21.7.8 CIM_LogicalPort	Mandatory	
21.7.9 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for ConnectivityCollections.Associates ProtocolEndpoints to the ConnectivityCollection

Table 218 - CIM Elements for SAS/SATA Initiator Ports

Element Name	Requirement	Description
21.7.10 CIM_SASSATAPort (Initiator ProtocolEndpoint)	Mandatory	
21.7.11 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Represents a path between a SCSI initiator, target, and logical unit.
21.7.12 CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)	Mandatory	Initiator SCSI endpoints
21.7.13 CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)	Optional	Remote SCSI endpoints
21.7.14 CIM_SystemDevice (Initiator Ports)	Mandatory	Associates system to initiator ports.
21.7.15 CIM_SystemDevice (Non-port devices)	Optional	Associates system to ports and optional logical unit LogicalDevices

21.7.1 CIM_ATAProtocolEndpoint (Initiator ProtocolEndpoint)

Initiator ATA endpoints

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

Table 219 describes class CIM_ATAProtocolEndpoint (Initiator ProtocolEndpoint).

Table 219 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Initiator ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ProtocolIFType	Mandatory	Shall be 1 (Other)
OtherTypeDescription	Mandatory	Shall be 'ATA'
ConnectionType	Mandatory	Shall be 3 (SATA).
Role	Mandatory	Shall be 3 (Target) or 4 (Both Initiator and Target)

21.7.2 CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint)

Remote ATA endpoints

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Optional

Table 220 describes class CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint).

Table 220 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Target or non-local ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ProtocolIFType	Mandatory	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be 'ATA'
ConnectionType	Mandatory	Shall be 3 (SATA).
Role	Mandatory	Should be set appropriately by the instrumentation. If not know, use 0 (Unknown).

21.7.3 CIM_ConnectivityCollection

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

Table 221 describes class CIM_ConnectivityCollection.

Table 221 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	

21.7.4 CIM_DeviceSAPImplementation

Connects Initiator LogicalPort and ProtocolEndpoint

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

Table 222 describes class CIM_DeviceSAPImplementation.

Table 222 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation

Properties	Requirement	Description & Notes
Dependent	Mandatory	Validation Property : restrict to initiators
Antecedent	Mandatory	Validation Property : restrict to back-end ports (initiators)

21.7.5 CIM_HostedAccessPoint (Initiator)

Associates system to initiator protocol endpoints.

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

Table 223 describes class CIM_HostedAccessPoint (Initiator).

Table 223 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Initiator)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Initiators

21.7.6 CIM_HostedAccessPoint (Target)

Associates system to optional remote protocol endpoints.

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

Table 224 describes class CIM_HostedAccessPoint (Target).

Table 224 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (Target)

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Validation Property : Limit PEs to Targets

21.7.7 CIM_HostedCollection

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

Requirement: Support for ConnectivityCollections.

Table 225 describes class CIM_HostedCollection.

Table 225 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

21.7.8 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 226 describes class CIM_LogicalPort.

Table 226 - SMI Referenced Properties/Methods for CIM_LogicalPort

Properties	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	Mandatory	Initiator port specialized profiles specify the appropriate subset of values.

21.7.9 CIM_MemberOfCollection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ConnectivityCollections.

Table 227 describes class CIM_MemberOfCollection.

Table 227 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Member	Mandatory	The reference to the ProtocolEndpoint
Collection	Mandatory	The reference to the ConnectivityCollection

21.7.10 CIM_SASSATAPort (Initiator ProtocolEndpoint)

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

Table 228 describes class CIM_SASSATAPort (Initiator ProtocolEndpoint).

Table 228 - SMI Referenced Properties/Methods for CIM_SASSATAPort (Initiator ProtocolEndpoint)

Properties	Requirement	Description & Notes
PortType	Mandatory	Shall be 95 (SASSATA).

21.7.11 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 229 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 229 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Requirement	Description & Notes
LogicalUnit	Mandatory	
Target	Mandatory	
Initiator	Mandatory	

21.7.12 CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)

Initiator SCSI endpoints

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 230 describes class CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint).

Table 230 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Initiator ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ProtocolIFType	Mandatory	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be the string 'SCSI'.
ConnectionType	Mandatory	Shall be 8 (SAS)
Role	Mandatory	Shall be 2 (Initiator) or 4 (Both Initiator and Target)

21.7.13 CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)

Remote SCSI endpoints

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

Table 231 describes class CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint).

Table 231 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Target or non-local ProtocolEndpoint)

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
ProtocolIFType	Mandatory	Shall be 1 (Other).
OtherTypeDescription	Mandatory	Shall be the string 'SCSI.
ConnectionType	Mandatory	Shall be 8 (SAS)
Role	Mandatory	Shall be 2 (Initiator) or 4 (Both Initiator and Target)

21.7.14 CIM_SystemDevice (Initiator Ports)

Associates system to initiator ports.

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

Table 232 describes class CIM_SystemDevice (Initiator Ports).

Table 232 - SMI Referenced Properties/Methods for CIM_SystemDevice (Initiator Ports)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem
PartComponent	Mandatory	Reference to LogicalPort.Validation Property : Restrict to back-end or unrestricted ports.

21.7.15 CIM_SystemDevice (Non-port devices)

Associates system to ports and optional logical unit LogicalDevices

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

Table 233 describes class CIM_SystemDevice (Non-port devices).

Table 233 - SMI Referenced Properties/Methods for CIM_SystemDevice (Non-port devices)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	Reference to non-port devices.

EXPERIMENTAL

DEPRECATED

Clause 22: Backend Ports Subprofile

The functionality of the Backend Ports Subprofile has been subsumed by Clause 17: Fibre Channel Initiator Port 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

STABLE

Clause 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 27.

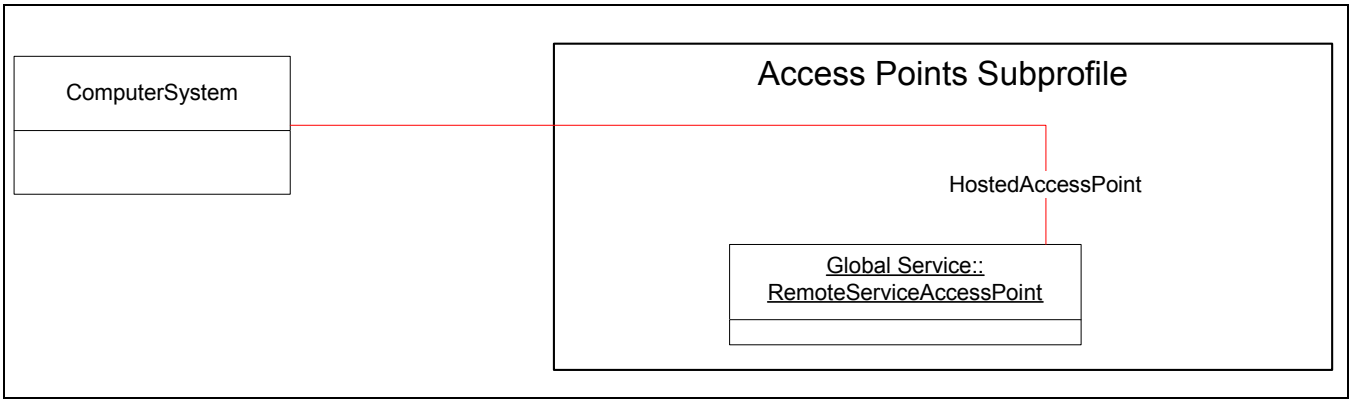

Figure 27 - System-wide Remote Access Point

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 28 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.

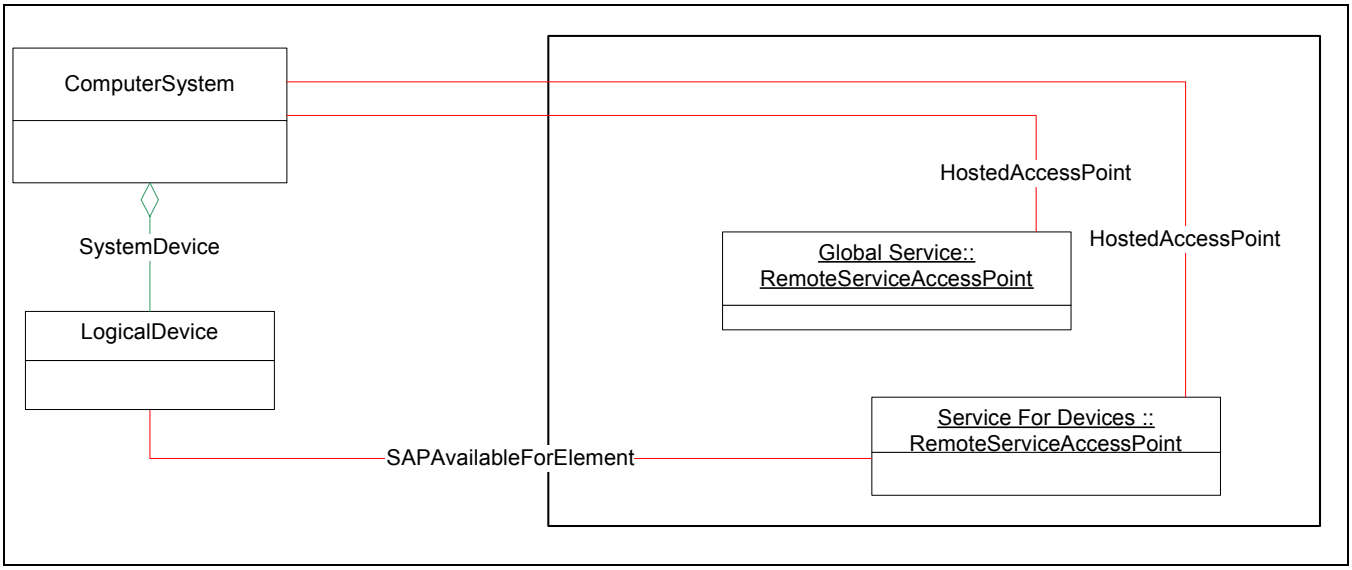

Figure 28 - Access Point Instance Diagram

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 234 specifies the requirements for InfoFormat, AccessInfo, and the scheme subset of a URL AccessInfo.

Table 234 - 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

23.8 CIM Elements

Table 235 describes the CIM elements for Access Points.

Table 235 - 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 236 describes class CIM_HostedAccessPoint.

Table 236 - 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 237 describes class CIM_RemoteServiceAccessPoint.

Table 237 - 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 238 describes class CIM_SAPAvailableForElement.

Table 238 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The managed element
AvailableSAP		Mandatory	The service access point

STABLE

EXPERIMENTAL

Clause 24: Cascading Subprofile

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, if 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 29 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.

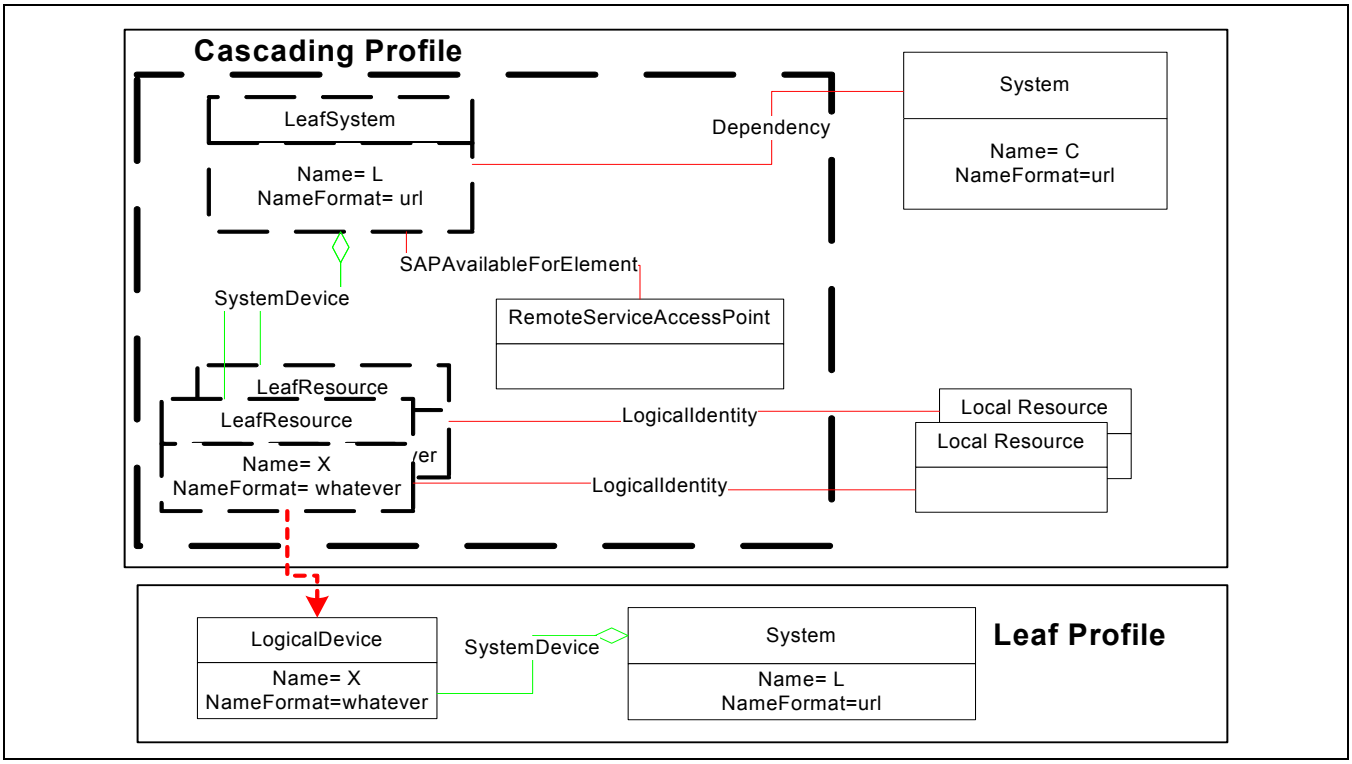

Figure 29 - Instance Diagram for Logical Topology

Note: The dashed classes in Figure 29 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 29 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 30.

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.

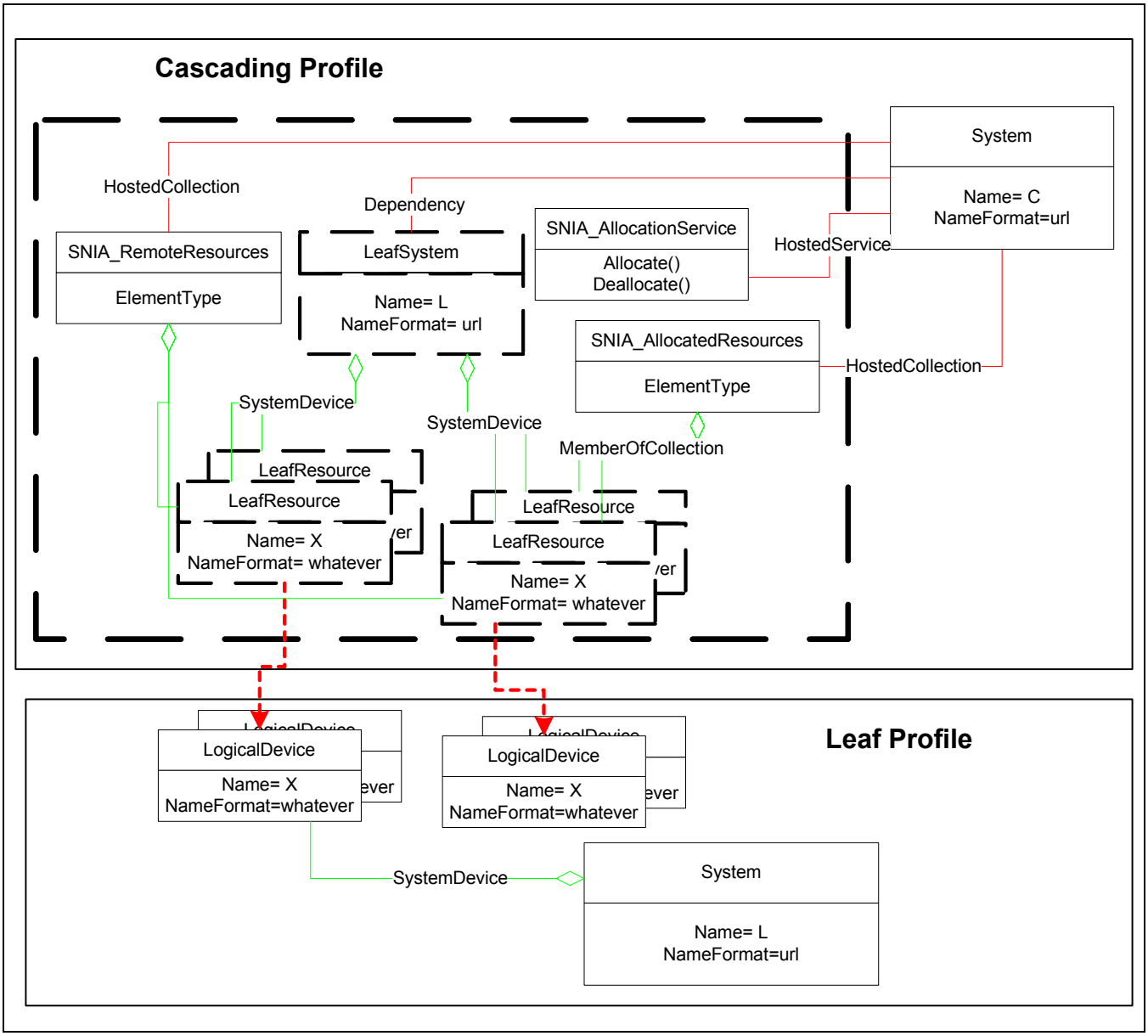

Figure 30 - Resource Allocation/Deallocation Instance Diagram

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 31.

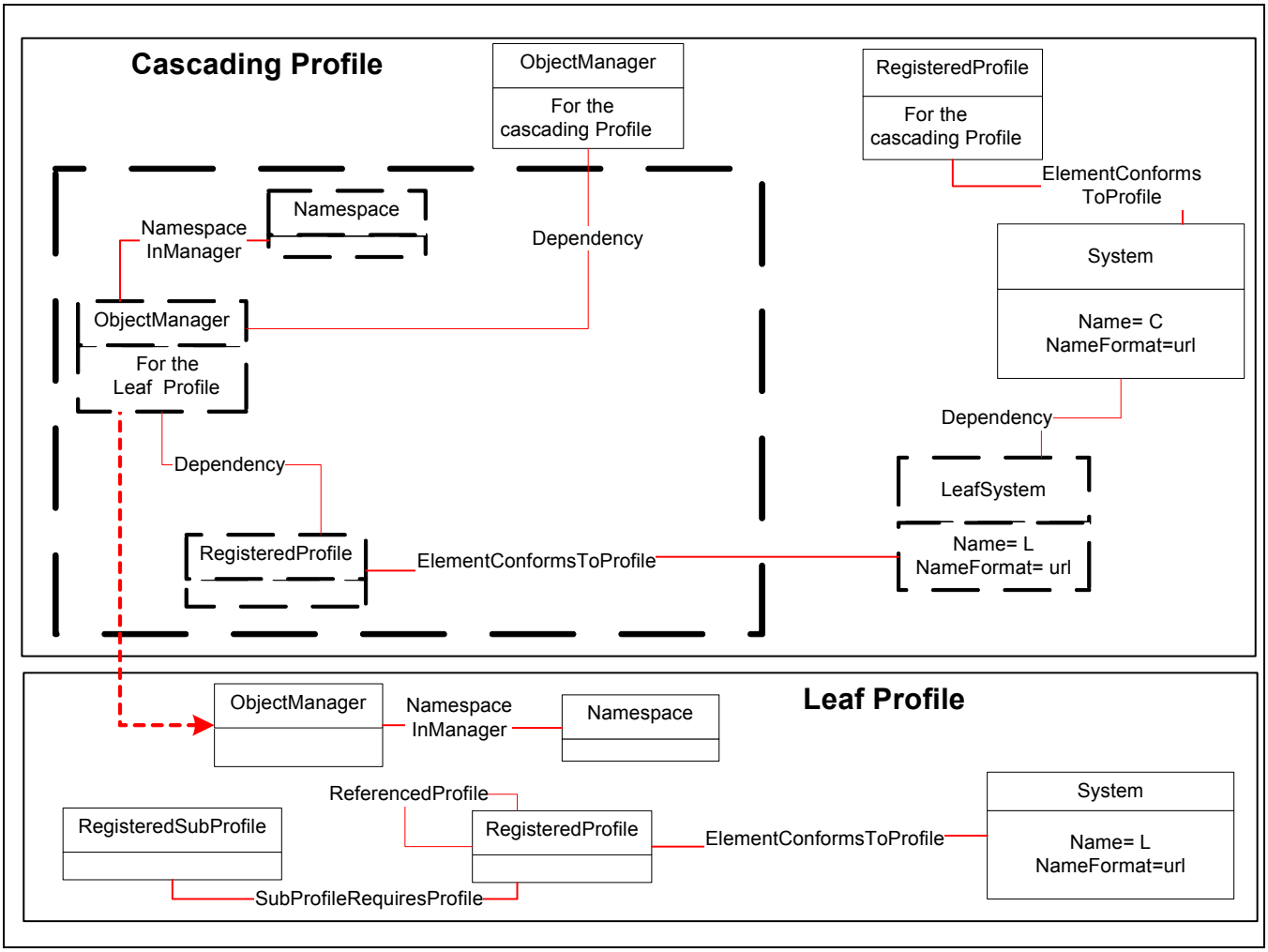

Figure 31 - Cascading Server Topology

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

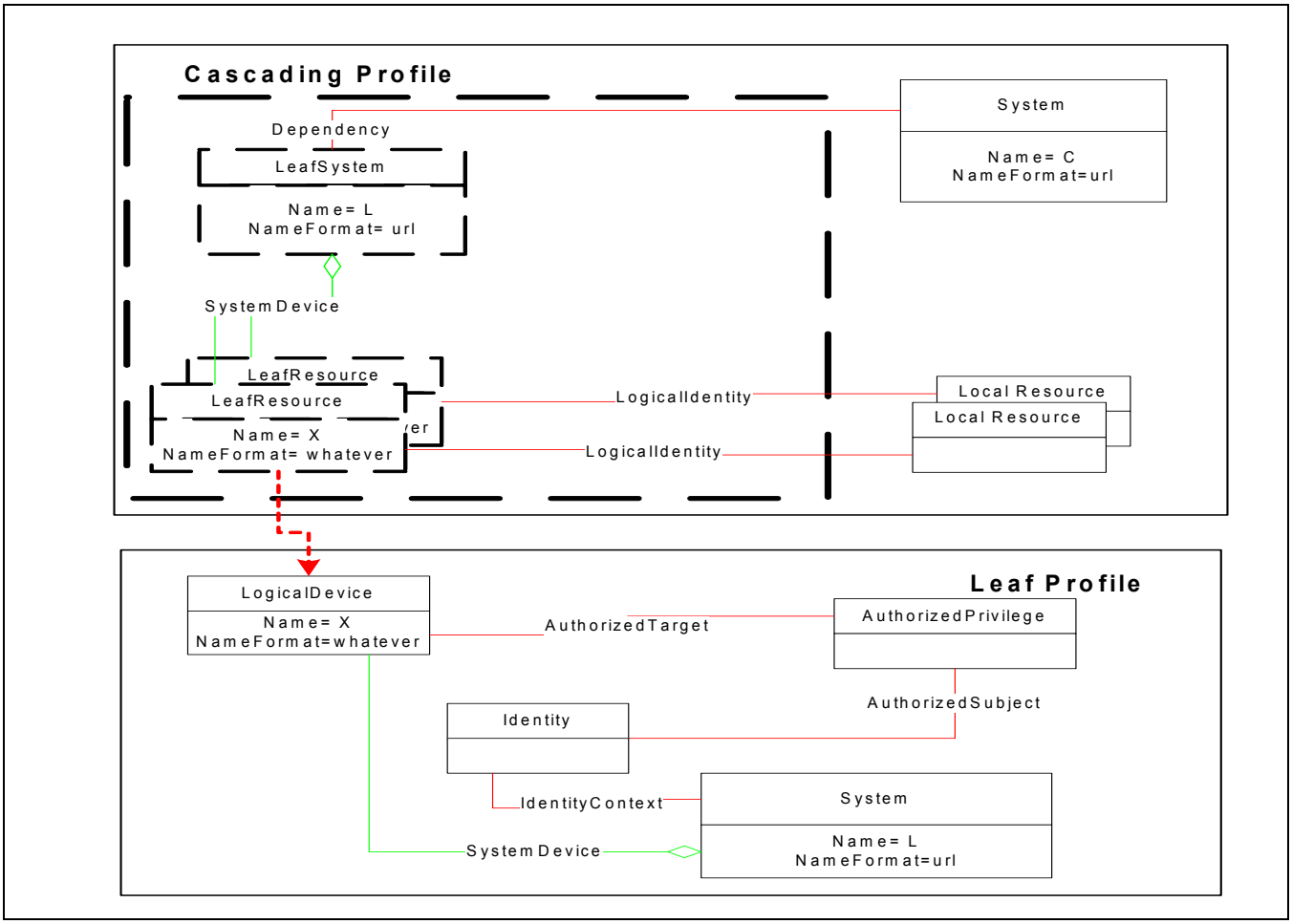

Figure 32 - Instance Diagram for Cascading with Resource Ownership

Figure 32 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 33.

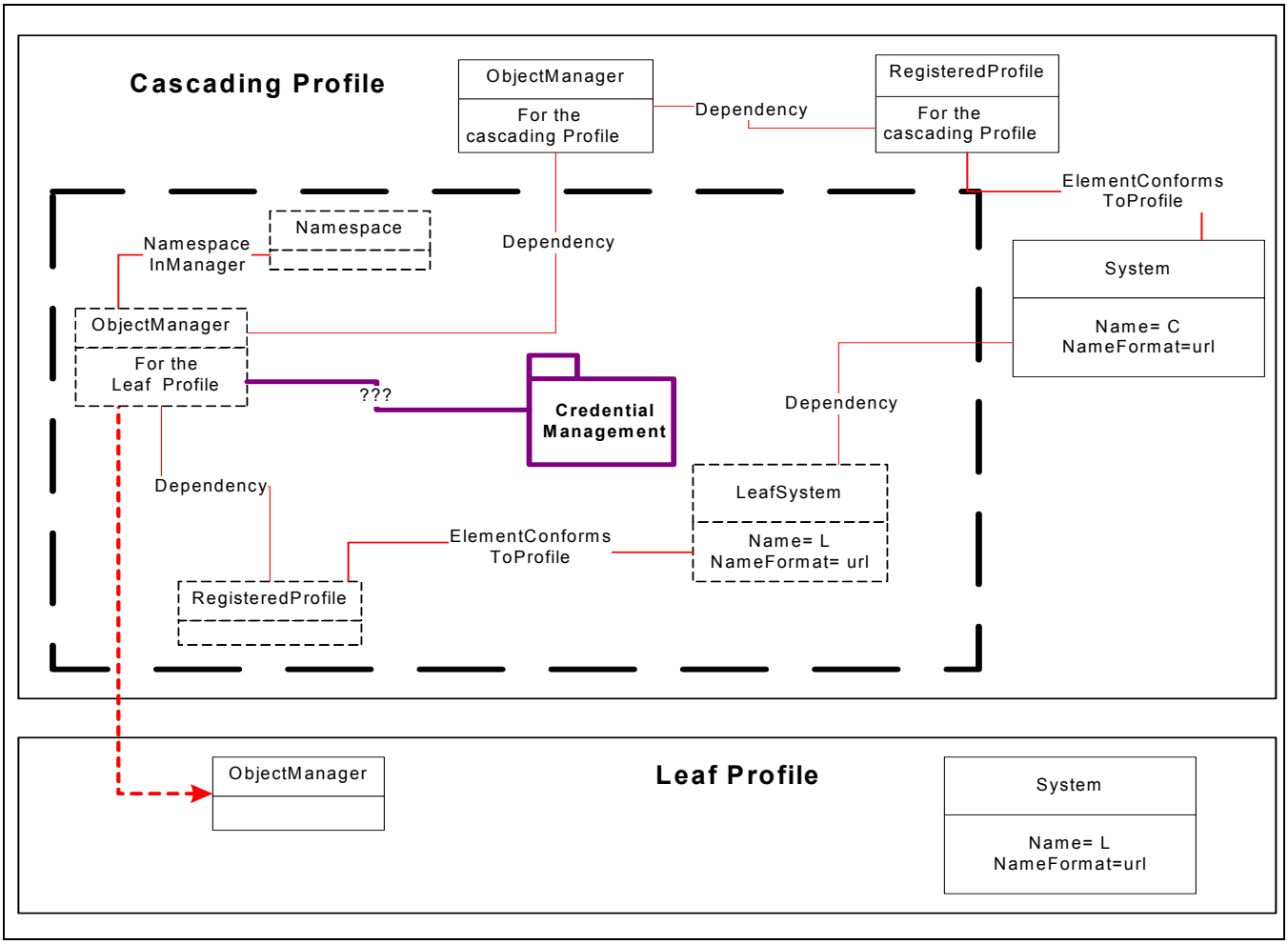

Figure 33 - Instance Diagram for Cascading with Credential Management Subprofile

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 241. 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 34.

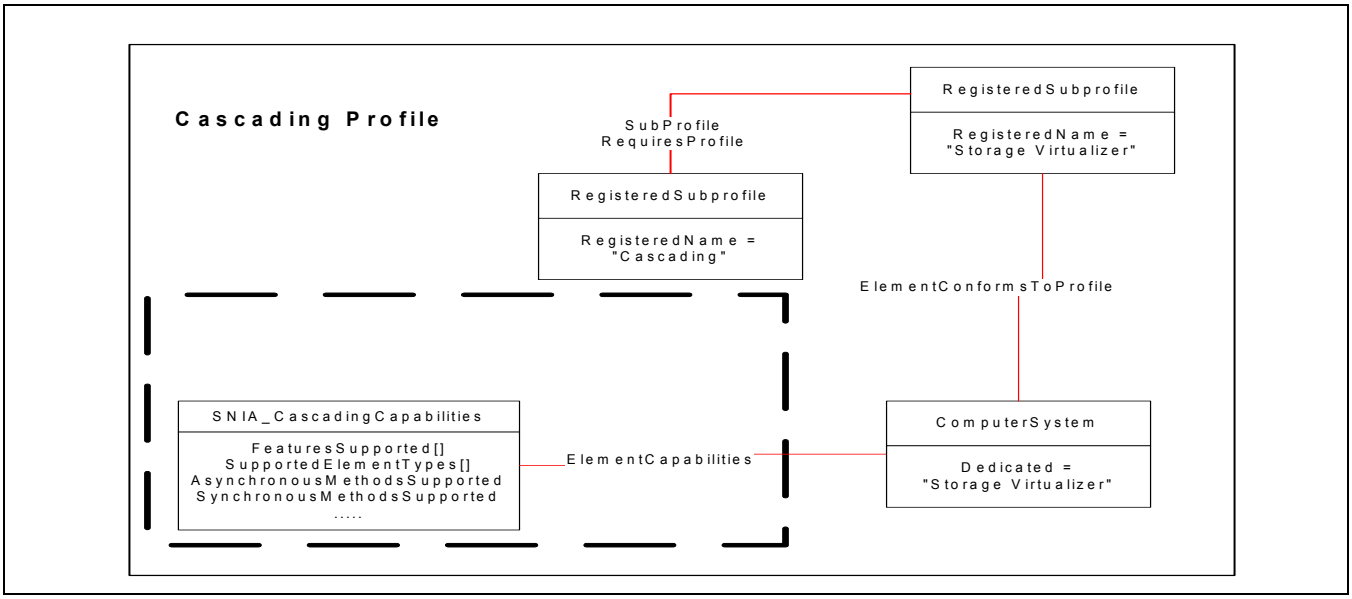

Figure 34 - Modeling of Cascading Capabilities

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 239 describes the supported profiles for Cascading.

Table 239 - Supported Profiles for Cascading

Registered Profile Names	Mandatory	Version
Server	Yes	1.3.0
Security Resource Ownership	No	1.3.0
Credential Management	No	1.3.0

24.5 Methods of this Subprofile

Table 240 summarized the extrinsic methods supported by the Cascading Subprofile.

Table 240 - 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 241 are formally recognized and supported by this version of SMI-S.

Table 241 - 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 241 - Cascading Capabilities Patterns (Continued)

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

24.8 CIM Elements

Table 242 describes the CIM elements for Cascading.

Table 242 - 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).

Table 242 - CIM Elements for Cascading

Element Name	Requirement	Description
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.
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.

Table 242 - CIM Elements for Cascading

Element Name	Requirement	Description
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.
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 243 describes class CIM_ComputerSystem (Leaf System).

Table 243 - 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 244 describes class CIM_Dependency (Object Managers).

Table 244 - 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 245 describes class CIM_Dependency (Profile to Object Manager).

Table 245 - 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 246 describes class CIM_Dependency (Systems).

Table 246 - 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 247 describes class CIM_ElementCapabilities.

Table 247 - 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 248 describes class CIM_ElementConformsToProfile (Leaf).

Table 248 - 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 249 describes class CIM_HostedCollection (Allocated Resources).

Table 249 - 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 250 describes class CIM_HostedCollection (Remote Resources).

Table 250 - 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 251 describes class CIM_HostedService (Allocation Service).

Table 251 - SMI Referenced Properties/Methods for CIM_HostedService (Allocation Service)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	>The AllocationService hosted on the System

24.8.10 CIM_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 252 describes class CIM_HostedService (Object Manager).

Table 252 - SMI Referenced Properties/Methods for CIM_HostedService (Object Manager)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

24.8.11 CIM_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 253 describes class CIM_LogicalDisk.

Table 253 - SMI Referenced Properties/Methods for CIM_LogicalDisk

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

Table 253 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
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	
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.

24.8.12 CIM_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 254 describes class CIM_LogicalIdentity (General).

Table 254 - 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.

24.8.13 CIM_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 255 describes class CIM_LogicalIdentity (LogicalDisk).

Table 255 - 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.

24.8.14 CIM_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 256 describes class CIM_LogicalIdentity (StorageVolume).

Table 256 - 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.

24.8.15 CIM_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 257 describes class CIM_MemberOfCollection (Allocated Resources).

Table 257 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

24.8.16 CIM_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 258 describes class CIM_MemberOfCollection (Remote Resources).

Table 258 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

24.8.17 CIM_Namespace (Leaf)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if Leaf ObjectManagers are modeled.

Table 259 describes class CIM_Namespace (Leaf).

Table 259 - 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'

24.8.18 CIM_NamespaceInManager (Leaf)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if Leaf ObjectManagers are modeled.

Table 260 describes class CIM_NamespaceInManager (Leaf).

Table 260 - SMI Referenced Properties/Methods for CIM_NamespaceInManager (Leaf)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

24.8.19 CIM_ObjectManager (Leaf)

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

Table 261 describes class CIM_ObjectManager (Leaf).

Table 261 - 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	

24.8.20 CIM_RegisteredProfile (Leaf)

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

Table 262 describes class CIM_RegisteredProfile (Leaf).

Table 262 - 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	

Table 262 - SMI Referenced Properties/Methods for CIM_RegisteredProfile (Leaf)

Properties	Flags	Requirement	Description & Notes
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

24.8.21 CIM_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 263 describes class CIM_RemoteServiceAccessPoint (Leaf).

Table 263 - 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.

24.8.22 CIM_SAPAvailableForElement

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_RemoteServiceAccessPoint is modeled.

Table 264 describes class CIM_SAPAvailableForElement.

Table 264 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Leaf System
AvailableSAP		Mandatory	

24.8.23 CIM_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 265 describes class CIM_StorageVolume.

Table 265 - 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.

Table 265 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
ConsumableBlocks		Mandatory	The number of blocks usable by consumers.
IsBasedOnUnderlyingRedundancy		Mandatory	
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.

24.8.24 CIM_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 266 describes class CIM_SystemDevice (Leaf Devices).

Table 266 - 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.

24.8.25 SNIA_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 267 describes class SNIA_AllocatedResources.

Table 267 - 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).

24.8.26 SNIA_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 268 describes class SNIA_AllocationService.

Table 268 - 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.

24.8.27 SNIA_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 269 describes class SNIA_CascadingCapabilities.

Table 269 - 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'}

Table 269 - SMI Referenced Properties/Methods for SNIA_CascadingCapabilities

Properties	Flags	Requirement	Description & Notes
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'}

24.8.28 SNIA_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 270 describes class SNIA_RemoteResources.

Table 270 - 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).

EXPERIMENTAL

STABLE

Clause 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 the Health and Fault Management (HFM) Clause. 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 implementation should 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 previous 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 are 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 271 - 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 271. 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 271.

The operational status combinations listed in Table 271 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 271 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

25.7 CIM Elements

Table 272 describes the CIM elements for Health.

Table 272 - CIM Elements for Health

Element Name	Requirement	Description
25.7.1 CIM_ComputerSystem	Mandatory	
25.7.2 CIM_LogicalDevice	Mandatory	

Table 272 - 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 273 describes class CIM_ComputerSystem.

Table 273 - 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 274 describes class CIM_LogicalDevice.

Table 274 - 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 275 describes class CIM_RelatedElementCausingError.

Table 275 - 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.

Table 275 - SMI Referenced Properties/Methods for CIM_RelatedElementCausingError

Properties	Flags	Requirement	Description & Notes
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

Clause 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 35.

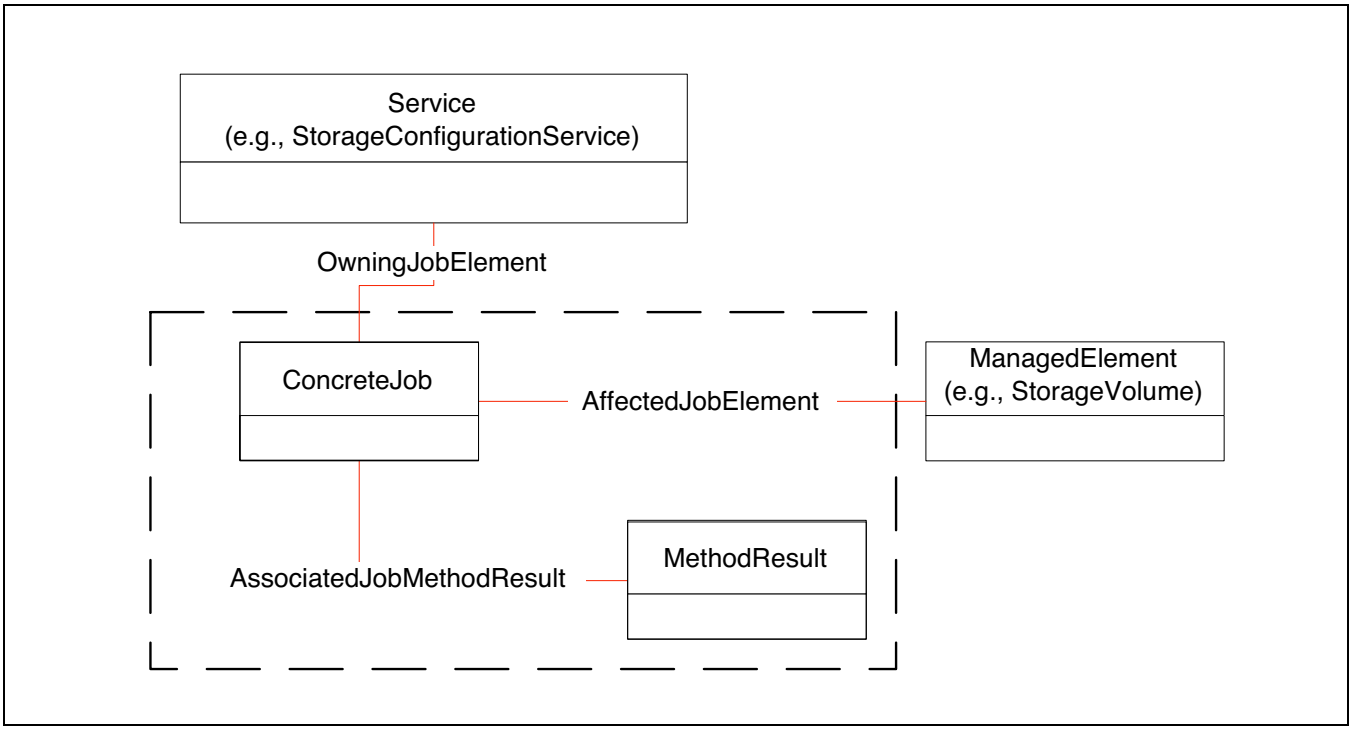

Figure 35 - Job Control Subprofile Model

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 276 maps the standard mapping between the OperationalStatus and JobState properties on ConcreteJob. The actual values of the properties are listed in Table 276with the associated value from the property’s ValueMap qualifier.

Table 276 - 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 276 - 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 Clause 25: Health Package for details.

EXPERIMENTAL

The standards messages specific to this profile are listed in Table 277. See Clause 9: Standard Messages in *Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6* for a description of standard messages and the list all standard messages

Table 277 - 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 Clause 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 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 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 36.
- It may take some time before the Job starts.
- A Job may be terminated or killed.
- Jobs may be modified.
- Jobs may be restarted.

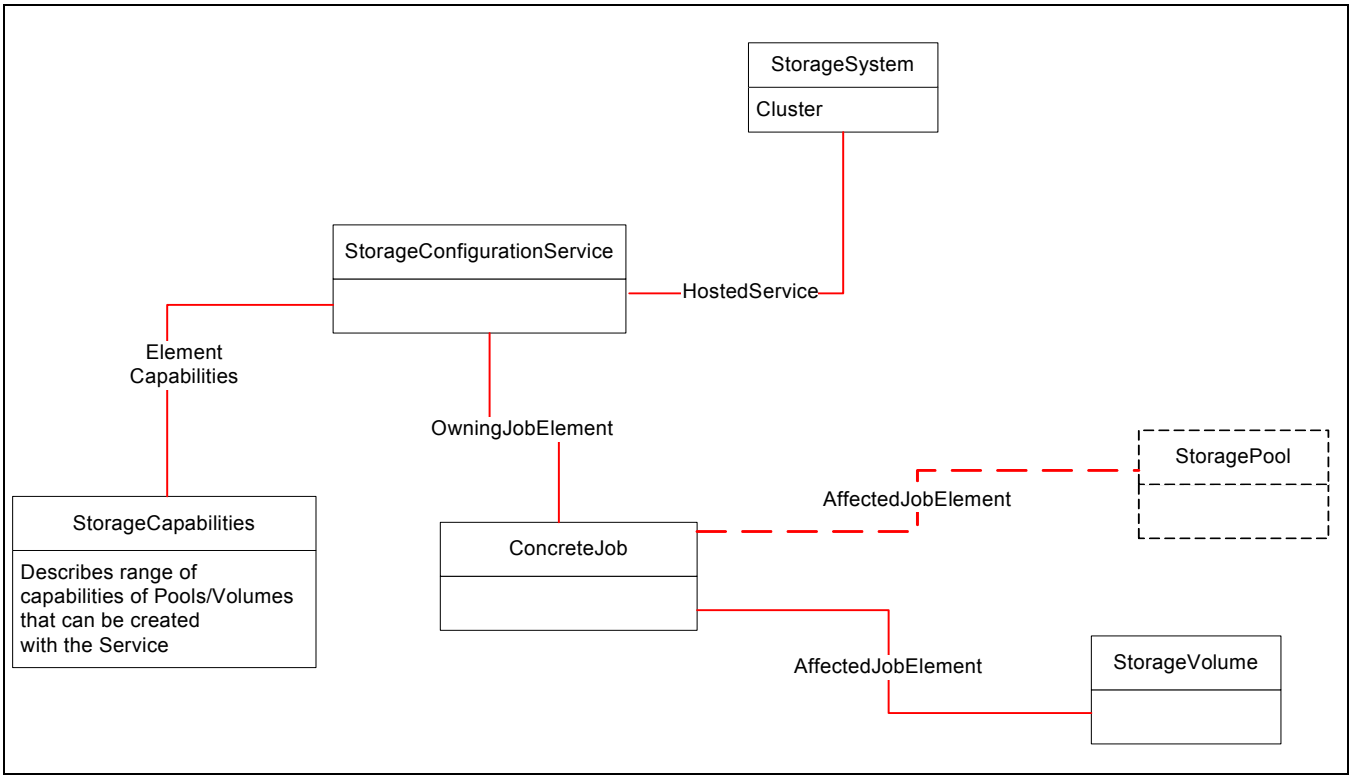

Figure 36 - Storage Configuration

26.7 Registered Name and Version

Job Control version 1.3.0

26.8 CIM Elements

Table 278 describes the CIM elements for Job Control.

Table 278 - 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	

Table 278 - CIM Elements for Job Control

Element Name	Requirement	Description
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
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'

Table 278 - CIM Elements for Job Control

Element Name	Requirement	Description
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 279 describes class CIM_AffectedJobElement.

Table 279 - 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 280 describes class CIM_AssociatedJobMethodResult.

Table 280 - 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 281 describes class CIM_ConcreteJob.

Table 281 - 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.
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.

Table 281 - SMI Referenced Properties/Methods for CIM_ConcreteJob

Properties	Flags	Requirement	Description & Notes
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 282 describes class CIM_MethodResult.

Table 282 - 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 283 describes class CIM_OwningJobElement.

Table 283 - 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

Clause 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 37 illustrates a typical instance diagram.

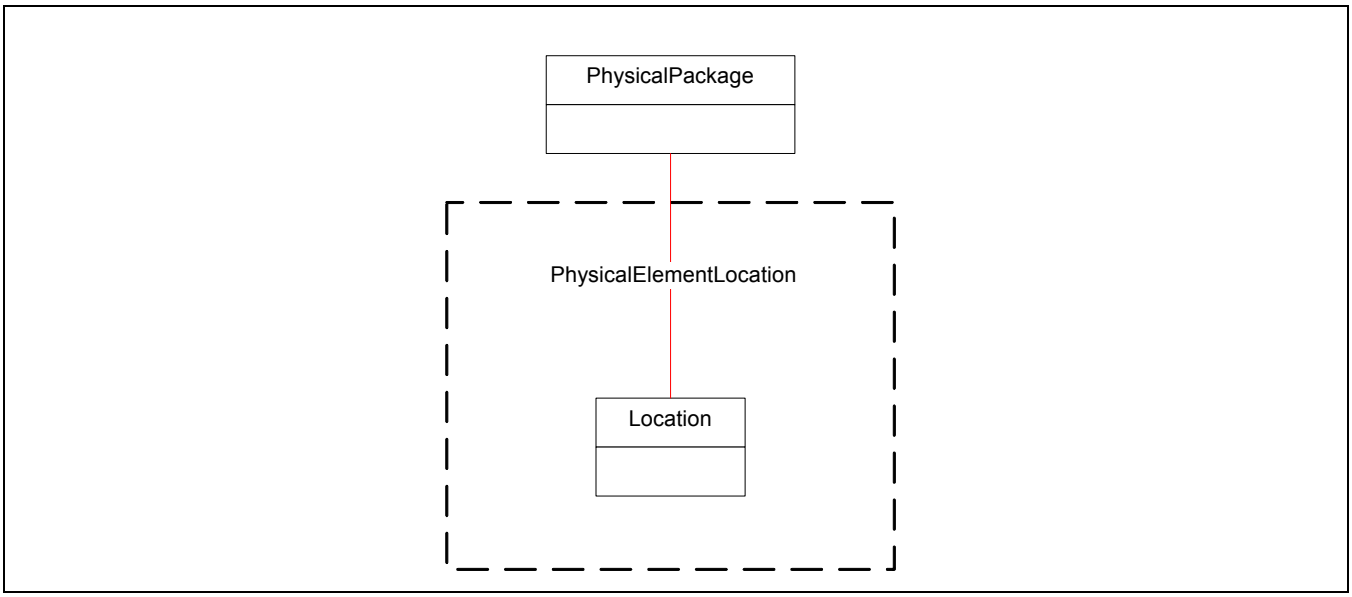

Figure 37 - Location Instance

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 1.3.0

27.8 CIM Elements

Table 284 describes the CIM elements for Location.

Table 284 - 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 285 describes class CIM_Location.

Table 285 - 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 286 describes class CIM_PhysicalElementLocation.

Table 286 - SMI Referenced Properties/Methods for CIM_PhysicalElementLocation

Properties	Flags	Requirement	Description & Notes
Element		Mandatory	
PhysicalLocation		Mandatory	

STABLE

DEPRECATED

Clause 28: Extra Capacity Set Subprofile

The functionality of the Extra Capacity Set Subprofile has been replaced by the Clause 30: Multiple Computer System Subprofile.

The Extra Capacity Set Subprofile is defined in section B.8 of SMI-S 1.0.2.

DEPRECATED

DEPRECATED

Clause 29: Cluster Subprofile

The functionality of the Cluster Subprofile has been subsumed by Clause 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

Clause 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 38.

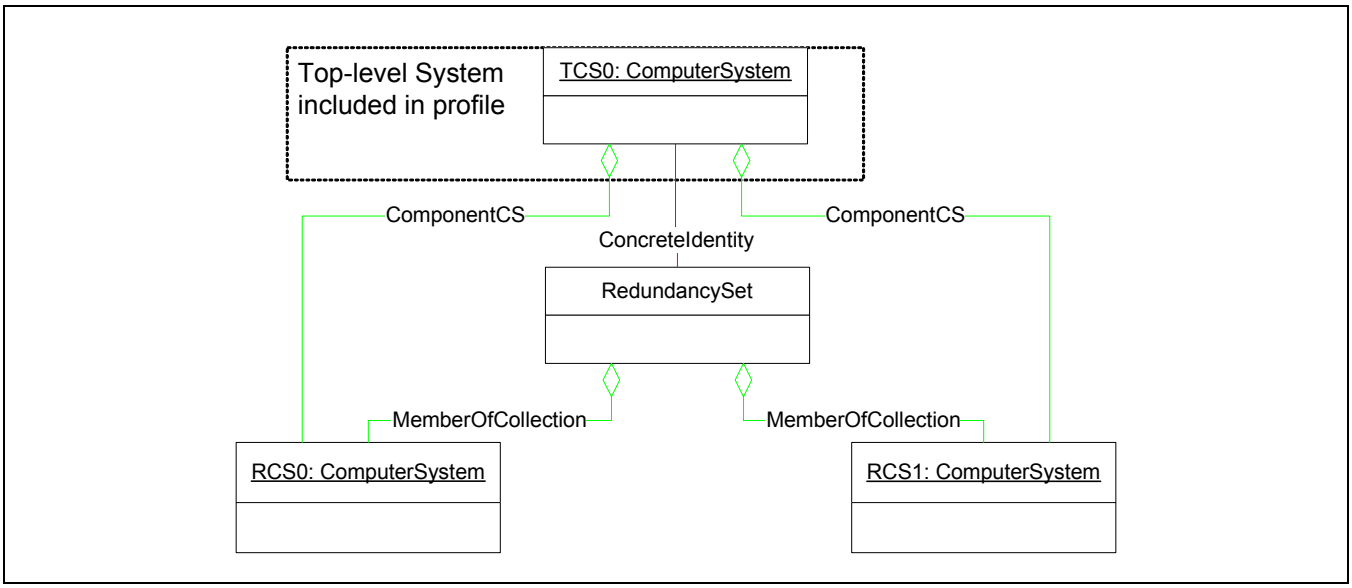

Figure 38 - Two Redundant Systems Instance Diagram

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 Clause 42: 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 38 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 287.

Table 287 - 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 39.

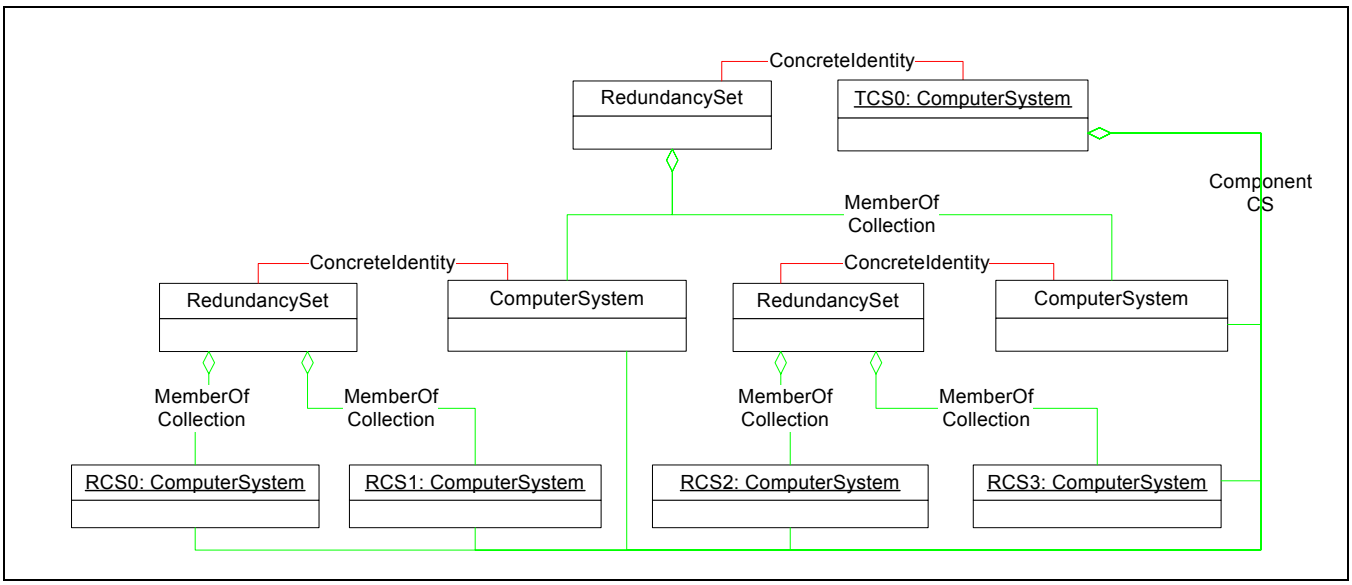

Figure 39 - Multiple Redundancy Tier Instance Diagram

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 40 demonstrates these system level numbers using the same configuration from Figure 39. Note that ComponentCS diagrams are omitted from this diagram to avoid clutter.

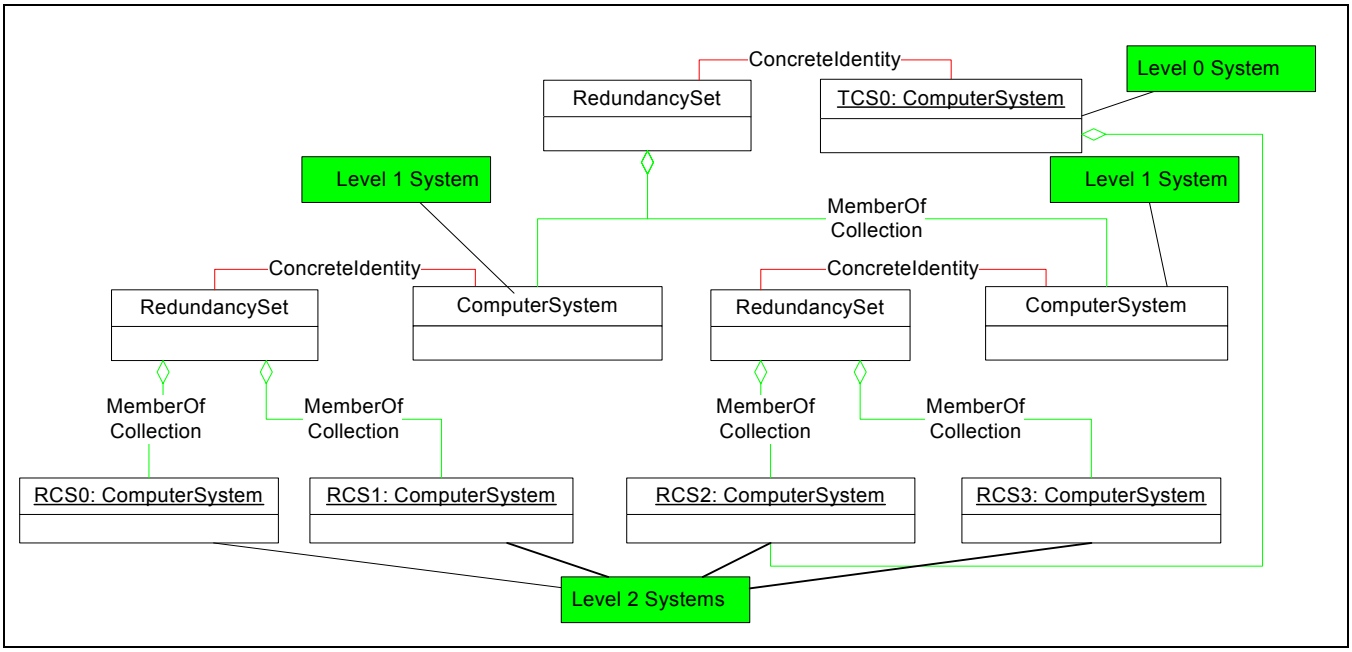

Figure 40 - System Level Numbers

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 39; 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 Clause 32: 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 38. 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 1 Common Architecture, 1.3.0 Rev 6* Clause 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 Clause 25: Health Package.

30.3 Cascading Considerations

None

30.4 Supported Subprofiles and Packages

Table 288 describes the supported profiles for Multiple Computer System.

Table 288 - Supported Profiles for Multiple Computer System

Registered Profile Names	Mandatory	Version
Storage Server Asymmetry	No	1.2.0

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 43.5.5.)

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

30.8 CIM Elements

Table 288 describes the CIM elements for Multiple Computer System.

Table 289 - 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
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 290 describes class CIM_ComponentCS.

Table 290 - 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 291 describes class CIM_ComputerSystem (Non-Top-Level System).

Table 291 - 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 292 describes class CIM_ConcreteIdentity.

Table 292 - 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 293 describes class CIM_IsSpare.

Table 293 - 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 294 describes class CIM_MemberOfCollection.

Table 294 - 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 295 describes class CIM_RedundancySet.

Table 295 - 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

IMPLEMENTED

Clause 31: Policy Package

31.1 Description

The Policy Package would be deployed by any profile or subprofile that provides Policy management capability. Any profile or subprofile that supports the Policy Package is referred to as a "**Policy based**" profile or subprofile. In the current version of SMI-S, there is no profile defined for a "**Global Policy Manager**" that provides policy management for a variety of other SMI-S profiles. The intent of this version of the SMI-S Policy Package is to support policy mechanisms "inside" Arrays, Storage Virtualizers, Volume Management, NAS, Storage Libraries, and Fabric components of a storage network. As a result, there are some limitations in the current version of the Policy Package and there are some simplifying assumptions that can be made about the Policy mechanisms. For example, most arrays today don't support providing a general policy mechanism for a storage network. The policies and the context of the execution of the policies are confined to the array.

There are, however, some complications that will be dealt with. In particular, cascading profiles, such as Volume Management, Storage Virtualizers and NAS Heads will have to deal with policies that derive context from other profiles (e.g., arrays and/or fabric). Note: In the future, the Policy Package will be expanded to support a Specific Policy Profile as implemented in a Global Policy Manager and this may raise additional requirements.

Note: This Package covers "Policy-Based" support. That is, it only covers implementation of Policy constructs (classes and associations) in the policy based profile or subprofile. It does not cover requirements on underlying profiles that may be used by the policy based profile or subprofile.

It is important to understand the limitations of the 1.1.0 Policy Package. While one could argue that a host based volume manager has a broad view of the storage network and could, in theory, perform policy based SAN management, there is no expectation that a volume manager will (or should) be the vehicle for SAN management. The policies that would be supported by a Volume Management Profile would be policies for automating certain administrative functions of the volume manager.

31.1.1 Instance Diagrams

Support for the Policy Package entails support for a number of constructs and the methods to support them. Any given implementation may support only a subset of the constructs and methods, based on how flexible their support is. This will be discussed in more detail in 31.6.1.

Policy constructs will be discussed in the following sections, starting with the basics and then building on those basics to describe more complicated functions and constructs.

31.1.1.1 Basics of Policy Support

The basic constructs used by the Policy Package are illustrated in Figure 41

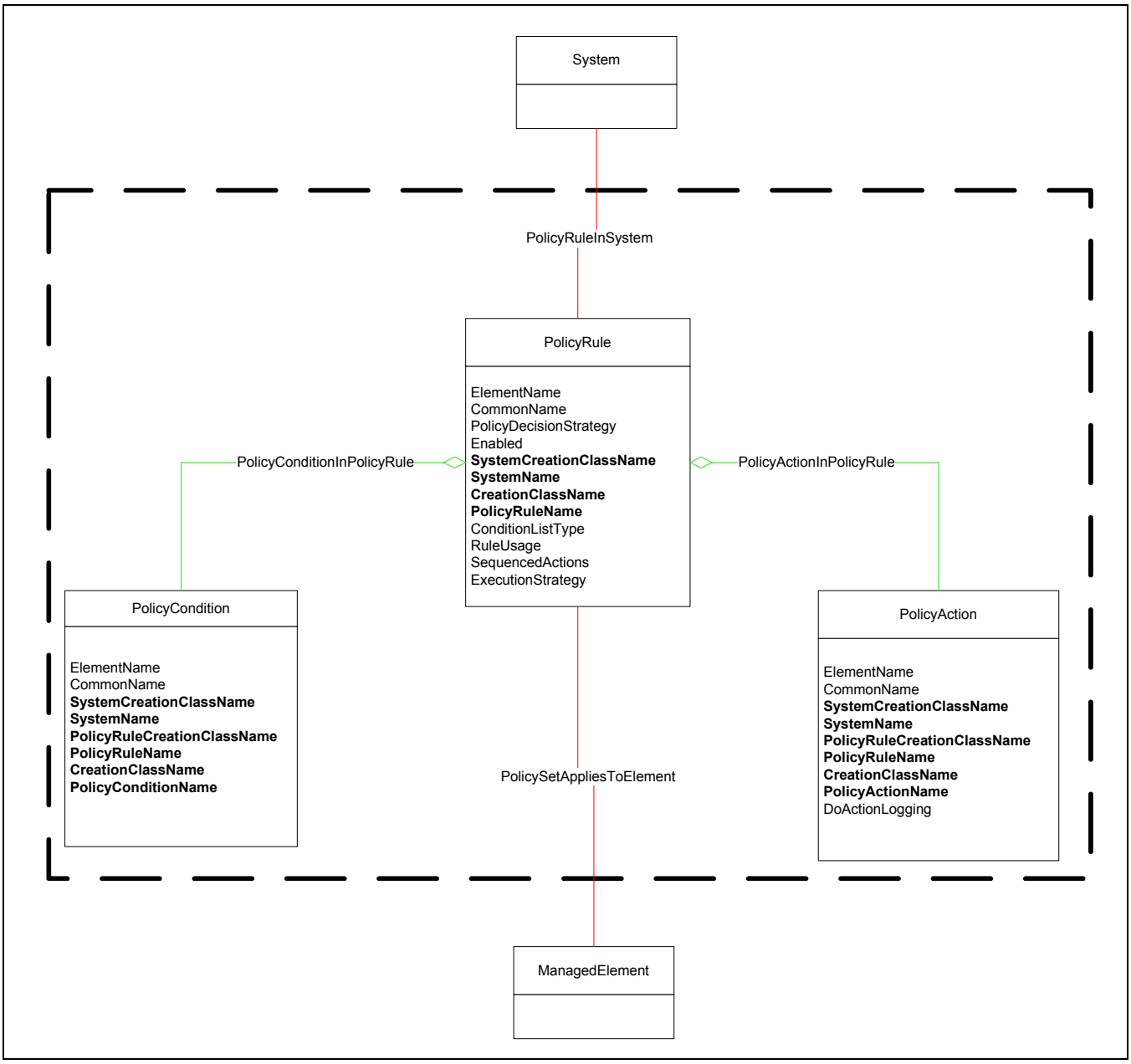

Figure 41 - Basic Policy Package Instance Diagram

There are five basic constructs that define a policy:

PolicyRule – This defines a policy to be applied. Specifically, it collects a number of other constructs that compose the policy.

PolicyCondition – A condition to be evaluated at the time the Policy Rule is checked. The PolicyCondition would be subclassed to a specific condition (e.g., QueryCondition) that can be evaluated in the context of the policy based profile or subprofile.

PolicyAction – An action to be executed based on conditions of the policy rule. The PolicyAction would be subclassed to a specific PolicyAction (e.g., a MethodAction) supported by the policy based profile or subprofile.

PolicySetAppliesToElement – An association that may be referenced by a PolicyCondition or PolicyAction (e.g., used as part of the query string in QueryConditions or MethodActions) to constrain the application of the PolicyRule. The “ManagedElement” would generally be any ManagedElement within the profile of the policy based profile or subprofile.

Note: In the case of a Policy-based cascading profile, the ManagedElement could be a reference to a ManagedElement in a leaf profile (see 31.3)

PolicyRuleInSystem – An association that is used to define the System scope of the PolicyRule. For policy based profiles or subprofiles, the system in question would be the “top level” system for the profile.

Note: In the case where a Policy-based cascading profile cascades to a Policy-based leaf profile, it is possible for a PolicyRule to be defined at the leaf and referenced by the cascading profile (i.e., cascading policy rules). See 31.3 for more information on this case.

In addition there are associations to define what Policy conditions are used in what Policy Rules (the PolicyConditionInPolicyRule association) and what Policy Actions are used by what Policy Rules (the PolicyActionInPolicyRule association).

A PolicyRule is the central class used for representing the 'If Condition then Action' semantics of a policy rule. A PolicyRule condition, in the most general sense, is represented as either an OR’ed set of AND’ed conditions (Disjunctive Normal Form, or DNF) or an AND’ed set of OR’ed conditions (Conjunctive Normal Form, or CNF). Individual conditions may either be negated (not C) or unnegated (C). The actions specified by a PolicyRule are to be performed if and only if the PolicyRule condition (whether it is represented in DNF or CNF) evaluates to TRUE.

The conditions and actions associated with a PolicyRule are modeled, respectively, with instances of PolicyCondition and PolicyAction. These condition and action objects are tied to instances of PolicyRule by the PolicyConditionInPolicyRule and PolicyActionInPolicyRule aggregations.

The PolicyRule class uses the property ConditionListType, to indicate whether the conditions for the rule are in DNF (disjunctive normal form), CNF (conjunctive normal form) or, in the case of a rule with no conditions, as an UnconditionalRule. The PolicyConditionInPolicyRule aggregation contains two additional properties to complete the representation of the Rule's conditional expression. The first of these properties is an integer to partition the referenced PolicyConditions into one or more groups, and the second is a Boolean to indicate whether a referenced Condition is negated.

31.1.1.2 Query Conditions

The basic constructs used by QueryConditions are illustrated in Figure 42

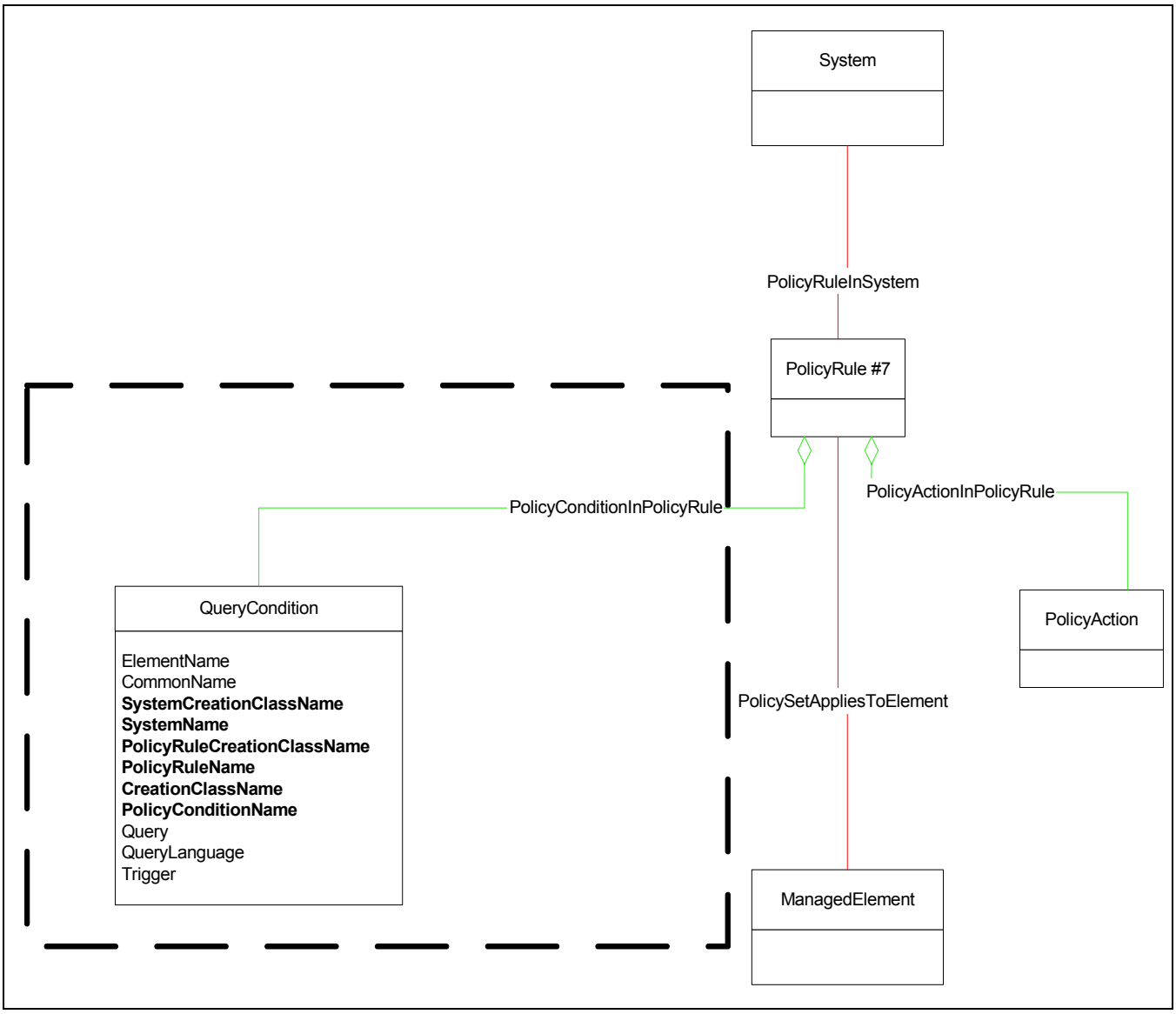

Figure 42 - Policy Package QueryCondition Support Instance Diagram

A QueryCondition is a subclass of PolicyCondition that defines the criteria for generating a set of instances that result from the contained query. If there are no instances returned from the query, then the result is false; otherwise, true.

Note: A QueryCondition instance has a Trigger property. This property indicates whether or not the query is to be used to trigger evaluation of all QueryConditions of the PolicyRule. If the Trigger Boolean is set to TRUE, then the QueryCondition is a trigger. When the QueryCondition evaluates to TRUE, then all the QueryConditions are evaluated.

Note: None, some or all query conditions in a PolicyRule may have the Trigger Boolean set to TRUE. If no Trigger property is set to TRUE, then the conditions are to be periodically evaluated (with the period selected by the policy based profile or subprofile). See 31.1.1.9.

The following query is an example of a QueryCondition query that might be used:

```
SELECT
     OBJECTPATH(primordial) AS POBJ,
     OBJECTPATH(concrete) AS COBJ,
     OBJECTPATH(service) AS SOBJ,
     concrete.TotalManagedSpace * .25 AS AmountToIncrease
FROM
     CIM_PolicyAppliesToElement applies,
     CIM_StoragePool concrete,
     CIM_StoragePool primordial.
     CIM_AllocatedFromStoragePool alloc,
     CIM_PolicySet policy,
     CIM_HostedService hosted,
     CIM_HostedStoragePool hostedpool,
     CIM_ComputerSystem, system,
     CIM_StorageConfigurationService service
WHERE    (concrete.RemainingManagedSpace/primordial.TotalManagedSpace * 100) < 75
    and concrete.Primordial = false
// Join Primordial Pool with Concrete Pools
    and OBJECTPATH(primordial) = alloc.Antecedent
    and OBJECTPATH(concrete) = alloc.Dependent
// Determine what concrete Pools the PolicySet applies to
    and  policy.CommonName = "Pool Exhausting Policy Condition"
    and  OBJECTPATH(policy) = element.PolicySet
    and  OBJECTPATH(concrete) = element.ManagedElement
// Join found primordial Pool with Service
    and  OBJECTPATH(primoridal) = hostedpool.PartComponent
    and  OBJECTPATH(system) = hostedpool.GroupComponent
    and OBJECTPATH(system) = hosted.Antecedent
    and  OBJECTPATH(service) = hosted.Dependent
    and  service ISA "CIM_StorageConfigurationService"
```

31.1.1.3 MethodActions

The basic constructs for MethodActions of the Policy Package are illustrated in Figure 43

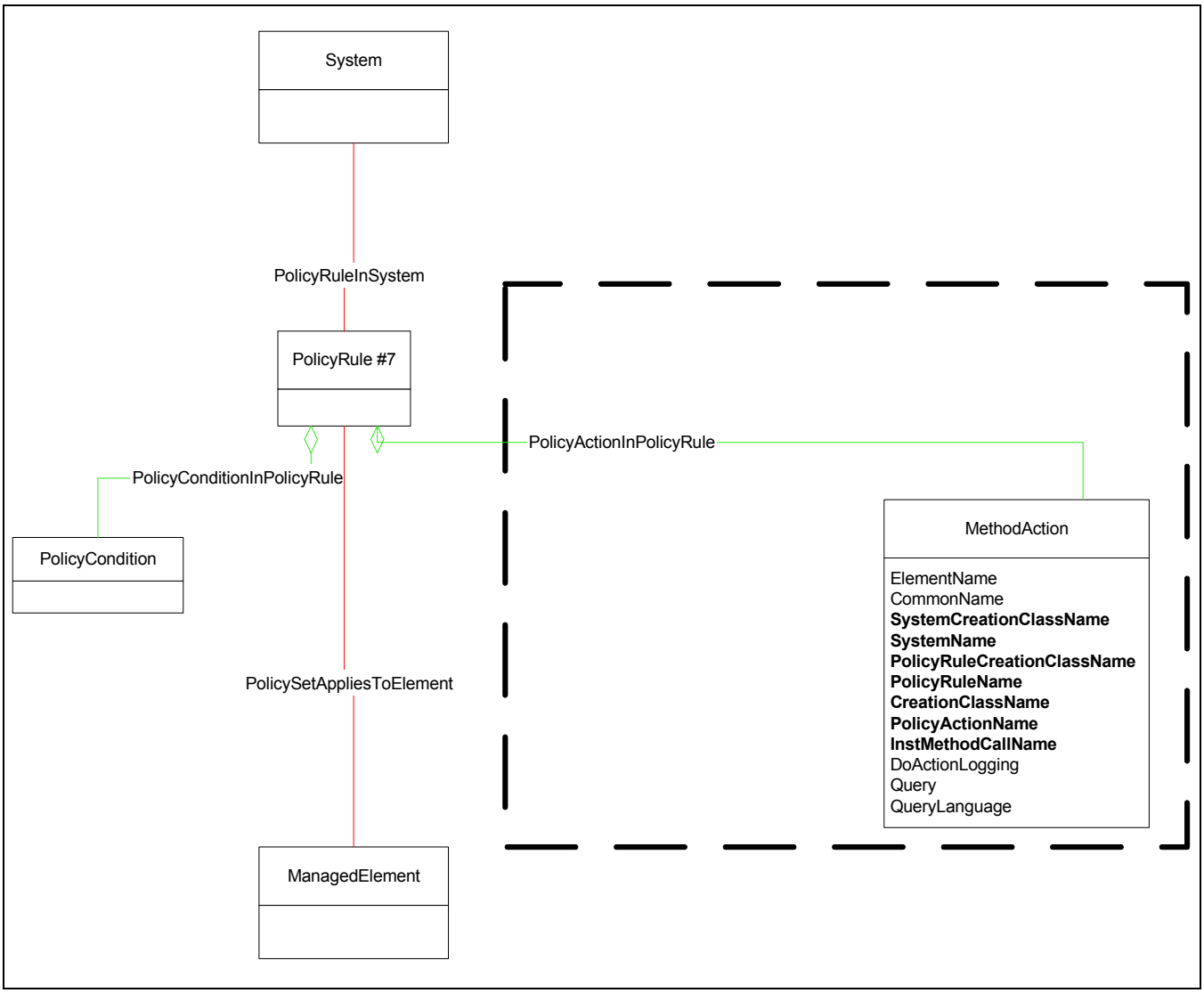

Figure 43 - Policy Package MethodAction Support Instance Diagram

A MethodAction is a PolicyAction that is a method that invokes an action defined by a query. The action is defined by a method of an ObjectName, which may be an intrinsic method of a CIM Namespace or an extrinsic method of a ManagedElement. The input parameters to the method are defined by the query and may be fixed values defined by literals or may be defined by reference to one or more properties of result instance from a QueryCondition query, a MethodAction query, or other instances.

The following query is an example of a MethodAction query that might be used:

```
SELECT
     SOBJ,     // Service object path
     'CreateOrModifyStoragePool',
     NULL,     // ElementName parameter
     NULL,     // Goal parameter, take default Setting
     AmountToIncrease, // Size parameter
```

```
     POBJ,     // InPools parameter
     NULL,     // InExtents parameter
     COBJ         // Pool parameter
FROM
     CIM_QueryCondition condition,
     CIM_QueryResult result,
     CIM_PolicySet policy,
     CIM_PolicyConditionInPolicyRule inpolicyset
WHERE
     policy.CommonName = "Pool Exhausting Policy Condition"
    and OBJECTPATH(policy) = inpolicyset.GroupComponent
    and OBJECTPATH(condition) = inpolicyset.PartComponent
    and CLASSNAME(result) = QueryResult.QueryResultSubclassName
```

31.1.1.4 PolicySetAppliesToElement

PolicySetAppliesToElement makes explicit which PolicyRules are currently applied to a particular Element. This association indicates that the PolicyRules that are appropriate for a ManagedElement (specified using the PolicyRoleCollection aggregation) have actually been implemented in the policy management infrastructure. One or more QueryCondition or MethodAction instances may reference the PolicySetAppliesToElement association as part of its query. PolicySetAppliesToElement shall not be used if the associated PolicyRule does not make use of the association. Note that if the named Element refers to a Collection, then the PolicyRule is assumed to be applied to all the members of the Collection.

PolicyRules are defined in the context of the System in which they apply. For policy based profiles or subprofiles, this is the “top level” system of the profile. The top level system can have many PolicyRules. A priority may be assigned to these rules using the Priority property of the PolicyRuleInSystem association.

31.1.1.5 Context Passing

The execution of a PolicyRule involves establishing and naming the results of Query execution in QueryConditions and Queries associated with MethodActions. These Query results are transient instances that only exist in the context of the PolicyRule. The QueryResultName is a Property of QueryCondition that identifies the output of the query in the QueryCondition instance. The InstMethodCallName is a Property of a MethodAction that identifies the output of the query in the MethodAction instance.

31.1.1.6 Static Rules Support

A policy based profile or subprofile may support a set of “Static” PolicyRules. These are PolicyRules that cannot be modified by a client (except for enabling or disabling the rule or defining a PolicySetAppliesToElement association). The constructs used for this are illustrated in Figure 44.

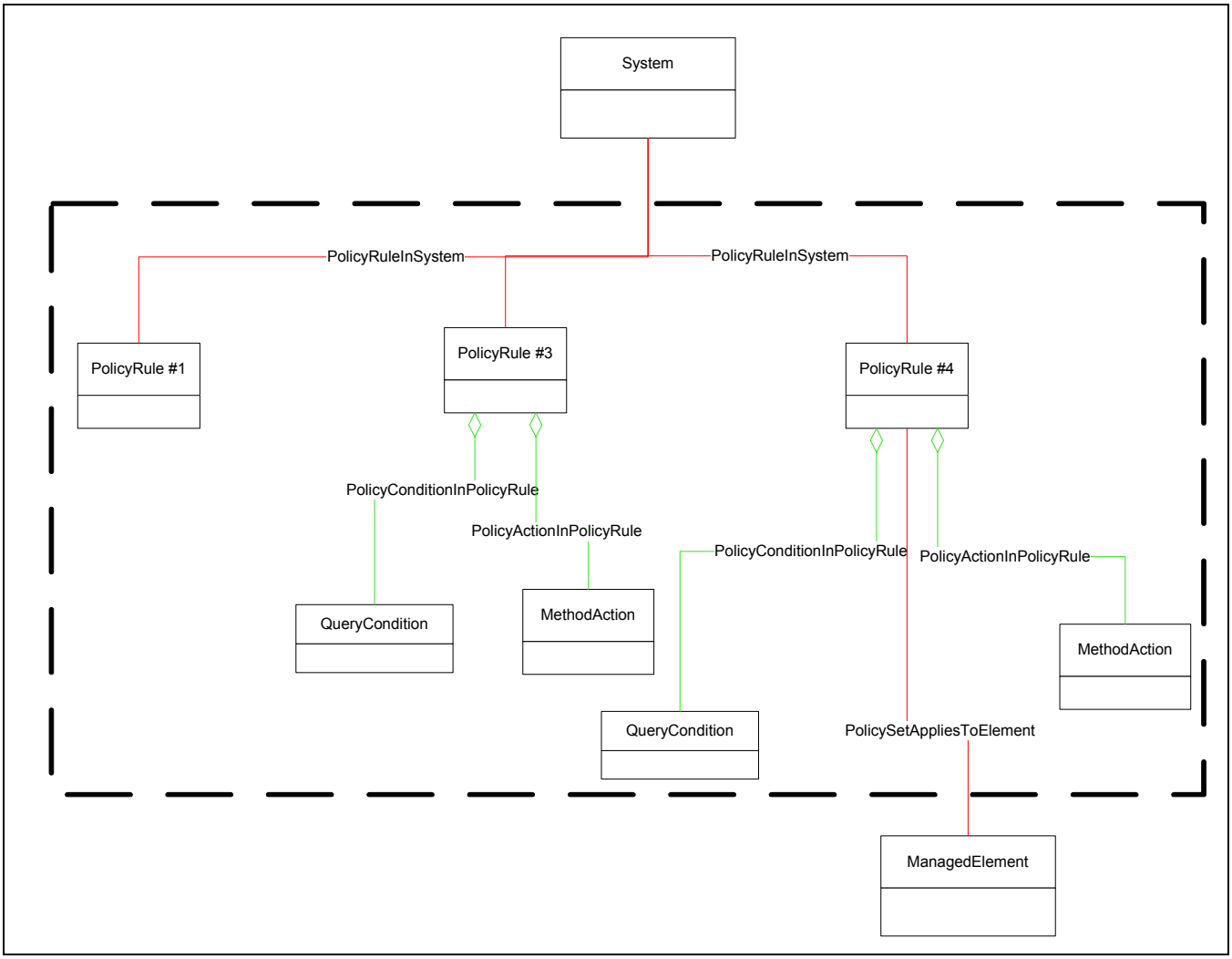

Figure 44 - Policy Package for Static Rules Instance Diagram

Figure 44 shows 3 static rules (PolicyRules #1, #3 and #4). These illustrate four distinct types of Static policy rules.

The first PolicyRule (PolicyRule #1) has no condition(s) and action(s) (or PolicySetAppliesToElement association). It merely names a specified policy rule. The only aspect of the PolicyRule that may (or may not) be changed is the “Enabled” property of the PolicyRule. This type of static policy rule is used to identify a behavior supported by the policy based profile. For example, Arrays might define a PolicyRule named “Controller Failover Type 1” or “Controller Failover Type 2” to indicate how controller failover works. Any particular Array Profile implementation would only support one of these PolicyRules. The client would determine behavior of failover by inspecting which PolicyRule is followed. But the actual behavior is not actually modeled in CIM. It is merely referenced using this simple form of static policy rules.

The second PolicyRule (PolicyRule #3) is has condition(s) and action(s), but is not referenced by any PolicySetAppliesToElement association. It behaves exactly like any other PolicyRule, except the QueryCondition(s) and MethodAction(s) are fixed and cannot be changed. The only aspects of the PolicyRule that

may (or may not) be changed is the “Enabled” property of the PolicyRule. This type of static policy rule is more descriptive than the first, in that it models conditions that are evaluated and actions that are taken.

The third PolicyRule (PolicyRule #4) has condition(s) and action(s), and is referenced by a PolicySetAppliesToElement association. It behaves exactly like any other PolicyRule, except the QueryCondition(s) and MethodAction(s) are fixed and cannot be changed. The only aspects of the PolicyRule that may be changed are the “Enabled” property of the PolicyRule and the PolicySetAppliesToElement association (to identify the managed element in which to apply the rule). In this case, the Query Condition or MethodAction refers to the PolicySetAppliesToElement association to constrain where or how the policy rule is applied. This type of static Policy Rule can be applied to specific managed elements in the profile. For example, an Array PolicyRule might define a policy for automatic extension of a StoragePool. The application of this policy to specific StoragePools would be governed by use of the PolicySetAppliesToElement.

Note: All PolicyRules have a PolicyRuleInSystem association to the System in which the PolicyRule is evaluated. In most cases, this will be the Top Level Object (System) for the policy based profile (i.e., the RegisteredProfile that a specific Policy RegisteredSubprofile supports). In order for the execution of the Policy to be constrained to the profile in question the QueryConditions and MethodActions should include a reference to PolicyRuleInSystem.

If any of these types of “Static Rules” are supported by a specific Policy Subprofile implementation then the PolicyFeaturesSupported array property of the PolicyCapabilities shall be set to include the “Static Rules” value.

31.1.1.7 Static Conditions and Actions

In addition to Static Rules, there are “Dynamic” PolicyRules that can be constructed using static conditions and static actions. The constructs used for this are illustrated in Figure 45.

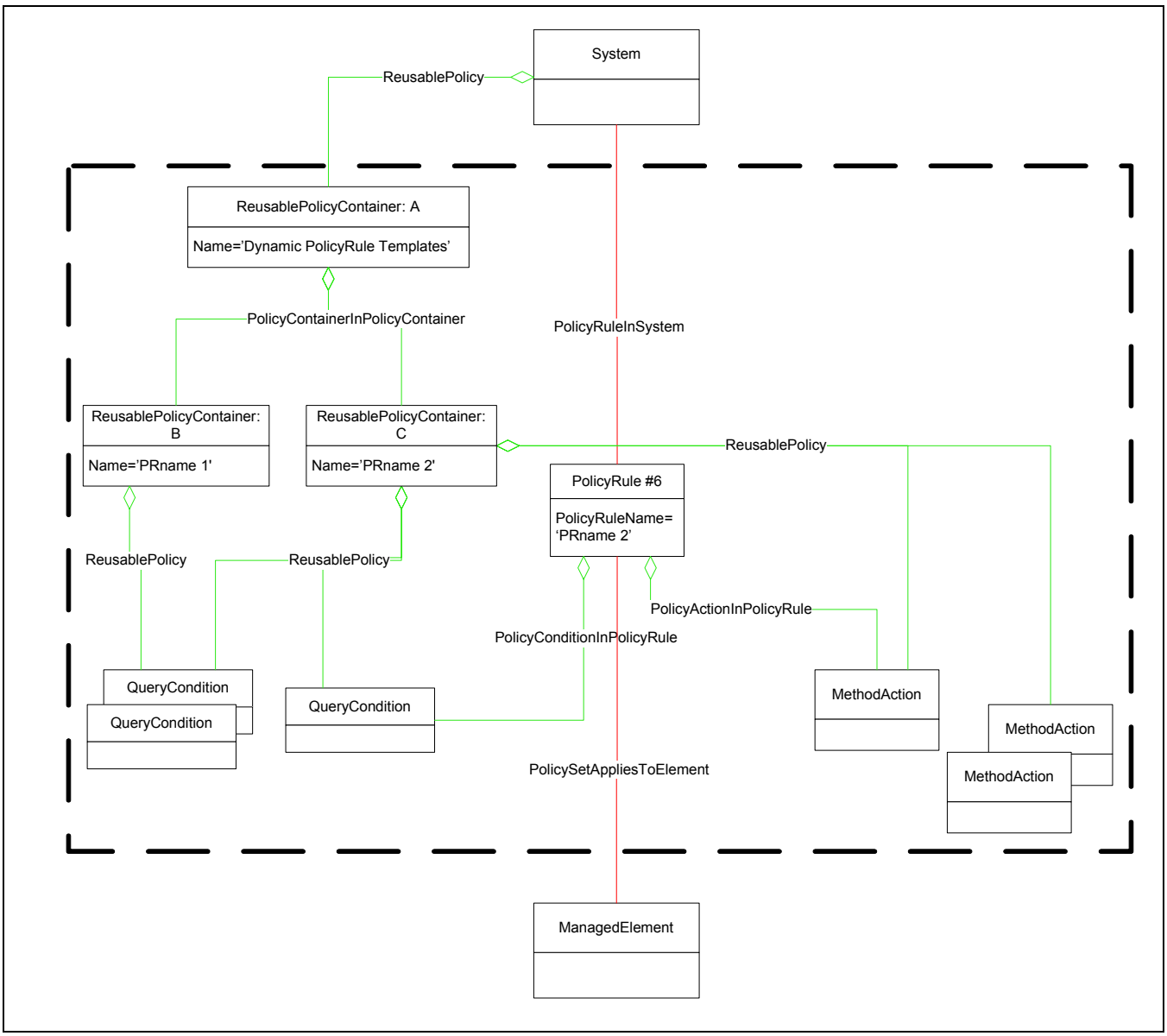

Figure 45 - Policy Package Support for Static Conditions and Actions Instance Diagram

Dynamic PolicyRules are constructed out of PolicyRule templates. In Figure 45, PolicyContainer C is a template, and PolicyRule #6 is the policy rule constructed from the template. The PolicyContainer C merely collects all the “static” Conditions and “Static” actions that may be used to construct the PolicyRule. The ReusablePolicy associations are what connects the QueryConditions and MethodActions to the ReusablePolicyContainer (template). Note that a QueryCondition or MethodAction may appear in multiple ReusablePolicyContainers (e.g., ReusablePolicyContainer B and ReusablePolicyContainer C share a common QueryCondition).

To construct PolicyRule #6, the client would need to create PolicyRule #6 (giving it a client defined name) and creating the associations to the conditions and actions that are desired.

Note: Creation of the PolicyRule and the associations to QueryConditions and MethodActions are done using the CreateInstance intrinsic. Until all associations are in place and correctly configured, the “Enabled” property of the PolicyRule should be “disabled.” Once everything is in place and correct, the client may enable the rule).

If any of these types of “Dynamic Rules” are supported by a specific Policy Subprofile implementation then the PolicyFeaturesSupported array property of the PolicyCapabilities shall be set to include the “Dynamic Rules” value.

31.1.1.8 Dynamic Conditions and Actions

The most general policy support includes support for dynamic conditions and actions. The constructs used for this are the basic policy constructs as illustrated in Figure 46.

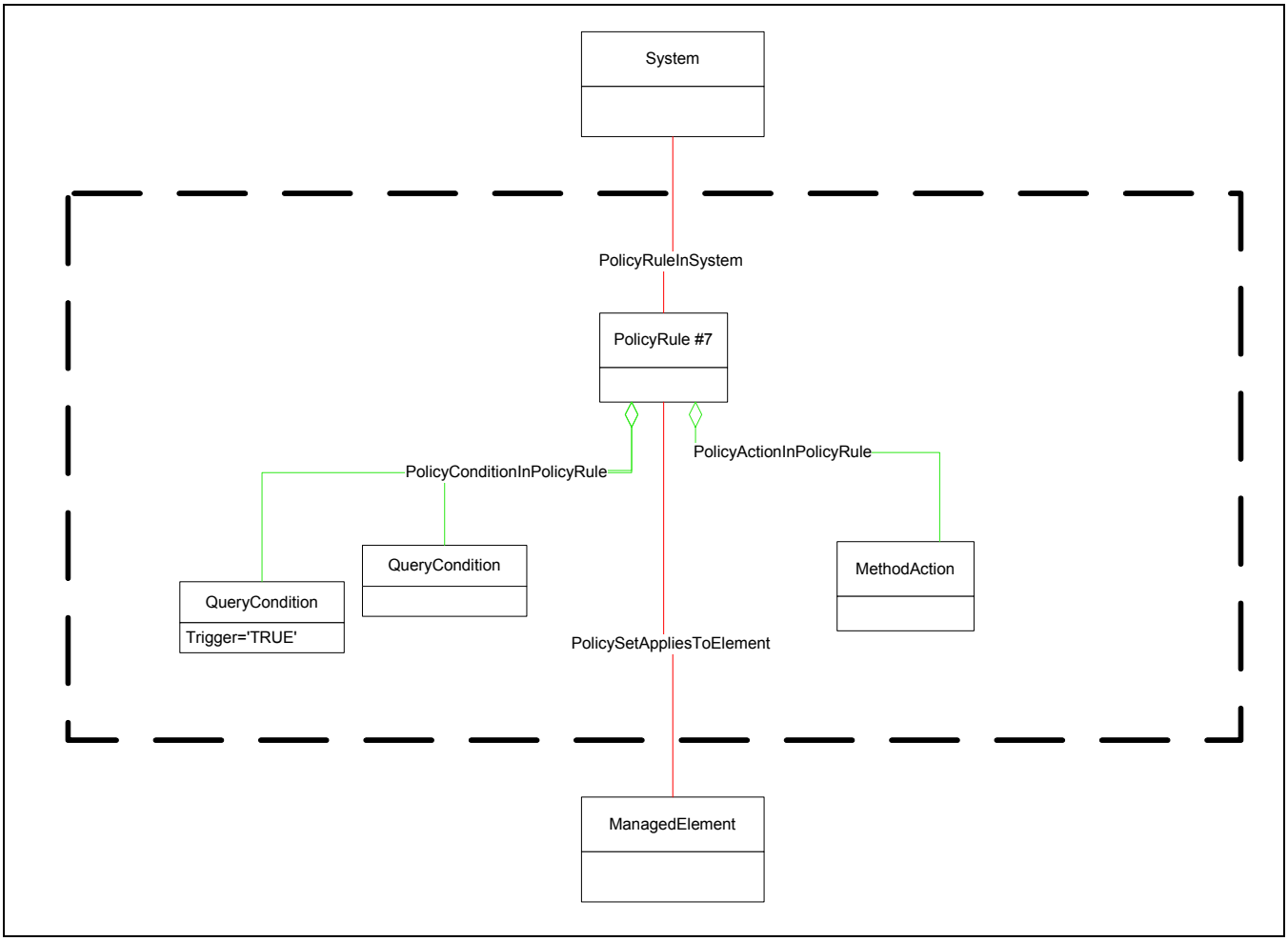

Figure 46 - Policy Package support for Dynamic Conditions and Actions Instance Diagram

In the dynamic conditions and actions case, all constructs are built using CreateInstance. The client would first create (and name) the PolicyRule, setting the Enabled property to ‘disabled’. Then the client would create the QueryConditions and MethodActions, and associate them to the PolicyRule.

Note: At least one QueryCondition should have a Trigger property of TRUE. If all the QueryConditions have a Trigger property of FALSE, the conditions will be evaluated at the convenience of the CIM server.

SMI-S only recognizes CQL Query statements in the QueryConditions. An implementation may support other QueryLanguages, but these would not be covered by SMI-S.

CQL defines “levels” of support. These levels are recognized for the purposes of Policy definitions. The CQL levels shall be identified in the CQLFeatures property of the QueryCapabilities instance associated to a specific Policy Subprofile (See 31.1.1.13.)

If this types of “Client defined rules” are supported by a specific Policy Subprofile implementation then the PolicyFeaturesSupported array property of the PolicyCapabilities shall be set to include the “Client Defined Rules” value.

31.1.1.9 Trigger Conditions

Trigger Conditions are QueryConditions that, when TRUE, cause evaluation of all conditions in the Policy Rule. A trigger condition is a QueryCondition with the Trigger property set to TRUE. This is illustrated in Figure 47.

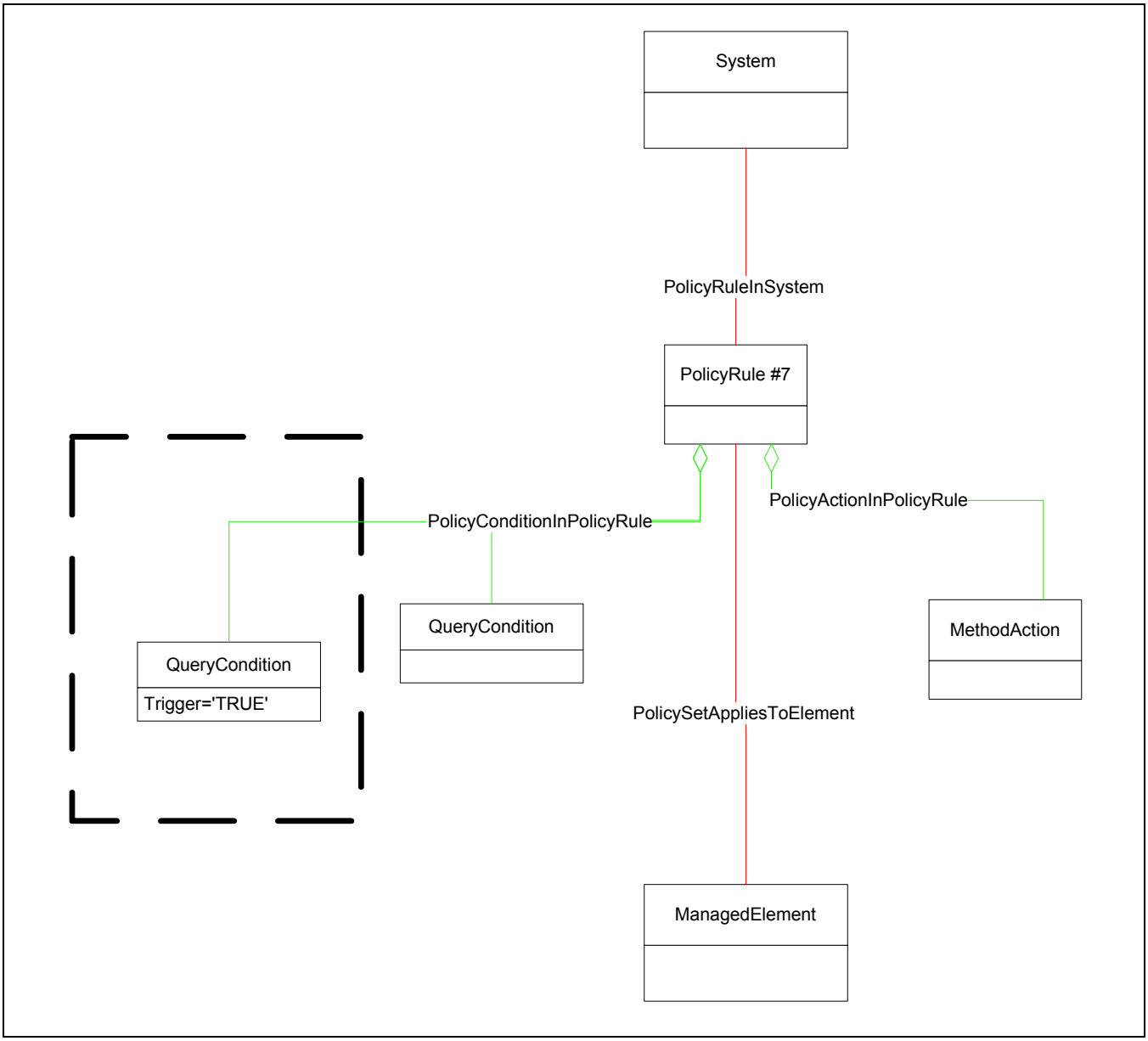

Figure 47 - Policy Package support for Trigger Conditions Instance Diagram

Figure 47 shows a PolicyRule with three QueryConditions. Two of the QueryConditions have Trigger set to TRUE. In the third, the Trigger property is set to FALSE. If either of the first two QueryConditions are true the third is evaluated.

31.1.1.10 TimePeriod Conditions

PolicyRules may be constrained by one or more time periods that define when the PolicyRule is to be active. The constructs used for this are illustrated in Figure 48 .

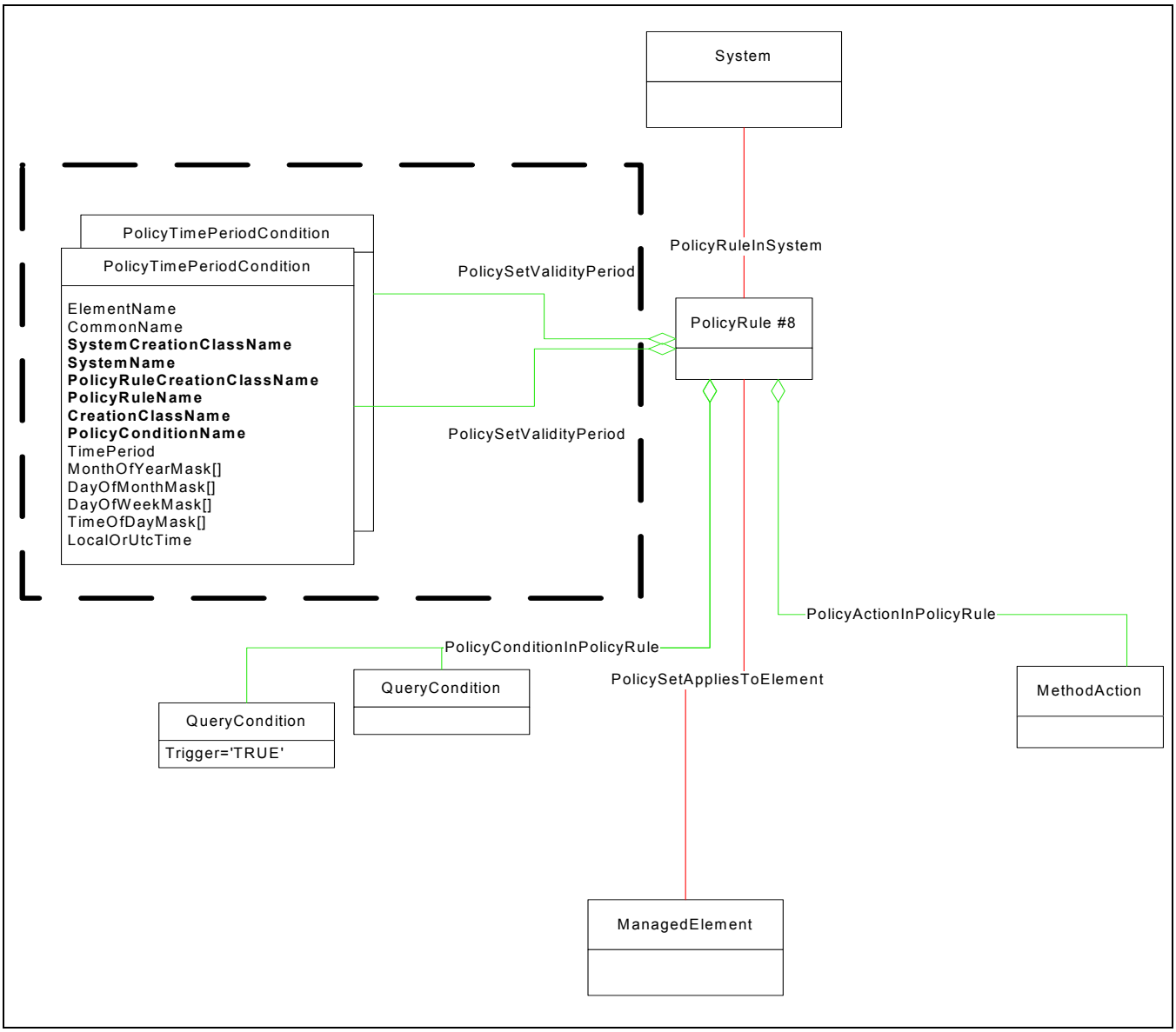

Figure 48 - Policy Package support for Time Periods Instance Diagram

A PolicyRule may also be associated with one or more policy time periods, indicating the schedule according to which the policy rule is active and inactive. In this case it is the PolicySetValidityPeriod aggregation that provides this linkage.

Evaluation of Policy conditions may be consider to be done in the following sequence:

1) Trigger Conditions - triggers are treated like indications to initiate evaluation of other conditions

2) TimePeriod Conditions - to determine if the remaining conditions need to be evaluated

3) Non-Trigger Conditions - the remaining Policy Conditions.

When there are compound conditions, the evaluation of each compound condition is evaluated independently. And the evaluation of a compound condition would follow the logical sequence described above.

When there are multiple PolicyTimePeriodConditions in a PolicyRule, then all shall evaluate to true. If there are no PolicyTimePeriodConditions specified in a PolicyRule, then all times are valid.

There are also two special cases in which one of the date/time strings is replaced with a special string defined in RFC 2445.

- If the first date/time is replaced with the string 'THISANDPRIOR', then the property indicates that a PolicyRule is valid [from now] until the date/time that appears after the '/'.
- If the second date/time is replaced with the string 'THISANDFUTURE', then the property indicates that a PolicyRule becomes valid on the date/time that appears before the '/', and remains valid from that point on.

31.1.1.11 Compound Conditions

QueryConditions may be aggregated into rules and into compound conditions. The constructs used for this are illustrated in Figure 49

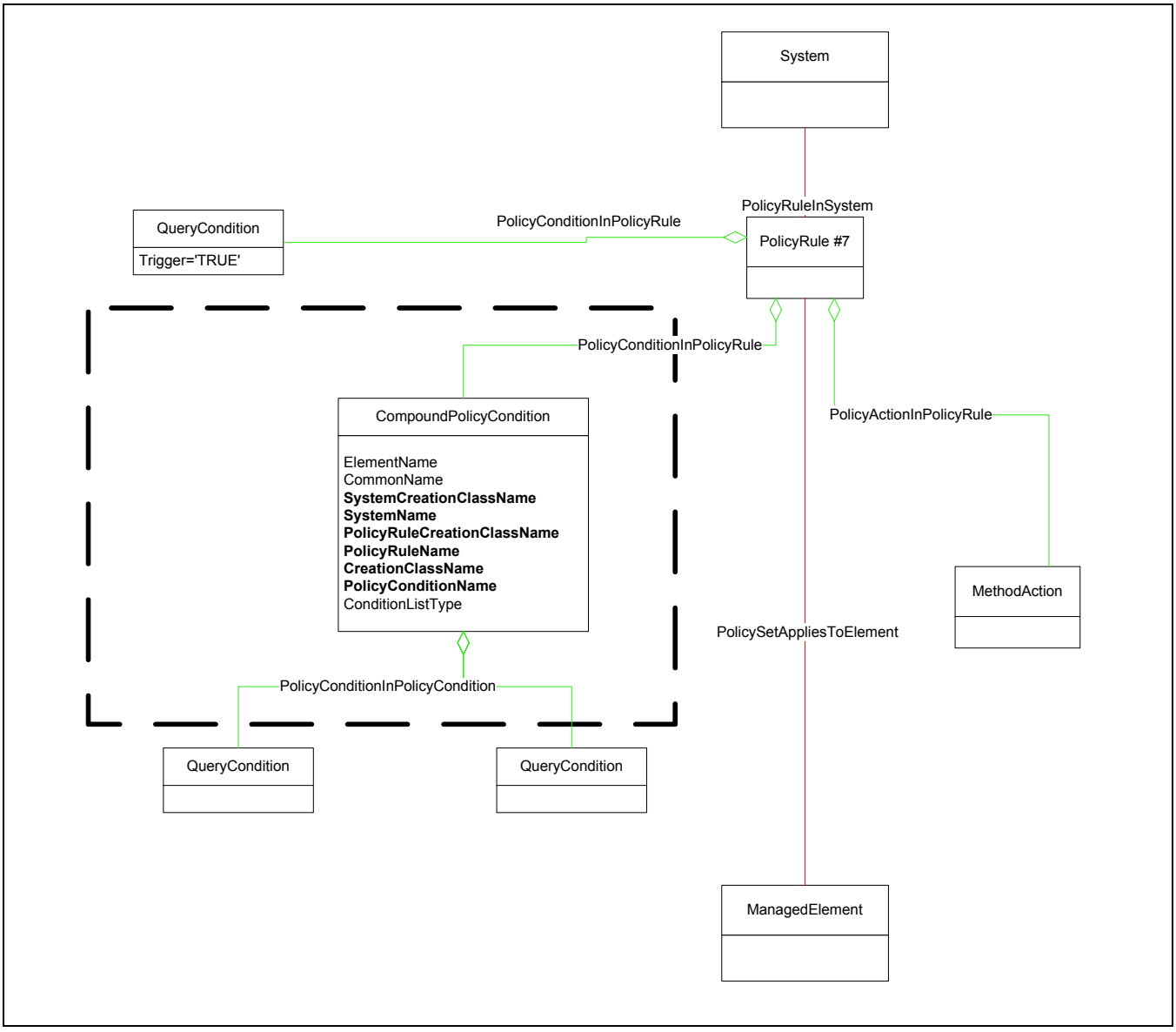

Figure 49 - Policy Package support for Compound Conditions Instance Diagram

A PolicyRule aggregates zero or more instances of the QueryCondition class, via the PolicyConditionInPolicyRule association. A Rule that aggregates zero Conditions is not valid; it may, however, be in the process of being defined. Note that a PolicyRule should have no effect until it is enabled.

QueryConditions may be aggregated into rules and into compound conditions. PolicyConditionStructure is the abstract aggregation class for the structuring of policy conditions.

The Conditions aggregated by a PolicyRule or CompoundPolicyCondition are grouped into two levels of lists: either an OR'ed set of AND'ed sets of conditions (DNF, the default) or an AND'ed set of OR'ed sets of conditions (CNF). Individual QueryConditions in these lists may be negated. The property ConditionListType specifies which of these two grouping schemes applies to a particular PolicyRule or CompoundPolicyCondition instance.

One or more PolicyTimePeriodConditions may be among the conditions associated with a PolicyRule or CompoundPolicyCondition via the PolicyConditionStructure subclass association. In this case, the time periods are simply additional Conditions to be evaluated along with any others that are specified.

A CompoundPolicyCondition aggregates zero or more instances of the QueryCondition class, via the PolicyConditionInPolicyCondition association. A CompoundPolicyCondition that aggregates zero Conditions is not valid; it may, however, be in the process of being defined. Note that a CompoundPolicyCondition should have no effect until it is valid.

31.1.1.12 Compound Actions

PolicyActions may be aggregated into rules and into compound actions. The constructs used for this are illustrated in Figure 50

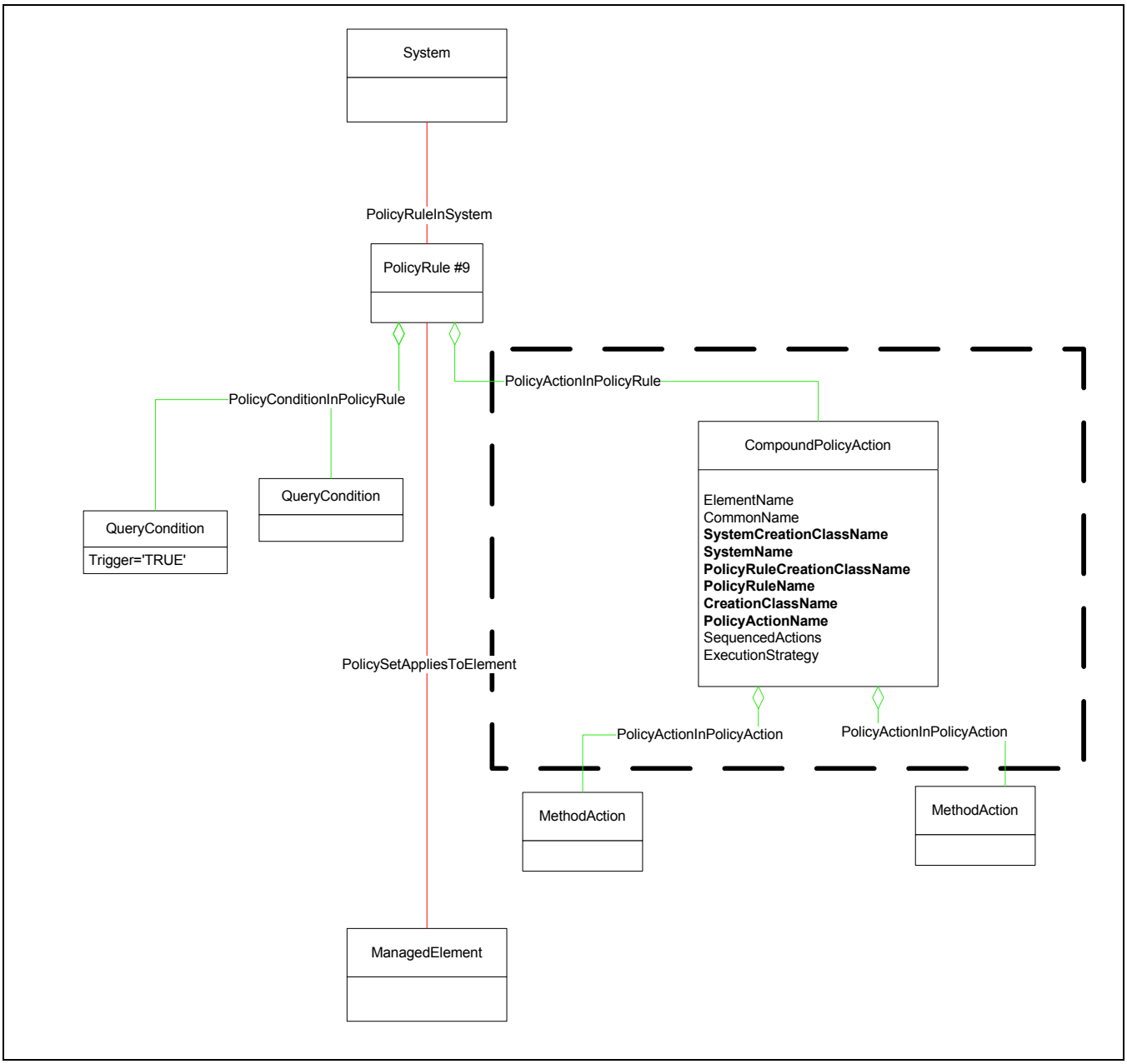

Figure 50 - Policy Package support for Compound Actions Instance Diagram

A PolicyRule aggregates zero or more instances of the PolicyAction class, via the PolicyActionInPolicyRule association. A Rule that aggregates zero Actions is not valid--it may, however, be in the process of being entered into a PolicyRepository or being defined for a System. Alternately, the actions of the policy may be explicit in the definition of the PolicyRule. Note that a PolicyRule should have no effect until it is valid.

The Actions associated with a PolicyRule may be given a required order, a recommended order, or no order at all. For Actions represented as separate objects, the PolicyActionInPolicyRule aggregation can be used to express an order.

This aggregation does not indicate whether a specified action order is required, recommended, or of no significance; the property SequencedActions in the aggregating instance of PolicyRule provides this indication.

A series of examples will make ordering of PolicyActions clearer: ActionOrder is an unsigned integer 'n' that indicates the relative position of a PolicyAction in the sequence of actions associated with a PolicyRule or CompoundPolicyAction. When 'n' is a positive integer, it indicates a place in the sequence of actions to be performed, with smaller integers indicating earlier positions in the sequence. The special value '0' indicates 'don't care'. If two or more PolicyActions have the same non-zero sequence number, they may be performed in any order, but they shall all be performed at the appropriate place in the overall action sequence.

If all actions have the same sequence number, regardless of whether it is '0' or non-zero, any order is acceptable.

The values:

1:ACTION A

2:ACTION B

1:ACTION C

3:ACTION D

indicate two acceptable orders: A,C,B,D or C,A,B,D,

since A and C can be performed in either order, but only at the '1' position.

The values:

0:ACTION A

2:ACTION B

3:ACTION C

3:ACTION D

require that B,C, and D occur either as B,C,D or as B,D,C. Action A may appear at any point relative to B, C, and D. Thus the complete set of acceptable orders is: A,B,C,D; B,A,C,D; B,C,A,D; B,C,D,A; A,B,D,C; B,A,D,C; B,D,A,C; B,D,C,A.

Note: The non-zero sequence numbers need not start with '1', and they need not be consecutive. All that matters is their relative magnitude.

EXPERIMENTAL

31.1.1.13 Policy (and Query) Capabilities

Implementations of a specific Policy Subprofile can vary in degree of support. The degree of support provided by an implementation can be determined by inspection of the QueryCapabilities and PolicyCapabilities. The constructs used for this are illustrated in Figure 51

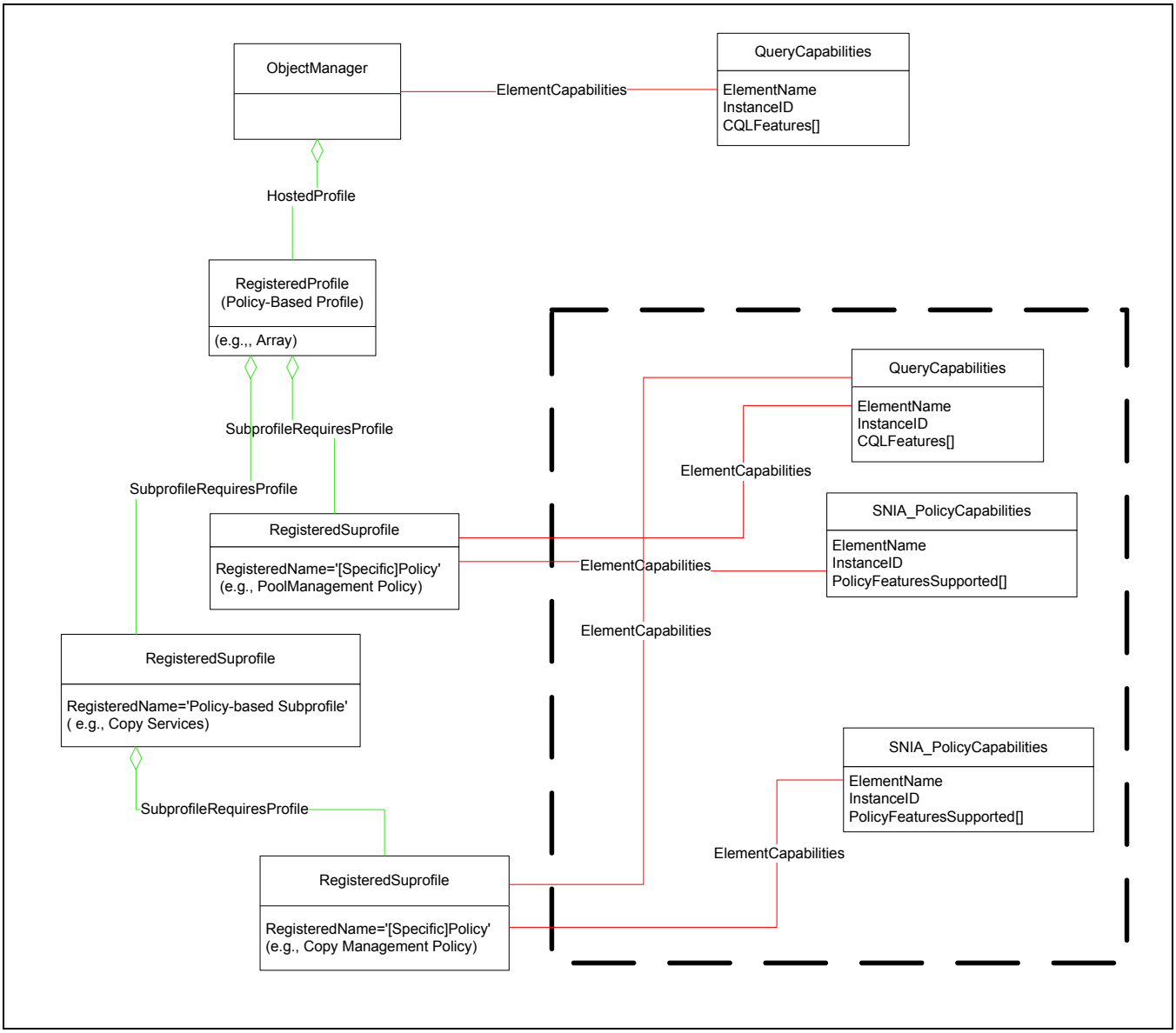

Figure 51 - Policy Package support for Policy Capabilities Instance Diagram

In this figure, the policy based profile is an Array Profile. And it has two Specific Policy Subprofiles:

1) a Pool Management Policy Subprofile and,
2) a Copy Management Policy.

Each of these subprofiles shall have their Policy capabilities defined by associating an instance of PolicyCapabilities to each. Similarly, each may refer to a QueryCapabilities instance.

A policy based profile or subprofile would identify its basic capabilities using 2 capabilities classes: A QueryCapabilities class instance and a PolicyCapabilities class instance. Both instances will be associated to a specific Policy RegisteredSubprofile of the Policy-based RegisteredProfile. These classes and associations should be populated in the InterOp Namespace (with the RegisteredSubprofile). If they are populated in the policy based profile namespace, then the ElementCapabilities associations shall (at least) be populated in the InteropNamespace.

Also shown in Figure 51 are the ObjectManager (representing the CIM Server) and its QueryCapabilities instance. The QueryCapabilities instance that is associated to the ObjectManager represents the general capabilities of the CIM Server and may offer more capabilities than are supported for defining QueryConditions for PolicyRules. This instance is not part of the Policy Package (or either of the specific Policy Subprofiles). The ObjectManager version of the QueryCapabilities need not be present. The QueryCapabilities associated with a Specific Policy Subprofile is mandatory if the profile supports “Client defined” QueryConditions. If Client defined QueryConditions are not supported by the profile or subprofile, then the QueryCapabilities instance is not needed for the Specific Policy Subprofile.

The QueryCapabilities that may be supported for the purpose of client defined policies are “Basic Query”, “Simple Join”, “Complex Join”, “Time”, “Basic Like”, ”Full Like”, “Array Elements”, ”Embedded Objects“, “Order By”, “Aggregations”, “Subduer”, “Satisfies Array”, “Distinct”, “Forestland “Path Functions”. For definitions of these values see the CIM Query Language Specification.

> Any or all of these may be specified in the QueryCapabilities associated with a specific Policy Subprofile.

The second capabilities instance associated to the specific Policy Subprofile is the PolicyCapabilities instance. The PolicyCapabilities class has the following properties that define the capabilities of the subprofile:

PolicyFeaturesSupported[]

“Static Rules” – Static rules are pre-defined by the profile implementation and are available to the Client to enable and disable (or set PolicySetAppliesToElement).

“Dynamic Rules” – Dynamic rules means that the profile implementation has populated PolicyContainers that include QueryConditions and MethodActions that can be constructed into Client specified PolicyRules (using the conditions and actions in the container).

“Client Defined Rules” – Client Defined Rules means that a Client may create its own PolicyRules, specifying its own (Client invented) QueryConditions and MethodActions. The QueryConditions and MethodActions shall, of course, be valid operations on the profile in question. That is, QueryConditions shall address class instances and properties that are part of the profile model and the MethodActions shall be actions supported by the profile.

EXPERIMENTAL

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

Table 301 describes the supported profiles for Policy.

Table 296 - Supported Profiles for Policy

Registered Profile Names	Mandatory	Version
Server	Yes	1.3.0

31.5 Methods of the Profile

31.5.1 Extrinsic Methods of the Profile (EXPERIMENTAL)

There are no Extrinsic methods defined for this Package. All Policy manipulation actions are done using intrinsic methods. These are described in 31.5.2 and illustrated in 31.6. However, it is recognized that some Extrinsic Methods may make Policy manipulation a lot easier and more efficient for clients. Such methods will be considered in a future release.

31.5.2 Intrinsic Methods of the Profile

Table 297 identifies how Policy constructs get created, deleted or modified. Any class not listed is assumed to be pre-existing (e.g., canned) or manipulated through another profile or subprofile.

Table 297 - Static Policy Instance Manipulation Methods

Method	Created Instances	Deleted Instances	Modified Instances
CreateInstance	PolicySetAppliesToElement	N/A	N/A
DeleteInstance	N/A	PolicySetAppliesToElement	N/A
SetProperty	N/A	N/A	PolicyRule (Enabled)

Table 298 identifies how Policy constructs get created, deleted or modified. Any class not listed is assumed to be pre-existing (e.g., canned) or manipulated through another profile or subprofile.

Table 298 - Dynamic Policy Instance Manipulation Methods

Method	Created Instances	Deleted Instances	Modified Instances
CreateInstance	PolicyRule	N/A	N/A
CreateInstance	PolicyConditionInPolicyRule	N/A	N/A
CreateInstance	PolicyActionInPolicyRule	N/A	N/A
DeleteInstance	N/A	PolicyRule	N/A
DeleteInstance	N/A	PolicyConditionInPolicyRule	N/A
DeleteInstance	N/A	PolicyActionInPolicyRule	N/A

Table 298 - Dynamic Policy Instance Manipulation Methods (Continued)

ModifyInstance	N/A	N/A	PolicyRule (Enabled)
ModifyInstance	N/A	N/A	QueryCondition (Trigger)

Table 299 identifies how Policy constructs get created, deleted or modified for dynamic policies.

Table 299 - Methods that cause Instance Creation, Deletion or Modification of Dynamic Policy Rules

Method	Created Instances	Deleted Instances	Modified Instances
CreateInstance	PolicyRule	N/A	N/A
CreateInstance	QueryCondition	N/A	N/A
CreateInstance	PolicyConditionInPolicyRule	N/A	N/A
CreateInstance	PolicyConditionIn PolicyCondition	N/A	N/A
CreateInstance	CompoundPolicyCondition	N/A	N/A
CreateInstance	PolicySetValidityPeriod	N/A	N/A
CreateInstance	PolicyTimePeriodCondition	N/A	N/A
CreateInstance	CompoundPolicyAction	N/A	N/A
CreateInstance	MethodAction	N/A	N/A
CreateInstance	PolicyActionInPolicyRule	N/A	N/A
CreateInstance	PolicyActionInPolicyAction	N/A	N/A
DeleteInstance	N/A	PolicyRule	N/A
DeleteInstance	N/A	QueryCondition	N/A
DeleteInstance	N/A	MethodAction	N/A
DeleteInstance	N/A	PolicyConditionIn PolicyRule	N/A
DeleteInstance	N/A	PolicyActionInPolicyRule	N/A
DeleteInstance	N/A	CompoundPolicyCondition	N/A
DeleteInstance	N/A	PolicyConditionIn PolicyCondition	N/A
DeleteInstance	N/A	CompoundPolicyAction	N/A
DeleteInstance	N/A	PolicyActionInPolicyAction	N/A
DeleteInstance	N/A	PolicySetValidityPeriod	N/A
DeleteInstance	N/A	PolicyTimePeriodCondition	N/A

Table 299 - Methods that cause Instance Creation, Deletion or Modification of Dynamic Policy Rules (Continued)

ModifyInstance	N/A	N/A	PolicyRule (Enabled)
ModifyInstance	N/A	N/A	PolicyRuleInSystem
ModifyInstance	N/A	N/A	QueryCondition (Trigger)
ModifyInstance	N/A	N/A	QueryCondition
ModifyInstance	N/A	N/A	MethodAction
ModifyInstance	N/A	N/A	PolicyConditionIn PolicyRule
ModifyInstance	N/A	N/A	PolicyActionInPolicyRul e
ModifyInstance	N/A	N/A	CompoundPolicy Condition
ModifyInstance	N/A	N/A	PolicyConditionIn PolicyCondition
ModifyInstance	N/A	N/A	CompoundPolicyAction
ModifyInstance	N/A	N/A	PolicyTimePeriod Condition
ModifyInstance	N/A	N/A	PolicyActionIn PolicyAction

CreateInstance

CreateInstance (

[IN] <instance> NewInstance

)

The CreateInstance intrinsic method is used for the creation of PolicyRules, QueryConditions, ReusablePolicyContainers and MethodActions, It is also used to create PolicyConditionInPolicyRule associations, PolicyConditionInPolicyCondition associations, ReusablePolicyComponent associations, PolicyActionInPolicyRule associations, PolicyActionInPolicyAction associations and PolicySetAppliesToElement associations.

Care should be taken when creating a policy. The following sequence should be followed for enabling **Static Policies**:

- Creation of the PolicyRule (disabled)
- Creation of the QueryCondition(s)
- Immediately followed by Creation of the PolicyConditionInPolicyRule association(s)
- Creation of the MethodAction(s)
- Immediately followed by Creation of the PolicyActionInPolicyRule association(s)
- Creation of one or more PolicySetAppliesToElement associations (if needed)

- ModifyInstance of the PolicyRule (to enable)

If this sequence in not followed, there is no guarantee that the desired Policy will be created. Also note that all steps would need to successfully execute to ensure creation of any of the instances involved in the PolicyRule.

If instances created are not immediately associated with an appropriate PolicyRule, they may be lost. A provider is not required to keep “dangling” instances around indefinitely. Indeed, they are expected to do periodic clean up of “dangling” instances.

The above sequence may not need to be done if there is no PolicySetAppliesToElement. In this case, all copies of the static policy are the same. All that is required is to enable (ModifyInstance) the PolicyRule.

The following sequence should be followed for creating **Dynamic PolicyRules**:

- Creation of the PolicyRule (disabled) (based on a ReusablePolicyContainer)
- Associate selected QueryConditions to the PolicyRule
- Associate selected MethodActions to the PolicyRule
- Create the appropriate PolicySetAppliesToElement associations
- ModifyInstance of the PolicyRule (to enable)

If this sequence in not followed, there is no guarantee that the desired Policy will be created. Also note that all steps would need to successfully execute to ensure creation of any of the instances involved in the PolicyRule

The following sequence should be followed for creating **Client Defined Policies**:

- Creation of the PolicyRule (disabled)
- Creation of the QueryCondition(s)
- Immediately followed by Creation of the PolicyConditionInPolicyRule association(s)
- Creation of the MethodAction(s)
- Immediately followed by Creation of the PolicyActionInPolicyRule association(s)
- Creation of one or more PolicySetAppliesToElement associations (if needed)
- ModifyInstance of the PolicyRule (to enable)

If this sequence in not followed, there is no guarantee that the desired Policy will be created. Also note that all steps would need to successfully execute to ensure creation of any of the instances involved in the PolicyRule

DeleteInstance

Not defined in this standard.

ModifyInstance

Not defined in this standard.

31.6 Client Considerations and Recipes

31.6.1 SMI-S Supported PolicyCapabilities and QueryCapabilities Patterns

The PolicyCapabilities patterns that are formally recognized by the current version of SMI-S are shown in Table 300.

Table 300 - SMI-S Supported PolicyCapabilities Patterns

PolicyLevels Supported
Static Rules
Static Rules, Dynamic Rules
Static Rules, Client Defined Rules
Static Rules, Dynamic Rules, Client Defined Rules
Dynamic Rules
Dynamic Rules, Client Defined Rules
Client Defined Rules

31.7 Registered Name and Version

Policy version 1.2.0

31.8 CIM Elements

Table 301 describes the CIM elements for Policy.

Table 301 - CIM Elements for Policy

Element Name	Requirement	Description
31.8.1 CIM_CompoundPolicyAction (Client defined)	Optional	A Client defined Policy action that groups multiple method actions as a unit.
31.8.2 CIM_CompoundPolicyAction (Pre-defined)	Optional	A predefined Policy action that groups multiple method actions as a unit.
31.8.3 CIM_CompoundPolicyCondition (Client defined)	Optional	A Client defined Policy condition that groups multiple query conditions as a unit.
31.8.4 CIM_CompoundPolicyCondition (Pre-defined)	Optional	A predefined Policy condition that groups multiple query conditions as a unit
31.8.5 CIM_ElementCapabilities (Policy Capabilities)	Optional	This associates the SNIA_PolicyCapabilities to the specific Policy RegisteredSubprofile.
31.8.6 CIM_ElementCapabilities (Query Capabilities)	Optional	This associates the QueryCapabilities to the specific Policy RegisteredSubprofile.
31.8.7 CIM_MethodAction (Client defined)	Optional	Defines a Method (Client defined) to be executed as part of a PolicyRule
31.8.8 CIM_MethodAction (Pre-defined)	Optional	Defines a Method (predefined) to be executed as part of a PolicyRule
31.8.9 CIM_PolicyActionInPolicyAction (Client defined)	Optional	Associates a MethodAction to a Client defined CompoundPolicyAction.
31.8.10 CIM_PolicyActionInPolicyAction (Pre-defined)	Optional	Associates a MethodAction to a predefined CompoundPolicyAction.
31.8.11 CIM_PolicyActionInPolicyRule (Client defined)	Optional	Associates a MethodAction to the Client defined PolicyRule of which it is a part.

Table 301 - CIM Elements for Policy

Element Name	Requirement	Description
31.8.12 CIM_PolicyActionInPolicyRule (Pre-defined)	Optional	Associates a MethodAction to the predefined PolicyRule of which it is a part.
31.8.13 CIM_PolicyConditionInPolicyCondition (Client defined)	Optional	Associates a QueryCondition to a Client defined CompoundPolicyCondition.
31.8.14 CIM_PolicyConditionInPolicyCondition (Pre-defined)	Optional	Associates a QueryCondition to a predefined CompoundPolicyCondition.
31.8.15 CIM_PolicyConditionInPolicyRule (Client defined)	Optional	Associates a Client defined QueryCondition to the PolicyRules of which it is part.
31.8.16 CIM_PolicyConditionInPolicyRule (Pre-defined)	Optional	Associates a predefined QueryCondition to the PolicyRules of which it is part.
31.8.17 CIM_PolicyContainerInPolicyContainer	Optional	Association that collects PolicyContainers in other PolicyContainers.
31.8.18 CIM_PolicyRule (Dynamic or Client defined)	Optional	Defines a PolicyRule created by a client (Dynamic or Client Defined policy).
31.8.19 CIM_PolicyRule (Pre-defined)	Optional	Defines a Static (predefined) PolicyRule.
31.8.20 CIM_PolicyRuleInSystem (Dynamic or Client defined)	Optional	Associates Dynamic or Client Defined PolicyRules to the System that hosts them.
31.8.21 CIM_PolicyRuleInSystem (Pre-defined)	Optional	Associates Static PolicyRules to the System that hosts them.
31.8.22 CIM_PolicySetAppliesToElement (Dynamic or Client defined)	Optional	An association that may be referenced in QueryConditions or MethodActions to constrain the application of a Dynamic or Client defined PolicyRule. It associates the PolicyRule to ManagedElements.
31.8.23 CIM_PolicySetAppliesToElement (Pre-defined)	Optional	An association that may be referenced in QueryConditions or MethodActions to constrain the application of a predefined PolicyRule. It associates the PolicyRule to ManagedElements.
31.8.24 CIM_PolicySetValidityPeriod (Dynamic or Client defined)	Optional	Associates a PolicyTimePeriodCondition to a Dynamic or client defined PolicyRule.
31.8.25 CIM_PolicySetValidityPeriod (Pre-defined)	Optional	Associates a PolicyTimePeriodCondition to a predefined PolicyRule.
31.8.26 CIM_PolicyTimePeriodCondition (Dynamic or Client defined)	Optional	A Dynamic or Client defined PolicyCondition that specifies the valid time period for Policy activation.
31.8.27 CIM_PolicyTimePeriodCondition (Pre-defined)	Optional	A predefined PolicyCondition that specifies the valid time period for Policy activation.

Table 301 - CIM Elements for Policy

Element Name	Requirement	Description
31.8.28 CIM_QueryCapabilities	Optional	Defines the Query execution capabilities of the profile or CIMOM.
31.8.29 CIM_QueryCondition (Dynamic or Client defined)	Optional	A Dynamic or Client defined Query that is used as a condition of a PolicyRule. A QueryCondition where Trigger=TRUE serves as an indication to drive evaluation of other QueryConditions in the PolicyRule.
31.8.30 CIM_QueryCondition (Pre-defined)	Optional	A predefined Query that is used as a condition of a PolicyRule. A QueryCondition where Trigger=TRUE serves as an indication to drive evaluation of other QueryConditions in the PolicyRule.
31.8.31 CIM_ReusablePolicy (Container to MethodAction)	Optional	ReusablePolicy associates Policy Conditions and Policy Actions to a ReusablePolicyContainer. It is used for Dynamic Policy support.
31.8.32 CIM_ReusablePolicy (Container to QueryCondition)	Optional	ReusablePolicy associates Policy Conditions and Policy Actions to a ReusablePolicyContainer. It is used for Dynamic Policy support.
31.8.33 CIM_ReusablePolicy (Container to System)	Optional	ReusablePolicy associates Policy Conditions and Policy Actions to a ReusablePolicyContainer. It is used for Dynamic Policy support.
31.8.34 CIM_ReusablePolicyContainer	Optional	A ReusablePolicyContainer collects all the Policy Conditions and Actions that may be used in constructing a Dynamic PolicyRule.
31.8.35 SNIA_PolicyCapabilities	Mandatory	Defines the Policy capabilities of the profile or CIMOM.

31.8.1 CIM_CompoundPolicyAction (Client defined)

CompoundPolicyAction is used to represent an expression consisting of an ordered sequence of action terms. Each action term is represented as a subclass of the PolicyAction class. Compound actions are constructed by associating dependent action terms together using the PolicyActionInPolicyAction aggregation.

CompoundPolicyAction is subclassed from PolicyAction.

An instance of CompoundPolicyAction will exist if any compound actions exist.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 302 describes class CIM_CompoundPolicyAction (Client defined).

Table 302 - SMI Referenced Properties/Methods for CIM_CompoundPolicyAction (Client defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another Client defined user-friendly name
CommonName		Optional	A client defined user-friendly name of the CompoundPolicyAction
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyAction is defined.
SystemName		Mandatory	The name of the System object in whose scope this PolicyAction is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyAction, the CreationClassName of the PolicyRule object with which this Action is associated. For a reusable PolicyAction, a special value, 'NO RULE', should be used.
CreationClassName		Mandatory	The name of the class or the subclass used in the creation of an instance.
PolicyRuleName		Mandatory	For a rule-specific PolicyAction, the name of the PolicyRule object with which this Action is associated. For a reusable PolicyAction, a special value, 'NO RULE', should be used.
PolicyActionName		Mandatory	A client defined user friendly name of this policy (method) action
DoActionLogging		Optional	

Table 302 - SMI Referenced Properties/Methods for CIM_CompoundPolicyAction (Client defined)

Properties	Flags	Requirement	Description & Notes
SequencedActions		Optional	This property gives a policy administrator (client) a way of specifying how the ordering of the PolicyActions associated with this PolicyRule is to be interpreted. Three values are supported: - mandatory(1): Do the actions in the indicated order, or don't do them at all. - recommended(2): Do the actions in the indicated order if you can, but if you can't do them in this order, do them in another order if you can. - dontCare(3): Do them -- I don't care about the order. The default value is 3 ("DontCare"). Values { "Mandatory", "Recommended", "Dont Care" }
ExecutionStrategy		Optional	ExecutionStrategy defines the strategy to be used in executing the sequenced actions aggregated by this CompoundPolicyAction. There are three execution strategies: Do Until Success - execute actions according to predefined order, until successful execution of a single action. Do All - execute ALL actions which are part of the modeled set, according to their predefined order. Continue doing this, even if one or more of the actions fails. Do Until Failure - execute actions according to predefined order, until the first failure in execution of an action instance. The default value is 2 ("Do All"). Values { "Do Until Success", "Do All", "Do Until Failure" }

31.8.2 CIM_CompoundPolicyAction (Pre-defined)

CompoundPolicyAction is used to represent an expression consisting of an ordered sequence of action terms. Each action term is represented as a subclass of the PolicyAction class. Compound actions are constructed by associating dependent action terms together using the PolicyActionInPolicyAction aggregation.

CompoundPolicyAction is subclassed from PolicyAction.

An instance of CompoundPolicyAction will exist if any compound actions exist.

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

Table 303 describes class CIM_CompoundPolicyAction (Pre-defined).

Table 303 - SMI Referenced Properties/Methods for CIM_CompoundPolicyAction (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another provider generated user-friendly name
CommonName		Optional	A provider generated user-friendly name of the CompoundPolicyAction
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyAction is defined.
SystemName		Mandatory	The name of the System object in whose scope this PolicyAction is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyAction, the CreationClassName of the PolicyRule object with which this Action is associated. For a reusable PolicyAction, a special value, 'NO RULE', should be used.
CreationClassName		Mandatory	The name of the class or the subclass used in the creation of an instance.
PolicyRuleName		Mandatory	For a rule-specific PolicyAction, the name of the PolicyRule object with which this Action is associated. For a reusable PolicyAction, a special value, 'NO RULE', should be used.
PolicyActionName		Mandatory	A provider generated user-friendly name of this policy (method) action
DoActionLogging		Optional	

Table 303 - SMI Referenced Properties/Methods for CIM_CompoundPolicyAction (Pre-defined)

Properties	Flags	Requirement	Description & Notes
SequencedActions		Optional	This property gives a profile designer a way of specifying how the ordering of the PolicyActions associated with this PolicyRule is to be interpreted. Three values are supported: - mandatory(1): Do the actions in the indicated order, or don't do them at all. - recommended(2): Do the actions in the indicated order if you can, but if you can't do them in this order, do them in another order if you can. - dontCare(3): Do them -- I don't care about the order. The default value is 3 ("DontCare"). Values { "Mandatory", "Recommended", "Dont Care" }
ExecutionStrategy		Optional	A profile designed ExecutionStrategy defines the strategy to be used in executing the sequenced actions aggregated by this CompoundPolicyAction. There are three execution strategies: Do Until Success - execute actions according to predefined order, until successful execution of a single action. Do All - execute ALL actions which are part of the modeled set, according to their predefined order. Continue doing this, even if one or more of the actions fails. Do Until Failure - execute actions according to predefined order, until the first failure in execution of an action instance. The default value is 2 ("Do All"). Values { "Do Until Success", "Do All", "Do Until Failure" }

31.8.3 CIM_CompoundPolicyCondition (Client defined)

CompoundPolicyCondition is used to represent compound conditions formed by aggregating simpler policy conditions. Compound conditions are constructed by associating subordinate condition terms together using the PolicyConditionInPolicyCondition aggregation.

CompoundPolicyCondition is subclassed from PolicyCondition.

An instance of CompoundPolicyCondition will exist if any client defined compound conditions exist.

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

Table 304 describes class CIM_CompoundPolicyCondition (Client defined).

Table 304 - SMI Referenced Properties/Methods for CIM_CompoundPolicyCondition (Client defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another client defined user friendly name
CommonName		Optional	A client defined user friendly name of the CompoundPolicyCondition.
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyCondition is defined.
SystemName		Mandatory	The name of the System object in whose scope this PolicyCondition is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyCondition, the CreationClassName of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
PolicyRuleName		Mandatory	For a rule-specific PolicyCondition, the name of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
CreationClassName		Mandatory	The name of the class or the subclass used in the creation of an instance.
PolicyConditionName		Mandatory	A client defined user friendly name of this PolicyCondition.
ConditionListType		Optional	Indicates whether the list of CompoundPolicyConditions associated with this PolicyRule is in disjunctive normal form (DNF) or conjunctive normal form (CNF). The default value is 1 ("DNF"). Values { "DNF", "CNF" }

31.8.4 CIM_CompoundPolicyCondition (Pre-defined)

CompoundPolicyCondition is used to represent compound conditions formed by aggregating simpler policy conditions. Compound conditions are constructed by associating subordinate condition terms together using the PolicyConditionInPolicyCondition aggregation.

CompoundPolicyCondition is subclassed from PolicyCondition.

An instance of CompoundPolicyCondition will exist if any predefined compound conditions exist.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 305 describes class CIM_CompoundPolicyCondition (Pre-defined).

Table 305 - SMI Referenced Properties/Methods for CIM_CompoundPolicyCondition (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another provider supplied user friendly name
CommonName		Optional	A provider supplied user friendly name of the CompoundPolicyCondition.
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyCondition is defined.
SystemName		Mandatory	The name of the System object in whose scope this PolicyCondition is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyCondition, the CreationClassName of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
PolicyRuleName		Mandatory	For a rule-specific PolicyCondition, the name of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
CreationClassName		Mandatory	The name of the class or the subclass used in the creation of an instance.
PolicyConditionName		Mandatory	A provider supplied user friendly name of this PolicyCondition.
ConditionListType		Optional	Indicates whether the list of CompoundPolicyConditions associated with this PolicyRule Is in disjunctive normal form (DNF) or conjunctive normal form (CNF). The default value is 1 ("DNF"). Values { "DNF", "CNF" }

31.8.5 CIM_ElementCapabilities (Policy Capabilities)

CIM_ElementCapabilities represents the association between ManagedElements (i.e.,CIM_RegisteredSubprofile) and their Capabilities (e.g., SNIA_PolicyCapabilities).

CIM_ElementCapabilities is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 306 describes class CIM_ElementCapabilities (Policy Capabilities).

Table 306 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Policy Capabilities)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

31.8.6 CIM_ElementCapabilities (Query Capabilities)

CIM_ElementCapabilities represents the association between ManagedElements (i.e.,CIM_RegisteredSubprofile) and their Capabilities (e.g., CIM_QueryCapabilities).

CIM_ElementCapabilities is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 307 describes class CIM_ElementCapabilities (Query Capabilities).

Table 307 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Query Capabilities)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

31.8.7 CIM_MethodAction (Client defined)

MethodAction is a PolicyAction that invokes an action defined by a query. The action is defined by a method of an ObjectName, which may be an intrinsic method of a CIM Namespace or an extrinsic method of a CIM_ManagedElement. The input parameters to the method are defined by the query and may be fixed values defined by literals or may be defined by reference to one or more properties of QueryConditionResult, MethodActionResult, or other instances.

MethodAction is subclassed from PolicyAction.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 308 describes class CIM_MethodAction (Client defined).

Table 308 - SMI Referenced Properties/Methods for CIM_MethodAction (Client defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another client defined user friendly name
CommonName		Optional	A client defined user friendly name of the MethodAction.

Table 308 - SMI Referenced Properties/Methods for CIM_MethodAction (Client defined)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyAction is defined.
SystemName		Mandatory	The name of the System object in whose scope this MethodAction is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific MethodAction, the CreationClassName of the PolicyRule object with which this Action is associated. For a reusable MethodAction, a special value, 'NO RULE', should be used.
PolicyRuleName		Mandatory	For a rule-specific MethodAction, the name of the PolicyRule object with which this Action is associated. For a reusable MethodAction, a special value, 'NO RULE', should be used.
CreationClassName		Mandatory	The name of the class or the subclass used in the creation of an instance.
PolicyActionName		Mandatory	A client defined user friendly name of this policy (method) action
DoActionLogging		Optional	
InstMethodCallName		Mandatory	In the context of the associated PolicyRule, InstMethodCallName defines a unique name for the query results that invoke the method specified in the Query string. It may be used in subsequent MethodActions of the same PolicyRule. This string is treated as a class name, in a query statement.
Query		Mandatory	The query that defines the method and the input parameters to that method.
QueryLanguage		Mandatory	This defines the query language being used, and for the current version of SMI-S, this shall be set to "2" (CQL).

31.8.8 CIM_MethodAction (Pre-defined)

MethodAction is a PolicyAction that invokes an action defined by a query. The action is defined by a method of an ObjectName, which may be an intrinsic method of a CIM Namespace or an extrinsic method of a CIM_ManagedElement. The input parameters to the method are defined by the query and may be fixed values defined by literals or may be defined by reference to one or more properties of QueryConditionResult, MethodActionResult, or other instances.

MethodAction is subclassed from PolicyAction.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 309 describes class CIM_MethodAction (Pre-defined).

Table 309 - SMI Referenced Properties/Methods for CIM_MethodAction (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another provider supplied user friendly name
CommonName		Optional	A provider supplied user friendly name of the MethodAction.
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyAction is defined.
SystemName		Mandatory	The name of the System object in whose scope this MethodAction is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific MethodAction, the CreationClassName of the PolicyRule object with which this Action is associated. For a reusable MethodAction, a special value, 'NO RULE', should be used.
PolicyRuleName		Mandatory	For a rule-specific MethodAction, the name of the PolicyRule object with which this Action is associated. For a reusable MethodAction, a special value, 'NO RULE', should be used.
CreationClassName		Mandatory	The name of the class or the subclass used in the creation of an instance.
PolicyActionName		Mandatory	A provider supplied user friendly name of this policy (method) action
DoActionLogging		Optional	
InstMethodCallName		Mandatory	In the context of the associated PolicyRule, InstMethodCallName defines a unique name for the query results that invoke the method specified in the Query string. It may be used in subsequent MethodActions of the same PolicyRule. This string is treated as a class name, in a query statement.
Query		Mandatory	The query that defines the method and the input parameters to that method.
QueryLanguage		Mandatory	This defines the query language being used, and for the current version of SMI-S, it shall be set to "2" (CQL).

31.8.9 CIM_PolicyActionInPolicyAction (Client defined)

PolicyActionInPolicyAction is used to represent the compounding of policy actions into a higher-level policy action.

PolicyActionInPolicyAction is subclassed from PolicyActionStructure.

This association will exist if there is a Client defined CompoundPolicyAction instance.

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

Requirement: Optional

Table 310 describes class CIM_PolicyActionInPolicyAction (Client defined).

Table 310 - SMI Referenced Properties/Methods for CIM_PolicyActionInPolicyAction (Client defined)

Properties	Flags	Requirement	Description & Notes
ActionOrder		Optional	ActionOrder is an unsigned integer 'n' that indicates the relative position of a PolicyAction in the sequence of actions associated with a PolicyRule or CompoundPolicyAction.
GroupComponent		Mandatory	This property represents the CompoundPolicyAction that contains one or more PolicyActions.
PartComponent		Mandatory	This property holds the name of a PolicyAction contained by one or more CompoundPolicyActions.

31.8.10 CIM_PolicyActionInPolicyAction (Pre-defined)

PolicyActionInPolicyAction is used to represent the compounding of policy actions into a higher-level policy action.

PolicyActionInPolicyAction is subclassed from PolicyActionStructure.

This association will exist if there is a predefined CompoundPolicyAction instance.

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

Table 311 describes class CIM_PolicyActionInPolicyAction (Pre-defined).

Table 311 - SMI Referenced Properties/Methods for CIM_PolicyActionInPolicyAction (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ActionOrder		Optional	ActionOrder is an unsigned integer 'n' that indicates the relative position of a PolicyAction in the sequence of actions associated with a PolicyRule or CompoundPolicyAction.
PartComponent		Mandatory	
GroupComponent		Mandatory	

31.8.11 CIM_PolicyActionInPolicyRule (Client defined)

A PolicyRule aggregates zero or more instances of the PolicyAction class, via the PolicyActionInPolicyRule association. A Rule that aggregates zero Actions is not valid--it may, however, be in the process of being entered into a PolicyRepository or being defined for a System. Alternately, the actions of the policy may be explicit in the definition of the PolicyRule. Note that a PolicyRule should have no effect until it is valid.

The Actions associated with a PolicyRule may be given a required order, a recommended order, or no order at all. For Actions represented as separate objects, the PolicyActionInPolicyRule aggregation can be used to express an order.

This aggregation does not indicate whether a specified action order is required, recommended, or of no significance; the property SequencedActions in the aggregating instance of PolicyRule provides this indication.

PolicyActionInPolicyRule is subclassed from PolicyActionStructure.

This association will exist if there are any Client defined PolicyRules that have MethodActions.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 312 describes class CIM_PolicyActionInPolicyRule (Client defined).

Table 312 - SMI Referenced Properties/Methods for CIM_PolicyActionInPolicyRule (Client defined)

Properties	Flags	Requirement	Description & Notes
ActionOrder		Optional	ActionOrder is an unsigned integer 'n' that indicates the relative position of a PolicyAction in the sequence of actions associated with a PolicyRule.
GroupComponent		Mandatory	This property represents the PolicyRule that contains one or more PolicyActions.
PartComponent		Mandatory	This property holds the name of a PolicyAction contained by one or more PolicyRules.

31.8.12 CIM_PolicyActionInPolicyRule (Pre-defined)

A PolicyRule aggregates zero or more instances of the PolicyAction class, via the PolicyActionInPolicyRule association. A Rule that aggregates zero Actions is not valid--it may, however, be in the process of being entered into a PolicyRepository or being defined for a System. Alternately, the actions of the policy may be explicit in the definition of the PolicyRule. Note that a PolicyRule should have no effect until it is valid.

The Actions associated with a PolicyRule may be given a required order, a recommended order, or no order at all. For Actions represented as separate objects, the PolicyActionInPolicyRule aggregation can be used to express an order.

This aggregation does not indicate whether a specified action order is required, recommended, or of no significance; the property SequencedActions in the aggregating instance of PolicyRule provides this indication.

PolicyActionInPolicyRule is subclassed from PolicyActionStructure.

This association will exist if there are any Static PolicyRules that have MethodActions.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 313 describes class CIM_PolicyActionInPolicyRule (Pre-defined).

Table 313 - SMI Referenced Properties/Methods for CIM_PolicyActionInPolicyRule (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ActionOrder		Optional	ActionOrder is an unsigned integer 'n' that indicates the relative position of a PolicyAction in the sequence of actions associated with a PolicyRule.
GroupComponent		Mandatory	
PartComponent		Mandatory	

31.8.13 CIM_PolicyConditionInPolicyCondition (Client defined)

A CompoundPolicyCondition aggregates zero or more instances of the PolicyCondition class, via the PolicyConditionInPolicyCondition association. A CompoundPolicyCondition that aggregates zero Conditions is not valid; it may, however, be in the process of being defined. Note that a CompoundPolicyCondition should have no effect until it is valid.

CIM_PolicyConditionInPolicyCondition is subclassed from CIM_PolicyConditionStructure.

There would be at least on instance of this association if there are any Client defined CompoundPolicyConditions.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 314 describes class CIM_PolicyConditionInPolicyCondition (Client defined).

Table 314 - SMI Referenced Properties/Methods for CIM_PolicyConditionInPolicyCondition (Client defined)

Properties	Flags	Requirement	Description & Notes
GroupNumber		Mandatory	Unsigned integer indicating the group to which the contained PolicyCondition belongs. This integer segments the Conditions into the ANDed sets (when the ConditionListType is "DNF") or, similarly, into the ORed sets (when the ConditionListType is "CNF").
ConditionNegated		Mandatory	Indication of whether the contained PolicyCondition is negated. TRUE indicates that the PolicyCondition IS negated, FALSE indicates that it IS not negated.
GroupComponent		Mandatory	This property represents the CompoundPolicyCondition that contains one or more PolicyConditions.
PartComponent		Mandatory	This property holds the name of a PolicyCondition contained by one or more CompoundPolicyConditions.

31.8.14 CIM_PolicyConditionInPolicyCondition (Pre-defined)

A CompoundPolicyCondition aggregates zero or more instances of the PolicyCondition class, via the PolicyConditionInPolicyCondition association. A CompoundPolicyCondition that aggregates zero Conditions is not

valid; it may, however, be in the process of being defined. Note that a CompoundPolicyCondition should have no effect until it is valid.

CIM_PolicyConditionInPolicyCondition is subclassed from CIM_PolicyConditionStructure.

There would be at least on instance of this association if there are any predefined CompoundPolicyConditions.

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

Table 315 describes class CIM_PolicyConditionInPolicyCondition (Pre-defined).

Table 315 - SMI Referenced Properties/Methods for CIM_PolicyConditionInPolicyCondition (Pre-defined)

Properties	Flags	Requirement	Description & Notes
GroupNumber		Mandatory	Unsigned integer indicating the group to which the contained PolicyCondition belongs. This integer segments the Conditions into the ANDed sets (when the ConditionListType is "DNF") or, similarly, into the ORed sets (when the ConditionListType is "CNF").
ConditionNegated		Mandatory	Indication of whether the contained PolicyCondition is negated. TRUE indicates that the PolicyCondition IS negated, FALSE indicates that it IS not negated.
PartComponent		Mandatory	
GroupComponent		Mandatory	

31.8.15 CIM_PolicyConditionInPolicyRule (Client defined)

A PolicyRule aggregates zero or more instances of the PolicyCondition class, via the PolicyConditionInPolicyRule association. A Rule that aggregates zero Conditions is not valid; it may, however, be in the process of being defined. Note that a PolicyRule should have no effect until it is valid.

CIM_PolicyConditionInPolicyRule is subclassed from CIM_PolicyConditionStructure.

There would be one instance of this association for each client defined PolicyCondition in a PolicyRule.

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

Table 316 describes class CIM_PolicyConditionInPolicyRule (Client defined).

Table 316 - SMI Referenced Properties/Methods for CIM_PolicyConditionInPolicyRule (Client defined)

Properties	Flags	Requirement	Description & Notes
GroupNumber		Mandatory	Unsigned integer indicating the group to which the contained PolicyCondition belongs. This integer segments the Conditions into the ANDed sets (when the ConditionListType is "DNF") or, similarly, into the ORed sets (when the ConditionListType is "CNF").
ConditionNegated		Mandatory	Indication of whether the contained PolicyCondition is negated. TRUE indicates that the PolicyCondition IS negated, FALSE indicates that it IS not negated.
GroupComponent		Mandatory	This property represents the PolicyRule that contains one or more PolicyConditions.
PartComponent		Mandatory	This property holds the name of a PolicyCondition contained by one or more PolicyRules.
PartComponent		Mandatory	
GroupComponent		Mandatory	

31.8.16 CIM_PolicyConditionInPolicyRule (Pre-defined)

A PolicyRule aggregates zero or more instances of the PolicyCondition class, via the PolicyConditionInPolicyRule association. A Rule that aggregates zero Conditions is not valid; it may, however, be in the process of being defined. Note that a PolicyRule should have no effect until it is valid.

CIM_PolicyConditionInPolicyRule is subclassed from CIM_PolicyConditionStructure.

There would be one instance of this association for each predefined PolicyCondition in a PolicyRule.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 317 describes class CIM_PolicyConditionInPolicyRule (Pre-defined).

Table 317 - SMI Referenced Properties/Methods for CIM_PolicyConditionInPolicyRule (Pre-defined)

Properties	Flags	Requirement	Description & Notes
GroupNumber		Mandatory	Unsigned integer indicating the group to which the contained PolicyCondition belongs. This integer segments the Conditions into the ANDed sets (when the ConditionListType is "DNF") or, similarly, into the ORed sets (when the ConditionListType is "CNF").
GroupComponent		Mandatory	
PartComponent		Mandatory	

31.8.17 CIM_PolicyContainerInPolicyContainer

A relationship that aggregates one or more lower-level ReusablePolicyContainer instances into a higher-level ReusablePolicyContainer.

CIM_PolicyContainerInPolicyContainer is subclassed form CIM_SystemComponent.

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

Table 318 describes class CIM_PolicyContainerInPolicyContainer.

Table 318 - SMI Referenced Properties/Methods for CIM_PolicyContainerInPolicyContainer

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

31.8.18 CIM_PolicyRule (Dynamic or Client defined)

Same rules as defined for predefined PolicyRules apply to Client Defined PolicyRules.

CIM_PolicyRule is subclassed from CIM_PolicySet.

There shall be at least one instance of PolicyRule for a policy based profile (a profile with an implementation of the Policy Package).

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

Table 319 describes class CIM_PolicyRule (Dynamic or Client defined).

Table 319 - SMI Referenced Properties/Methods for CIM_PolicyRule (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another client defined user friendly name
CommonName		Optional	A client defined user friendly name of policy rule.
PolicyDecisionStrategy		Mandatory	PolicyDecisionStrategy defines the evaluation method used for policies contained in the PolicySet. FirstMatching enforces the actions of the first rule that evaluates to TRUE. It is the only value currently defined. Values { "First Matching" }

Table 319 - SMI Referenced Properties/Methods for CIM_PolicyRule (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
Enabled		Mandatory	Indicates whether this PolicySet is administratively enabled or administratively disabled. SMI-S does not define a usage for 'Enabled for Debug', but it may be supported by an implementation. ValueMap { "1", "2", "3" }, Values { "Enabled", "Disabled", "Enabled For Debug" }
SystemCreationClassName		Mandatory	The scoping System's CreationClassName.
SystemName		Mandatory	The scoping System's Name.
CreationClassName		Mandatory	CreationClassName indicates the name of the class or the subclass used in the creation of an instance.
PolicyRuleName		Mandatory	A user-friendly name of this PolicyRule.
ConditionListType		Optional	Indicates whether the list of PolicyConditions associated with this PolicyRule is in disjunctive normal form (DNF), conjunctive normal form (CNF), or has no conditions (i.e., is an UnconditionalRule) and is automatically evaluated to "True." The default value is 1 ("DNF"). Values { "Unconditional Rule", "DNF", "CNF" }
RuleUsage		Optional	A free-form string that can be used to provide guidelines on how this PolicyRule should be used.

Table 319 - SMI Referenced Properties/Methods for CIM_PolicyRule (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
SequencedActions		Optional	This property gives a policy administrator a way of specifying how the ordering of the PolicyActions associated with this PolicyRule is to be interpreted. Three values are supported: - mandatory(1): Do the actions in the indicated order, or don't do them at all. - recommended(2): Do the actions in the indicated order if you can, but if you can't do them in this order, do them in another order if you can. - dontCare(3): Do them -- I don't care about the order. The default value is 3 ("DontCare"). Values { "Mandatory", "Recommended", "Dont Care" }
ExecutionStrategy		Mandatory	ExecutionStrategy defines the strategy to be used in executing the sequenced actions aggregated by this PolicyRule. There are three execution strategies: Do Until Success - execute actions according to predefined order, until successful execution of a single action. Do All - execute ALL actions which are part of the modeled set, according to their predefined order. Continue doing this, even if one or more of the actions fails. Do Until Failure - execute actions according to predefined order, until the first failure in execution of an action instance. Values { "Do Until Success", "Do All", "Do Until Failure" }

31.8.19 CIM_PolicyRule (Pre-defined)

The central class used for representing the 'If Condition then Action' semantics of a policy rule. A PolicyRule condition, in the most general sense, is represented as either an ORed set of ANDed conditions (Disjunctive Normal Form, or DNF) or an ANDed set of ORed conditions (Conjunctive Normal Form, or CNF). Individual conditions may either be negated (not C) or unnegated (C). The actions specified by a PolicyRule are to be performed if and only if the PolicyRule condition (whether it is represented in DNF or CNF) evaluates to TRUE.

The conditions and actions associated with a PolicyRule are modeled, respectively, with subclasses of PolicyCondition and PolicyAction. These condition and action objects are tied to instances of PolicyRule by the PolicyConditionInPolicyRule and PolicyActionInPolicyRule aggregations.

A PolicyRule may also be associated with one or more policy time periods, indicating the schedule according to which the policy rule is active and inactive. In this case it is the PolicySetValidityPeriod aggregation that provides this linkage.

The PolicyRule class uses the property ConditionListType, to indicate whether the conditions for the rule are in DNF (disjunctive normal form), CNF (conjunctive normal form) or, in the case of a rule with no conditions, as an UnconditionalRule. The PolicyConditionInPolicyRule aggregation contains two additional properties to complete the representation of the Rule's conditional expression. The first of these properties is an integer to partition the referenced PolicyConditions into one or more groups, and the second is a Boolean to indicate whether a

referenced Condition is negated. An example shows how ConditionListType and these two additional properties provide a unique representation of a set of PolicyConditions in either DNF or CNF.

Suppose we have a PolicyRule that aggregates five PolicyConditions C1 through C5, with the following values in the properties of the five PolicyConditionInPolicyRule associations:

C1: GroupNumber = 1, ConditionNegated = FALSE

C2: GroupNumber = 1, ConditionNegated = TRUE

C3: GroupNumber = 1, ConditionNegated = FALSE

C4: GroupNumber = 2, ConditionNegated = FALSE

C5: GroupNumber = 2, ConditionNegated = FALSE

If ConditionListType = DNF, then the overall condition for the PolicyRule is:

(C1 AND (not C2) AND C3) OR (C4 AND C5)

On the other hand, if ConditionListType = CNF, then the overall condition for the PolicyRule is:

(C1 OR (not C2) OR C3) AND (C4 OR C5)

In both cases, there is an unambiguous specification of the overall condition that is tested to determine whether to perform the PolicyActions associated with the PolicyRule.

PolicyRule instances may also be used to aggregate other PolicyRules and/or PolicyGroups. When used in this way to implement nested rules, the conditions of the aggregating rule apply to the subordinate rules as well. However, any side effects of condition evaluation or the execution of actions shall not affect the result of the evaluation of other conditions evaluated by the rule engine in the same evaluation pass. That is, an implementation of a rule engine may evaluate all conditions in any order before applying the priority and determining which actions are to be executed.

CIM_PolicyRule is subclassed from CIM_PolicySet.

There shall be at least one instance of PolicyRule for a policy based profile (a profile with an implementation of the Policy Package).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 320 describes class CIM_PolicyRule (Pre-defined).

Table 320 - SMI Referenced Properties/Methods for CIM_PolicyRule (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another provider supplied user friendly name
CommonName		Optional	A provider supplied user friendly name of the policy rule
PolicyDecisionStrategy		Mandatory	PolicyDecisionStrategy defines the evaluation method used for policies contained in the PolicySet. FirstMatching enforces the actions of the first rule that evaluates to TRUE. It is the only value currently defined. Values { "First Matching" }

Table 320 - SMI Referenced Properties/Methods for CIM_PolicyRule (Pre-defined)

Properties	Flags	Requirement	Description & Notes
Enabled		Mandatory	Indicates whether this PolicySet is administratively enabled or administratively disabled. SMI-S does not define a usage for 'Enabled for Debug', but it may be supported by an implementation. ValueMap { "1", "2", "3" }, Values { "Enabled", "Disabled", "Enabled For Debug" }
SystemCreationClassName		Mandatory	The scoping System's CreationClassName.
SystemName		Mandatory	The scoping System's Name.
CreationClassName		Mandatory	CreationClassName indicates the name of the class or the subclass used in the creation of an instance.
PolicyRuleName		Mandatory	A user-friendly name of this PolicyRule.
ConditionListType		Optional	Indicates whether the list of PolicyConditions associated with this PolicyRule is in disjunctive normal form (DNF), conjunctive normal form (CNF), or has no conditions (i.e., is an UnconditionalRule) and is automatically evaluated to "True." The default value is 1 ("DNF"). Values { "Unconditional Rule", "DNF", "CNF" }
RuleUsage		Optional	A free-form string that can be used to provide guidelines on how this PolicyRule should be used.

Table 320 - SMI Referenced Properties/Methods for CIM_PolicyRule (Pre-defined)

Properties	Flags	Requirement	Description & Notes
SequencedActions		Optional	This property gives a policy administrator a way of specifying how the ordering of the PolicyActions associated with this PolicyRule is to be interpreted. Three values are supported: - mandatory(1): Do the actions in the indicated order, or don't do them at all. - recommended(2): Do the actions in the indicated order if you can, but if you can't do them in this order, do them in another order if you can. - dontCare(3): Do them -- I don't care about the order. The default value is 3 ("DontCare"). Values { "Mandatory", "Recommended", "Dont Care" }
ExecutionStrategy		Mandatory	ExecutionStrategy defines the strategy to be used in executing the sequenced actions aggregated by this PolicyRule. There are three execution strategies: Do Until Success - execute actions according to predefined order, until successful execution of a single action. Do All - execute ALL actions which are part of the modeled set, according to their predefined order. Continue doing this, even if one or more of the actions fails. Do Until Failure - execute actions according to predefined order, until the first failure in execution of an action instance. Values { "Do Until Success", "Do All", "Do Until Failure" }

31.8.20 CIM_PolicyRuleInSystem (Dynamic or Client defined)

An association that links a PolicyRule to the System in whose scope the Rule is defined. It represents a relationship between a System and a PolicyRule used in the administrative scope of that system (e.g., AdminDomain, ComputerSystem). The Priority property is used to assign a relative priority to a PolicyRule within the administrative scope in contexts where it is not a component of another PolicySet.

CIM_PolicyRuleInSystem is subclassed from CIM_PolicySetInSystem.

There shall be at least one instance of this association for each Dynamic or Client Defined Policy Rule.

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

Table 321 describes class CIM_PolicyRuleInSystem (Dynamic or Client defined).

Table 321 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
Priority		Optional	The Priority property is used to specify the relative priority of the referenced PolicySet (PolicyRule) when there are more than one PolicySet instances applied to a managed resource that are not PolicySetComponents and, therefore, have no other relative priority defined. The priority is a non-negative integer; a larger value indicates a higher priority.
Antecedent		Mandatory	The System in whose scope a PolicyRule is defined.
Dependent		Mandatory	A PolicyRule named within the scope of a System.

31.8.21 CIM_PolicyRuleInSystem (Pre-defined)

An association that links a PolicyRule to the System in whose scope the Rule is defined. It represents a relationship between a System and a PolicyRule used in the administrative scope of that system (e.g., AdminDomain, ComputerSystem). The Priority property is used to assign a relative priority to a PolicyRule within the administrative scope in contexts where it is not a component of another PolicySet.

CIM_PolicyRuleInSystem is subclassed from CIM_PolicySetInSystem.

There shall be at least one instance of this association for each Static Policy rule.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 322 describes class CIM_PolicyRuleInSystem (Pre-defined).

Table 322 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem (Pre-defined)

Properties	Flags	Requirement	Description & Notes
Priority		Optional	The Priority property is used to specify the relative priority of the referenced PolicySet (PolicyRule) when there are more than one PolicySet instances applied to a managed resource that are not PolicySetComponents and, therefore, have no other relative priority defined. The priority is a non-negative integer; a larger value indicates a higher priority.
Antecedent		Mandatory	
Dependent		Mandatory	

31.8.22 CIM_PolicySetAppliesToElement (Dynamic or Client defined)

PolicySetAppliesToElement makes explicit which PolicySets (i.e., policy rules and groups of rules) ARE CURRENTLY applied to a particular Element. This association indicates that the PolicySets that are appropriate for a ManagedElement(specified using the PolicyRoleCollection aggregation) have actually been deployed in the policy management infrastructure. One or more QueryCondition or MethodAction instances may reference the

PolicySetAppliesToElement association as part of its query. PolicySetAppliesToElement shall not be used if the associated PolicySet, (collectively though its rules, conditions, and actions), does not make use of the association. Note that if the named Element refers to a Collection, then the PolicySet is assumed to be applied to all the members of the Collection.

CIM_PolicySetAppliesToElement is not subclassed from anything.

An instance of this class may or may not exist.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 323 describes class CIM_PolicySetAppliesToElement (Dynamic or Client defined).

Table 323 - SMI Referenced Properties/Methods for CIM_PolicySetAppliesToElement (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	The PolicyRules and/or groups of rules that are currently applied to an Element.
ManagedElement		Mandatory	The ManagedElement to which the PolicySet applies.

31.8.23 CIM_PolicySetAppliesToElement (Pre-defined)

PolicySetAppliesToElement makes explicit which PolicySets (i.e., policy rules and groups of rules) ARE CURRENTLY applied to a particular Element. This association indicates that the PolicySets that are appropriate for a ManagedElement(specified using the PolicyRoleCollection aggregation) have actually been deployed in the policy management infrastructure. One or more QueryCondition or MethodAction instances may reference the PolicySetAppliesToElement association as part of its query. PolicySetAppliesToElement shall not be used if the associated PolicySet, (collectively though its rules, conditions, and actions), does not make use of the association. Note that if the named Element refers to a Collection, then the PolicySet is assumed to be applied to all the members of the Collection.

CIM_PolicySetAppliesToElement is not subclassed from anything.

An instance of this class may or may not exist.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 324 describes class CIM_PolicySetAppliesToElement (Pre-defined).

Table 324 - SMI Referenced Properties/Methods for CIM_PolicySetAppliesToElement (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
PolicySet		Mandatory	

31.8.24 CIM_PolicySetValidityPeriod (Dynamic or Client defined)

The rules for client defined PolicySetValidityPeriods are the same as those for predefined PolicySetValidityPeriods.

CIM_PolicySetValidityPeriod is subclassed from CIM_PolicyComponent.

An instance of this class may or may not exist.

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

Table 325 describes class CIM_PolicySetValidityPeriod (Dynamic or Client defined).

Table 325 - SMI Referenced Properties/Methods for CIM_PolicySetValidityPeriod (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	This property contains the name of a PolicySet that contains one or more PolicyTimePeriodConditions.
PartComponent		Mandatory	This property contains the name of a PolicyTimePeriodCondition defining the valid time periods for one or more PolicySets.

31.8.25 CIM_PolicySetValidityPeriod (Pre-defined)

The PolicySetValidityPeriod aggregation represents scheduled activation and deactivation of a PolicySet. A PolicySet is considered "active" if it is both "Enabled" and in a valid time period.

If a PolicySet is associated with multiple policy time periods via this association, then the Set is in a valid time period if at least one of the time periods evaluates to TRUE. If a PolicySet is contained in another PolicySet via the PolicySetComponent aggregation (e.g., a PolicyRule in a PolicyGroup), then the contained PolicySet (e.g., PolicyRule) is in a valid period if at least one of the aggregate's PolicyTimePeriodCondition instances evaluates to TRUE and at least one of its own PolicyTimePeriodCondition instances also evaluates to TRUE. (In other words, the PolicyTimePeriodConditions are ORed to determine whether the PolicySet is in a valid time period and then ANDed with the ORed PolicyTimePeriodConditions of each of PolicySet instances in the PolicySetComponent hierarchy to determine if the PolicySet is in a valid time period and, if also "Enabled", therefore, active, i.e., the hierarchy ANDs the ORed PolicyTimePeriodConditions of the elements of the hierarchy.

A Time Period may be aggregated by multiple PolicySets. A Set that does not point to a PolicyTimePeriodCondition via this association, from the point of view of scheduling, is always in a valid time period.

CIM_PolicySetValidityPeriod is subclassed from CIM_PolicyComponent.

An instance of this class may or may not exist.

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

Table 326 describes class CIM_PolicySetValidityPeriod (Pre-defined).

Table 326 - SMI Referenced Properties/Methods for CIM_PolicySetValidityPeriod (Pre-defined)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

31.8.26 CIM_PolicyTimePeriodCondition (Dynamic or Client defined)

The rules for client defined PolicyTimePeriodCondition are the same as those described for predefined PolicyTimePeriodCondition.

CIM_PolicyTimePeriodCondition is subclassed from CIM_PolicyCondition.

An instance of this class may or may not exist. If they exist, they can be found by following PolicyConditionInRule associations from PolicyRule instances or ReusablePolicy associations from ReusablePolicyContainer instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 327 describes class CIM_PolicyTimePeriodCondition (Dynamic or Client defined).

Table 327 - SMI Referenced Properties/Methods for CIM_PolicyTimePeriodCondition (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another client defined user friendly name.
CommonName		Optional	A client defined user friendly name of policy object
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyCondition is defined.
SystemName		Mandatory	The name of the System object in whose scope this PolicyCondition is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyCondition, the CreationClassName of the PolicyRule object with which this Condition is associated. For a reusable Policy Condition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
PolicyRuleName		Mandatory	For a rule-specific PolicyCondition, the name of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
CreationClassName		Mandatory	CreationClassName indicates the name of the class or the subclass used in the creation of an instance.

Table 327 - SMI Referenced Properties/Methods for CIM_PolicyTimePeriodCondition (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
PolicyConditionName		Mandatory	A user-friendly name of this PolicyCondition.
TimePeriod		Optional	
MonthOfYearMask		Optional	
DayOfMonthMask		Optional	
DayOfWeekMask		Optional	
TimeOfDayMask		Optional	
LocalOrUtcTime		Optional	

31.8.27 CIM_PolicyTimePeriodCondition (Pre-defined)

This class provides a means of representing the time periods during which a PolicySet is valid, i.e., active. At all times that fall outside these time periods, the PolicySet has no effect. A PolicySet is treated as valid at ALL times, if it does not specify a PolicyTimePeriodCondition.

In some cases a Policy Consumer may need to perform certain setup / cleanup actions when a PolicySet becomes active / inactive. For example, sessions that were established while a PolicySet was active might need to be taken down when the PolicySet becomes inactive. In other cases, however, such sessions might be left up. In this case, the effect of deactivating the PolicySet would just be to prevent the establishment of new sessions.

Setup / cleanup behaviors on validity period transitions are not currently addressed by the Policy Model, and must be specified in 'guideline' documents or via subclasses of CIM_PolicySet, CIM_PolicyTimePeriod Condition or other concrete subclasses of CIM_Policy. If such behaviors need to be under the control of the policy administrator, then a mechanism to allow this control shall also be specified in the subclasses.

PolicyTimePeriodCondition is defined as a subclass of PolicyCondition. This is to allow the inclusion of time based criteria in the AND/OR condition definitions for a PolicyRule.

Instances of this class may have up to five properties identifying time periods at different levels. The values of all the properties present in an instance are ANDed together to determine the validity period(s) for the instance. For example, an instance with an overall validity range of January 1, 2000 through December 31, 2000; a month mask that selects March and April; a day-of-the-week mask that selects Fridays; and a time of day range of 0800 through 1600 would be represented using the following time periods:

Friday, March 5, 2000, from 0800 through 1600;

Friday, March 12, 2000, from 0800 through 1600;

Friday, March 19, 2000, from 0800 through 1600;

Friday, March 26, 2000, from 0800 through 1600;

Friday, April 2, 2000, from 0800 through 1600;

Friday, April 9, 2000, from 0800 through 1600;

Friday, April 16, 2000, from 0800 through 1600;

Friday, April 23, 2000, from 0800 through 1600;

Friday, April 30, 2000, from 0800 through 1600.

Properties not present in an instance of PolicyTimePeriodCondition are implicitly treated as having their value 'always enabled'. Thus, in the example above, the day-of-the-month mask is not present, and so the validity period for the instance implicitly includes a day-of-the-month mask that selects all days of the month. If this 'missing property' rule is applied to its fullest, we see that there is a second way to indicate that a PolicySet is always enabled: associate with it an instance of PolicyTimePeriodCondition whose only properties with specific values are its key properties.

CIM_PolicyTimePeriodCondition is subclassed from CIM_PolicyCondition.

An instance of this class may or may not exist. If they exist, they can be found by following PolicyConditionInRule associations from PolicyRule instances or ReusablePolicy associations from ReusablePolicyContainer instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 328 describes class CIM_PolicyTimePeriodCondition (Pre-defined).

Table 328 - SMI Referenced Properties/Methods for CIM_PolicyTimePeriodCondition (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another provider supplied user friendly name.
CommonName		Optional	A provider supplied user friendly name of policy object.
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyCondition is defined.
SystemName		Mandatory	The name of the System object in whose scope this PolicyCondition is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyCondition, the CreationClassName of the PolicyRule object with which this Condition is associated. For a reusable Policy Condition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
PolicyRuleName		Mandatory	For a rule-specific PolicyCondition, the name of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
CreationClassName		Mandatory	CreationClassName indicates the name of the class or the subclass used in the creation of an instance.
PolicyConditionName		Mandatory	A user-friendly name of this PolicyCondition.

Table 328 - SMI Referenced Properties/Methods for CIM_PolicyTimePeriodCondition (Pre-defined)

Properties	Flags	Requirement	Description & Notes
TimePeriod		Optional	This property identifies an overall range of calendar dates and times over which a PolicySet is valid. It is formatted as a string representing a start date and time, in which the character 'T' indicates the beginning of the time portion, followed by the solidus character '/', followed by a similar string representing an end date and time. The first date indicates the beginning of the range, while the second date indicates the end. Thus, the second date and time shall be later than the first. Date/times are expressed as substrings of the form yyyymmddThhmmss.
MonthOfYearMask		Optional	The purpose of this property is to refine the valid time period that is defined by the TimePeriod property, by explicitly specifying in which months the PolicySet is valid. These properties work together, with the TimePeriod used to specify the overall time period in which the PolicySet is valid, and the MonthOfYearMask used to pick out the months during which the PolicySet is valid.
DayOfMonthMask		Optional	The purpose of this property is to refine the valid time period that is defined by the TimePeriod property, by explicitly specifying in which days of the month the PolicySet is valid. These properties work together, with the TimePeriod used to specify the overall time period in which the PolicySet is valid, and the DayOfMonthMask used to pick out the days of the month during which the PolicySet is valid.
DayOfWeekMask		Optional	The purpose of this property is to refine the valid time period that is defined by the TimePeriod property, by explicitly specifying in which days of the week the PolicySet is valid. These properties work together, with the TimePeriod used to specify the overall time period in which the PolicySet is valid, and the DayOfWeekMask used to pick out the days of the week during which the PolicySet is valid.

Table 328 - SMI Referenced Properties/Methods for CIM_PolicyTimePeriodCondition (Pre-defined)

Properties	Flags	Requirement	Description & Notes
TimeOfDayMask		Optional	The purpose of this property is to refine the valid time period that is defined by the TimePeriod property, by explicitly specifying a range of times in a day during which the PolicySet is valid. These properties work together, with the TimePeriod used to specify the overall time period in which the PolicySet is valid, and the TimeOfDayMask used to pick out the range of time periods in a given day of during which the PolicySet is valid.
LocalOrUtcTime		Optional	This property indicates whether the times represented in the TimePeriod property and in the various Mask properties represent local times or UTC times. There is no provision for mixing of local times and UTC times: the value of this property applies to all of the other time-related properties. TimePeriods are synchronized worldwide by using the enumeration value 'UTCTime'.

31.8.28 CIM_QueryCapabilities

This class defines the capabilities of the Specific Policy Subprofile associated via ElementCapabilities.

CIM_QueryCapabilities is subclassed from CIM_Capabilities.

An instance of this class may or may not exist. An instance of CIM_QueryCapabilities shall exist for each Specific Policy Subprofile that supports client defined queries.

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

Table 329 describes class CIM_QueryCapabilities.

Table 329 - SMI Referenced Properties/Methods for CIM_QueryCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	This is a user friendly name of the capabilities instance.
CQLFeatures		Mandatory	Enumeration of CQL features supported by an Object Manager or Provider associated via ElementCapabilities. (See DSP0202 CIM Query Language Specification for a normative definition of each feature.) Values {"Basic Query", "Simple Join", "Complex Join", "Time", "Basic Like", "Full Like", "Array Elements", "Embedded Objects", "Order By", "Aggregations", "Subquery", "Satisfies Array", "Distinct", "First", "Path Functions"}

31.8.29 CIM_QueryCondition (Dynamic or Client defined)

QueryCondition defines the criteria for generating a set of QueryConditionResult instances that result from the contained query. If there are no instances returned from the query, then the result is false; otherwise, true.

CIM_QueryCondition is subclassed from CIM_PolicyCondition.

QueryCondition instances may or may not exist. If they exist, they can be found by following PolicyConditionInRule associations from PolicyRule instances or ReusablePolicy associations from ReusablePolicyContainer instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 330 describes class CIM_QueryCondition (Dynamic or Client defined).

Table 330 - SMI Referenced Properties/Methods for CIM_QueryCondition (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another user-friendly name.
CommonName		Optional	User-friendly name of the QueryCondition.
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyCondition is defined.
SystemName		Mandatory	The name of the System object in whose scope this PolicyCondition is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyCondition, the CreationClassName of the PolicyRule object with which this Condition is associated. For a reusable Policy Condition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
PolicyRuleName		Mandatory	For a rule-specific PolicyCondition, the name of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
CreationClassName		Mandatory	CreationClassName indicates the name of the class or the subclass used in the creation of an instance.
PolicyConditionName		Mandatory	A user-friendly name of this PolicyCondition.
QueryResultName		Mandatory	In the context of the associated PolicyRule, QueryResultName defines a unique alias for the query results that may be used in subsequent QueryConditions or MethodActions of the same PolicyRule. This string is treated as a class name, in a query statement.

Table 330 - SMI Referenced Properties/Methods for CIM_QueryCondition (Dynamic or Client defined)

Properties	Flags	Requirement	Description & Notes
Query		Mandatory	A query expression that defines the condition(s) under which QueryConditionResult instances will be generated. The FROM clause may reference any class, including QueryConditionResult. NOTE that the property name, "QueryConditionPath", shall not be used as the name of a select-list entry in the select-criteria clause of the query.
QueryLanguage		Mandatory	The language in which the query is expressed. SMI-S only recognizes "CQL" Other query languages may be encoded for vendor specific support, but only CQL is supported for SMI-S interoperability. Values {"CQL", "DMTF Reserved", "Vendor Reserved"}
Trigger		Mandatory	If Trigger = true, and with the exception of any PolicyTimePeriodConditions, PolicyConditions of this PolicyRule are not evaluated until this "triggering" condition query is true. There shall be no more than one QueryCondition with Trigger = true associated with a particular PolicyRule.

31.8.30 CIM_QueryCondition (Pre-defined)

QueryCondition defines the criteria for generating a set of QueryConditionResult instances that result from the contained query. If there are no instances returned from the query, then the result is false; otherwise, true.

CIM_QueryCondition is subclassed from CIM_PolicyCondition.

QueryCondition instances may or may not exist. If they exist, they can be found by following PolicyConditionInRule associations from PolicyRule instances or ReusablePolicy associations from ReusablePolicyContainer instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 331 describes class CIM_QueryCondition (Pre-defined).

Table 331 - SMI Referenced Properties/Methods for CIM_QueryCondition (Pre-defined)

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	Another provider supplied user friendly name
CommonName		Optional	A provider supplied user friendly name of the QueryCondition
SystemCreationClassName		Mandatory	The name of the class or the subclass used in the creation of the System object in whose scope this PolicyCondition is defined.

Table 331 - SMI Referenced Properties/Methods for CIM_QueryCondition (Pre-defined)

Properties	Flags	Requirement	Description & Notes
SystemName		Mandatory	The name of the System object in whose scope this PolicyCondition is defined.
PolicyRuleCreationClassName		Mandatory	For a rule-specific PolicyCondition, the CreationClassName of the PolicyRule object with which this Condition is associated. For a reusable Policy Condition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
PolicyRuleName		Mandatory	For a rule-specific PolicyCondition, the name of the PolicyRule object with which this Condition is associated. For a reusable PolicyCondition, a special value, 'NO RULE', should be used to indicate that this Condition is reusable and not associated with a single PolicyRule.
CreationClassName		Mandatory	CreationClassName indicates the name of the class or the subclass used in the creation of an instance.
PolicyConditionName		Mandatory	A user-friendly name of this PolicyCondition.
QueryResultName		Mandatory	In the context of the associated PolicyRule, QueryResultName defines a unique alias for the query results that may be used in subsequent QueryConditions or MethodActions of the same PolicyRule. This string is treated as a class name, in a query statement.
Query		Mandatory	A query expression that defines the condition(s) under which QueryConditionResult instances will be generated. The FROM clause may reference any class, including QueryConditionResult. NOTE that the property name, "QueryConditionPath", shall not be used as the name of a select-list entry in the select-criteria clause of the query.
QueryLanguage		Mandatory	The language in which the query is expressed. SMI-S only recognizes "CQL". Other query languages may be encoded for vendor specific support, but only CQL is supported for SMI-S interoperability. Values {"CQL", "DMTF Reserved", "Vendor Reserved"}
Trigger		Mandatory	If Trigger = true, and with the exception of any PolicyTimePeriodConditions, PolicyConditions of this PolicyRule are not evaluated until this 'triggering' condition query is true. There shall be no more than one QueryCondition with Trigger = true associated with a particular PolicyRule.

31.8.31 CIM_ReusablePolicy (Container to MethodAction)

The ReusablePolicy association provides for the reuse of any subclass of Policy in a ReusablePolicyContainer. It is used in the Policy Package to associate the ReusablePolicyContainer (Dynamic PolicyRule templates) to the System in which the Dynamic PolicyRule can be defined.

CIM_ReusablePolicy is subclassed from CIM_PolicyInSystem.

This would only be supported if the Policy Package supports Dynamic PolicyRules, as defined in the PolicyFeaturesSupported property of the PolicyCapabilities instance for the Package. There would be one instance of ReusablePolicy for every Dynamic PolicyRule template supported by the profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 332 describes class CIM_ReusablePolicy (Container to MethodAction).

Table 332 - SMI Referenced Properties/Methods for CIM_ReusablePolicy (Container to MethodAction)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

31.8.32 CIM_ReusablePolicy (Container to QueryCondition)

The ReusablePolicy association provides for the reuse of any subclass of Policy in a ReusablePolicyContainer. It is used in the Policy Package to associate the ReusablePolicyContainer (Dynamic PolicyRule templates) to the System in which the Dynamic PolicyRule can be defined.

CIM_ReusablePolicy is subclassed from CIM_PolicyInSystem.

This would only be supported if the Policy Package supports Dynamic PolicyRules, as defined in the PolicyFeaturesSupported property of the PolicyCapabilities instance for the Package. There would be one instance of ReusablePolicy for every Dynamic PolicyRule template supported by the profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 333 describes class CIM_ReusablePolicy (Container to QueryCondition).

Table 333 - SMI Referenced Properties/Methods for CIM_ReusablePolicy (Container to QueryCondition)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

31.8.33 CIM_ReusablePolicy (Container to System)

The ReusablePolicy association provides for the reuse of any subclass of Policy in a ReusablePolicyContainer. It is used in the Policy Package to associate the ReusablePolicyContainer (Dynamic PolicyRule templates) to the System in which the Dynamic PolicyRule can be defined.

CIM_ReusablePolicy is subclassed from CIM_PolicyInSystem.

This would only be supported if the Policy Package supports Dynamic PolicyRules, as defined in the PolicyFeaturesSupported property of the PolicyCapabilities instance for the Package. There would be one instance of ReusablePolicy for every Dynamic PolicyRule template supported by the profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 334 describes class CIM_ReusablePolicy (Container to System).

Table 334 - SMI Referenced Properties/Methods for CIM_ReusablePolicy (Container to System)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

31.8.34 CIM_ReusablePolicyContainer

ReusablePolicyContainer is a class representing an administratively defined container for reusable policy-related information. This class does not introduce any additional properties beyond those in its superclass AdminDomain. It does, however, participate in a unique association for containing policy elements that may be used in constructing Dynamic PolicyRules.

An instance of this class uses the NameFormat value "ReusablePolicyContainer".

CIM_ReusablePolicyContainer is subclassed from CIM_AdminDomain.

This would only be supported if the Policy Package supports Dynamic PolicyRules, as defined in the PolicyFeaturesSupported property of the PolicyCapabilities instance for the Package. There would be one instance of ReusablePolicyContainer for every Dynamic PolicyRule template supported by the profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 335 describes class CIM_ReusablePolicyContainer.

Table 335 - SMI Referenced Properties/Methods for CIM_ReusablePolicyContainer

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	This should be the Name of the PolicyRule Template as specified in the profile.
NameFormat		Mandatory	This shall be set to "ReusablePolicyContainer"

31.8.35 SNIA_PolicyCapabilities

This class defines the policy capabilities of the Specific Policy Subprofile associated via ElementCapabilities.

SNIA_PolicyCapabilities is subclassed from CIM_Capabilities.

An instance of SNIA_PolicyCapabilities shall exist for each Specific Policy Subprofile.

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

Table 336 describes class SNIA_PolicyCapabilities.

Table 336 - SMI Referenced Properties/Methods for SNIA_PolicyCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	This is a user friendly name of the capabilities instance.
PolicyFeaturesSupported		Mandatory	This array identifies the Policy features supported by the profile associated via ElementCapabilities. Values {"Static Rules", "Dynamic Rules", "Client Defined Rules"}

IMPLEMENTED

STABLE

Clause 32: Physical Package Package

32.1 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 52. 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.

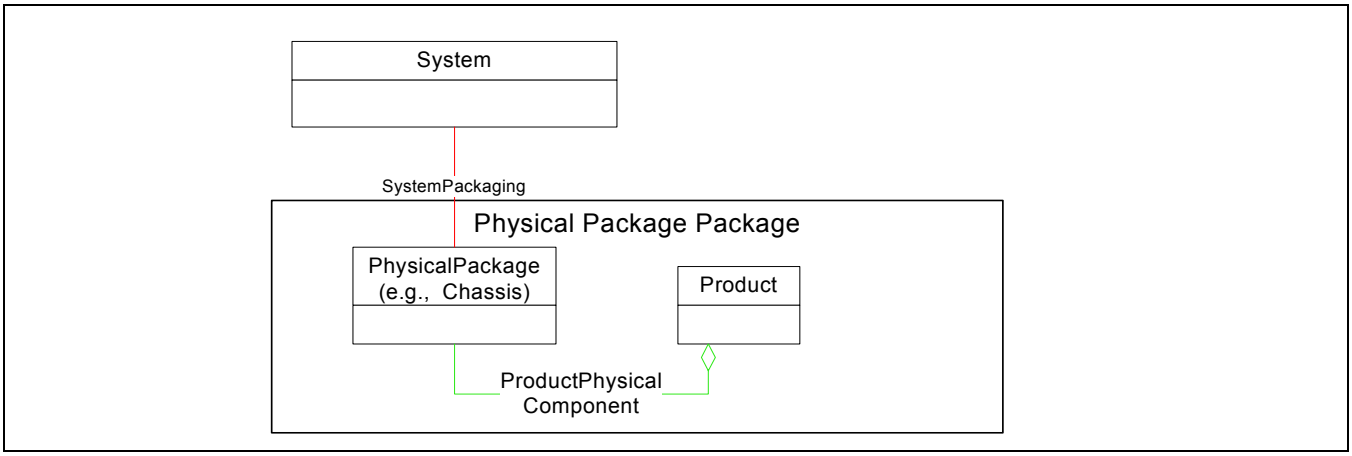

Figure 52 - Physical Package Package Mandatory Classes

32.1.1 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 shall 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 controller is realized by a card. The cards are contained in a controller chassis.

When establishing physical packages for subcomponents (e.g., disk drives, changers, etc.) the provider shall populate both Container and Realizes associations. Similarly, when establishing the Product instances for the packages the provider shall populate the ProductParentChild association to the parent product.

32.1.2 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 53 shows an example of the use of mandatory and optional physical package classes.

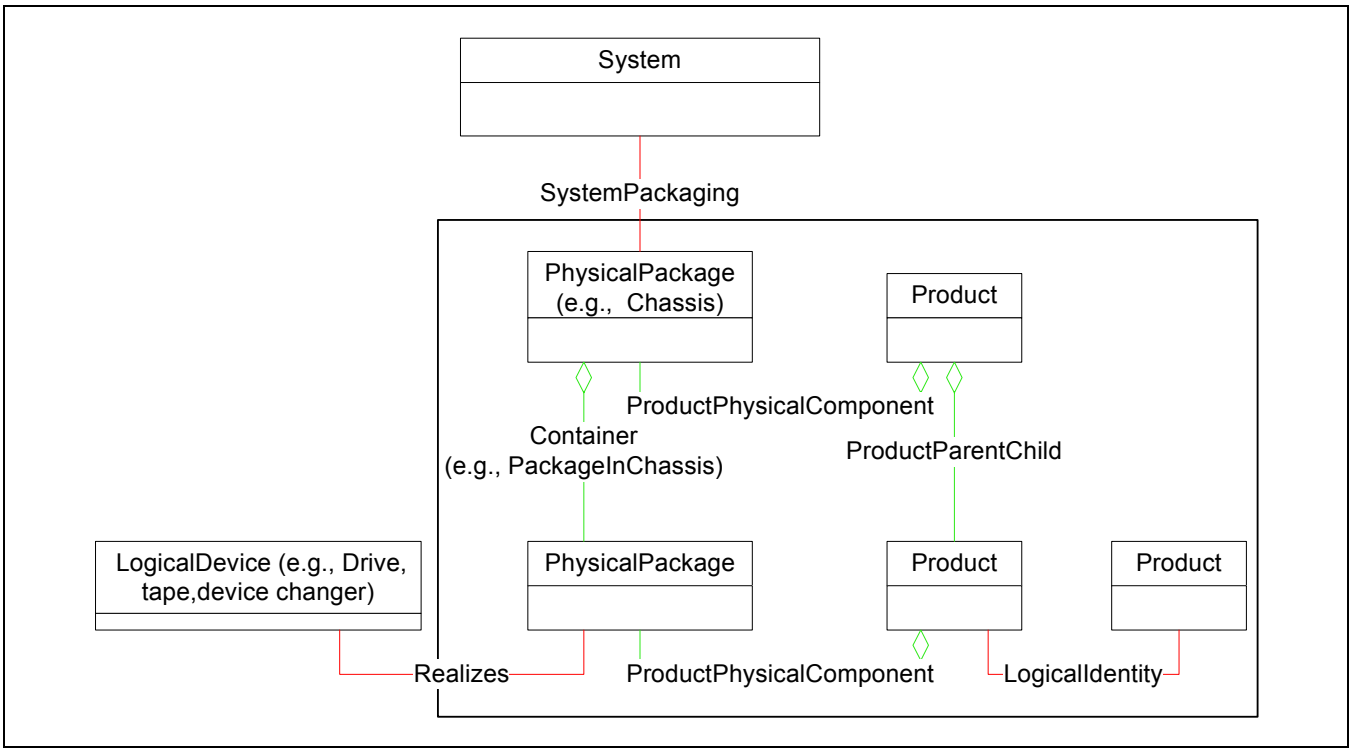

Figure 53 - Physical Package Package with Optional Classes

32.2 Health and Fault Management Considerations

Not defined in this standard.

32.3 Cascading Considerations

Not defined in this standard.

32.4 Supported Subprofiles and Packages

Not defined in this standard.

32.5 Methods of this Profile

Not defined in this standard.

32.6 Client Considerations and Recipes

32.6.1 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.

32.6.2 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.

32.6.3 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.

32.7 Registered Name and Version

Physical Package version 1.3.0

32.8 CIM Elements

Table 337 describes the CIM elements for Physical Package.

Table 337 - CIM Elements for Physical Package

Element Name	Requirement	Description
32.8.1 CIM_Card	Optional	A subclass of PhysicalPackage which may be used to appropriately model a specific implementation
32.8.2 CIM_Chassis	Optional	A subclass of PhysicalPackage which may be used to appropriately model a specific implementation
32.8.3 CIM_Container	Optional	Associates a PhysicalPackage to its component physical packages (e.g., Drives in a Storage System). This may be subclassed (e.g., PackageInChassis), but only the Container properties are required
32.8.4 CIM_LogicalIdentity	Optional	
32.8.5 CIM_PackageInChassis	Optional	Provided to allow component hierarchies
32.8.6 CIM_PhysicalConnector	Optional	
32.8.7 CIM_PhysicalElementLocation	Conditional	Conditional requirement: Support for the Location profile..
32.8.8 CIM_PhysicalPackage	Mandatory	
32.8.9 CIM_Product	Mandatory	

Table 337 - CIM Elements for Physical Package

Element Name	Requirement	Description
32.8.10 CIM_ProductParentChild	Optional	If more than one product comprises a system, this association should be used to indicate the 'parent' product
32.8.11 CIM_ProductPhysicalComponent	Mandatory	
32.8.12 CIM_SystemPackaging	Mandatory	Associates a system and its physical components. The ComputerSystemPackage subclass should be used if the referenced system is subclassed as ComputerSystem.

32.8.1 CIM_Card

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

32.8.2 CIM_Chassis

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

32.8.3 CIM_Container

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

Table 338 describes class CIM_Container.

Table 338 - SMI Referenced Properties/Methods for CIM_Container

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

32.8.4 CIM_LogicalIdentity

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

32.8.5 CIM_PackageInChassis

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

32.8.6 CIM_PhysicalConnector

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

32.8.7 CIM_PhysicalElementLocation

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for the Location profile..

Table 339 describes class CIM_PhysicalElementLocation.

Table 339 - SMI Referenced Properties/Methods for CIM_PhysicalElementLocation

Properties	Flags	Requirement	Description & Notes
PhysicalLocation		Mandatory	
Element		Mandatory	

32.8.8 CIM_PhysicalPackage

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 340 describes class CIM_PhysicalPackage.

Table 340 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Tag		Mandatory	
ElementName		Optional	
Name		Optional	
Manufacturer		Mandatory	
Model		Mandatory	
SerialNumber		Optional	
Version		Optional	
PartNumber		Optional	

32.8.9 CIM_Product

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

Table 341 describes class CIM_Product.

Table 341 - SMI Referenced Properties/Methods for CIM_Product

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
IdentifyingNumber		Mandatory	
Vendor		Mandatory	
Version		Mandatory	
ElementName		Mandatory	

32.8.10 CIM_ProductParentChild

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Optional

Table 342 describes class CIM_ProductParentChild.

Table 342 - SMI Referenced Properties/Methods for CIM_ProductParentChild

Properties	Flags	Requirement	Description & Notes
Parent		Mandatory	
Child		Mandatory	

32.8.11 CIM_ProductPhysicalComponent

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

Table 343 describes class CIM_ProductPhysicalComponent.

Table 343 - SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

32.8.12 CIM_SystemPackaging

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

Table 344 describes class CIM_SystemPackaging.

Table 344 - SMI Referenced Properties/Methods for CIM_SystemPackaging

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

STABLE

EXPERIMENTAL

Clause 33: Power Supply Profile

33.1 Synopsis

Profile Name: Power Supply

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.11.0

Table 345 describes the related profiles for Power Supply.

Table 345 - Related Profiles for Power Supply

Profile Name	Organization	Version	Requirement	Description
Server	SNIA	1.3.0	Mandatory	

Specializes: DMTF Power Supply Profile

The SNIA Power Supply profile specializes DSP1015: the DMTF Power Supply profile by adding indications.

33.2 Description

The SNIA Power Supply profile specializes the DMTF Power Supply profile by adding indications. No other changes are made to the DMTF profile.

33.3 Implementation

See DSP1015: the DMTF Power Supply Profile.

33.3.1 Health and Fault Management Consideration

None

33.3.2 Cascading Considerations

None

33.4 Methods

See DSP1015: the DMTF Power Supply Profile.

33.5 Use Cases

See DSP1015: the DMTF Power Supply Profile.

33.6 CIM Elements

Table 345 describes the CIM elements for Power Supply.

Table 346 - CIM Elements for Power Supply

Element Name	Requirement	Description
33.6.1 CIM_ElementCapabilities	Conditional	Conditional requirement: EnabledLogicalElementCapabilities
33.6.2 CIM_EnabledLogicalElementCapabilities	Optional	
33.6.3 CIM_IsSpare	Optional	
33.6.4 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for Power Supply redundancy.
33.6.5 CIM_OwningCollectionElement	Conditional	Conditional requirement: Support for Power Supply redundancy.
33.6.6 CIM_PowerSupply	Mandatory	
33.6.7 CIM_RedundancySet	Optional	
33.6.8 CIM_SuppliesPower	Optional	
33.6.9 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: Conditional on support for Power Supply RedundancyCQL - Change of redundancy status

33.6.1 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: null

Table 347 describes class CIM_ElementCapabilities.

Table 347 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	
Capabilities	Mandatory	

33.6.2 CIM_EnabledLogicalElementCapabilities

CIM_EnabledLogicalElementCapabilities represents the capabilities of the power supply.

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

Table 348 describes class CIM_EnabledLogicalElementCapabilities.

Table 348 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
RequestedStatesSupported	Mandatory	Array that contains the supported requested states for the instance of CIM_PowerSupply.
ElementNameEditSupported	Mandatory	
MaxElementNameLen	Conditional	Conditional requirement: Support for Element Name editing.Conditional on Support for Element Name editing.

33.6.3 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 349 describes class CIM_IsSpare.

Table 349 - SMI Referenced Properties/Methods for CIM_IsSpare

Properties	Requirement	Description & Notes
SpareStatus	Mandatory	
FailoverSupported	Mandatory	
Antecedent	Mandatory	The RedundancySet
Dependent	Mandatory	PowerSupply

33.6.4 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 350 describes class CIM_MemberOfCollection.

Table 350 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Collection	Mandatory	
Member	Mandatory	

33.6.5 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 351 describes class CIM_OwningCollectionElement.

Table 351 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement

Properties	Requirement	Description & Notes
OwnedElement	Mandatory	
OwningElement	Mandatory	

33.6.6 CIM_PowerSupply

CIM_PowerSupply is used to represent the power supply.

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

Table 352 describes class CIM_PowerSupply.

Table 352 - SMI Referenced Properties/Methods for CIM_PowerSupply

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
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	
RequestedState	Mandatory	
RequestStateChange()	Mandatory	The implementation shall support this method, but the method may always return 'Not Supported.'

33.6.7 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 353 describes class CIM_RedundancySet.

Table 353 - SMI Referenced Properties/Methods for CIM_RedundancySet

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	shall be formatted as a free formed string of variable length (pattern ".+")

Table 353 - SMI Referenced Properties/Methods for CIM_RedundancySet

Properties	Requirement	Description & Notes
RedundancyStatus	Mandatory	
TypeOfSet	Mandatory	
MinNumberNeeded	Mandatory	shall match 0 when the minimum number of power supplies needed for the redundancy is unknown.
Failover()	Optional	

33.6.8 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 354 describes class CIM_SuppliesPower.

Table 354 - SMI Referenced Properties/Methods for CIM_SuppliesPower

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Shall reference the instance of the subclass of CIM_ManagedSystemElement representing element receiving the power.

33.6.9 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 355 describes class CIM_SystemDevice.

Table 355 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 34: Fan Profile

34.1 Synopsis

Profile Name: Fan

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.11.0

Table 356 describes the related profiles for Fan.

Table 356 - Related Profiles for Fan

Profile Name	Organization	Version	Requirement	Description
Server	SNIA	1.3.0	Mandatory	

Specializes: DMTF Fan Profile

The SNIA Fan profile specializes DSP1013: the DMTF Fan profile by adding indications.

34.2 Description

The SNIA Fan profile specializes the DMTF Fan profile by adding indications. No other changes are made to the DMTF profile.

34.3 Implementation

See DSP1013: the DMTF Fan Profile.

34.3.1 Health and Fault Management Consideration

None

34.3.2 Cascading Considerations

None

34.4 Methods

See DSP1013: the DMTF Fan Profile.

34.5 Use Cases

See DSP1013: the DMTF Fan Profile.

34.6 CIM Elements

Table 356 describes the CIM elements for Fan.

Table 357 - CIM Elements for Fan

Element Name	Requirement	Description
34.6.1 CIM_AssociatedCooling	Optional	
34.6.2 CIM_ElementCapabilities	Conditional	Conditional requirement: EnabledLogicalElementCapabilities
34.6.3 CIM_EnabledLogicalElementCapabilities	Optional	
34.6.4 CIM_Fan	Mandatory	
34.6.5 CIM_IsSpare	Optional	
34.6.6 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for Fan redundancy.
34.6.7 CIM_NumericSensor	Optional	
34.6.8 CIM_OwningCollectionElement	Conditional	Conditional requirement: Support for Fan redundancy.
34.6.9 CIM_RedundancySet	Optional	
34.6.10 CIM_Sensor	Optional	
34.6.11 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	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

34.6.1 CIM_AssociatedCooling

CIM_AssociatedCooling associates CIM_Fan with a subclass of CIM_ManagedSystemElement.

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

Table 358 describes class CIM_AssociatedCooling.

Table 358 - SMI Referenced Properties/Methods for CIM_AssociatedCooling

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	Shall reference an instance of a subclass of CIM_ManagedSystemElement for which the fan is providing cooling.

34.6.2 CIM_ElementCapabilities

CIM_ElementCapabilities is used to associate CIM_Fan with the CIM_EnabledLogicalElementCapabilities instance that describes the capabilities of the fan.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: null

Table 359 describes class CIM_ElementCapabilities.

Table 359 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	
Capabilities	Mandatory	

34.6.3 CIM_EnabledLogicalElementCapabilities

CIM_EnabledLogicalElementCapabilities represents the capabilities of the Fan.

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

Table 360 describes class CIM_EnabledLogicalElementCapabilities.

Table 360 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
RequestedStatesSupported	Mandatory	Array that contains the supported requested states for the instance of CIM_Fan.
ElementNameEditSupported	Mandatory	
MaxElementNameLen	Conditional	Conditional requirement: Support for Element Name editing.Conditional on Support for Element Name editing.

34.6.4 CIM_Fan

CIM_Fan is used to represent the fan.

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

Table 361 describes class CIM_Fan.

Table 361 - SMI Referenced Properties/Methods for CIM_Fan

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
CreationClassName	Mandatory	Key
DeviceID	Mandatory	Key
ElementName	Mandatory	
OperationalStatus	Mandatory	
HealthState	Mandatory	
EnabledState	Mandatory	
VariableSpeed	Mandatory	
DesiredSpeed	Conditional	Conditional requirement: Support for the SetSpeed method..
ActiveCooling	Mandatory	Shall have the value TRUE
RequestedState	Mandatory	
SetSpeed()	Optional	
RequestStateChange()	Mandatory	The implementation shall support this method, but the method may always return 'Not Supported.'

34.6.5 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 362 describes class CIM_IsSpare.

Table 362 - SMI Referenced Properties/Methods for CIM_IsSpare

Properties	Requirement	Description & Notes
SpareStatus	Mandatory	
FailoverSupported	Mandatory	
Antecedent	Mandatory	The RedundancySet
Dependent	Mandatory	The Fan

34.6.6 CIM_MemberOfCollection

CIM_MemberOfCollection is used to associate CIM_Fan with CIM_RedundancySet that the CIM_Fan is a member of.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for Fan redundancy.

Table 363 describes class CIM_MemberOfCollection.

Table 363 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Collection	Mandatory	
Member	Mandatory	

34.6.7 CIM_NumericSensor

The CIM_NumericSensor class is defined by the Sensors Profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 364 describes class CIM_NumericSensor.

Table 364 - SMI Referenced Properties/Methods for CIM_NumericSensor

Properties	Requirement	Description & Notes
SensorType	Mandatory	Shall be set to 5 (Tachometer)
BaseUnits	Mandatory	Shalll be 19 (RPM)
RateUnits	Mandatory	Shall be 0 (None)

34.6.8 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 Fan redundancy.

Table 365 describes class CIM_OwningCollectionElement.

Table 365 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement

Properties	Requirement	Description & Notes
OwnedElement	Mandatory	
OwningElement	Mandatory	

34.6.9 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 366 describes class CIM_RedundancySet.

Table 366 - SMI Referenced Properties/Methods for CIM_RedundancySet

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	shall be formatted as a free formed string of variable length (pattern ".+")
RedundancyStatus	Mandatory	
TypeOfSet	Mandatory	

Table 366 - SMI Referenced Properties/Methods for CIM_RedundancySet

Properties	Requirement	Description & Notes
MinNumberNeeded	Mandatory	shall match 0 when the minimum number of power supplies needed for the redundancy is unknown.
Failover()	Optional	

34.6.10 CIM_Sensor

The CIM_Sensor class is defined by the Sensors Profile.

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

Table 367 describes class CIM_Sensor.

Table 367 - SMI Referenced Properties/Methods for CIM_Sensor

Properties	Requirement	Description & Notes
SensorType	Mandatory	Shall be set to 5 (Tachometer)

34.6.11 CIM_SystemDevice

CIM_SystemDevice is used to associate CIM_Fan with CIM_ComputerSystem that the CIM_Fan is a member of.

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

Table 368 describes class CIM_SystemDevice.

Table 368 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 35: Sensors Profile

35.1 Synopsis

Profile Name: Sensors

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.11.0

Table 369 describes the related profiles for Sensors.

Table 369 - Related Profiles for Sensors

Profile Name	Organization	Version	Requirement	Description
Server	SNIA	1.3.0	Mandatory	

Specializes: DMTF Sensors Profile

The SNIA Sensors profile specializes DSP1009: the DMTF Sensors profile by adding indications.

35.2 Description

The SNIA Sensors profile specializes the DMTF Sensors profile by adding indications. No other changes are made to the DMTF profile.

35.3 Implementation

See DSP1009: the DMTF Sensors Profile.

35.3.1 Health and Fault Management Consideration

None

35.3.2 Cascading Considerations

None

35.4 Methods

See DSP1009: the DMTF Sensors Profile.

35.5 Use Cases

See DSP1009: the DMTF Sensors Profile.

35.6 CIM Elements

Table 369 describes the CIM elements for Sensors.

Table 370 - CIM Elements for Sensors

Element Name	Requirement	Description
35.6.1 CIM_AssociatedSensor	Optional	
35.6.2 CIM_ElementCapabilities	Conditional	Conditional requirement: EnabledLogicalElementCapabilities
35.6.3 CIM_EnabledLogicalElementCapabilities	Optional	
35.6.4 CIM_NumericSensor	Conditional	Conditional requirement: Absence of Support for CIM_Sensor.
35.6.5 CIM_Sensor	Conditional	Conditional requirement: Absence of Support for CIM_NumericSensor.
35.6.6 CIM_SystemDevice	Mandatory	
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

35.6.1 CIM_AssociatedSensor

CIM_AssociatedSensor associates CIM_Sensor or CIM_NumericSensor with a subclass of CIM_ManagedSystemElement.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 371 describes class CIM_AssociatedSensor.

Table 371 - SMI Referenced Properties/Methods for CIM_AssociatedSensor

Properties	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_ManagedSystemElement for which the sensor is monitoring.

35.6.2 CIM_ElementCapabilities

CIM_ElementCapabilities is used to associate CIM_Sensor with the CIM_EnabledLogicalElementCapabilities instance that describes the capabilities of the fan.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: null

Table 372 describes class CIM_ElementCapabilities.

Table 372 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	
Capabilities	Mandatory	

35.6.3 CIM_EnabledLogicalElementCapabilities

CIM_EnabledLogicalElementCapabilities represents the capabilities of the Fan.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 373 describes class CIM_EnabledLogicalElementCapabilities.

Table 373 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
RequestedStatesSupported	Mandatory	Array that contains the supported requested states for the instance of CIM_Sensor.
ElementNameEditSupported	Mandatory	
MaxElementNameLen	Conditional	Conditional requirement: EditSupportConditional on Support for Element Name editing.

35.6.4 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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Absence of Support for CIM_Sensor.

Table 374 describes class CIM_NumericSensor.

Table 374 - SMI Referenced Properties/Methods for CIM_NumericSensor

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
CreationClassName	Mandatory	Key
DeviceID	Mandatory	Key
BaseUnits	Mandatory	
UnitModifier	Mandatory	
RateUnits	Mandatory	
CurrentReading	Mandatory	
LowerThresholdNonCritical	Conditional	See DMTF Sensors Profile
UpperThresholdNonCritical	Conditional	See DMTF Sensors Profile
LowerThresholdCritical	Conditional	See DMTF Sensors Profile
UpperThresholdCritical	Conditional	See DMTF Sensors Profile
LowerThresholdFatal	Conditional	See DMTF Sensors Profile
UpperThresholdFatal	Conditional	See DMTF Sensors Profile

Table 374 - SMI Referenced Properties/Methods for CIM_NumericSensor

Properties	Requirement	Description & Notes
SupportedThresholds	Mandatory	See DMTF Sensors Profile
SettableThresholds	Mandatory	See DMTF Sensors Profile
SensorType	Mandatory	
PossibleStates	Mandatory	
CurrentState	Mandatory	
ElementName	Mandatory	
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	
RequestedState	Mandatory	
OperationalStatus	Mandatory	
HealthState	Mandatory	
RequestStateChange()	SensorState Management	
RestoreDefaultThresholds()	RestoreDefaultThreshold Support	

35.6.5 CIM_Sensor

CIM_Sensor is used to represent a discrete sensor. The CIM_Sensor class is mandatory if the CIM_NumericSensor class is not implemented.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Absence of Support for CIM_NumericSensor.

Table 375 describes class CIM_Sensor.

Table 375 - SMI Referenced Properties/Methods for CIM_Sensor

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

Table 375 - SMI Referenced Properties/Methods for CIM_Sensor

Properties	Requirement	Description & Notes
SensorType	Mandatory	
PossibleStates	Mandatory	See DMTF Sensors Profile
CurrentState	Mandatory	See DMTF Sensors Profile
ElementName	Mandatory	See DMTF Sensors Profile
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 \.*\')..'See DMTF Sensors Profile
EnabledState	Mandatory	See DMTF Sensors Profile
RequestedState	Mandatory	See DMTF Sensors Profile
OperationalStatus	Mandatory	
HealthState	Mandatory	
RequestStateChange()	SensorState Management	

35.6.6 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 376 describes class CIM_SystemDevice.

Table 376 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 36: Base Server Profile

36.1 Synopsis

Profile Name: Base Server

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.15.0

Table 378 describes the related profiles for Base Server.

Table 377 - Related Profiles for Base Server

Profile Name	Organization	Version	Requirement	Description
Storage HBA	SNIA	1.3.0	Optional	
Host Hardware RAID Controller	SNIA	1.3.0	Optional	
Storage Enclosure	SNIA	1.3.0	Optional	

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.

36.2 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 represents a physical system. The Virtual System profile is an alternatative for virtual systems.

36.3 Implementation

See DSP1004, DMTF Base Server profile for details on the model.

In a storage context, there are several related deployment options.

36.3.1 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.

36.3.2 Host Hardware RAID Instrumentation

Host Hardware RAID vendors have the same deployment options as HBA vendors (see 36.3.1)

36.3.3 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.

36.3.4 Health and Fault Management Consideration

36.3.5 Cascading Considerations

None

36.4 Methods

See DSP1004, DMTF Base Server profile.

36.5 Use Cases

See DSP1004, DMTF Base Server profile.

36.6 CIM Elements

Table 378 describes the CIM elements for Base Server.

Table 378 - CIM Elements for Base Server

Element Name	Requirement	Description
36.6.1 CIM_ComputerSystem	Mandatory	The hosting system for the Storage Elements.
36.6.2 CIM_ComputerSystemPackage	Mandatory	
36.6.3 CIM_EnabledLogicalElementCapabilities	Optional	
36.6.4 CIM_PhysicalPackage	Mandatory	

36.6.1 CIM_ComputerSystem

The hosting system for the Storage Elements.

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

Table 379 describes class CIM_ComputerSystem.

Table 379 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Requirement	Description & Notes
CreationClassName	Mandatory	
Name	Mandatory	Unique identifier for the hosting system.
ElementName	Mandatory	User friendly name
NameFormat	Mandatory	
OtherIdentifyingInfo	Mandatory	
Dedicated	Mandatory	0 (Not Dedicated)
OtherDedicatedDescriptions	Optional	
OperationalStatus	Mandatory	
EnabledState	Mandatory	
RequestedState	Mandatory	

36.6.2 CIM_ComputerSystemPackage

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

Table 380 describes class CIM_ComputerSystemPackage.

Table 380 - SMI Referenced Properties/Methods for CIM_ComputerSystemPackage

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

36.6.3 CIM_EnabledLogicalElementCapabilities

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

Table 381 describes class CIM_EnabledLogicalElementCapabilities.

Table 381 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Requirement	Description & Notes
RequestedStatesSupported	Mandatory	

36.6.4 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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

EXPERIMENTAL

EXPERIMENTAL

Clause 37: Media Access Device Profile

37.1 Synopsis

Profile Name: Media Access Device

Version: 1.0.0

Organization: SNIA

CIM Schema Version: 2.11.0

Table 384 describes the related profiles for Media Access Device.

Table 382 - Related Profiles for Media Access Device

Profile Name	Organization	Version	Requirement	Description
Software Inventory	SNIA	1.3.0	Mandatory	
Software Update	DMTF	1.0.0	Optional	
Indication	SNIA	1.3.0	Optional	

The Media Access Device profile models media access devices - such as tape and CD drives.

37.2 Description

The Media Access Device profile models media access devices - such as tape and CD drives.

37.2.1 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).

37.2.2 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

See Table 383.

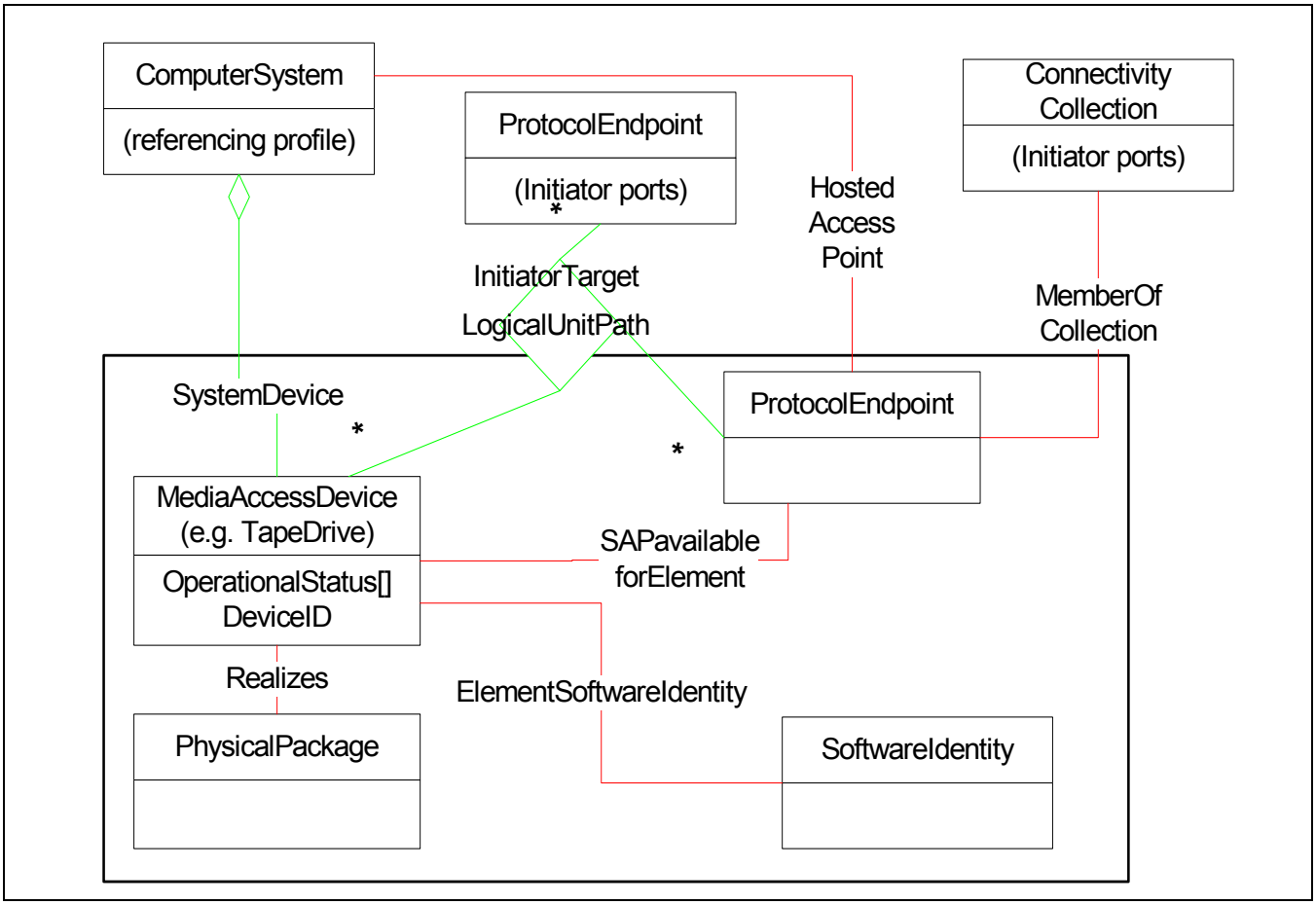

Figure 54 - Media Access Device Class Diagram

37.3 Implementation

37.3.1 Health and Fault Management Consideration

The MediaAccessDevice.OperationalStatus contains the overall status of the disk, summarized in Table 383.

Table 383 - 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.

Table 383 - OperationalStatus For MediaAccessDevice (Continued)

Primary Operational Status	Subsidiary Operational Status	Description
9 “Stopping”		Media Access Device is being disabled.
10 “Stopped”		Media Access Device is disabled.

37.3.2 Cascading Considerations

Not defined in this standard.

37.3.3 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.

37.4 Methods

37.4.1 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.

37.5 Use Cases

Not defined in this standard.

37.6 CIM Elements

Table 384 describes the CIM elements for Media Access Device.

Table 384 - CIM Elements for Media Access Device

Element Name	Requirement	Description
37.6.1 CIM_EnabledLogicalElementCapabilities	Mandatory	
37.6.2 CIM_HostedAccessPoint	Optional	ComputerSystem to storage ProtocolEndpoint
37.6.3 CIM_MediaAccessDevice	Mandatory	Represents a tape or optical drive.

Table 384 - CIM Elements for Media Access Device

Element Name	Requirement	Description
37.6.4 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.
37.6.5 CIM_ProtocolEndpoint	Optional	
37.6.6 CIM_Realizes	Mandatory	Associates MediaAccessDevice and PhysicalPackage.
37.6.7 CIM_SAPAvailableForElement	Conditional	Conditional requirement: Support for ProtocolEndpoints..Associates MediaAccessDevice to ProtocolEndpoint
37.6.8 CIM_SystemDevice	Mandatory	ComputerSystem to MediaAccessDevice.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_MediaAccessDevice	Optional	MediaAccessDevice Creation. See37.3.3 Hot swap insertion or Removal of Drives
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_MediaAccessDevice	Optional	MediaAccessDevice Removal. See37.3.3 Hot swap insertion or Removal of Drives

37.6.1 CIM_EnabledLogicalElementCapabilities

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

Table 385 describes class CIM_EnabledLogicalElementCapabilities.

Table 385 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	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.

37.6.2 CIM_HostedAccessPoint

ComputerSystem to storage ProtocolEndpoint

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

Table 386 describes class CIM_HostedAccessPoint.

Table 386 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

37.6.3 CIM_MediaAccessDevice

Represents a tape or optical drive.

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

Table 387 describes class CIM_MediaAccessDevice.

Table 387 - SMI Referenced Properties/Methods for CIM_MediaAccessDevice

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
DeviceID	Mandatory	
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()	RequestedStatesSupported	

37.6.4 CIM_PhysicalPackage

The physical aspects of the drive.

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

Table 388 describes class CIM_PhysicalPackage.

Table 388 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	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	

37.6.5 CIM_ProtocolEndpoint

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

Table 389 describes class CIM_ProtocolEndpoint.

Table 389 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint

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

37.6.6 CIM_Realizes

Associates MediaAccessDevice and PhysicalPackage.

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

Table 390 describes class CIM_Realizes.

Table 390 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

37.6.7 CIM_SAPAvailableForElement

Associates MediaAccessDevice to ProtocolEndpoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for ProtocolEndpoints..

Table 391 describes class CIM_SAPAvailableForElement.

Table 391 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Requirement	Description & Notes
AvailableSAP	Mandatory	
ManagedElement	Mandatory	

37.6.8 CIM_SystemDevice

ComputerSystem to MediaAccessDevice.

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

Table 392 describes class CIM_SystemDevice.

Table 392 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 38: Storage Enclosure Profile

Synopsis

Table 393 describes the supported profiles for Storage Enclosure.

Table 393 - Supported Profiles for Storage Enclosure

Registered Profile Names	Mandatory	Version
Power Supply	No	1.0.0
Fan	No	1.0.0
Sensors	No	1.0.0
Disk Drive Lite	No	1.3.0
Media Access Device	No	1.0.0
Switch	No	1.3.0

The Storage Enclosure profile describes an enclosure that houses storage components.

38.1 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.

38.1.1 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:

38.1.1.1 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.

38.1.1.2 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.

38.1.1.3 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 55 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.

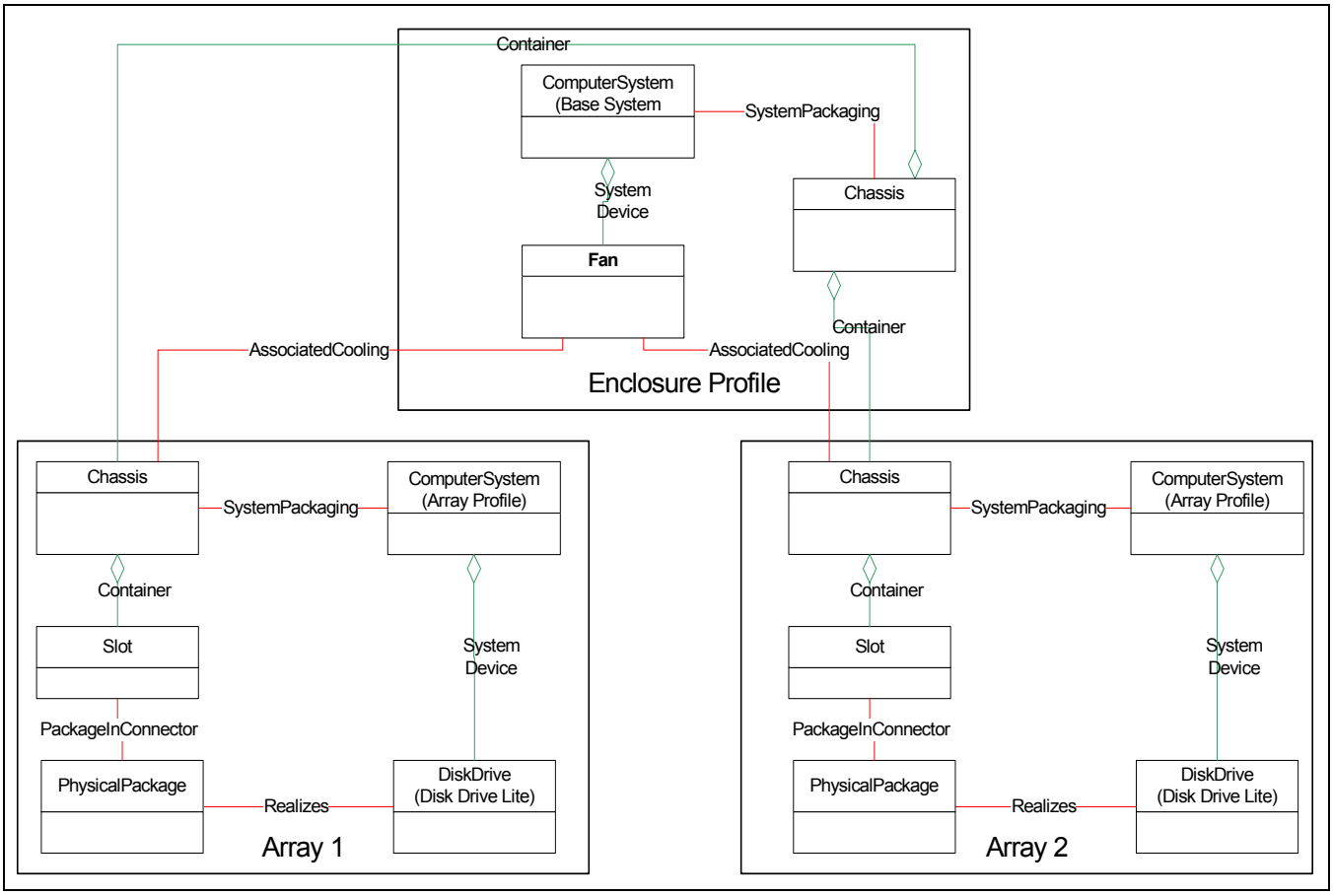

Figure 55 - Enclosure with Two Arrays

38.1.2 Examples of Storage Enclosure Configurations

38.1.2.1 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 38.2.5.2 is mandatory.

38.1.2.2 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 38.2.5.2 is mandatory.

38.1.2.3 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).

38.1.2.4 Enclosure Dedicated to non-RAID Controllers on Multiple Servers

Guideline 2 applies. The referencing profile is the base system profile.

Guideline 3 may apply.

38.1.2.5 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;

38.1.2.6 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.

38.2 Implementation

38.2.1 Health and Fault Management Consideration

See the component profiles.

38.2.2 Cascading Considerations

Not defined in this standard.

38.2.3 Enclosure Elements

38.2.3.1 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.

38.2.3.2 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.

38.2.3.3 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.

38.2.4 Storage Elements

38.2.4.1 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.

38.2.4.2 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 38.2.4.1.

38.2.4.3 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.

38.2.4.4 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.

38.2.4.5 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 38.3.1

38.2.5 Physical Assets

The physical representation of the storage enclosure is mandatory. The core frame of the storage enclosure is described by CIM_Chassis.

38.2.5.1 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.

38.2.5.2 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 56 illustrates the model.

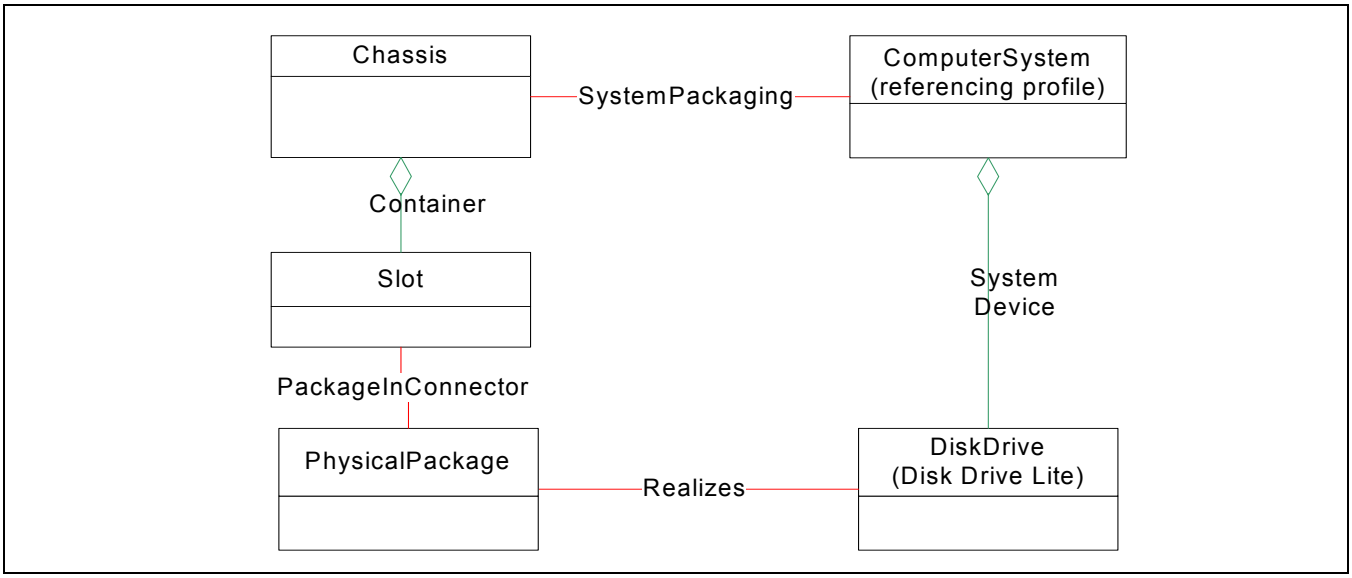

Figure 56 - Model for Disk in Enclosure

38.3 Methods

38.3.1 Extrinsic Methods of the Profile

38.3.1.1 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.

38.3.1.2 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.

38.3.1.3 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.

38.3.2 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

38.4 Use Cases

38.5 Registered Name and Version

Storage Enclosure version 1.3.0

Specialized DMTF Physical Asset version 1.0.0a

38.6 CIM Elements

Table 394 describes the CIM elements for Storage Enclosure.

Table 394 - CIM Elements for Storage Enclosure

Element Name	Requirement	Description
38.6.1 CIM_Card	Optional	CIM_Card is used to represent the card and its FRU data.
38.6.2 CIM_Chassis	Optional	CIM_Chassis is used to represent the chassis and its FRU data.
38.6.3 CIM_Chip	Optional	CIM_Chip is used to represent the chip and its FRU data.
38.6.4 CIM_ComputerSystemPackage	Conditional	Conditional requirement: Support for a Chassis instance.
38.6.5 CIM_ConfigurationCapacity	Optional	
38.6.6 CIM_ConfigurationReportingService	Mandatory	
38.6.7 CIM_ConnectedTo	Optional	

Table 394 - CIM Elements for Storage Enclosure

Element Name	Requirement	Description
38.6.8 CIM_Container	Optional	CIM_Container is used to associate a Physical Package with Physical Elements representing the physical elements that reside within the package.
38.6.9 CIM_ElementCapabilities	Conditional	Conditional requirement: Support for a PhysicalAssetCapabilities instance.
38.6.10 CIM_ElementCapacity	Conditional	Conditional requirement: Support for a ConfigurationCapacity instance.
38.6.11 CIM_HostedService	Mandatory	Associates the CIM_ConfigurationReportingService to the System in the referencing profile.
38.6.12 CIM_PackageInConnector	Optional	CIM_PackageInConnector is used to associate a CIM_PhysicalConnector or CIM_Slot instance, representing the connector or slot, with Physical Packages.
38.6.13 CIM_PhysicalAssetCapabilities	Optional	
38.6.14 CIM_PhysicalComponent	Optional	CIM_PhysicalComponent is used to represent any physical element that cannot be further decomposed, such as ASIC or tape, and its FRU data.
38.6.15 CIM_PhysicalConnector	Optional	CIM_PhysicalConnector is used to represent the physical connector.
38.6.16 CIM_PhysicalElement	Mandatory	At least one PhysicalElement subclass is mandatory; see the subclasses for details.
38.6.17 CIM_PhysicalFrame	Optional	CIM_PhysicalFrame is used to represent the frame and its FRU data.
38.6.18 CIM_PhysicalMemory	Optional	CIM_PhysicalMemory is used to represent the physical memory and its FRU data.
38.6.19 CIM_PhysicalPackage	Mandatory	
38.6.20 CIM_Rack	Optional	CIM_Rack is used to represent the rack and its FRU data.
38.6.21 CIM_Realizes	Optional	
38.6.22 CIM_Slot	Optional	CIM_Slot is used to represent the slot and its FRU data.
38.6.23 CIM_SystemPackaging	Optional	Associates a system and its physical components. The ComputerSystemPackage subclass should be used if the referenced system is subclassed as ComputerSystem.

38.6.1 CIM_Card

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

Table 395 describes class CIM_Card.

Table 395 - SMI Referenced Properties/Methods for CIM_Card

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
HostingBoard		Mandatory	
PackageType		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
CanBeFRUed		Optional	
VendorCompatibilityStrings		Mandatory	
ElementName		Mandatory	

38.6.2 CIM_Chassis

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

Table 396 describes class CIM_Chassis.

Table 396 - SMI Referenced Properties/Methods for CIM_Chassis

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	

Table 396 - SMI Referenced Properties/Methods for CIM_Chassis

Properties	Flags	Requirement	Description & Notes
PackageType		Mandatory	Shall be 3 (Chassis/Frame)
ChassisPackageType		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
CanBeFRUed		Optional	should be implemented when the PhysicalElement can be field replaced.
VendorCompatibilityStrings		Mandatory	
ElementName		Mandatory	

38.6.3 CIM_Chip

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

Table 397 describes class CIM_Chip.

Table 397 - SMI Referenced Properties/Methods for CIM_Chip

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
CanBeFRUed		Optional	
ElementName		Mandatory	

38.6.4 CIM_ComputerSystemPackage

CIM_ComputerSystemPackage is used to associate CIM_ComputerSystem, representing the managed system, with a System Chassis.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for a Chassis instance.

Table 398 describes class CIM_ComputerSystemPackage.

Table 398 - SMI Referenced Properties/Methods for CIM_ComputerSystemPackage

Properties	Flags	Requirement	Description & Notes
PlatformGUID		Mandatory	
Antecedent		Mandatory	Shall reference the System Chassis.
Dependent		Mandatory	

38.6.5 CIM_ConfigurationCapacity

CIM_ConfigurationCapacity is used to advertise the possible configuration of a System Chassis.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 399 describes class CIM_ConfigurationCapacity.

Table 399 - SMI Referenced Properties/Methods for CIM_ConfigurationCapacity

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
ElementName		Mandatory	
ObjectType		Mandatory	
OtherTypeDescription		Conditional	Conditional requirement: ConfigurationCapacity ObjectType set to 0 (Other).\.'
MinimumCapacity		Optional	
MaximumCapacity		Mandatory	
Increment		Mandatory	
VendorCompatibilityStrings		Mandatory	

38.6.6 CIM_ConfigurationReportingService

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

Table 400 describes class CIM_ConfigurationReportingService.

Table 400 - SMI Referenced Properties/Methods for CIM_ConfigurationReportingService

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
Name		Mandatory	
CreationClassName		Mandatory	
GetClassTypes()		Mandatory	
GetUnitTypes()		Mandatory	
ReportCapacity()		Mandatory	

38.6.7 CIM_ConnectedTo

CIM_ConnectedTo is used to associate the CIM_PhysicalConnector or CIM_Slot instances that represent connectors that are connected together.

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

Table 401 describes class CIM_ConnectedTo.

Table 401 - SMI Referenced Properties/Methods for CIM_ConnectedTo

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

38.6.8 CIM_Container

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

Table 402 describes class CIM_Container.

Table 402 - SMI Referenced Properties/Methods for CIM_Container

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

38.6.9 CIM_ElementCapabilities

CIM_ElementCapabilities is used to associate Physical Elements with the CIM_PhysicalAssetCapabilities instances that advertise the physical capabilities. CIM_ElementCapabilities shall be instantiated when an instance of CIM_PhysicalAssetCapabilities exists.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for a PhysicalAssetCapabilities instance.

Table 403 describes class CIM_ElementCapabilities.

Table 403 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

38.6.10 CIM_ElementCapacity

CIM_ElementCapacity is used to associate CIM_ConfigurationCapacity instances with a System Chassis.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for a ConfigurationCapacity instance.

Table 404 describes class CIM_ElementCapacity.

Table 404 - SMI Referenced Properties/Methods for CIM_ElementCapacity

Properties	Flags	Requirement	Description & Notes
Capacity		Mandatory	
Element		Mandatory	

38.6.11 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 405 describes class CIM_HostedService.

Table 405 - 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.

38.6.12 CIM_PackageInConnector

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

Table 406 describes class CIM_PackageInConnector.

Table 406 - SMI Referenced Properties/Methods for CIM_PackageInConnector

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Antecedent		Mandatory	

38.6.13 CIM_PhysicalAssetCapabilities

CIM_PhysicalAssetCapabilities is used to advertise whether the associated instance of subclass of CIM_PhysicalElement contains FRU data.

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

Table 407 describes class CIM_PhysicalAssetCapabilities.

Table 407 - SMI Referenced Properties/Methods for CIM_PhysicalAssetCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
FRUInfoSupported		Mandatory	

38.6.14 CIM_PhysicalComponent

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

Table 408 describes class CIM_PhysicalComponent.

Table 408 - SMI Referenced Properties/Methods for CIM_PhysicalComponent

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
CanBeFRUed		Optional	
ElementName		Mandatory	

38.6.15 CIM_PhysicalConnector

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

Table 409 describes class CIM_PhysicalConnector.

Table 409 - SMI Referenced Properties/Methods for CIM_PhysicalConnector

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
ConnectorLayout		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo

Table 409 - SMI Referenced Properties/Methods for CIM_PhysicalConnector

Properties	Flags	Requirement	Description & Notes
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
ElementName		Mandatory	

38.6.16 CIM_PhysicalElement

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

38.6.17 CIM_PhysicalFrame

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

Table 410 describes class CIM_PhysicalFrame.

Table 410 - SMI Referenced Properties/Methods for CIM_PhysicalFrame

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
PackageType		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
VendorCompatibilityStrings		Mandatory	
ElementName		Mandatory	

38.6.18 CIM_PhysicalMemory

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

Table 411 describes class CIM_PhysicalMemory.

Table 411 - SMI Referenced Properties/Methods for CIM_PhysicalMemory

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
FormFactor		Mandatory	
MemoryType		Mandatory	
Speed		Mandatory	
Capacity		Mandatory	
BankLabel		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
CanBeFRUed		Optional	
ElementName		Mandatory	

38.6.19 CIM_PhysicalPackage

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

Table 412 describes class CIM_PhysicalPackage.

Table 412 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
VendorCompatibilityStrings		Mandatory	
CanBeFRUed		Optional	
ElementName		Mandatory	

38.6.20 CIM_Rack

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

Table 413 describes class CIM_Rack.

Table 413 - SMI Referenced Properties/Methods for CIM_Rack

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
TypeOfRack		Mandatory	
PackageType		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo

Table 413 - SMI Referenced Properties/Methods for CIM_Rack

Properties	Flags	Requirement	Description & Notes
VendorCompatibilityStrings		Mandatory	
ElementName		Mandatory	

38.6.21 CIM_Realizes

CIM_Realizes is used to associate an instance of subclass of CIM_LogicalDevice, representing the logical device, with a Physical Element.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 414 describes class CIM_Realizes.

Table 414 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Shall reference the System Chassis.
Dependent		Mandatory	

38.6.22 CIM_Slot

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 415 describes class CIM_Slot.

Table 415 - SMI Referenced Properties/Methods for CIM_Slot

Properties	Flags	Requirement	Description & Notes
Tag		Mandatory	
CreationClassName		Mandatory	
Number		Mandatory	
ConnectorLayout		Mandatory	
Manufacturer		Conditional	Conditional requirement: FRUinfo
Model		Conditional	Conditional requirement: FRUinfo
SerialNumber		Conditional	Conditional requirement: FRUinfo

Table 415 - SMI Referenced Properties/Methods for CIM_Slot

Properties	Flags	Requirement	Description & Notes
PartNumber		Conditional	Conditional requirement: FRUinfo
SKU		Conditional	Conditional requirement: FRUinfo
VendorCompatibilityStrings		Mandatory	
ElementName		Mandatory	

38.6.23 CIM_SystemPackaging

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

Table 416 describes class CIM_SystemPackaging.

Table 416 - SMI Referenced Properties/Methods for CIM_SystemPackaging

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

EXPERIMENTAL

STABLE

Clause 39: Software Subprofile

39.1 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 57 contains the instance diagram for the Software Profile.

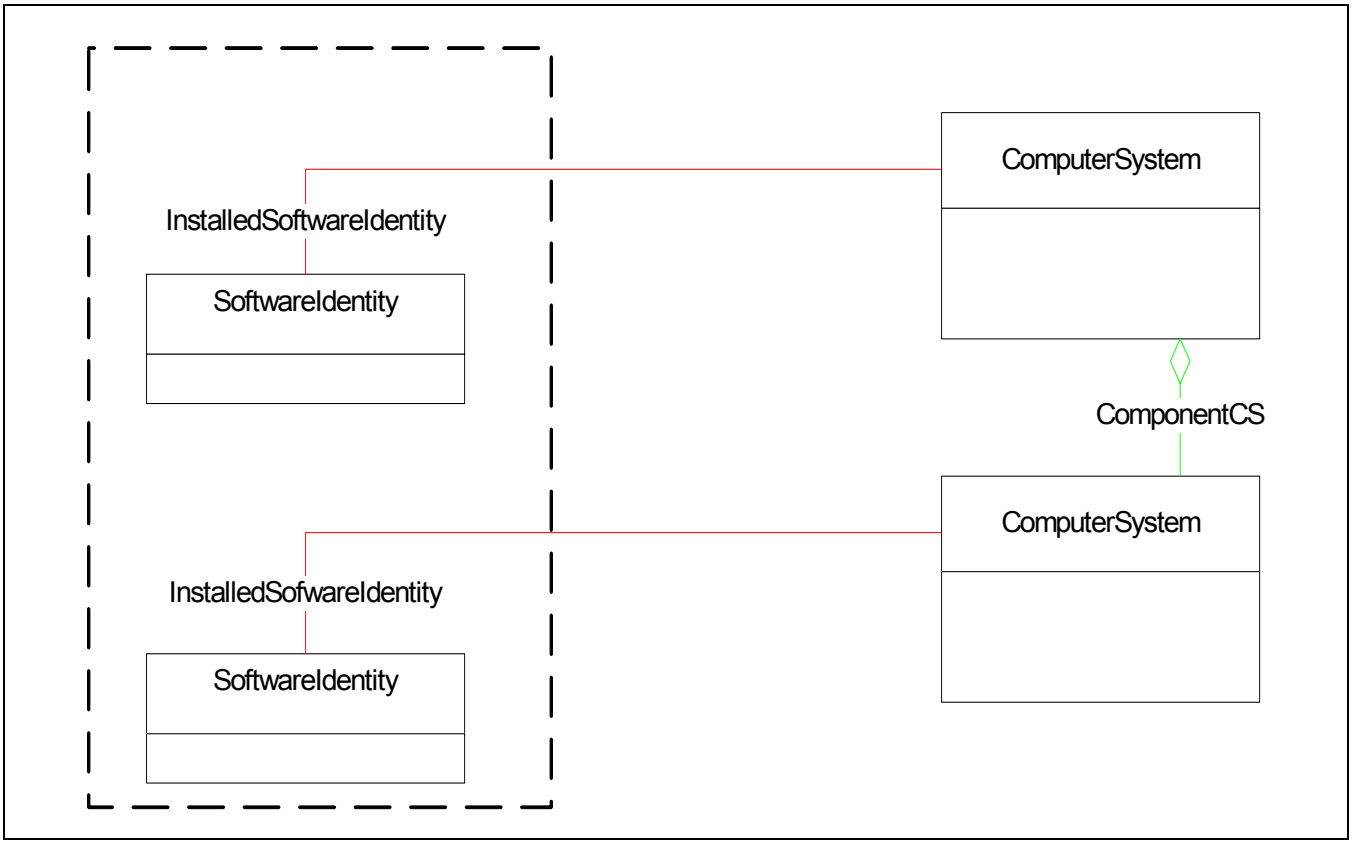

Figure 57 - Software Instance Diagram

39.2 Health and Fault Management Considerations

Not defined in this standard.

39.3 Cascading Considerations

Not defined in this standard.

39.4 Supported Subprofiles, and Packages

None.

39.5 Methods of the Profile

None.

39.6 Client Considerations and Recipes

None.

39.7 Registered Name and Version

Software version 1.3.0

39.8 CIM Elements

Table 417 describes the CIM elements for Software.

Table 417 - CIM Elements for Software

Element Name	Requirement	Description
39.8.1 CIM_InstalledSoftwareIdentity	Mandatory	
39.8.2 CIM_SoftwareIdentity	Mandatory	

39.8.1 CIM_InstalledSoftwareIdentity

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

Table 418 describes class CIM_InstalledSoftwareIdentity.

Table 418 - SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity

Properties	Flags	Requirement	Description & Notes
System		Mandatory	
InstalledSoftware		Mandatory	

39.8.2 CIM_SoftwareIdentity

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 419 describes class CIM_SoftwareIdentity.

Table 419 - 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

Clause 40: Software Inventory Profile

40.1 Synopsis

Profile Name: Software Inventory

Version: 1.3.0

Organization: SNIA

CIM Schema Version: 2.15.0

Table 421 describes the related profiles for Software Inventory.

Table 420 - Related Profiles for Software Inventory

Profile Name	Organization	Version	Requirement	Description
Indication	SNIA	1.3.0	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.

40.2 Description

The Software Inventory profile models installed and available software and firmware. The SNIA version specializes the DMTF profile in order to add indications.

40.2.1 Relationship to the SMI-S Software Profile

SMI-S defined a similar profile, the Software Subprofile (see Clause 39: 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 proerties beyond those in the SNIA Software subprofile.

40.3 Implementation

See DSP1023, DMTF Software Inventory profile.

40.3.1 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)

40.3.2 Health and Fault Management Consideration

None

40.3.3 Cascading Considerations

None

40.4 Methods

See DSP1023, DMTF Software Inventory profile.

40.5 Use Cases

See DSP1023, DMTF Software Inventory profile.

40.6 CIM Elements

Table 421 describes the CIM elements for Software Inventory.

Table 421 - CIM Elements for Software Inventory

Element Name	Requirement	Description
40.6.1 CIM_ElementSoftwareIdentity	Optional	
40.6.2 CIM_HostedAccessPoint	Optional	

Table 421 - CIM Elements for Software Inventory

Element Name	Requirement	Description
40.6.3 CIM_HostedCollection	Conditional	Conditional requirement: Support for collection of SoftwareIdentity instances.
40.6.4 CIM_InstalledSoftwareIdentity	Optional	
40.6.5 CIM_MemberOfCollection	Conditional	Conditional requirement: Support for collection of SoftwareIdentity instances.
40.6.6 CIM_OrderedComponent	Optional	
40.6.7 CIM_OrderedDependency	Optional	
40.6.8 CIM_SAPAvailableForElement	Conditional	Conditional requirement: Support for SoftwareIdentityResource instances.
40.6.9 CIM_SoftwareIdentity	Mandatory	
40.6.10 CIM_SoftwareIdentityResource	Optional	
40.6.11 CIM_SystemSpecificCollection	Optional	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SoftwareIdentity	Mandatory	Creation of a SoftwareIdentity. See 40.3.1 Software Installation and Update
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SoftwareIdentity	Mandatory	Delete of a SoftwareIdentity. See 40.3.1 Software Installation and Update
SELECT * FROM CIM_AlertIndication WHERE OwningEntity=SNIA and MessageID=\Core1\"	Mandatory	In-place update of Software (or Firmware). See 40.3.1 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 40.3.1 Software Installation and Update

40.6.1 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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 422 describes class CIM_ElementSoftwareIdentity.

Table 422 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity

Properties	Requirement	Description & Notes
ElementSoftwareStatus	Mandatory	
Antecedent	Mandatory	
Dependent	Mandatory	

40.6.2 CIM_HostedAccessPoint

CIM_HostedAccessPoint is used to associate CIM_System and CIM_SoftwareIdentityResource when an instance of CIM_SoftwareIdentityResource is instrumented.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 423 describes class CIM_HostedAccessPoint.

Table 423 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

40.6.3 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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for collection of SoftwareIdentity instances.

Table 424 describes class CIM_HostedCollection.

Table 424 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

40.6.4 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 425 describes class CIM_InstalledSoftwareIdentity.

Table 425 - SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity

Properties	Requirement	Description & Notes
System	Mandatory	
InstalledSoftware	Mandatory	

40.6.5 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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for collection of SoftwareIdentity instances.

Table 426 describes class CIM_MemberOfCollection.

Table 426 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Requirement	Description & Notes
Collection	Mandatory	
Member	Mandatory	

40.6.6 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

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 427 describes class CIM_OrderedComponent.

Table 427 - SMI Referenced Properties/Methods for CIM_OrderedComponent

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

40.6.7 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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 428 describes class CIM_OrderedDependency.

Table 428 - SMI Referenced Properties/Methods for CIM_OrderedDependency

Properties	Requirement	Description & Notes
AssignedSequence	Mandatory	
Antecedent	Mandatory	
Dependent	Mandatory	

40.6.8 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

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for SoftwareIdentityResource instances.

Table 429 describes class CIM_SAPAvailableForElement.

Table 429 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Requirement	Description & Notes
AvailableSAP	Mandatory	
ManagedElement	Mandatory	

40.6.9 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 430 describes class CIM_SoftwareIdentity.

Table 430 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	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	

40.6.10 CIM_SoftwareIdentityResource

CIM_SoftwareIdentityResource is used to represent the location of a Software Identity, which could be used as input to the software installation service.

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

Table 431 describes class CIM_SoftwareIdentityResource.

Table 431 - SMI Referenced Properties/Methods for CIM_SoftwareIdentityResource

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

Table 431 - SMI Referenced Properties/Methods for CIM_SoftwareIdentityResource

Properties	Requirement	Description & Notes
InfoFormat	Mandatory	
AccessInfo	Mandatory	
ResourceType	Optional	

40.6.11 CIM_SystemSpecificCollection

CIM_SystemSpecificCollection is used to represent a collection of Available Software.

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

Table 432 describes class CIM_SystemSpecificCollection.

Table 432 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection

Properties	Requirement	Description & Notes
InstanceID	Mandatory	
ElementName	Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 41: Software Repository Subprofile

41.1 Description

This profile provides the ability to expose a collection of SoftwareIdentity instances representing software installation packages that can be used in conjunction with Clause 41: Software Installation Service Subprofile. These two profiles form a 'pair' that can be used together within a single system or independently on different unaware systems. The different use cases covered are shown in Figure 58.

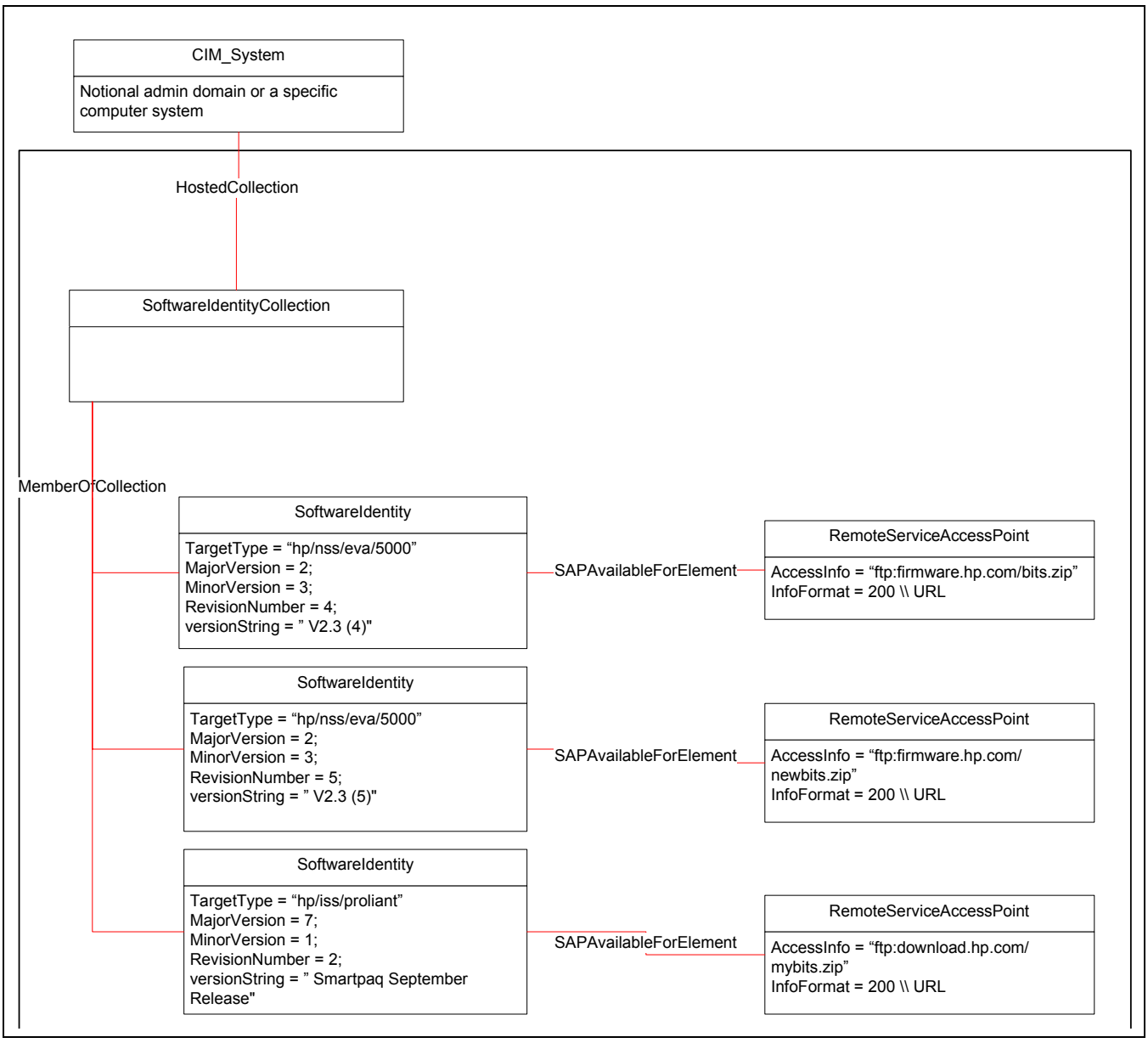

Figure 58 - Software Repository Instance Diagram

A typical implementation of a representation would consist of multiple SoftwareIdentitys representing potential upgrades associated by MemberOfCollection to an instance of a SoftwareIdentityCollection which represents the

collection itself. The ‘location’ of the bits needed to install a specific SoftwareIdentity are represented as RemoteServiceAccessPoint instances one per URL) associated to the SoftwareIdentity by SAPAvailableForElement.

41.1.1 Durable Names and Correlatable IDs of the Profile

Software Identity.TargetType is the only correlatable ID introduced by this subprofile. The TargetType parameter is a correlatable identifier that indicates the ‘type’ of SoftwareIdentity. It allows a ‘repository’ to be queried for applicable software/firmware.

The same format shall be used for the Software Repository and for the Software Installation Service so that correlation can be performed.

Since the SoftwareInstallationService may be able to handle multiple TargetTypes, SoftwareInstallationServiceCapabilities includes an array of supported TargetTypes that indicates the types supported by the service.

41.2 Health and Fault Management Considerations

Not defined in this standard.

41.3 Cascading Considerations

Not defined in this standard.

41.4 Methods of the Profile

None.

41.5 Supported Subprofiles, and Packages

None.

41.6 Client Considerations and Recipes

None.

41.7 Registered Name and Version

Software Repository version 1.3.0

41.8 CIM Elements

Table 433 describes the CIM elements for Software Repository.

Table 433 - CIM Elements for Software Repository

Element Name	Requirement	Description
41.8.1 CIM_HostedCollection	Mandatory	The SoftwareIdentityCollection is scoped to a system.
41.8.2 CIM_MemberOfCollection	Mandatory	Associates SoftwareIdentities to the collection

Table 433 - CIM Elements for Software Repository

Element Name	Requirement	Description
41.8.3 CIM_RemoteServiceAccessPoint	Mandatory	Used to express the location of the 'bits' for a software update as an URL
41.8.4 CIM_SAPAvailableForElement	Mandatory	Links one or more URLS to a SoftwareIdentity.
41.8.5 CIM_SoftwareIdentity	Mandatory	The information for an available software/ firmware update
41.8.6 CIM_SoftwareIdentityCollection	Mandatory	A collection of SoftwareIdentities that forms the repository
41.8.7 CIM_System	Mandatory	Represents the system hosting the Software Repository.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SoftwareIdentity	Mandatory	Addition of Software Identity
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SoftwareIdentity	Mandatory	Delete SoftwareIdentity

41.8.1 CIM_HostedCollection

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

Table 434 describes class CIM_HostedCollection.

Table 434 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

41.8.2 CIM_MemberOfCollection

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

Table 435 describes class CIM_MemberOfCollection.

Table 435 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

41.8.3 CIM_RemoteServiceAccessPoint

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

Table 436 describes class CIM_RemoteServiceAccessPoint.

Table 436 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	
ElementName		Mandatory	
AccessInfo		Mandatory	
InfoFormat		Mandatory	

41.8.4 CIM_SAPAvailableForElement

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

Table 437 describes class CIM_SAPAvailableForElement.

Table 437 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

41.8.5 CIM_SoftwareIdentity

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

Table 438 describes class CIM_SoftwareIdentity.

Table 438 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
TargetType	C	Mandatory	
SerialNumber		Optional	
ReleaseDate		Optional	

41.8.6 CIM_SoftwareIdentityCollection

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

Table 439 describes class CIM_SoftwareIdentityCollection.

Table 439 - SMI Referenced Properties/Methods for CIM_SoftwareIdentityCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Optional	

41.8.7 CIM_System

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

Table 440 describes class CIM_System.

Table 440 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Name of Class
Name		Mandatory	System hosting the Software Repository

EXPERIMENTAL

STABLE

Clause 42: Server Profile

42.1 Description

A CIM Server is anything that supports the CIM-XML protocol or other WBEM protocols and supports the basic read functional profile as defined by the CIM Operations over HTTP specification.

The Server Profile is mandatory for all compliant SMI-S servers.

The object manager part of the model, shown in Figure 59, 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.

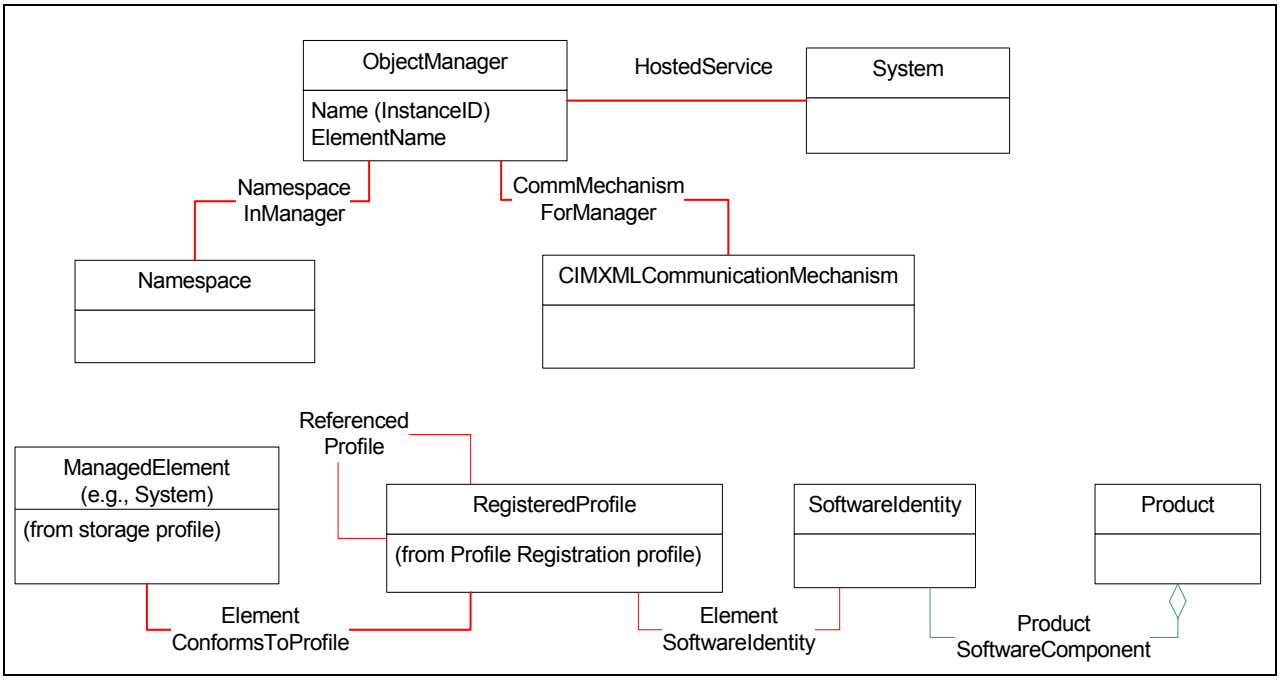

Figure 59 - Server Model

A Server is modeled as a System with a HostedService association to an ObjectManager. The ObjectManager is subclassed from Service.

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.

Each RegisteredProfile and RegisteredSubprofile instance (from the Profile Registration 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.

EXPERIMENTAL

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

EXPERIMENTAL

42.2 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

Service-location-tcp: The location of one service access point offered by the CIM Server over TCP transport. This attribute shall provide sufficient addressing information that the CIM Server can be addressed directly using only this attribute.

CommunicationMechanism: ObjectManagerCommunicationMechanism.CommunicationMechanism

OtherCommunicationMechanism: ObjectManagerCommunicationMechanism.OtherCommunicationMechanism

InteropSchemaNamespace: Namespace.Name for the InteropNamespace

ProtocolVersion: ObjectManagerCommunicationMechanism.Version

FunctionalProfilesSupported: ObjectManagerCommunicationMechanism.FunctionalProfilesSupported

FunctionalProfileDescriptions: ObjectManagerCommunicationMechanism.FunctionalProfileDescriptions

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.

42.2.1 HTTP Security Background

Section 4.4 of “Specification for CIM Operations over HTTP, Version 1.1” from DMTF describes the requirements for CIM clients and servers. The authentication methods referred to in the above specification are described in the IETF RFCs 1945, 2616, and 2617. Transport Layer Security (TLS) is defined by IETF RFC 4346 which contains specifications for both versions 1.0 and 1.1. The Secure Sockets Layer 3.0 is defined in reference SSL 3.0.

Section 4.4 of “Specification for CIM Operations over HTTP, Version 1.1" defines additional requirements for HTTP authentication, above those found in IETF RFC 2616, or the HTTP authentication documents [IETF RFC 2617]. HTTP authentication generally starts with an HTTP client request, such as “GET Request-URI” (where Request-URI is the resource requested). If the client request does not include an “Authorization” header line and authentication is required, the server responds with a “401 unauthorized” status code, and a “WWW-Authenticate” header line. The HTTP client shall then respond with the appropriate “Authorization” header line in a subsequent request. The format of the “WWW-Authenticate” and “Authorization” header lines varies depending on the type of authentication required: basic authentication or digest authentication. If the authentication is successful, the HTTP server will respond with a status code of “200 OK”.

Basic authentication involves sending the user name and password in the clear, and should only be used on a secure network, or in conjunction with a mechanism that ensures confidentiality, such as TLS. Digest authentication sends a secure digest of the user name and password (and other information including a nonce value), so that the password is not revealed. “401Unauthorized” responses should not include a choice of authentication

SSL 3.0 and TLS provide both confidentiality and integrity in communication, which precludes eavesdropping, tampering, and message forgery. While TLS 1.1 and TLS 1.0 are based on SSL 3.0 and the differences between them are not dramatic, it is important to note that these differences are significant enough that TLS 1.1, TLS 1.0 and SSL 3.0 will not interoperate. However, both versions of TLS do provide mechanisms for backwards compatibility with the earlier versions.

Both TLS and SSL 3.0 package one key establishment, confidentiality, signature and hash algorithm into a “cipher suite.” A registered 16-bit (4 hexadecimal digit) number, called the cipher suite index, is assigned for each defined cipher suite. For example, RSA key agreement, RSA signature, Triple Data Encryption Standard (3DES) using Encryption-Decryption-Encryption (EDE) and Cipher Block Chaining (CBC) confidentiality, and the Secure Hash Algorithm (SHA-1) hash is assigned the hexadecimal value {0x000A} for TLS. Note especially that TLS 1.1 requires (IEFT RFC 4346, Section 9 – Mandatory Cipher Suites): “In the absence of an application profile standard specifying otherwise, a TLS compliant application shall implement the cipher suite TLS_RSA_WITH_3DES_EBE_CBC_ SHA” described above.

The client always initiates the TLS and SSL 3.0 session and starts cipher suite negotiation by transmitting a handshake message that lists the cipher suites (by index value) that it will accept. The server responds with a handshake message indicating which cipher suite it selected from the list or an “abort” as described below. Although the client is required to order its list by increasing “strength” of cipher suite, the server may choose ANY of the cipher suites proposed by the client. Therefore, there is NO guarantee that the negotiation will select the strongest suite. If no cipher suites are mutually supported, the connection is aborted. When the negotiated options, including optional public key certificates and random data for developing keying material to be used by the cryptographic algorithms, are complete, messages are exchanged to place the communications channel in a secure mode.

SMI-S clients and servers may be attacked by setting up a false SMI-S server to capture userids and passwords or to insert itself as an undetected proxy between an SMI-S client and server. The most effective countermeasure for this attack is the controlled use of server certificates with SSL 3.0 or TLS, matched by client controls on certificate acceptance on the assumption that the false server will be unable to obtain an acceptable certificate. Specifically, this could be accomplished by configuring clients to always use SSL 3.0 or TLS underneath HTTP authentication, and only accept certificates from a specific local certificate authority. See 42.2.2 for requirements in this area. In the absence of this countermeasure, some protection can by obtained by limiting the scope of SMI-S discovery, including SLP, by IP address range (this involves client configuration plus SLP DA configuration, if any SLP DA is used), and the use of firewalls to block ports used by SMI-S and SLP in order to prevent SMI-S access to/from points outside a protected area of the network.

42.2.2 HTTP Security

This section specifies security requirements on the protocol for communication between a Client and an SMI-S Server, but not the mechanism of authentication used by the SMI-S Server.

Client authentication to the SMI-S Server is based on an authentication service. Differing authentication schemes may be supported, including host-based authentication, Kerberos, PKI, or other.

For the purposes of SMI-S, basic strength ciphersuites include 512-bit (or longer) asymmetric algorithms (RSA or Diffie-Hellman), combined with 40-bit (or longer) symmetric algorithms (Triple DES, IDEA, RC4-128) and either SHA-1 or MD5. Enhanced strength ciphersuites combine 1 024-bit (or longer) asymmetric algorithms (RSA or

Diffie-Hellman) with 128-bit (or longer) symmetric algorithms (Triple DES, IDEA, RC4-128, AES) and either SHA-1 or MD5.

42.2.2.1 General Requirements

The following are general requirements for the support of security when using HTTP.

a) SMI-S Servers and Clients shall conform to DMTF DSP0200 CIM Operations over HTTP 1.1 section 4.4.

b) HTTP Basic Authentication shall be implemented. HTTP Digest Authentication should be implemented.

c) To minimize compromising user identities, and credentials such as passwords, implementers should use HTTP Basic Authentication ONLY in conjunction with SSL 3.0 or TLS and an enhanced strength cipher-suite.

d) Where neither SSL 3.0 or TLS are used, or where they are used with a basic strength ciphersuite, implementers should utilize HTTP Digest Authentication.

IMPLEMENTED

e) To ensure a minimum level of security and interoperability between implementations, support for the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite shall be included in all implementation. Implementers are free to include additional cipher suites.

IMPLEMENTED

EXPERIMENTAL

When such cipher suites are supported, SSL_RSA_WITH_3DES_EDE_CBC_SHA for SSL 3.0 and TLS_RSA_WITH_3DES_EDE_CBC_SHA for TLS shall be supported at a minimum. Additionally, Table 441 identifies the SSL and TLS cipher suites (in order of descending preference) that should be supported and used by SMI-S implementations:

Table 441 - SSL and TLS Cipher Suites

TLS 1.0 & 1.1.	SSL 3.0
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	SSL_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	SSL_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA

The order of the cipher suites in Table 441 is the order of preference (i.e., cipher suites higher in the table are preferred over those lower in the table) when multiple cipher suites are offered unless overridden by local security policy. Within each pair of cipher suites, the "_DHE_" suite uses a Diffie-Hellman exchange to provide forward secrecy so that future disclosure of the RSA key(s) used will not compromise previous secured traffic.

Recognizing that implementers are likely to start with the least preferred 3DES-based cipher suites and then consider the AES suites, it is important to note that the National Institute of Standards and Technology (NIST) is currently encouraging transition to AES. Implementers should be aware that AES_128 is not only a stronger encryption algorithm than 3DES, but also that AES_128 tends to be more efficient and of higher performance when implemented.

For these reasons, if an SMI-S implementation supports 3DES, then support of AES_128 is strongly recommended. It is reasonable to expect that a future version of SMI-S will include a mandatory AES_128-based cipher suite.

EXPERIMENTAL

f) If no enhanced strength ciphersuite is supported, then HTTP Digest Authentication shall be implemented.

g) A user identity and credential used with one type of HTTP Authentication (i.e., Basic or Digest) shall not ever be subsequently used with the other type of HTTP Authentication. To avoid compromising the integrity of a stronger scheme, established good security practices avoids the reuse of identity & credential information across schemes of different strengths.

h) SSL 3.0 and TLS 1.0 shall be supported; TLS 1.1 is currently an allowed option that is strongly recommended. SSL support is currently required for backwards compatibility as described in Appendix E of RFC 4346.

EXPERIMENTAL

Additionally, SMI-S implementations shall have configurable mechanisms to only use cipher suites that include RSA, DHE-RSA, or DHE-DSS key establishment mechanisms and RSA or DSA signature mechanisms (i.e., only certificate-based cipher suites). These mechanisms shall further prevent the negotiation of the “EXPORT” cipher suites (identified in Section A.5 of RFC 4346 as TLS 1.1 shall not negotiate cipher suites; in addition, SMI-S prohibits use of "EXPORT" ciphersuites with SSL 3.0 and TLS 1.0).

Although DES is an allowed cipher when used with the appropriate key exchange mechanism, DES is vulnerable to brute-force attacks. When such an attack is a concern, a stronger cipher should be used.

It is important to recognize that maintaining security often requires changing requirements to reflect advances in technology, discovery of vulnerabilities, and defenses against new attacks. Consequently, it is expected that future versions of SMI-S will require TLS 1.1 to be implemented, deprecate support for SSL 3.0, deprecate cipher suites that include DES (any key size) as the cipher, and deprecate cipher suites that include MD5 as the hash.

EXPERIMENTAL

i) Clients that fail to contact an SMI-S server via HTTP over SSL 3.0 or TLS on TCP port 5 989 should retry with HTTP on TCP port 5 988 if their security policy allows it.

j) In order for Clients and Servers to communicate, they need to be using a consistent approach to security. It is possible for properly configured Clients and Servers to fail to communicate if one is relying upon port 5 989 and the other on port 5 988.

k) Servers can accelerate discovery that a secure channel is needed by responding to HTTP contacts on TCP port 5 988 with an HTTP REDIRECT to the appropriate HTTPS: URL (HTTP over SSL or TLS on TCP port 5 989) to avoid the need for clients to timeout the HTTP contact attempt. Clients should honor such redirects in this situation.

l) Anonymous SSL/TLS ciphersuites should not be offered or used for CIM operation invocation by SMI-S Clients. Anonymous SSL/TLS ciphersuites should not be used for indication delivery to indication listeners that do not have certificates - see 42.2.2.

42.2.2.2 Requirements for the support of HTTP Realm

The relationship of the realm-value to an authentication service, and one or more sets of user identity and credential, is determined separately by the configuration of each SMI-S client, and configurations may differ between multiple SMI-S clients in the same system. The means of creating this configuration in the SMI-S client is outside of the scope of this specification. The client configuration is expected to a contain at least a default set of user identity and credential per realm-value. When the configuration associates a single realm-value with multiple sets of user identity and credential, the basis on which a single set is selected is also outside of the scope of this specification (and may include considerations such as the need to assert elevated privilege at the server to perform specific operations.)

Where the Realm field is not used, or the realm-value is unrecognized, the SMI-S Client may use means outside of the scope of this specification to identify the user identity and credential to be used, including the use of information obtained during Service Discovery.

For this revision of the specification, it is recommended that a single realm-value per SMI-S Server be defined by means such as a configuration file. In future revisions, the definition of multiple and dynamic user identities and credentials per SMI-S Server will be addressed, and may use other communication methods in addition to, or in place of, the Realm field.

a) The Realm field defined by HTTP Version 1.1 (see RFC 2617 section 1.2 and RFC 2616) shall be implemented by the SMI-S Server, and should be used to identify to the Client the authentication service to be used to access the server.

b) The realm-value contains information to help determine which specific user identity and credential (e.g. user ID & password) and are to be used with the authentication service, but shall not contain any portion of an identity or a credential itself.

c) The exact form of the authentication service is not defined by SMI-S, and may either be part of the configuration of an SMI-S Server, or may involve an external entity such as a RADIUS server. A single authentication service may be utilized by multiple SMI-S Servers. Realm-values shall be unique throughout the scope of the authentication service.

d) When provided, the realm-value shall meet all of the requirements contained in RFC 2616 and RFC 2617, with the exception of the specific requirement in section 3.2.1 of RFC 2617 that the realm-value “be displayed to users”. In SMI-S, the realm-value may be handled by the SMI-S Client without reference to a user.

e) Where no format for the realm-value has been defined by other standards or conventions, and where an authentication is handled autonomously by an SMI-S server, then a string in the format defined in 42.2.2.3, "SMI-S defined format for HTTP Realm"” is recommended.

f) Where a single authentication service is utilized by multiple SMI-S Servers, the SMI-S recommended format defined in 42.2.2.3, "SMI-S defined format for HTTP Realm" should not be used, and use of SHA-1 in the creation of realm-values is recommended.

42.2.2.3 SMI-S defined format for HTTP Realm

The format is based on components of the definition of the Uniform Resource Identifier (URI) in IETF RFC 2396 and extended in IETF RFC 3986, and is described using the BNF-like grammar of those documents as:

```
[1*(unreserved) "."] "smis@" host
```

where:

- unreserved is as defined in section 2.3 of IETF RFC 2396

- "." is a dot
- "smis@" is a string literal
- host is as defined in section 3 of IETF RFC 3986

The combination of the unreserved and host portions should be defined in a manner that allows an administrator to quickly identify a specific SMI-S Server in his configuration. Note that some portion of unreserved could be generated randomly in the SMI-S Server to reduce the chance of accidental realm collisions.

An example of the use of the recommended format defined above is as follows: Consider a single server system labelled Server6 owned by Widgets Inc. (owner of the example.com domain) that hosts two SMI-S Servers, one from Acme Inc., and the other from XYZ Ltd. The realm-value reported by the Acme SMI-S Server might be "ug723.acme.net.smis@server6.example.com". In the configuration of a specific SMI-S client accessing the Acme SMI-S Server, this realm-value might identify a server-specific authentication service and a user identity of "arrayuser74" and a password of "YT56z". Similarly, the realm-value reported by the XYZ Ltd. SMI-S Server might be "bx48d.xyz.co.uk.smis@server6.example.com". In the configuration of a different SMI-S client accessing the XYZ SMI-S Server, this realm-value might identify a SMI-S-server-specific authentication service and a user identity of "42fred" and a password of "OTH3afa".

42.2.2.4 Certificate Usage with SSL 3.0 and TLS

Within SMI-S, SSL 3.0 and TLS are used with public key certificates (or identity certificates) for authentication. These X.509 certificates conform to the format and semantics specified in IETF RFC 3280 and use a digital signature to bind together a public key with an identity. These signatures will often be issued by a certification authority (CA) associated with an internal or external public key infrastructure (PKI); however, an alternate approach uses self-signed certificates (the certificate is digitally signed by the very same key-pair whose public part appears in the certificate data). The trust models associated with these two approaches are very different. In the case of PKI certificates, there is a hierarchy of trust and a trusted third-party that can be consulted in the certificate validation process, which enhances security at the expense of increased complexity. The self-signed certificates can be used to form a web of trust (trust decisions are in the hands of individual users/administrators), but is considered less secure as there is no central authority for trust (e.g., no identity assurance or revocation). This reduction in overall security, which may still offer adequate protections for some environments, is accompanied by an easing of the overall complexity of implementation.

With PKI certificates, it is often necessary to traverse the hierarchy or chain of trust in search of a root of trust or trust anchor (a trusted CA). This trust anchor may be an internal CA, which has a certificate signed by a higher ranking CA, or it may be the end of a certificate chain with the highest ranking CA. This highest ranking CA provides the ultimate in attestation authority in a particular PKI scheme and its certificate, known as a root certificate, can only be self-signed. Establishing a trust anchor at the root certificate level, especially for commercial CAs, can have undesirable side effects resulting from the implicit trust afforded all certificates issued by that commercial CA. Ideally the trust anchor should be established with the lowest ranking CA that is practical.

The remainder of this subsection provides certificate related requirements that apply to any SMI-S implementation that supports SSL 3.0 or TLS.

42.2.2.4.1 Require support for existing common practice for certificate usage.

SMI-S uses X.509 version 3 public key certificates that are conformant with the Certificate and Certificate Extension Profile defined in Section 4 of IETF RFC 3280. This profile specifies the mandatory fields that shall be included in the certificate as well as optional fields and extensions that may be included in the certificate.

Server certificates shall be supported and client certificates MAY be supported. A server certificate is presented by the server to authenticate the server to the client; likewise, a client certificate is presented by the client to authenticate itself to the server. For public web sites offering secure communications via SSL 3.0 or TLS, server certificate usage is quite common, but client certificates are rarely used.

SMI-S clients and servers shall perform basic path validation, extension path validation, and CRL validation as specified in Section 6 of IETF RFC 3280 for all presented certificates. These validations include, but are not limited to, the following:

- The certificate is a validly constructed certificate
- The signature is correct for the certificate
- The date of its use is within the validity period (i.e., it has not expired)
- The certificate has not been revoked (applies only to PKI certificates)
- The certificate chain is validly constructed (considering the peer certificate plus valid issuer certificates up to the maximum allowed chain depth; applies only to PKI certificates).

When SMI-S clients and servers use certificate revocation lists (CRL), they shall uses X.509 version 2 CRLs that are conformant with the CRL and CRL Extension Profile defined in Section 5 of IETF RFC 3280.

When PKI certificates and self-signed certificates are used together in a single management domain, it is important to recognize that the level of security is lowered to that afforded by self-signed certificates.

42.2.2.4.2 Allow customers to enforce their own certificate usage and acceptance policies.

All certificates identifying SMI-S management entities and their associated private keys shall be replaceable. SMI-S clients and servers shall either 1) have the ability to import an externally generated certificate and corresponding private key or 2) have the ability to generate and install a new self-signed certificate along with its corresponding private key.

When PKI certificates are used by SMI-S clients and servers, the implementations shall include the ability to import, install/store, and remove the CA root certificates; support for multiple trusted issuing CAs shall be included. CA certificates are used to verify that a certificate has been signed by a key from an acceptable certification authority.

To facilitate the use of certificates, SMI-S implementations should include configurable mechanisms that allow for one of the following mutually exclusive operating modes to be in force at any point in time for end-entity certificates (i.e., not CA certificates):

- Unverifiable end-entity (self-signed) certificates are automatically installed as trust anchors when they are presented; such certificates shall be determined to not be CA root certificates prior to being installed as trust anchors and shall not serve as trust anchors to verify any other certificates. If a CA certificate is presented as an end-entity certificate in this mode, it shall be rejected. For SMI-S clients, a variant of this option, which consults the user before taking action, should be implemented and used when possible.

  Note: The use of this operating mode should be limited to a learning or enrollment period during which communication is established with all other SMI-S systems with which security communication is desired. Use of a timeout to force automatic exit from this mode is recommended.

- Unverifiable end-entity (self-signed) certificates can be manually imported and installed as trust anchors (in a fashion similar to manually importing and installing a CA root certificate), but they are not automatically added when initially encountered. Administrative privilege may be required to import and install an end-entity certificate as a trust anchor. NOTE: This is considered the normal operating mode.

All certificate acceptance policies for SMI-S clients and servers shall be configurable. The configurable mechanisms determine how the SMI-S implementation handles presented certificates. Under normal operating mode, SMI-S servers should not accept certificates from unknown trust authorities (i.e., the CA root certificate has not been installed).

When self-signed certificates are used in conjunction with SLPv2, the trustworthiness of these certificates becomes an important factor in preventing SLPv2 from becoming an attack vector.

42.2.2.4.2.1 Default to facilitating interoperability where not specifically disallowed by security policy.

Interactive clients should provide a means to query the user about acceptance of a certificate from an unrecognized certificate authority (no corresponding CA root certificate installed in client), and accept responses allowing use of the certificate presented, or all certificates from the issuing CA. Servers should not support acceptance of unrecognized certificates; it is expected that a limited number of CAs will be acceptable for client certificates in any site that uses them.

Pre-configuring root certificates from widely used CAs is OPTIONAL, but simplifies initial configuration of certificate-based security, as certificates from those CAs will be accepted. These CA root certificates can be exported from widely available web browsers.

42.2.2.4.3 Require support for certificate acquisition from and revocation by common PKI/CA software.

All interfaces for certificate configuration in b and c of 42.2.2.4 shall support the following certificate formats:

- DER encoded X.509
  International Telecommunications Union Telecommunication Standardization Sector (ITU-T), Recommendation X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks, May 2000.
  Specification and technical corrigenda can be obtained from:
  http://www.itu.int/ITU-T/publications/recs.html;

- Base64 encoded X.509 (often called PEM)
  N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, IETF RFC 2045, November 1996, Section 6.8.
  Available at: http://www.ietf.org/rfc/rfc2045.txt;

- PKCS#12
  RSA Laboratories, PKCS #12: Personal Information Exchange Syntax, Version 1.0, June 1999. Specification and Technical Corrigendum. Available at:
  http://www.rsasecurity.com/rsalabs/pkcs/pkcs-12/index.html.

All certificate validation software shall support local certificate revocation lists, and at least one list per CA root certificate supported. Support is REQUIRED for both DER encoded X.509 and Base64 encoded X.509 formats, but this support MAY be provided by using one format in the software and providing a tool to convert lists from the other format. OCSP and other means of immediate online verification of certificate validity are OPTIONAL, as connectivity to the issuing Certificate Authority cannot be assured.

42.2.2.4.4 Allow security policy control to be restricted to security administrators.

All certificate interfaces required above shall support access restrictions that permit access only by suitably privileged administrators. A suitably privileged security administrator shall be able to disable functionality for acceptance of unrecognized certificates described in 42.2.2.4.3.

The above requirements can be satisfied via appropriate use of the readily-available OpenSSL toolkit software (www.openssl.org). Support for PKCS#7 certificate format was deliberately omitted from the requirements. This format is primarily used for online interaction with certificate authorities; such functionality is not appropriate to require of all SMI-S storage management software, and tools are readily available to convert PKCS#7 certificates to or from other certificate formats.

42.3 Health and Fault Management

Not defined in this standard.

42.4 Cascading Considerations

Not defined in this standard.

42.5 Supported Subprofiles and Packages

Table 442 describes the supported profiles for Server.

Table 442 - Supported Profiles for Server

Registered Profile Names	Mandatory	Version
Object Manager Adapter	No	1.3.0
Indication	No	1.3.0
Profile Registration	Yes	1.3.0

42.6 Methods of the Profile

None.

42.7 Client Considerations and Recipes

42.7.1 Applicability of Security Considerations

The security requirements for HTTP implementation given in 42.2.2, "HTTP Security" apply to both SMI-S servers and clients. An SMI-S client shall comply with all security requirements for HTTP specified in 42.2.1, "HTTP Security Background" that are applicable to clients.

SMI-S Client support for HTTP security is *required*. This includes the following requirements applicable to clients:

- SSL 3.0 and TLS shall be supported.
- HTTP Basic Authentication shall be supported. HTTP Digest Authentication should be supported.
- HTTP Realms shall be supported.
- All certificates, including CA Root Certificates used by clients for certificate validation, shall be replaceable.
- The DER encoded X.509, Base64 encoded X.509 and PKCS#12 certificate formats shall be supported.
- Certificate Revocation Lists shall be supported in the DER encoded X.509 and Base64 encoded X.509 formats.

The above list is not comprehensive; see 42.2.2, "HTTP Security" for the complete requirements. If there is any conflict between this text and 42.2.2, "HTTP Security", the text in 42.2.2, "HTTP Security" is the final specification of the requirements.

42.7.2 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.
```

42.8 Registered Name and Version

Server version 1.3.0

42.9 CIM Elements

Table 442 describes the CIM elements for Server.

Table 443 - CIM Elements for Server

Element Name	Requirement	Description
42.9.1 CIM_CIMXMLCommunicationMechanism	Mandatory	
42.9.2 CIM_CommMechanismForManager	Mandatory	This associates the ObjectManager and the communication classes it supports
42.9.3 CIM_HostedAccessPoint	Mandatory	This associates the communication mechanisms with the hosting System
42.9.4 CIM_HostedService	Mandatory	Connects the ObjectManager to the System that is hosting the ObjectManager.
42.9.5 CIM_Namespace	Mandatory	This is a namespace within the Object Manager.

Table 443 - CIM Elements for Server

Element Name	Requirement	Description
42.9.6 CIM_NamespaceInManager	Mandatory	This associates the namespace to the ObjectManager.
42.9.7 CIM_ObjectManager	Mandatory	This is the Object Manager service of the CIM Server.
42.9.8 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
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ObjectManager AND SourceInstance.CIM_ObjectManager::Started <> PreviousInstance.CIM_ObjectManager::Started	Optional	CQL -Start of object manager

42.9.1 CIM_CIMXMLCommunicationMechanism

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

Table 444 describes class CIM_CIMXMLCommunicationMechanism.

Table 444 - SMI Referenced Properties/Methods for CIM_CIMXMLCommunicationMechanism

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
ElementName		Mandatory	
CommunicationMechanism		Mandatory	
Version		Mandatory	'CIM Operations over HTTP' version. Shall be '1.0' or '1.1' or '1.2'.

Table 444 - SMI Referenced Properties/Methods for CIM_CIMXMLCommunicationMechanism

Properties	Flags	Requirement	Description & Notes
CIMValidated		Mandatory	
FunctionalProfilesSupported		Mandatory	
MultipleOperationsSupported		Mandatory	
AuthenticationMechanismsSupported		Mandatory	
OtherCommunicationMechanismDescription		Conditional	Conditional requirement: CIM_CIMXMLCommunicationMechanism requires the unicationMechanismDescription property be populated if the CommunicationMechanism property has a value of 1 (\Other\')..'This shall not be NULL if 'Other' is identified in CommunicationMechanism
OperationalStatus		Mandatory	
StatusDescriptions		Optional	
FunctionalProfileDescriptions		Optional	

42.9.2 CIM_CommMechanismForManager

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

Table 445 describes class CIM_CommMechanismForManager.

Table 445 - SMI Referenced Properties/Methods for CIM_CommMechanismForManager

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

42.9.3 CIM_HostedAccessPoint

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

Table 446 describes class CIM_HostedAccessPoint.

Table 446 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

42.9.4 CIM_HostedService

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

Table 447 describes class CIM_HostedService.

Table 447 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

42.9.5 CIM_Namespace

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

Table 448 describes class CIM_Namespace.

Table 448 - SMI Referenced Properties/Methods for CIM_Namespace

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

Table 448 - SMI Referenced Properties/Methods for CIM_Namespace

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
ClassType		Mandatory	
DescriptionOfClassType		Conditional	Conditional requirement: CIM_Namespace requires the DescriptionOfClassType property be populated if the ClassType property has a value of 1 (\Other\')..'Mandatory if ClassType is set to 1 ('Other')
ClassInfo		Optional	Deprecated.Deprecated in the MOF, but required for 1.0 compatibility. Not required if all hosted profiles are new in 1.1
DescriptionOfClassInfo		Optional	Deprecated.Deprecated in the MOF, but mandatory for 1.0 compatibility. Mandatory if ClassInfo is set to 'Other'

42.9.6 CIM_NamespaceInManager

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

Table 449 describes class CIM_NamespaceInManager.

Table 449 - SMI Referenced Properties/Methods for CIM_NamespaceInManager

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

42.9.7 CIM_ObjectManager

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

Table 450 describes class CIM_ObjectManager.

Table 450 - SMI Referenced Properties/Methods for CIM_ObjectManager

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
ElementName		Mandatory	
Description		Mandatory	
OperationalStatus		Mandatory	
Started		Mandatory	
StopService()		Optional	

42.9.8 CIM_System

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

Table 451 describes class CIM_System.

Table 451 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	
Description		Mandatory	
ElementName		Mandatory	
OperationalStatus		Mandatory	
NameFormat		Mandatory	

STABLE

EXPERIMENTAL

Clause 43: Profile Registration Profile

43.1 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.

43.2 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

43.3 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 60 shows the Profile Registration Model.

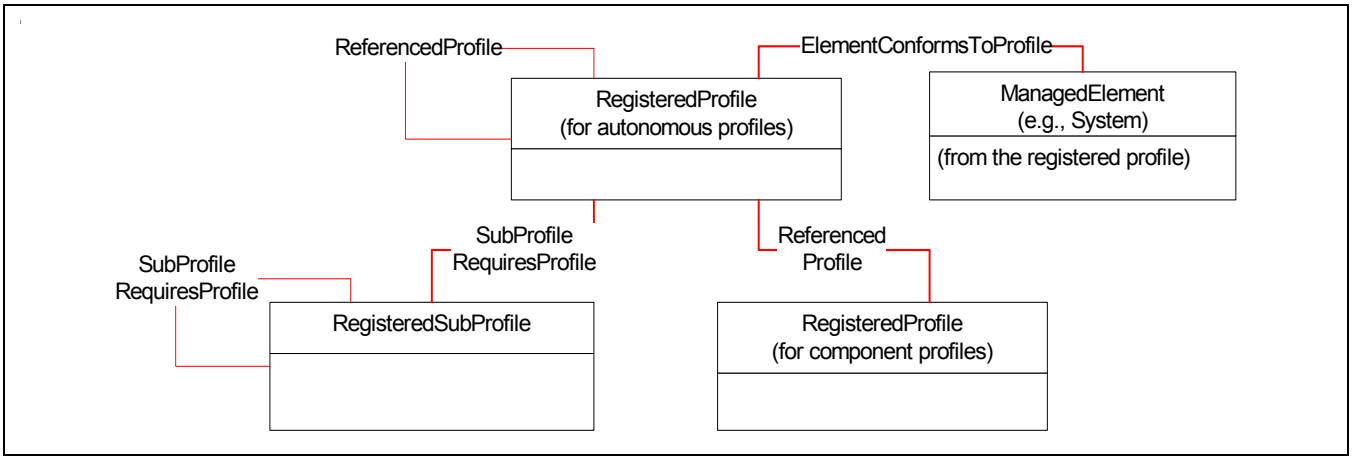

Figure 60 - Profile Registration Model

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.

43.3.1 ElementConformsToProfile Association

In addition, the ElementConformsToProfile association ties the “top-level” autonomous profile (RegisteredProfile) to scoping managed elements.

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 Rules for Combining (Autonomous) Profiles (B.5) in *Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6*.

43.3.2 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 61, the Array profile suports 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.

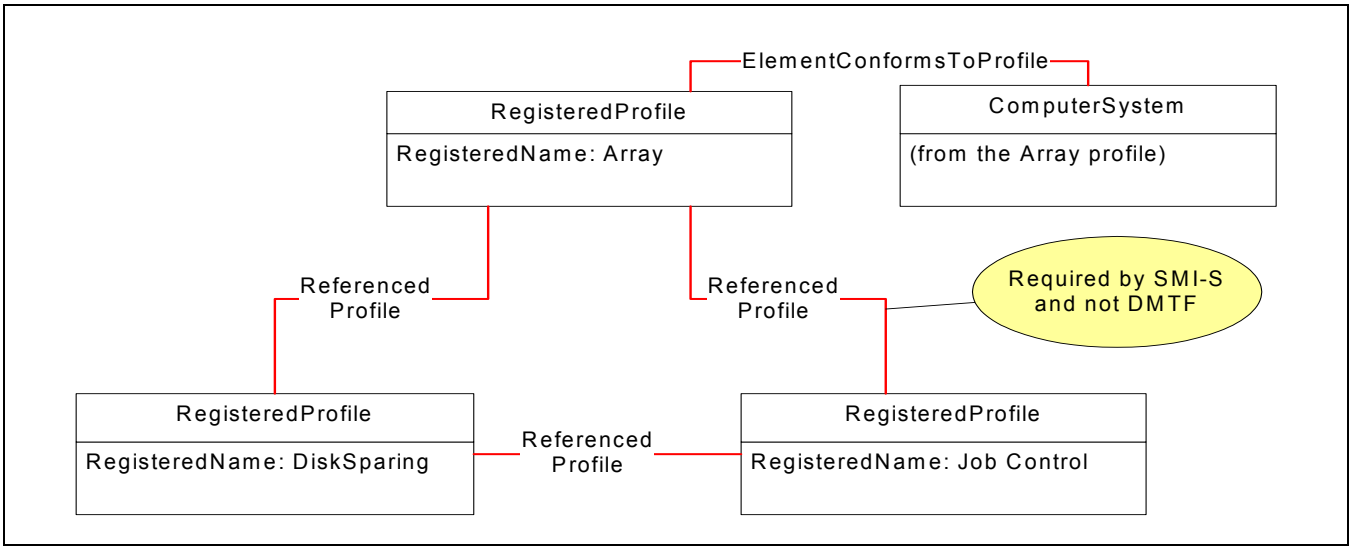

Figure 61 - Associations between RegisteredProfile instances

Each RegisteredProfile instance referenced by ElementConformsToProfile may have a set of supported profiles with RegisteredProfile instances associated using ReferencedProfile or SubProfileReuiredProfile. 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 implementaions shall support all the profiles supported by the RegisteredProfile.

43.3.3 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 62. 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 62 depicts the RegisteredProfile representing the SMI-S standard on the left, and RegisteredProfiles representing autonomous and component storage profiles in the middle.

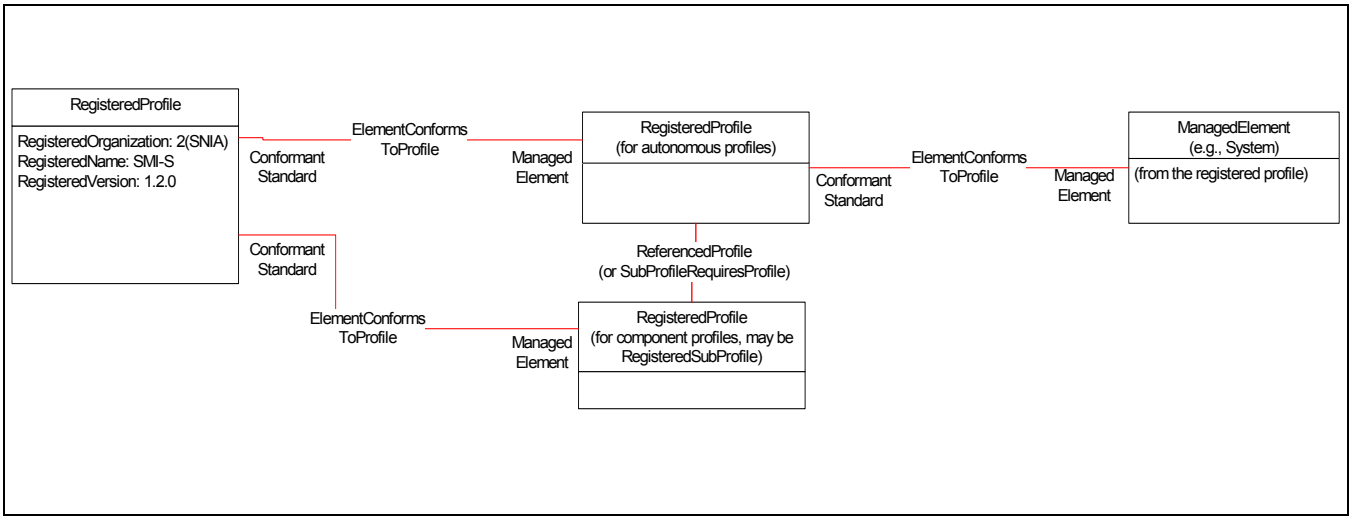

Figure 62 - Model for SMI-S Registered Profile

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 62. The role names are under the ends of the ElementConformsToProfile lines.

43.3.3.1 Provider Versions

Each RegisteredProfile and RegisteredSubprofile instance (from the Profile Registration 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 63.

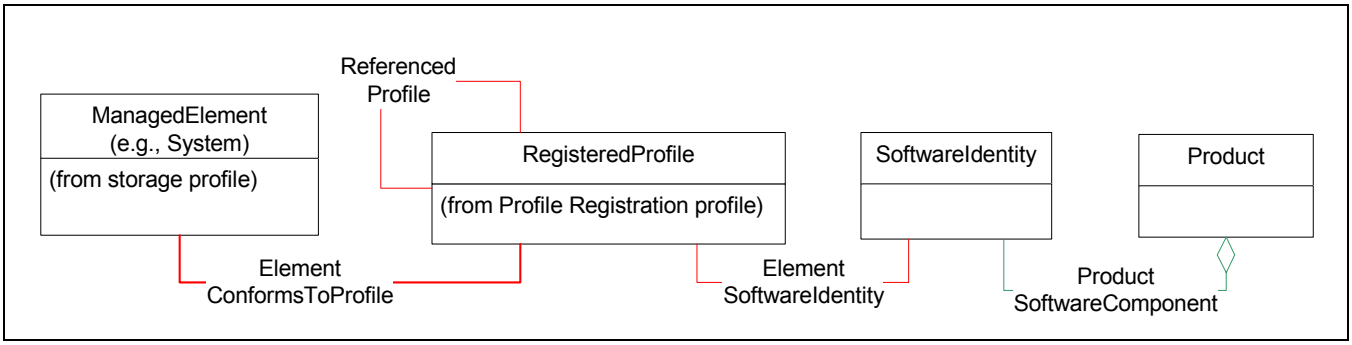

Figure 63 - Model for Provider Versions

43.3.3.2 Abstract Profile and Profile Registration

When profiles are defined for specialization, they may be defined as abstract and include this text in the Synposis 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 instaces shall not be instantiated for abstract profiles. Information about abstract profiles shall not be included in the SLP template.

43.3.4 Health and Fault Management Consideration

None

43.3.5 Cascading Considerations

None

43.4 Methods

None

43.5 Use Cases

43.5.1 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.

43.5.2 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.

43.5.3 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.
```

43.5.4 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
```

43.5.5 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()
```

43.5.6 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
}
```

43.5.7 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)
}
```

43.6 Registered Name and Version

Profile Registration version 1.3.0

43.7 CIM Elements

Table 452 describes the CIM elements for Profile Registration.

Table 452 - CIM Elements for Profile Registration

Element Name	Requirement	Description
43.7.1 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
43.7.2 CIM_ElementConformsToProfile (Associates RegisteredProfiles for SMI-S and domain profiles)	Mandatory	Associates RegisteredProfiles for SMI-S and domain profiles
43.7.3 CIM_ElementSoftwareIdentity (Profile and SW identity)	Mandatory	Associates a domain RegisteredProfile and SoftwareIdentity instances
43.7.4 CIM_ElementSoftwareIdentity (Subprofile and SW identity)	Conditional	Conditional requirement: Support for instances of RegisteredSubprofile.Associates the subprofile and SoftwareIdentity instances.
43.7.5 CIM_Product	Optional	Represents a software product aggregating SoftwareIdentity instaces with provider versions.
43.7.6 CIM_ProductSoftwareComponent	Optional	Associates Product and SoftwareIdentity.
43.7.7 CIM_ReferencedProfile	Optional	Associates referenced profiles using the DMTF Profile Registration profile
43.7.8 CIM_RegisteredProfile (Domain Registered Profile)	Mandatory	An object representing a domain (e.g. Array or Switch) profile.
43.7.9 CIM_RegisteredProfile (The SMI-S Registered Profile)	Mandatory	A registered profile that provides the version of the SMI-S standard

Table 452 - CIM Elements for Profile Registration

Element Name	Requirement	Description
43.7.10 CIM_RegisteredSubProfile	Optional	Specialization of RegisteredProfile for legacy SMI-S subprofiles.
43.7.11 CIM_SoftwareIdentity	Mandatory	A representation of some bundle of providers and supporting software that shares a version number.
43.7.12 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
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_RegisteredProfile	Optional	Deletion of a registered profile instance

43.7.1 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 453 describes class CIM_ElementConformsToProfile (Associates Domain object (e.g. System) to RegisteredProfile).

Table 453 - 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.

43.7.2 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 454 describes class CIM_ElementConformsToProfile (Associates RegisteredProfiles for SMI-S and domain profiles).

Table 454 - 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

43.7.3 CIM_ElementSoftwareIdentity (Profile and SW identity)

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

Table 455 describes class CIM_ElementSoftwareIdentity (Profile and SW identity).

Table 455 - 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 RegegisteredProfile.

43.7.4 CIM_ElementSoftwareIdentity (Subprofile and SW identity)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for instances of RegisteredSubprofile.

Table 456 describes class CIM_ElementSoftwareIdentity (Subprofile and SW identity).

Table 456 - 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

43.7.5 CIM_Product

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

Table 457 describes class CIM_Product.

Table 457 - 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

43.7.6 CIM_ProductSoftwareComponent

Associates Product and SoftwareIdentity.

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

Table 458 describes class CIM_ProductSoftwareComponent.

Table 458 - SMI Referenced Properties/Methods for CIM_ProductSoftwareComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to Product
PartComponent		Mandatory	Reference to SoftwareIdentity

43.7.7 CIM_ReferencedProfile

Associates referenced profiles using the DMTF Profile Registration profile

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

Table 459 describes class CIM_ReferencedProfile.

Table 459 - SMI Referenced Properties/Methods for CIM_ReferencedProfile

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

43.7.8 CIM_RegisteredProfile (Domain Registered Profile)

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

Table 460 describes class CIM_RegisteredProfile (Domain Registered Profile).

Table 460 - 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

43.7.9 CIM_RegisteredProfile (The SMI-S Registered Profile)

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

Table 461 describes class CIM_RegisteredProfile (The SMI-S Registered Profile).

Table 461 - 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.

43.7.10 CIM_RegisteredSubProfile

Specialization of RegisteredProfile for legacy SMI-S subprofiles.

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

Table 462 describes class CIM_RegisteredSubProfile.

Table 462 - 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').

Table 462 - SMI Referenced Properties/Methods for CIM_RegisteredSubProfile

Properties	Flags	Requirement	Description & Notes
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.
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.

43.7.11 CIM_SoftwareIdentity

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

Table 463 describes class CIM_SoftwareIdentity.

Table 463 - 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').

43.7.12 CIM_SubProfileRequiresProfile

Specialization of ReferencedProfile referencing a SubProfile

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

Table 464 describes class CIM_SubProfileRequiresProfile.

Table 464 - SMI Referenced Properties/Methods for CIM_SubProfileRequiresProfile

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to a RegisteredSubProfile.
Antecedent		Mandatory	Reference to a RegisteredProfile

EXPERIMENTAL

STABLE

Clause 44: Indication Profile

44.1 Description

Indications are support for event notifications. Each profile that supports event notification through CIM indications would support this profile and its classes and associations.

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).

Note: Refer to individual profile definitions to see whether or not the Indication Profile is mandatory. Figure 64 illustrates the structure of profiles, the Indication Profile and indication instances implied by an Array's support for the Indication Profile.

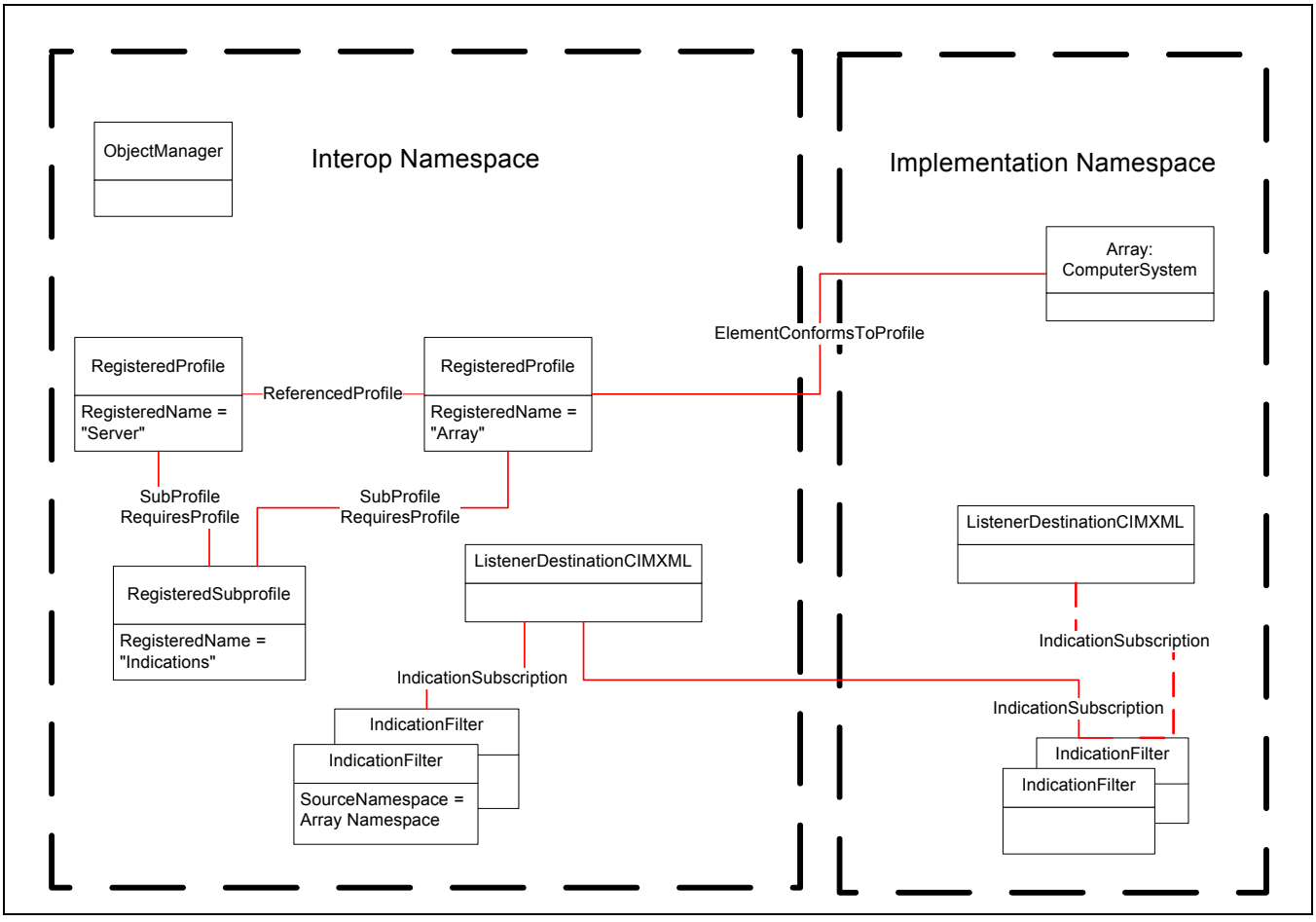

Figure 64 - Indication Profile and Namespaces

Indication filters are defined in the context of the namespace in which they are implemented. In Figure 64, 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. 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).

The RegisteredProfile for the Array is associated to the ComputerSystem that is the top level system for the Array. This is done via the ElementConformsToProfile association, which is a cross namespace association (populated by the provider). The IndicationFilters may also 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 “source” implementation namespace. And there would be two instantiations of the IndicationSubscription association: one in the Interop Namespace and one in the implementation namespace of the array.

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 “Source” implementation namespace. ListenerDestinationCIMXML instances should be created in the InteropNamespace, but may also be created in the “Source” implementation namespace (for SMI-S 1.0.x compatibility reasons). If a ListenerDestinationCIMXML instance is created in the “Source” 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 “source” 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.

44.1.1 Basic Indication Classes and Association

Figure 65 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 44.8 CIM Elements to determine the types of indications supported).

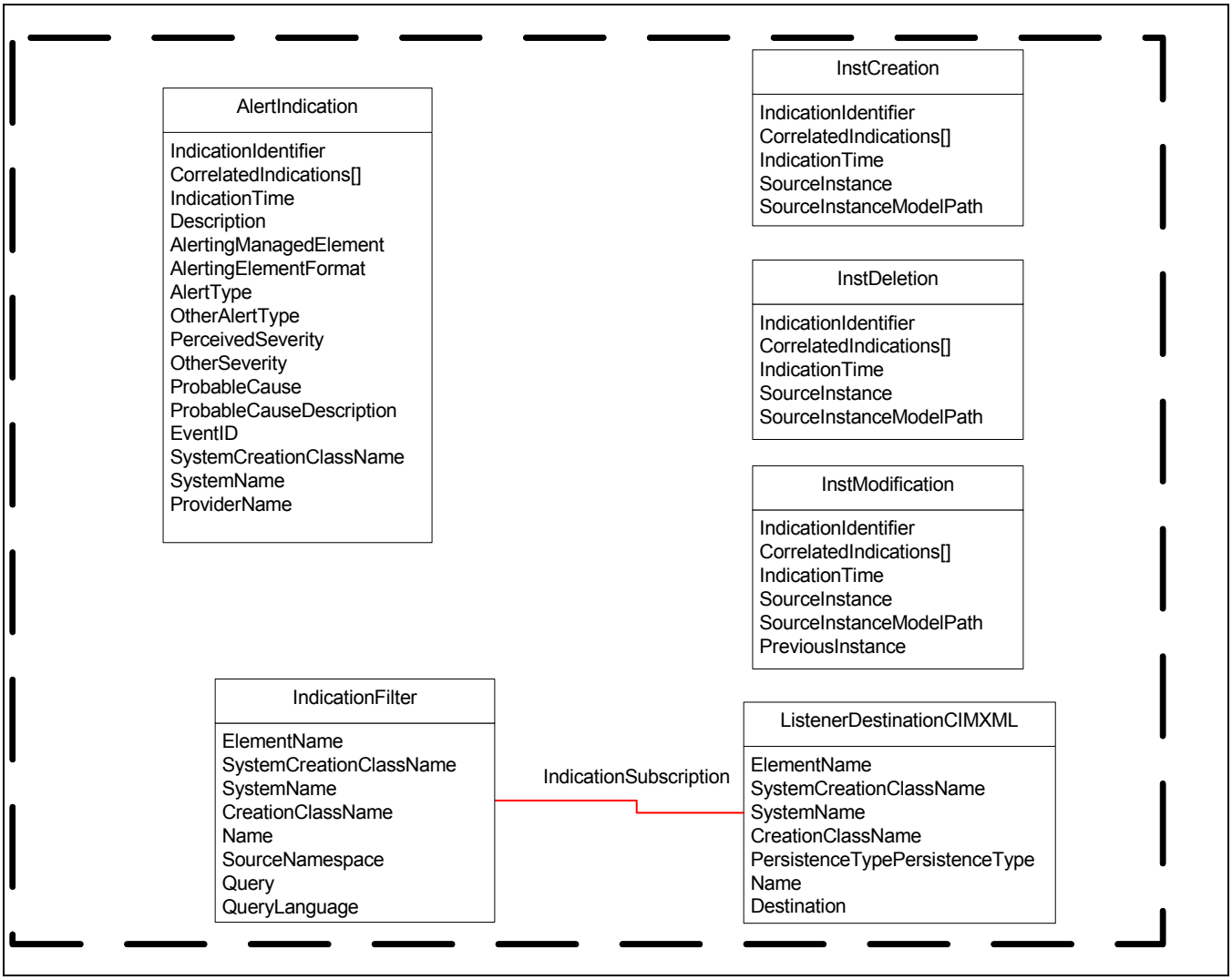

Figure 65 - Indication Profile Instance Diagram

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 1 Common Architecture, 1.3.0 Rev 6* 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.

EXPERIMENTAL

The scheme (protocol prefix) portion of that URL value defines the protocol to be used by the SMI-S Server (CIMOM) to deliver the indication. When the scheme (protocol prefix) is "https:", the SMI-S Server shall connect using SSL or TLS when delivering the indication to the destination. General Requirements i), j), and k) in 42.2.2.1, "General Requirements" shall not apply to indication delivery because the URL specifies the protocol to use. When the "https:" protocol scheme is used, the Indication Listener should have a certificate that it will use as the SSL or TLS server certificate; if the Indication Listener does not have such a certificate, SSL or TLS are forced to use Anonymous cipher suites and no assurance can be provided that the indication was delivered to the intended destination due to the lack of authentication of the Listener end of the secure channel.

EXPERIMENTAL

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.

44.1.2 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.

44.1.3 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 1 Common Architecture, 1.3.0 Rev 6* Clause 9: 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:

- IndicationIdentifier - An identifier for the Indication that can be used for identification when correlating Indications (see the CorrelatedIndications array).
- 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.

EXPERIMENTAL

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.

EXPERIMENTAL

- 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:

- CorrelatedIndications[]
- 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 9.3 in the *Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6*.

44.1.4 Special handling for Multiple events of the same type

When a client creates a subscription (using CreateInstance), the provider may fill in the RepeatNotificationPolicy and related properties. This information describes the policy used by the implementation for reporting multiple events of the same type (multiple events for the subscription). If the RepeatNotificationPolicy is “None”, then the client will receive all indications. If the RepeatNotificationPolicy is “Suppress”, then all indications after the first ‘n’ (where ‘n’ is defined by the RepeatNotificationCount) are not sent (within the RepeatNotificationInterval time). If the RepeatNotificationPolicy is “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.

44.1.5 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.

44.1.6 Instrumentation Requirements

44.1.6.1 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.

EXPERIMENTAL

44.1.6.2 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, shown in Figure 66 for the IndicationIdentifier property 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.

```
<Trademark>:<Object Name of IndicationFilter>#<production sequence number>#<delivery sequence number>
```

Figure 66 - 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 218 (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

44.1.6.3 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.

44.1.6.4 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.

44.1.6.5 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 1 Common Architecture, 1.3.0 Rev 6* for information on the use of indications filter strings.

44.1.6.6 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.

44.1.6.7 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 any one of three techniques:

- use the RepeatNotificationPolicy (and related properties) of the IndicationSubscription.

EXPERIMENTAL

- Use of Bellwether events (if they are defined by the profile) (EXPERIMENTAL)
- Use of batching (EXPERIMENTAL)

EXPERIMENTAL

44.1.6.8 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), 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.

If the value of RepeatNotificationPolicy is 4 (Delay) and an Indication is received, this Indication shall be suppressed if, including this Indication, RepeatNoticationCount 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 IndictationFilter). 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.

EXPERIMENTAL

44.1.6.9 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.

44.1.6.10 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.

EXPERIMENTAL

44.1.6.11 Clarification of indication generation

44.1.6.11.1 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.

44.1.6.11.2 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.

44.1.6.11.3 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.

44.1.6.11.4 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.

EXPERIMENTAL

44.1.6.12 HTTP Security

HTTP security shall be implemented for Indications as specified in 42.2.2, "HTTP Security" with additional requirements specified in this section. For applying the requirements in 42.2.2, "HTTP Security" to Indications, the term "SMI-S Client" shall be read to mean "any SMI-S entity that can function as an Indication Listener." HTTP security support for Indications is a mandatory part of Indications support for SMI-S Servers. An SMI-S Client that does not support certificates may omit SSL/TLS support for reception of Indications, but shall comply with all other requirements. The following security requirements based on 42.2.2, "HTTP Security" apply to Indications:

- SSL 3.0 and TLS shall be supported.
- HTTP Basic Authentication shall be supported. HTTP Digest Authentication should be supported.
- HTTP Realms shall be supported.
- SMI-S Servers shall support certificates, SMI-S Clients may support certificates. This includes Indication functionality in both cases.
- All certificates, including CA Root Certificates used for certificate validation, shall be replaceable.
- The DER encoded X.509, Base64 encoded X.509 and PKCS#12 certificate formats shall be supported.
- Certificate Revocation Lists shall be supported in the DER encoded X.509 and Base64 encoded X.509 formats.

The above list is not comprehensive - see 42.2.2, "HTTP Security" for the complete requirements. If there is any conflict between the above list and 42.2.2, the text in 42.2.2 is the final specification of the requirements. In addition the remainder of this section states additional requirements, some of which modify the requirements in 42.2.2.1, "General Requirements".

Determination of whether to use SSL/TLS is based on the scheme of the URL in the ListenerDestinationCIMXML property of the indication - see 44.1.1. General Requirements i), j), and k) in 42.2.2.1, "General Requirements" shall not apply to indication delivery because the URL specifies the protocol to use.

The SSL/TLS roles of client and server for Indication delivery are reversed for Indications; the SMI-S Server (CIMOM) is the SSL/TLS client and the Indication Listener (e.g., management application) is the SSL/TLS server. Hence for indications, the certificate support requirements are:

- SMI-S Servers shall support certificates for sending Indications. These are SSL/TLS client certificates.
- SMI-S Clients that function as Indication Listeners may support certificates for receiving Indications. SMI-S Servers that can function as Indication Listeners shall support certificates for receiving indications. These are SSL/TLS server certificates in both cases.

In order to use SSL or TLS for indication delivery, the Indication Listener is required to have a certificate; since the SMI-S Server should also have a certificate, mutual SSL/TLS Authentication is possible. SMI-S Servers should not use SSL or TLS for indication delivery when the Indication Listener does not present a certificate, and shall support a configurable operating mode in which indication delivery is not performed via SSL or TLS when the Listener does not present a certificate. This can be accomplished by preventing the use of Anonymous SSL/TLS cipher suites.

Mutual authentication can be achieved in the two certificate case. All SMI-S entities shall use certificates consistently - the certificate used for CIM operation invocation over SSL/TLS shall be used for indication delivery when SSL/TLS is employed for indication delivery. For SMI-S Servers, this requires that the SSL/TLS server certificate used to receive CIM operations via SSL/TLS shall be provided as the SSL/TLS client certificate for

indication delivery when mutual authentication is used (i.e., when an anonymous SSL/TLS cipher suite is not used). For SMI-S Clients that support certificates and can function as Indication Listeners, this means that the SSL/TLS client certificate used for CIM operation invocation over SSL/TLS shall be used as the SSL/TLS server certificate for receiving indications.

EXPERIMENTAL

44.2 Health and Fault Management Considerations

44.2.1 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.

44.2.2 Health State Transformations and Dependencies

No Indications class have OperationalStatus or HealthState properties.

44.2.3 Standard Errors Produced

All manipulation of Indication classes and associations are done using intrinsic methods. The errors produced are those listed for intrinsic methods.

44.2.4 Cause and effect associations

Cause and effect associations are defined as part of the Health and Fault Management Package.

EXPERIMENTAL

44.2.5 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

44.3 Cascading Considerations

Not Applicable.

44.4 Supported Profiles, Subprofiles and Packages

Related Profiles for Indication: Not defined in this standard.

44.5 Methods of the Profile

44.5.1 Extrinsic Methods of the Profile

No extrinsics are specified on the Indication Profile.

44.5.2 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 shown in Table 465.

Table 465 - 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
 )
```

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 the “Source” namespace. If the client creates a ListenerDestinationCIMXML instance in the “Source” namespace, then a duplicate ListenerDestinationCIMXML instance will be created 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 another namespace (there is nothing that would indicate which Namespace would be the Source namespace).

IndicationFilters shall be created in either the Interop Namespace or the 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 namespace).

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).

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.

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)

44.6 Client Considerations and Recipes

44.6.1 Use of Profile Specific Recipes

See Recipes in related profile sections.

44.6.2 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.

44.6.3 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).

44.6.4 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”.

44.6.5 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])
        }
     }
}
```

44.6.6 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
```

44.6.7 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
```

44.6.8 Subscription 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
```

44.6.9 Listenable 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
```

44.6.10 Subscribe 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
```

44.7 Registered Name and Version

Indication version 1.3.0

44.8 CIM Elements

Table 466 describes the CIM elements for Indication.

Table 466 - CIM Elements for Indication

Element Name	Requirement	Description
44.8.1 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.
44.8.2 CIM_IndicationFilter (client defined)	Optional	This is for 'client defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the criteria for generating an Indication and what data should be returned in the Indication.
44.8.3 CIM_IndicationFilter (pre-defined)	Optional	This is for 'pre-defined' CIM_IndicationFilter instances. CIM_IndicationFilter defines the criteria for generating an Indication and what data should be returned in the Indication.
44.8.4 CIM_IndicationSubscription	Mandatory	This association defines a subscription to a specific IndicationFilter instance by a specific indication handler (as represented by a ListenerDestinationCIMXML instance).
44.8.5 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).
44.8.6 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).
44.8.7 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).
44.8.8 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.

44.8.1 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 467 describes class CIM_AlertIndication.

Table 467 - SMI Referenced Properties/Methods for CIM_AlertIndication

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		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	Recommendation.ITU|X733.Additional text
AlertingManagedElement		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	
SystemCreationClassName		Mandatory	
SystemName		Mandatory	The scoping System's Name for the Provider generating this Indication. The SystemName would typically be the same name that for a system in the Implementation Namespace (unless the Indication is an indication generated for Server Profile).

Table 467 - SMI Referenced Properties/Methods for CIM_AlertIndication

Properties	Flags	Requirement	Description & Notes
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.

44.8.2 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 468 describes class CIM_IndicationFilter (client defined).

Table 468 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

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

Table 468 - SMI Referenced Properties/Methods for CIM_IndicationFilter (client defined)

Properties	Flags	Requirement	Description & Notes
SourceNamespace	N	Optional	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.
Query		Mandatory	
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.

44.8.3 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 469 describes class CIM_IndicationFilter (pre-defined).

Table 469 - SMI Referenced Properties/Methods for CIM_IndicationFilter (pre-defined)

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

Table 469 - SMI Referenced Properties/Methods for CIM_IndicationFilter (pre-defined)

Properties	Flags	Requirement	Description & Notes
SourceNamespace	N	Optional	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.
Query		Mandatory	
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	This should be NULL for pre-defined indication filters.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

44.8.4 CIM_IndicationSubscription

A CIM_IndicationSubscription is not subclassed from anything.

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

Table 470 describes class CIM_IndicationSubscription.

Table 470 - 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

Table 470 - SMI Referenced Properties/Methods for CIM_IndicationSubscription

Properties	Flags	Requirement	Description & Notes
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.
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	

44.8.5 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 471 describes class CIM_InstCreation.

Table 471 - SMI Referenced Properties/Methods for CIM_InstCreation

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		Mandatory	An identifier for the Indication used for correlated indications.
CorrelatedIndications		Optional	IndicationIdentifiers whose notifications are correlated with this one.

Table 471 - SMI Referenced Properties/Methods for CIM_InstCreation

Properties	Flags	Requirement	Description & Notes
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.

44.8.6 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 472 describes class CIM_InstDeletion.

Table 472 - SMI Referenced Properties/Methods for CIM_InstDeletion

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		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.

44.8.7 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 473 describes class CIM_InstModification.

Table 473 - SMI Referenced Properties/Methods for CIM_InstModification

Properties	Flags	Requirement	Description & Notes
IndicationIdentifier		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.
OtherSeverity	N	Optional	Not Specified in this version of the Profile.
SourceInstanceHost	N	Optional	Not Specified in this version of the Profile.

44.8.8 CIM_ListenerDestinationCIMXML (Indication Handler)

CIM_ListenerDestinationCIMXML is subclassed from CIM_ListenerDestination.

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

Table 474 describes class CIM_ListenerDestinationCIMXML (Indication Handler).

Table 474 - 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.

STABLE

STABLE

Clause 45: Object Manager Adapter Subprofile

45.1 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 the “required elements” table.

45.1.1 Instance Diagram

ObjectManagerAdapter subprofile is not advertised. Figure 67 illustrates the model.

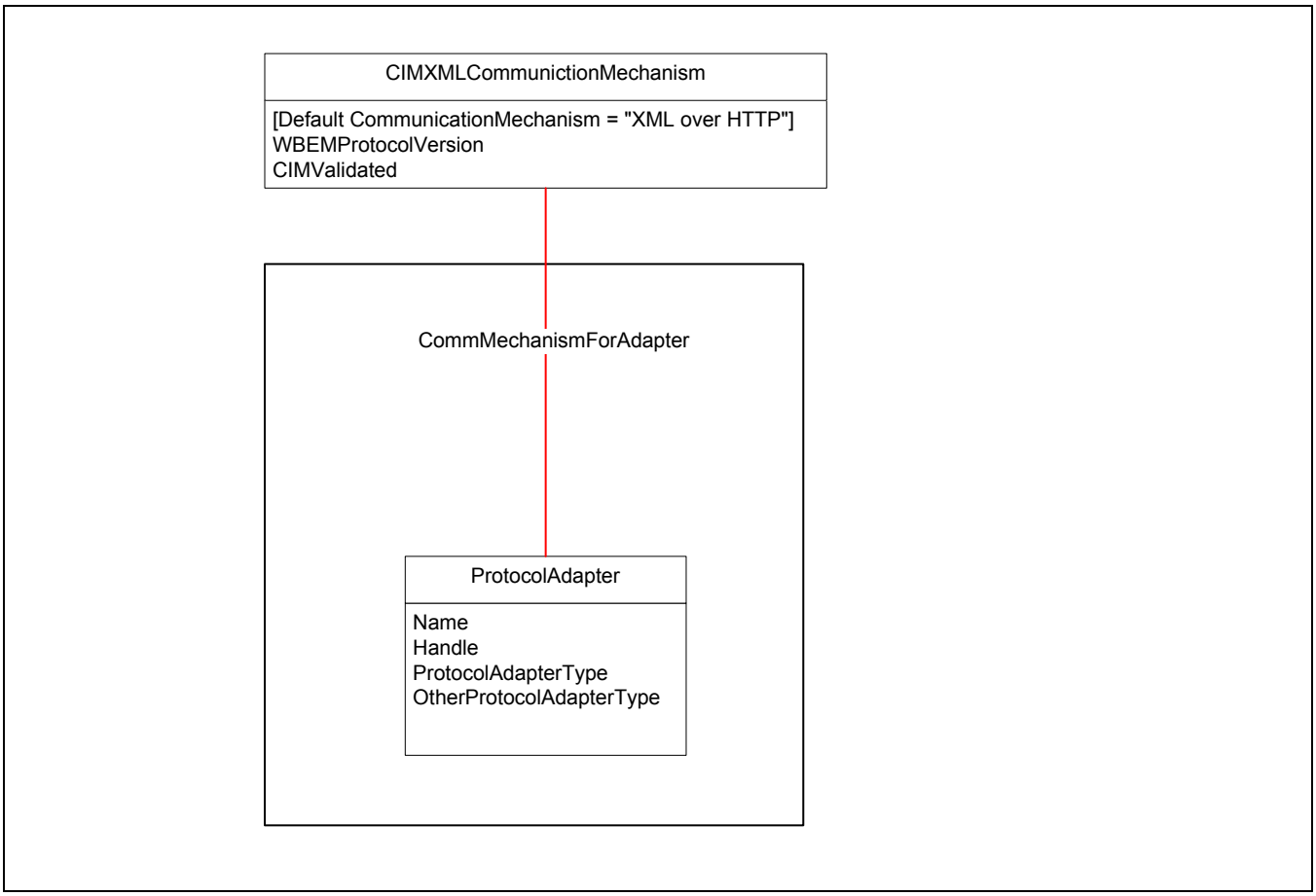

Figure 67 - ObjectManagerAdapter Subprofile Model

45.2 Health and Fault Management

Not defined in this standard.

45.3 Cascading Considerations

Not defined in this standard.

45.4 Supported Subprofiles and Packages

None.

45.5 Methods of the Profile

None.

45.6 Client Considerations and Recipes

None.

45.7 Registered Name and Version

Object Manager Adapter version 1.3.0

45.8 CIM Elements

Table 475 describes the CIM elements for Object Manager Adapter.

Table 475 - CIM Elements for Object Manager Adapter

Element Name	Requirement	Description
45.8.1 CIM_CommMechanismForObjectManagerAdapter	Mandatory	
45.8.2 CIM_ObjectManagerAdapter	Mandatory	

45.8.1 CIM_CommMechanismForObjectManagerAdapter

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

Table 476 describes class CIM_CommMechanismForObjectManagerAdapter.

Table 476 - 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.

45.8.2 CIM_ObjectManagerAdapter

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

Table 477 describes class CIM_ObjectManagerAdapter.

Table 477 - SMI Referenced Properties/Methods for CIM_ObjectManagerAdapter

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

Clause 46: Proxy Server System Management Subprofile

46.1 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 (Architecture book, pg124-125). Proxy servers 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 Embeded 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.

46.1.1 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.

46.1.2 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 68 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.

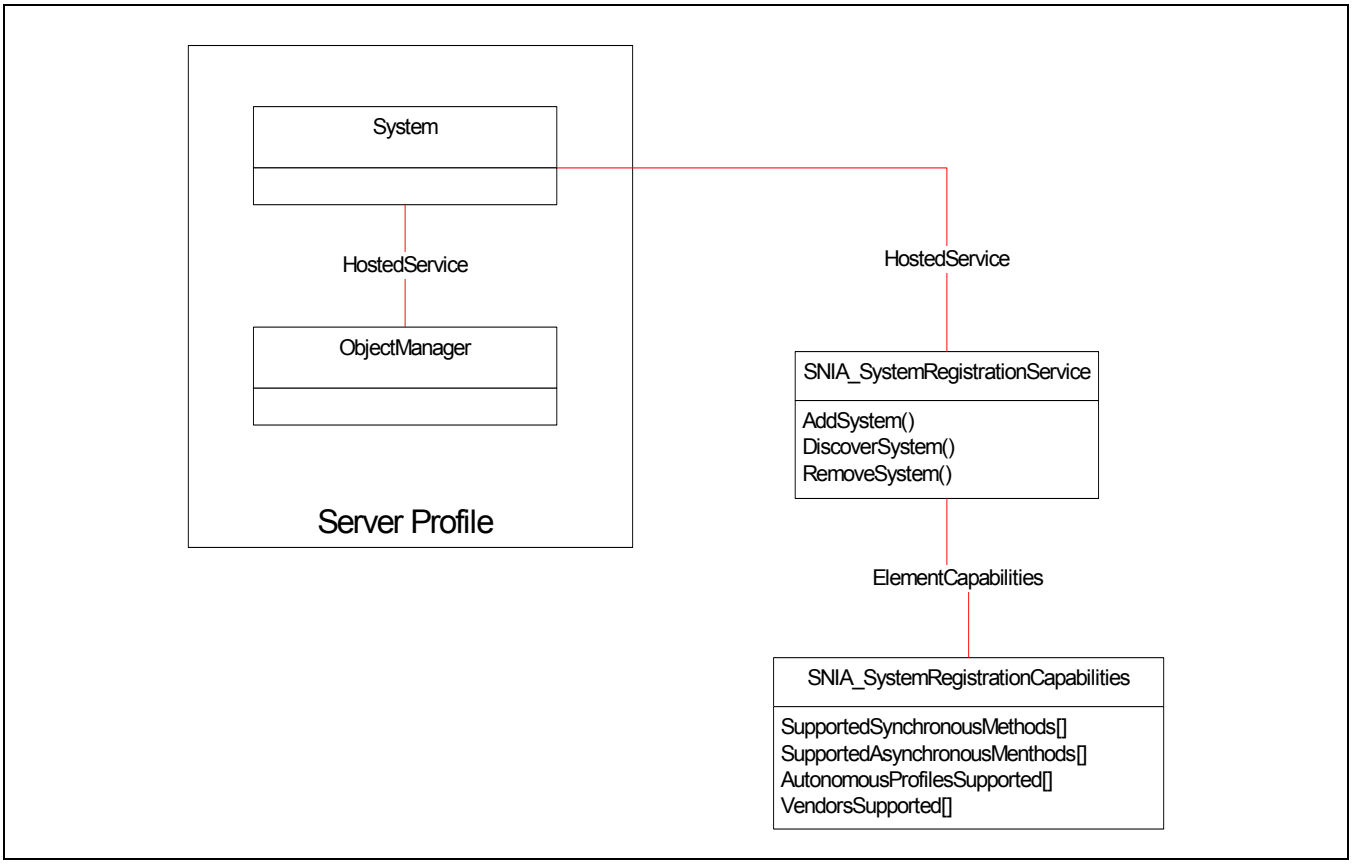

Figure 68 - Proxy Server System Management Model

Table 478 describes each associated capability.

Table 478 - 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

Table 478 - Capabilities

Capability	Description
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)

46.1.3 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.

46.2 Health and Fault Management Consideration

Not defined in this standard.

46.3 Cascading Considerations

Not applicable

46.4 Supported Profiles, Subprofiles, and Packages

Related Profiles for Proxy Server System Management: Not defined in this standard.

46.5 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.

46.5.1 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 Client Considerations for more details on what happens with 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 479 - AddSystem

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[]
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

Return codes:

Table 480 - 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

Table 480 - Return Codes

Value	Description
Device profile not supported	Device at the address specified does not support any of the automonous 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 credentails for device
4096: Method Parameters Checked - Job started	Job was started

46.5.2 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 Client Considerations 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 481 - DiscoverSystem

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 482 - 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 automonous 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 credentails for device
4096: Method Parameters Checked - Job started	Job was started

46.5.3 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 483 - RemoveSystem

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

46.6 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.

46.6.1 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

46.6.2 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

46.7 CIM Element

46.8 Registered Name and Version

Proxy Server System Management version 1.3.0

46.9 CIM Elements

Table 484 describes the CIM elements for Proxy Server System Management.

Table 484 - CIM Elements for Proxy Server System Management

Element Name	Requirement	Description
46.9.1 CIM_HostedService	Mandatory	
46.9.2 SNIA_SystemRegistrationCapabilities	Mandatory	
46.9.3 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)

46.9.1 CIM_HostedService

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

Table 485 describes class CIM_HostedService.

Table 485 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

46.9.2 SNIA_SystemRegistrationCapabilities

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

Requirement: Mandatory

Table 486 describes class SNIA_SystemRegistrationCapabilities.

Table 486 - SMI Referenced Properties/Methods for SNIA_SystemRegistrationCapabilities

Properties	Flags	Requirement	Description & Notes
AutonomousProfiles Supported		Mandatory	This property identifies the profiles that this service is capable of discovering and managing. For example, a block device could potentially list ‚ÄúArray‚Ä? and ‚ÄúStorage Virtualizer‚Ä?
VendorsSupported		Mandatory	This property identifies the vendors whose devices this service can discover. For example, if the list contains ‚ÄúVendor A‚Ä? then only ‚ÄúVendor A‚Ä? 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.

46.9.3 SNIA_SystemRegistrationService

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

Table 487 describes class SNIA_SystemRegistrationService.

Table 487 - SMI Referenced Properties/Methods for SNIA_SystemRegistrationService

Properties	Flags	Requirement	Description & Notes
AddSystem()		Mandatory	
DiscoverSystems()		Optional	
RemoveSystem()		Mandatory	

EXPERIMENTAL

STABLE

Clause 47: Device Credentials Subprofile

47.1 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.

47.1.1 Instance Diagram

Figure 69 provides a sample instance diagram.

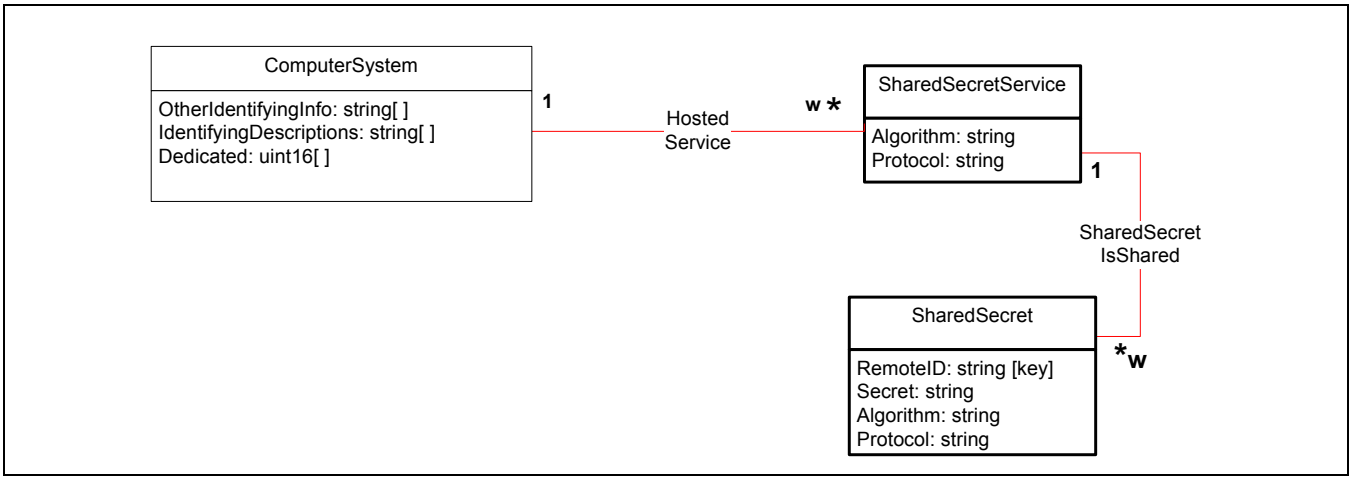

Figure 69 - DeviceCredentials Subprofile Model

47.2 Health and Fault Management Considerations

Not defined in this standard.

47.3 Cascading Considerations

Not defined in this standard.

47.4 Supported Subprofiles and Packages

Not defined in this standard.

47.5 Extrinsic Methods of this Profile

Not defined in this standard.

47.6 Client Considerations and Recipes

None.

47.7 Registered Name and Version

Device Credentials version 1.3.0

47.8 CIM Elements

Table 488 describes the CIM elements for Device Credentials.

Table 488 - CIM Elements for Device Credentials

Element Name	Requirement	Description
47.8.1 CIM_HostedService	Mandatory	
47.8.2 CIM_SharedSecret	Mandatory	
47.8.3 CIM_SharedSecretIsShared	Mandatory	
47.8.4 CIM_SharedSecretService	Mandatory	

47.8.1 CIM_HostedService

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

Table 489 describes class CIM_HostedService.

Table 489 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

47.8.2 CIM_SharedSecret

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

Table 490 describes class CIM_SharedSecret.

Table 490 - SMI Referenced Properties/Methods for CIM_SharedSecret

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
ServiceCreationClassName		Mandatory	
ServiceName		Mandatory	
RemoteID		Mandatory	
Secret		Mandatory	

47.8.3 CIM_SharedSecretIsShared

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

Table 491 describes class CIM_SharedSecretIsShared.

Table 491 - SMI Referenced Properties/Methods for CIM_SharedSecretIsShared

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

47.8.4 CIM_SharedSecretService

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

Table 492 describes class CIM_SharedSecretService.

Table 492 - SMI Referenced Properties/Methods for CIM_SharedSecretService

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

STABLE

EXPERIMENTAL

Clause 48: Security Profile

48.1 Description

48.1.1 Overview

Security requirements can be divided into four major categories: authentication, authorization, confidentiality, and integrity (including non-repudiation), brief definitions follow. Authentication is verifying the identity of an entity (client or server). Authorization is deciding if an entity is allowed to perform a given operation. Confidentiality is restricting information to only those intended recipients. Integrity is guaranteeing that information, passed between entities, has not been modified.

This top level Security Profile primarily addresses authentication, 42.2.1 HTTP Security Background addresses confidentiality, and authorization is addressed by Clause 49: Authorization Subprofile.

Issues not covered include threat models, protection against specific attack vectors, (such as denial of service, replay, buffer overflow, man in the middle, etc.), topics related to key management, and data integrity. Development of threat models, and specific attack countermeasures required for robust security elements, such as integrity has been left for future work.

Security concerns occur in three areas of an SMI-S implementation:

1) First an SMI-S Server may also be a client of other services, (sometimes conceptualized as a devices.). Those services, (or devices), may require a login before discovery or operations are allowed to be performed. The information needed to perform this login is generically referred to as “credentials”, (or in the case of devices as “device credentials”). An SMI-S server or provider needs to obtain these credentials in order to talk to the service, and they should be provided confidentially.

2) Second, an SMI-S Server may need to authenticate an SMI-S Client. Not all Clients may be allowed to query the object model, and not all Clients may be allowed to perform operations on objects in the model. The SMI-S Server is responsible for the process of authenticating credentials received from an SMI-S Client. Successful authentication establishes a trust relationship, which is represented on the SMI-S Server by an authenticated Identity. Authenticating the client is the first step in determining what that Client is allowed to do.

3) Thirdly, should implementers of an SMI-S Server be unaware of secure development practices, attackers may be able to exploit insecurely developed implementations. (Note, potential attacks might include, but not be limited to buffer overflows, obtaining secure information handled by the SMI-S implementation, like passwords, etc.) In an effort to increase the general knowledge of SMI-S developers, for secure development practices, one resources is referenced: Building Secure Software by Gary McGraw and John Viega (ISBN: 020172152X).

48.1.2 Security Subprofiles

This profile describes minimum requirements on Authentication and Authorization services of an SMI-S Server, where an authenticated Identity is assumed to be authorized. This capability is then extended and constrained by various subprofiles. These are summarized in Table 493.

.

Table 493 - Security Subprofiles

Security Subprofile	Depends on	References	Description
3rdPartyAuthentication	IdentityManagement Security	CredentialManagement	Specifies additional requirements on an SMI-S Server when it is also a client of a 3rd party authentication service
Authorization	Security		Specifies additional requirements on an SMI-S Server that supports an authorization service
CredentialManagement	Security		Specifies additional requirements on an SMI-S Server that is also a client of some other service that enforces security
IdentityManagement	Security		Specifies additional requirements on an SMI-S Server that supports the management of Identities, including establishing Accounts, and defining User and Organizational entities and Groups of those entities.
RBAC	Authorization Security		Specifies additional requirements on an SMI-S Server that supports Role Based Access Control.
ResourceOwnership	Authorization Security	RBAC	Specifies additional requirements on an SMI-S Server that supports the capability to restrict authorization rights.

The purpose of the Security profile is to enable the monitoring and management an entity's rights to act on, (including to view or detect), the operational or management aspects of particular objects within a System. Such an entity is known in CIM by an instance of Identity. With respect to the particular objects, at any point in time an entity is either authenticated or not. This is tracked in the Identity instance as CurrentlyAuthenticated. An Identity with CurrentlyAuthenticated set to True represents a security principal. Authentication is a key criteria for Authorization, where authenticated entities are granted rights to act on particular objects.

Support for this profile declares the ability to discover Identities maintained on an SMI-S Server. Unless modified by a subprofile, entities represented by authenticated Identities are granted all rights to all objects within the scope that the identified entity is known.

This profile contains a number of options. It is up to the profile or subprofile that depends on this profile to specify which options are acceptable

48.1.2.1 Selecting an Identity

To act on a system which enforces security, a requestor needs to be authenticated. The process of authentication maps a requestor to a well-defined Identity. From a management point of view, rights to act on particular resources of a system are granted to Identities.

Figure 70 shows that an Identity instance may be associated with the entity being identified via AssignedIdentity. Commonly this ManagedElement will be an instance of UserContact. UserContact provides information about a user, including UserID.

If AssignedIdentity is not used, an alternative is to use a subclass of Identity with additional properties and to algorithmically equate those properties to a requesting entity in a known way. StorageHardwareID instances are an example of the second option. Each StorageHardwareID contains a StorageID that uniquely identifies a requesting port.

An Identity is only valid within some scope. This is defined by an IdentityContext association, typically to a System or RemoteServiceAccessPoint. If there is more than one System or if there are RemoteServiceAccessPoint instances in the Profile namespace, then IdentityContext is mandatory for this profile.

In all cases, the InstanceID of an Identity should be treated as opaque.

Two options are available for managing the Authentication process within a System.

One option is to use the Identity aspect of Account via ConcreteIdentity. The UserID and UserPassword properties of Account are matched to the authentication information provided by a requestor and the associated Identity instances are selected.

The other option is to associate an AuthenticationRule via PolicySetAppliesToElement.

An Account may be used together with an AuthenticationRule.

See Clause 53: IdentityManagement Subprofile for specification of the ability to add Accounts, UserContacts, and Identities to an SMI-S Server.

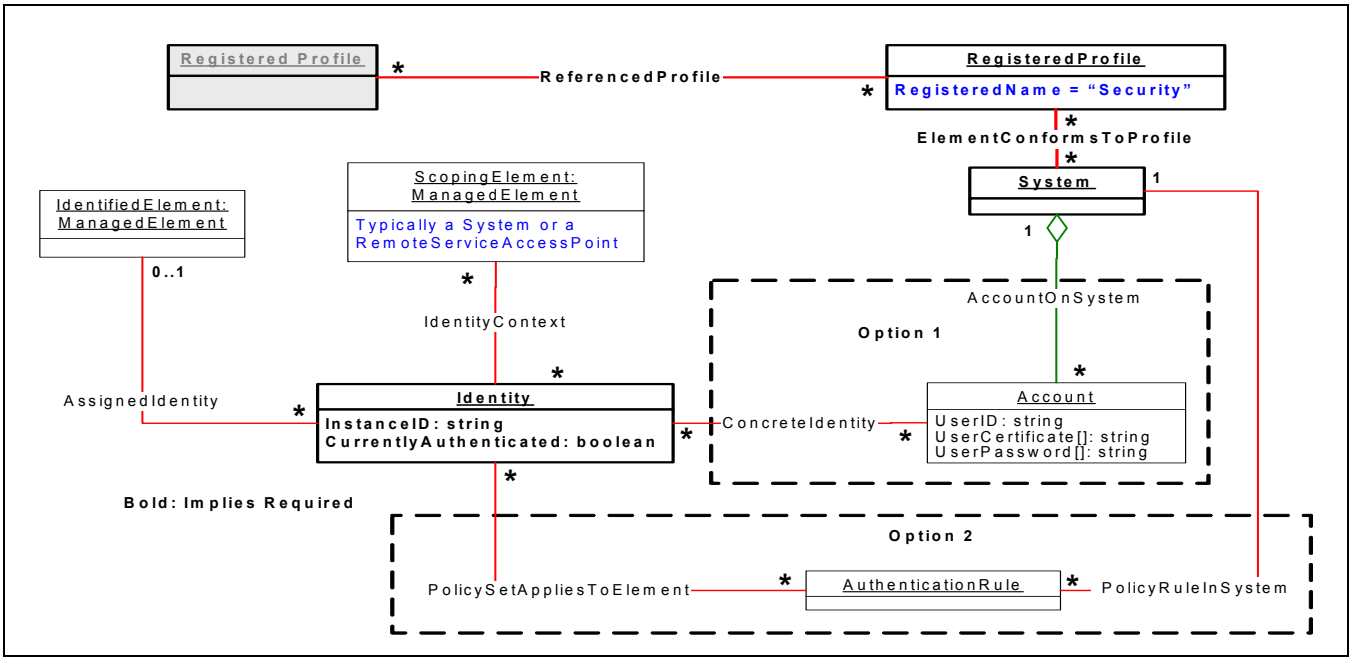

Figure 70 - Identity

48.1.2.2 Authentication Policy

If an AuthenticationRule is not associated with an Identity, then CurrentlyAuthenticated property of Identity is set to True whenever a requestor authenticates to an Identity, and False otherwise.

An AuthenticationRule may be associated with Identity via PolicySetAppliesToElement.

If specified, it further defines or constrains the authentication for the associated Identity. For instance, a PolicyTimePeriodCondition may be associated to the AuthenticationRule via PolicySetValidationPeriod. Additionally, there are a number of specific subclasses of AuthenticationCondition which may be used to further qualify the AuthenticationRule. The CurrentlyAuthenticated property of one of these Identity instances is set to True whenever a requestor matches to an Identity and the conditions of the AuthenticationRule are met, and is set to False otherwise.

The incorporating profile or subprofile shall specify which subclasses of Identity and AuthenticationRule are allowable.

48.1.2.3 Authorization

Unless further constrained by a subprofile or by an incorporating profile, if the CurrentlyAuthenticated property of Identity is set to True, then the identified requesting entity is granted permission to perform any supported action on all elements of the System that conforms to this profile.

See Clause 49: Authorization Subprofile and Clause 52: Security Role Based Access Control Subprofile for additional specification of SMI-S conformant authorization rules.

48.2 Health and Fault Management Considerations

Not defined in this standard.

48.3 Cascading Considerations

Not defined in this standard.

48.4 Supported Subprofiles and Packages

Table 494 describes the supported profiles for Security.

Table 494 - Supported Profiles for Security

Registered Profile Names	Mandatory	Version
Security Identity Management	No	1.1.0
Security Credential Management	No	1.1.0
Security Authorization	No	1.1.0

48.5 Methods of the Profile

None.

48.6 Client Considerations and Recipes

Included is one recipe to list and classify Identities.

48.6.1 List and classify Identities

```
// DESCRIPTION
// This recipe describes how to identify existing Identities and classify them
// by type. The current authentication status of each Identity is determined.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS:
// 1. The name of a top-level System instance in the Security Profile has
// previously been discovered via SLP and is known as $System->.

// MAIN
// Step 1. Locate the known Identities on the system.
$Identities[] = Associators($System->,
     “CIM_IdentityContext”,
     “CIM_Identity”,
     “ElementProvidingContext”,
     “ElementInContext”,
     false,
     false,
     {“CurrentlyAuthenticated”})
// Verify that one or more Identities exist on the system.
if ($Identities[] == null || $Identities[].length < 1) {
    <ERROR! No known Identities on the system>
}
```

```
// Step 2. Create a list entry for each Identity and classify it by type.
#IdentityType[]// contains {“HardwareID”, “Entity”, “Unknown”}
#IdentityUserID[]// contains UserID if the Identity is for an Account.
for (#i in $Identities[]) {

    #IsAuthenticated[#i] = $Identities[#i].CurrentlyAuthenticated

    $Identity-> = $Identities[#i].getObjectPath()
    if ($Identity-> ISA CIM_StorageHardwareID) {
     #IdentityType[#i] = “HardwareID”
     #IdentityUserID[#i] = “”
    } else if ($Identity-> ISA CIM_IPNetworkID) {
     #IdentityType[#i] = “IPNetworkID”
     #IdentityUserID[#i] = “”
    } else {

     // Determine the matching entity type
     $Entity[] = Associators($Identity->,
        “CIM_AssignedIdentity”,
        “CIM_ManagedElement”,
        “IdentityInfo”,
        “ManagedElement”,
        false,
        false,
        {“UserID”})

     // There will be at most one matching entity
     if ($Entity[] == null || $Entity[].length == 0) {
         // Not enough information present to determine type of Identity
         #IdentityType[#i] = “Unknown”
         #IdentityUserID[#i] = “”
     } else {
         // Determine the matching entity type.
         if ($Identity[#i] ISA CIM_UserContact) {
        // Identity of a User
        #IdentityType[#i] = “User”
        #IdentityUserID[#i] = $Entity[0].UserID
         } else {
        // Identity of some other type of Entity
        #IdentityType[#i] = “Entity”
        #IdentityUserID[#i] = “”
         }
     }
    }
    // Determine if there is an associated Account.
    $Entity[] = Associators($Identity->,
```

```
                "CIM_ConcreteIdentity",
                "CIM_Account",
                "SameElement",
                "SystemElement",
                null,
                null,
                {"UserID"})
            if ($Entity[] != null && $Entity[].length = 1) {
             #IdentityUserID[#i] = Entity[1].UserID
            }
        }
```

48.7 Registered Name and Version

Security version 1.1.0

48.8 CIM Elements

Table 494 describes the CIM elements for Security.

Table 495 - CIM Elements for Security

Element Name	Requirement	Description
48.8.1 CIM_Account	Optional	Represents information about an entity that may act on resources
48.8.2 CIM_AccountOnSystem	Optional	Identifies the conformant element
48.8.3 CIM_AssignedIdentity	Optional	Identifies the conformant element
48.8.4 CIM_AuthenticationRule	Optional	A policy the defines the rules for authenticating an Identity
48.8.5 CIM_ConcreteIdentity	Optional	Identifies the conformant element
48.8.6 CIM_Identity	Optional	Represents an entity that may act on resources
48.8.7 CIM_IdentityContext	Optional	Identifies the conformant element
48.8.8 CIM_ManagedElement	Optional	Represents either an entity or a resource
48.8.9 CIM_PolicyRuleInSystem	Optional	Identifies the System which supports the associated PolicyRule.
48.8.10 CIM_PolicySetAppliesToElement	Optional	Identifies the conformant element
48.8.11 CIM_System	Mandatory	System containing elements supporting Authentication and basic Authorization

48.8.1 CIM_Account

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

Table 496 describes class CIM_Account.

Table 496 - SMI Referenced Properties/Methods for CIM_Account

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
CreationClassName		Mandatory	Key
Name		Mandatory	Key
UserID		Mandatory	
UserPassword		Mandatory	
OrganizationName		Mandatory	

48.8.2 CIM_AccountOnSystem

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

Table 497 describes class CIM_AccountOnSystem.

Table 497 - SMI Referenced Properties/Methods for CIM_AccountOnSystem

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

48.8.3 CIM_AssignedIdentity

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

Table 498 describes class CIM_AssignedIdentity.

Table 498 - SMI Referenced Properties/Methods for CIM_AssignedIdentity

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
IdentityInfo		Mandatory	

48.8.4 CIM_AuthenticationRule

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

Table 499 describes class CIM_AuthenticationRule.

Table 499 - SMI Referenced Properties/Methods for CIM_AuthenticationRule

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

48.8.5 CIM_ConcreteIdentity

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

Table 500 describes class CIM_ConcreteIdentity.

Table 500 - SMI Referenced Properties/Methods for CIM_ConcreteIdentity

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	
SameElement		Mandatory	

48.8.6 CIM_Identity

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

Table 501 describes class CIM_Identity.

Table 501 - SMI Referenced Properties/Methods for CIM_Identity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	Indicates whether or not an entity has been authenticated to use this Identity.

48.8.7 CIM_IdentityContext

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

Table 502 describes class CIM_IdentityContext.

Table 502 - SMI Referenced Properties/Methods for CIM_IdentityContext

Properties	Flags	Requirement	Description & Notes
ElementInContext		Mandatory	
ElementProvidingContext		Mandatory	

48.8.8 CIM_ManagedElement

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

48.8.9 CIM_PolicyRuleInSystem

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

Table 503 describes class CIM_PolicyRuleInSystem.

Table 503 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

48.8.10 CIM_PolicySetAppliesToElement

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

Table 504 describes class CIM_PolicySetAppliesToElement.

Table 504 - SMI Referenced Properties/Methods for CIM_PolicySetAppliesToElement

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

48.8.11 CIM_System

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

Table 505 describes class CIM_System.

Table 505 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

EXPERIMENTAL

EXPERIMENTAL

Clause 49: Authorization Subprofile

49.1 Description

The Authorization subprofile extends the Security profile. The Authorization subprofile specifies base support to enable management of the rights of particular subjects to perform specific operations on selected target elements within a CIM Service.

49.1.1 Authorization

Assuming successful authentication, the system needs to assure that the requestor is authorized to perform the request. Figure 71 shows the elements needed to manage authorization. This subprofile constrains the Security profile. When applied, authenticated requestors are not automatically granted all rights. Instead, this subprofile automatically denies all rights unless specifically granted. See 49.1.2, "Authorization Rights", for a detailed description of rights.

Rights to act on a resource are granted or denied to entities using the ChangeAccess method of a PrivilegeManagementService instance. Resources and entities are represented by ManagedElements and Identities, respectively. Granted rights are displayed using the ShowAccess method.

In complex environments two additional associations are used to select the correct PrivilegeManagementService:

- The first is ServiceAvailableToElement, which is not mandatory unless there are more than one System instances in the profile namespace. If there are more than one System, then a ServiceAvailableToElement association between the applicable System and the PrivilegeManagementService is mandatory.
- The second is ServiceAffectsElement associations, which are not mandatory unless there are more than one PrivilegeManagementService instances in the profile namespace. If there are more than one PrivilegeManagementService, then a ServiceAffectsElement association between the PrivilegeManagementService and elements that it can operate on is mandatory.

Sets of rights are represented by Privilege instances. An implementation may publish Privilege instances to use as templates for granting rights. This is done by associating Privilege instances to a PrivilegeManagementService instance via ConcreteDependency.

When a set of rights are granted, the implementation may make this concrete by instantiating an AuthorizedPrivilege instance to represent the set of rights and then using AuthorizedSubject and AuthorizedTarget to associate the authorized Identity and resource. Profiles that incorporate this subprofile may require these associations to be made explicit.

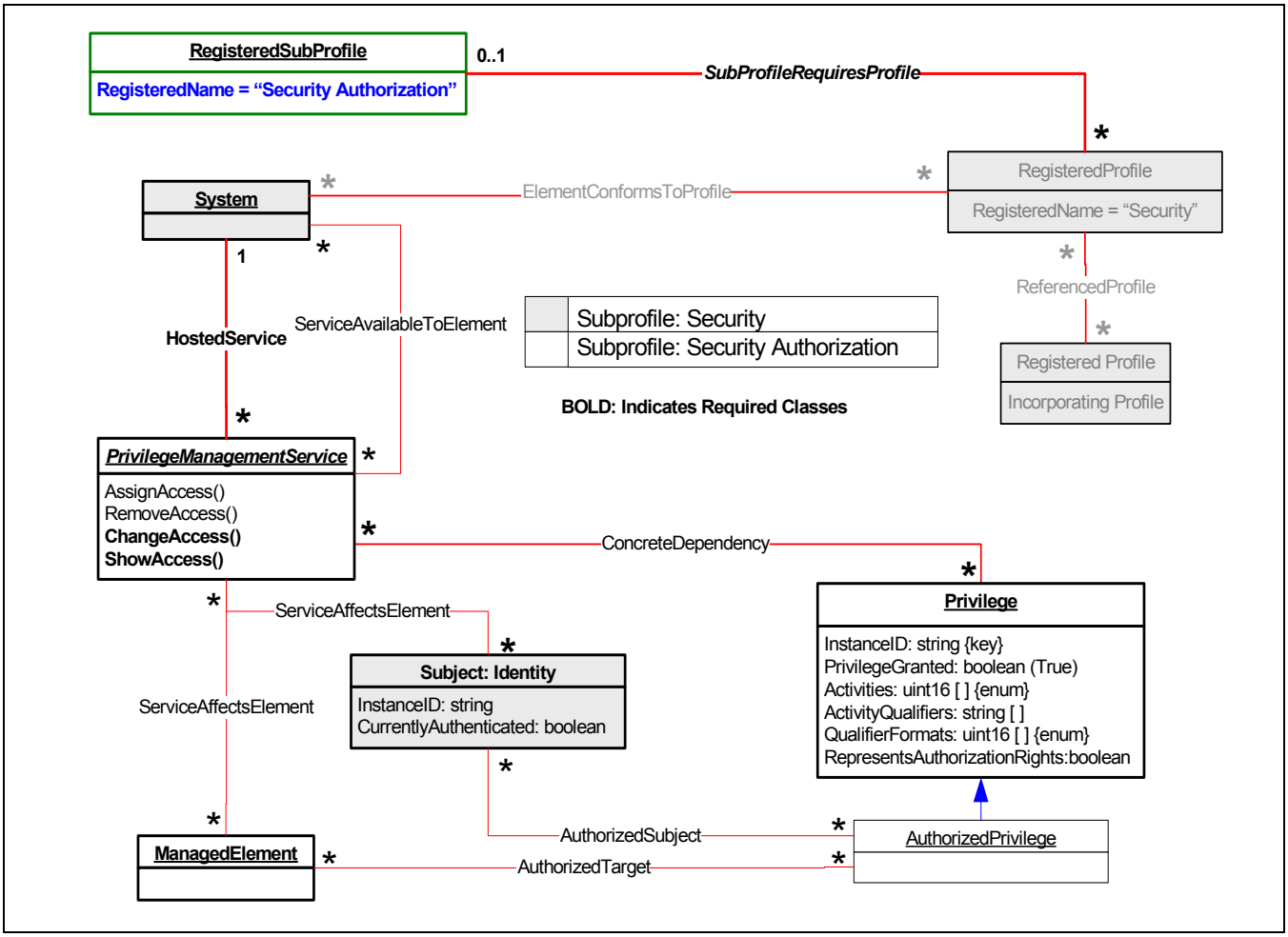

Figure 71 - Authorization

A request is made to act on some element. In the case of Intrinsic Methods, this is first a Namespace, which may or may not be modeled, and which may propagate sub-requests to one or more other ManagedElements published in that Namespace. In the case of Extrinsic Methods, the element shall be the ManagedElement which supports the method.

If it is desired to place restrictions on all elements within a Namespace, then modeling the Namespace is required. The Namespace instance is used as the "ManagedElement" instance shown in Figure 71.

Good practice requires the implementation of each ManagedElement to enforce authorization. A simpler, but less robust model allows the ObjectManager or the Provider of the ManagedElement to authorize the request. Since enforcement at either the ObjectManager or Provider level does not assure there are no back-doors to the implementation, and since the ObjectManager has limited semantic information about the model elements, (and therefore the meaning of the rights passed in Privilege instances,) these simpler schemes are not always applicable. As a result, this Profile RECOMMENDS the more general model.

When the request is delivered, the Identity of the requestor shall be available to the AuthorizationService. The Provider for a ManagedElement can then ask the AuthorizationService to verify that the requested action is allowed. The AuthorizationService maps the request to the rights specified by the Activities, ActivityQualifiers, and QualifierFormat properties of AuthorizedPrivilege. The means for the Provider of a ManagedElement to ask this question of the AuthorizationService is not specified by this Profile.

The client shall use either ChangeAccess (recommended), or AssignAccess and RemoveAccess to grant or deny rights.

49.1.2 Authorization Rights

Rights are encoded within the properties of Privilege, two of which operate on all rights defined by the Privilege instance and three of which define a set of rights. The Privilege global properties are:

- PrivilegeGranted: This boolean controls whether the rights defined by the instance are granted or denied[2]. The default is TRUE.
- RepresentsAuthorizationRights: This boolean controls whether the rights defined by the instance specifies access rights or authorization rights. Access rights grant a subject access to a target. Authorization rights grant a subject the right to assign, change, or remove the specified rights for a target to other subjects. The default is FALSE.

The properties which define rights are each an indexed array. Corresponding array entries across all three represent a single access or authorization right. These properties are:

- Activities: Each entry is an enumeration that specifies whether the corresponding right is “Read”. “Write”, “Execute”, “Create”, “Delete”, or “Detect”.
- ActivityQualifiers: Each entry is a string that qualifies the corresponding Activity entry. For instance, if the Activities is “Execute”, then the corresponding entry might be a comma separated list of method names. An entry may be NULL which specifies that the corresponding Activity is not qualified.
- QualifierFormats: Each entry is an enumeration that specifies the format of the string in the corresponding ActivityQualifiers entry. If an ActivityQualifiers entry is not NULL then the corresponding QualifierFormats entry shall be specified. Otherwise it shall be NULL. Possible enumerations are: “Class Name”, “<Class.>Property”, “<Class.>Method”, “Object Reference”, “Namespace”, “URL”, “Directory/File Name”, “Command Line Instruction”, “SCSI Command”, and “Packet”. In the “Execute” example above, the QualifierFormats entry shall be “<Class.>Method”.

Specification of allowable combinations of rights is left to the profiles or subprofiles that incorporate this subprofile.

49.1.3 Authorization Policy

The default authorization policy is to deny all requests that are not explicitly granted via either an AuthorizationPolicy or by an explicit ChangeAccess or AssignAccess method.

An AuthorizationRule may be specified as part of a ChangeAccess method. The AuthorizationRule may then grant rights implicitly.

Identities, Privileges, and target ManagedElements may be associated to an AuthorizationRule by AuthorizationRuleAppliesToIdentity, AuthorizationRuleAppliesTo-AuthorizedPrivilege, and AuthorizationRuleAppliesToTarget respectively. This is shown in Figure 72. When an AuthorizedPrivilege, is added to the AuthorizationRule, an AuthorizedSubject or AuthorizedTarget may be instantiated.

2.When used with ChangeAccess, the meaning of PrivilegeGranted changes to specify whether the rights defined by the instance are added or subtracted.

The details of the specification of AuthorizationRules are left to the profiles and subprofiles that reference this subprofile.

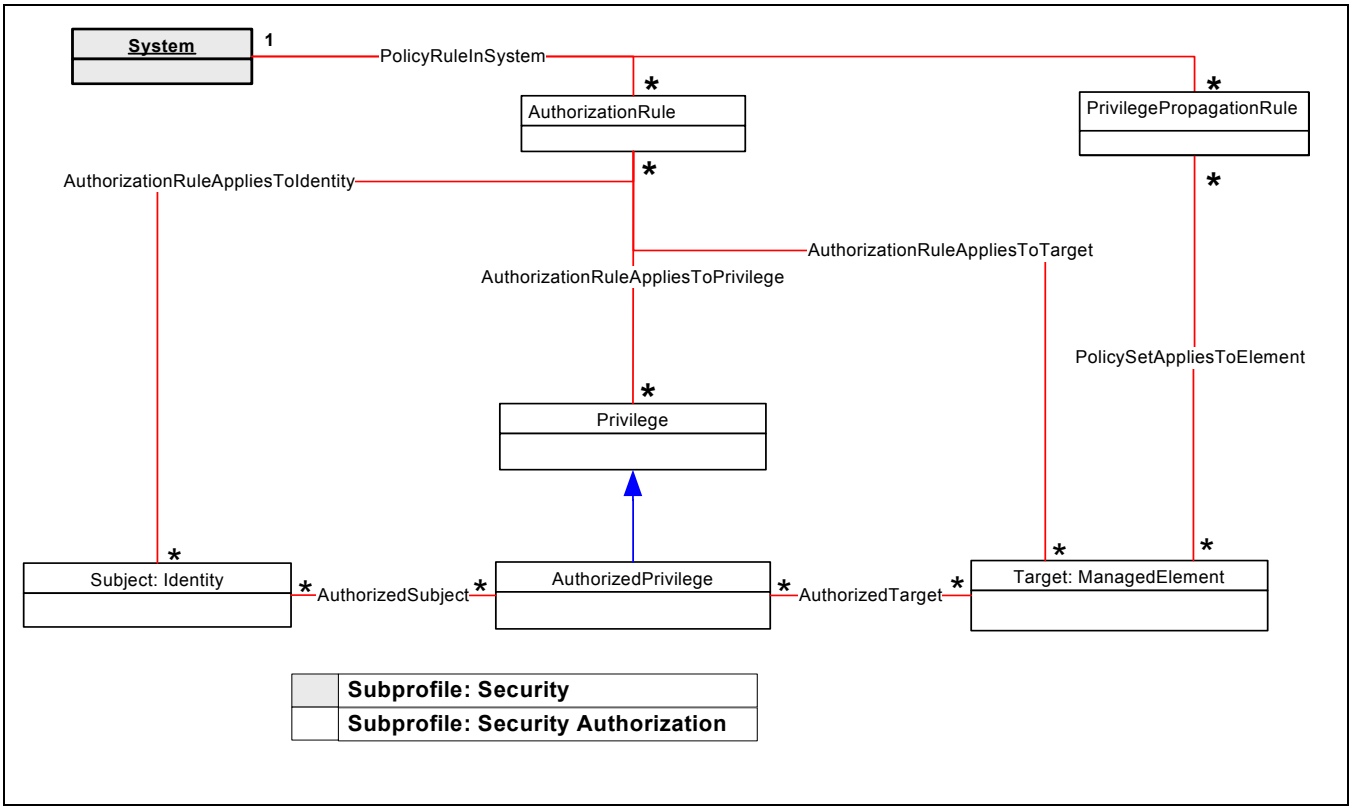

Figure 72 - Policy Rules

49.1.4 Privilege Propagation Policies

In most instances, the propagation rules for a particular type of target element are clear and apply to all subjects. In this case, the semantics of the target element can imply a particular propagation policy. When a subject may select from multiple possible propagation strategies for a target element, there needs to be a means to specify the propagation strategy. Subclasses of PrivilegePropagationRule provide this ability. When associated with a target element via PolicySetAppliesToElement, the PrivilegePropagationRule specifies the default policy to apply. When associated to an AuthorizedPrivilege, via PolicySetAppliesToElement, the PrivilegePropagationRule specifies the policy used to propagate the named rights.

When an AuthorizedPrivilege instance representing propagated rights is returned, it will have the IsPropagated boolean set to True.

The details of the specification of PrivilegePropagationRules are left to the profiles and subprofiles that reference this subprofile.

For illustrative purposes only, the following example illustrates the creation of a PrivilegePropagationRule using QueryCondition (not shown) and MethodAction (not shown) classes associated via PolicyConditionInPolicyRule (not shown) and PolicyActionInPolicyRule (not shown) respectively. The QueryLanguage property of the QueryCondition and MethodAction instances shall be set to “2”, meaning “CQL”. Assume the QueryCondition.QueryResultName is set to “SNIA_AuthorizationConditionExample” and its Query property set to

```
"SELECT (M.SourceInstanceHost || "/" || M.SourceInstanceModelPath) AS PMSPath,
                    M.MethodParameters.Subject,
                    ObjectPath(E) AS Target,
```

```
                    M MethodParameters.Privileges
FROM
     CIM_InstMethodCall M,
     CIM_Collection C,
     CIM_MemberOfCollection MoC,
     CIM_ManagedElement E
     CIM_PolicySetAppliesToElement PSATE
     CIM_PolicyConditionInPolicyRule PCIPR
     CIM_PrivilegePropagationRule PPR
WHERE
     M.MethodName = "ChangeAccess"
AND M.ReturnValue = 0
AND M.PreCall = FALSE
AND M.MethodParameters.Target ISA CIM_Collection
AND M.Target = MoC.Collection
AND ObjectPath(E) = MoC.Element
AND ObjectPath() = PCIPR.PartComponent
AND ObjectPath(PPR) = PCIPR.GroupComponent
AND ObjectPath(PPR) = PSATE.PolicySet
AND ObjectPath(E) = PSATE.ManagedElement"
```

This assures that this query is being run on behalf of a PrivilegePropagationRule that is applied to the Collection. This assures that propagation does not pass through collections that are not appropriate.

The corresponding MethodAction instance would have its Query property set to

"SELECT (Ex.PMSPath || "." || "ChangeAccess") AS Methadone,
Ex.Subject AS Subject,
Ex.Target AS Target,
NULL AS PropagationPolicies,
Ex.Privileges AS Privileges
FROM SNIA_AuthorizationConditionExample Ex"

The ChangeAccess method enables a client to specify a PrivilegePropagationRule to use while assigning rights. (See Figure 72.)

49.1.5 Reporting Granted Rights

Granted rights are reported using the ShowAccess method. (See Figure 71.) This method takes as input one or both of a subject Identity and target ManagedElement. Output is a list of Identity, Privilege, target triples that represent granted Privileges. This output shall reflect a consistent current state at the time of the call, regardless of whether or not corresponding instances of AuthorizedPrivilege, AuthorizedTarget, and AuthorizedSubject have been instantiated.

49.2 Health and Fault Management Considerations

Not defined in this standard.

49.3 Cascading Considerations

Not defined in this standard.

49.4 Supported Subprofiles and Packages

None.

49.5 Methods of the Profile

None.

49.6 Client Considerations and Recipes

49.6.1 Show access rights

```
// DESCRIPTION
// This recipe describes how to identify the authorized subjects and their
// rights for a specified resource.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The name of a top-level System instance in the Security Profile has
// previously been discovered via SLP and is known as $System->.
// 2. The name of a managed element on $System-> whose authorized subjects and
// rights has previously been discovered and is known as $Resource->.

// This function locates the PrivilegeManagementService that manages the
// specified managed element. If no such service is located, null is returned.
sub CIMObjectPath GetPrivilegeServiceForElement(CIMObjectPath[] $Services->[],
     CIMObjectPath $Resource->) {

    $Service-> = null
    // Verify that there is one or more instance of PrivilegeManagementService
    // hosted by the system.
    if ($Services->[] != null && $Services->[] > 0) {
     // Locate the service that manages the privileges of the specified
     // managed element.
     $ResourceServices->[] = AssociatorNames($Resource->,
         "CIM_ServiceAffectsElement",
         "CIM_PrivilegeManagementService",
         "UserOfService",
         "ServiceProvided")
     if ($ResourceServices->[] != null || $ResourceServices->[].length > 0) {
         for (#i in $ResourceServices->[]) {
         for (#j in $Services->[]) {
             if ($ResourceServices->[#i] == $Services->[#j)) {
            $Service-> = Services->[#j]
            break
             }
         }
          }
     }
    }
```

```
    return $Service->

}

// MAIN
// Step 1. Locate the PrivilegeManagementServices on the system.
$PrivilegeServices->[] = AssociatorNames($System->,
     "CIM_HostedService",
     "CIM_PrivilegeManagementService",
     "Antecedent",
     "Dependent")

// There must be exactly one PrivilegeManagementService for the managed element.
$PrivilegeService-> = &GetPrivilegeServiceForElement($PrivilegeServices->[],
     $Resource->)
if ($PrivilegeService-> == null) {
    <EXIT! The required PrivilegeManagementService was not found>
}

// Step 2. Retrieve the authorized subjects and their rights for the specified
// resource.
%InArgs["Subject"] = null
%InArgs["Target"] = $Resource->
#Result = InvokeMethod($PrivilegeService->,
     "ShowAccess",
     %InArgs[],
     %OutArgs[])

// Verify that the operation performed successfully.
if (#Result != 0) {
    <EXIT! Retrieving access for the specified resource failed>
}

// Step 3. Retrieve the references to the Identities (or other subjects)
// authorized for the resource.
$OutSubjects->[] = %OutArgs["OutSubjects"]

// Step 4. Retrieve the references to the Privileges corresponding to the
// subject entries.
$OutPrivileges->[] = %OutArgs["Privileges"]
```

49.6.2 Grant an access right

```
// DESCRIPTION
// This recipe describes how to apply a set of rights to a given resource
// and subject.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
```

```
// 1. The name of a top-level System instance in the Security Profile has
// previously been discovered via SLP and is known as $System->.
// 2. The name of a managed element on $System-> has previously been
// discovered and is known as $Resource->.
// 3. The name of a subject has previously been discovered and is known as
// $Subject->.
// 4. A container of activities to be granted or denied is known as #Activity[].
// 5. A container of additional information related to the activities is known
// as #ActivityQualifiers[].
// 6. A container of sematic descriptions of the formats of the elements in
// #ActivityQualifiers[] is known as #QualifierFormats[].

// MAIN
// Step 1. Locate the PrivilegeManagementServices on the system.
$PrivilegeServices->[] = AssociatorNames($System->,
     "CIM_HostedService",
     "CIM_PrivilegeManagementService",
     "Antecedent",
     "Dependent")

// There must be exactly one PrivilegeManagementService for the managed element.
$PrivilegeService-> = &GetPrivilegeServiceForElement($PrivilegeServices->[],
     $Resource->)
if ($PrivilegeService-> == null) {
    <EXIT! The required PrivilegeManagementService was not found>
}

// Step 2. Create an Access Privilege
$Privilege = newInstance("CIM_Privilege")
$Privilege.PrivilegeGranted = true
$Privilege.RepresentsAuthorizationRights = false
$Privilege.Activity[] = #Activity[]
$Privilege.ActivityQualifiers[] = #ActivityQualifiers[]
$Privilege.QualifierFormats[] = #QualifierFormats[]

// Step 3. Add the right and get the resultant rights.
%InArgs["Subject"] = $Subject->
%InArgs["Target"] = $Resource->
%InArgs["PropagationPolicies"] = null
$Privileges[0] = $Privilege
%InArgs["Privileges"] = $Privileges[]
#Result = InvokeMethod($PrivilegeService->,
     "ChangeAccess",
     %InArgs[],
     %OutArgs[])

// Verify that the operation performed successfully.
```

```
if (#Result != 0) {
    <EXIT! Changing access for the specified resource failed>
}

// Step 4. Retrieve the references to the Privileges that represent the
// resulting rights between the subject and target instances.
$OutPrivileges->[] = %OutArgs["Privileges"]
```

49.6.3 Deny a right

```
// DESCRIPTION
// This recipe describes how to remove a right from a given resource.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The name of a top-level System instance in the Security Profile has
// previously been discovered via SLP and is known as $System->.
// 2. The name of a managed element on $System-> has previously been
// discovered and is known as $Resource->.
// 3. The name of a subject has previously been discovered and is known as
// $Subject->.
// 4. A container of activities to be granted or denied is known as #Activity[].
// 5. A container of additional information related to the activities is known
// as #ActivityQualifiers[].
// 6. A container of sematic descriptions of the formats of the elements in
// #ActivityQualifiers[] is known as #QualifierFormats[].

// MAIN
// Step 1. Locate the PrivilegeManagementServices on the system.
$PrivilegeServices->[] = AssociatorNames($System->,
     "CIM_HostedService",
     "CIM_PrivilegeManagementService",
     "Antecedent",
     "Dependent")

// There must be exactly one PrivilegeManagementService for the managed element.
$PrivilegeService-> = &GetPrivilegeServiceForElement($PrivilegeServices->[],
     $Resource->)
if ($PrivilegeService-> == null) {
    <EXIT! The required PrivilegeManagementService was not found>
}

// Step 2. Create an Access Privilege
$Privilege = newInstance("CIM_Privilege")
$Privilege.PrivilegeGranted = false
$Privilege.RepresentsAuthorizationRights = false
$Privilege.Activity[] = #Activity[]
$Privilege.ActivityQualifiers[] = #ActivityQualifiers[]
$Privilege.QualifierFormats[] = #QualifierFormats[]
```

```
$Privilege[1] = $Privilege

// Step 3. Remove the right and get the resultant rights.
%InArgs[“Subject”] = $Subject->
%InArgs[“Target”] = $Resource->
%InArgs[“PropagationPolicies”] = null
$Privileges[0] = $Privilege
%InArgs[“Privileges”] = $Privileges[]
#Result = InvokeMethod($PrivilegeService->,
     “ChangeAccess”,
     %InArgs[],
     %OutArgs[])

// Verify that the operation performed successfully.
if (#Result != 0) {
    <EXIT! Changing access for the specified resource failed>
}

// Step 4. Retrieve the references to the Privileges that represent the
// resulting rights between the subject and target instances.
$OutPrivileges->[] = %OutArgs[“Privileges”]
```

49.7 Registered Name and Version

Security Authorization version 1.1.0

49.8 CIM Elements

Table 506 describes the CIM elements for Security Authorization.

Table 506 - CIM Elements for Security Authorization

Element Name	Requirement	Description
49.8.1 CIM_AuthorizationRule	Optional	
49.8.2 CIM_AuthorizationRuleAppliesToIdentity	Optional	
49.8.3 CIM_AuthorizationRuleAppliesToPrivilege	Optional	
49.8.4 CIM_AuthorizationRuleAppliesToTarget	Optional	
49.8.5 CIM_AuthorizedPrivilege	Optional	
49.8.6 CIM_AuthorizedSubject	Optional	
49.8.7 CIM_AuthorizedTarget	Optional	

Table 506 - CIM Elements for Security Authorization

Element Name	Requirement	Description
49.8.8 CIM_ConcreteDependency	Optional	
49.8.9 CIM_ConcreteDependency	Optional	
49.8.10 CIM_HostedService	Mandatory	
49.8.11 CIM_Identity	Optional	
49.8.12 CIM_ManagedElement	Optional	
49.8.13 CIM_PolicyRuleInSystem	Optional	
49.8.14 CIM_PolicySetAppliesToElement	Optional	
49.8.15 CIM_Privilege	Optional	
49.8.16 CIM_PrivilegeManagementService	Mandatory	
49.8.17 CIM_PrivilegePropagationRule	Optional	
49.8.18 CIM_ServiceAffectsElement (Service to AuthorizedProvolege)	Optional	
49.8.19 CIM_ServiceAffectsElement (Service to Iden tity)	Optional	
49.8.20 CIM_ServiceAffectsElement (Service to ManagedElement)	Optional	
49.8.21 CIM_ServiceAffectsElement (Service to Privilege)	Optional	
49.8.22 CIM_ServiceAvailableToElement	Mandatory	
49.8.23 CIM_System	Mandatory	

49.8.1 CIM_AuthorizationRule

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

Table 507 describes class CIM_AuthorizationRule.

Table 507 - SMI Referenced Properties/Methods for CIM_AuthorizationRule

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

Table 507 - SMI Referenced Properties/Methods for CIM_AuthorizationRule

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
PolicyRuleName		Mandatory	Key

49.8.2 CIM_AuthorizationRuleAppliesToIdentity

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

Table 508 describes class CIM_AuthorizationRuleAppliesToIdentity.

Table 508 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToIdentity

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

49.8.3 CIM_AuthorizationRuleAppliesToPrivilege

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

Table 509 describes class CIM_AuthorizationRuleAppliesToPrivilege.

Table 509 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToPrivilege

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

49.8.4 CIM_AuthorizationRuleAppliesToTarget

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

Table 510 describes class CIM_AuthorizationRuleAppliesToTarget.

Table 510 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToTarget

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

49.8.5 CIM_AuthorizedPrivilege

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

Table 511 describes class CIM_AuthorizedPrivilege.

Table 511 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
RepresentsAuthorizationRights		Mandatory	Must be an Access right for this subprofile.
PrivilegeGranted		Mandatory	Only Grant type privileges are allowed.

49.8.6 CIM_AuthorizedSubject

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

Table 512 describes class CIM_AuthorizedSubject.

Table 512 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject

Properties	Flags	Requirement	Description & Notes
Privilege		Mandatory	
PrivilegedElement		Mandatory	

49.8.7 CIM_AuthorizedTarget

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

Table 513 describes class CIM_AuthorizedTarget.

Table 513 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget

Properties	Flags	Requirement	Description & Notes
Privilege		Mandatory	
TargetElement		Mandatory	

49.8.8 CIM_ConcreteDependency

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

Table 514 describes class CIM_ConcreteDependency.

Table 514 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

49.8.9 CIM_ConcreteDependency

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

Table 515 describes class CIM_ConcreteDependency.

Table 515 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

49.8.10 CIM_HostedService

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

Table 516 describes class CIM_HostedService.

Table 516 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Dependent		Mandatory	

49.8.11 CIM_Identity

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

Table 517 describes class CIM_Identity.

Table 517 - SMI Referenced Properties/Methods for CIM_Identity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	The Identified entity is authenticated or not

49.8.12 CIM_ManagedElement

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

49.8.13 CIM_PolicyRuleInSystem

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Optional

Table 518 describes class CIM_PolicyRuleInSystem.

Table 518 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

49.8.14 CIM_PolicySetAppliesToElement

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

Table 519 describes class CIM_PolicySetAppliesToElement.

Table 519 - SMI Referenced Properties/Methods for CIM_PolicySetAppliesToElement

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

49.8.15 CIM_Privilege

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

Table 520 describes class CIM_Privilege.

Table 520 - SMI Referenced Properties/Methods for CIM_Privilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
RepresentsAuthorizationRights		Mandatory	Indicates the privilege is to assign the named rights to subjects.
PrivilegeGranted		Optional	Only Grant type privileges are allowed.

49.8.16 CIM_PrivilegeManagementService

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

Table 521 describes class CIM_PrivilegeManagementService.

Table 521 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
CreationClassName		Mandatory	Key
Name		Mandatory	Key
ChangeAccess()		Optional	
ShowAccess()		Optional	

49.8.17 CIM_PrivilegePropagationRule

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

Table 522 describes class CIM_PrivilegePropagationRule.

Table 522 - SMI Referenced Properties/Methods for CIM_PrivilegePropagationRule

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

49.8.18 CIM_ServiceAffectsElement (Service to AuthorizedProvolege)

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 523 describes class CIM_ServiceAffectsElement (Service to AuthorizedProvolege).

Table 523 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to AuthorizedProvolege)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	
AffectedElement		Mandatory	

49.8.19 CIM_ServiceAffectsElement (Service to Iden tity)

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

Table 524 describes class CIM_ServiceAffectsElement (Service to Iden tity).

Table 524 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to Iden tity)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

49.8.20 CIM_ServiceAffectsElement (Service to ManagedElement)

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

Table 525 describes class CIM_ServiceAffectsElement (Service to ManagedElement).

Table 525 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to ManagedElement)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

49.8.21 CIM_ServiceAffectsElement (Service to Privilege)

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

Table 526 describes class CIM_ServiceAffectsElement (Service to Privilege).

Table 526 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to Privilege)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	
AffectedElement		Mandatory	

49.8.22 CIM_ServiceAvailableToElement

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

Table 527 describes class CIM_ServiceAvailableToElement.

Table 527 - SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement

Properties	Flags	Requirement	Description & Notes
ServiceProvided		Mandatory	
UserOfService		Mandatory	

49.8.23 CIM_System

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

Table 528 describes class CIM_System.

Table 528 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

EXPERIMENTAL

EXPERIMENTAL

Clause 50: Credential Management Subprofile

50.1 Description

This subprofile provides for management of credentials used by a client to establish its identity to a serving system. An administrator of both the client and server systems establishes an Identity for the client on the server system and creates a credential for the client on the client system.

Note: SMI-S Servers are often clients of other services. For instance, a device that is managed by an SMI-S Server may require a login before it allows a client to discover or manage its components. Credentials used to access devices are known within this specification as “device credentials”.

As shown in Figure 73, this subprofile applies to a System as a whole.

Credentials are created by a LocalCredentialManagementService. There shall be a one or more LocalCredentialManagementService instances on a System conforming to this subprofile.

The Credentials are intended to authenticate a client on this System to a service running on a remote system. There shall be one or more RemoteServiceAccessPoint instances for each of the Systems to which Credentials may be presented.

50.1.1 Credential setup

The administrator needs to have prior knowledge about the type of Credential required by the remote system. The LocalCredentialManagementService is subclassed into two types, a SharedSecretService and a PublicKeyManagementService. If the latter is present, then UnsignedPubliKey Credentials are supported. If the former, then SharedSecretService.Protocol = “SharedSecret” specifies that SharedSecret credentials are supported and SharedSecretService.Protocol = “IKE” specifies that NamedSharedIKESecret credentials are supported

The administrator uses CreateInstance and DeleteInstance to create or delete Credentials. The details of each are described below. In common to all is that the key properties: SystemCreationClassName, SystemName, ServiceCreationClassName, and ServiceName of each Credential shall be fully specified at creation time. This information is used by the system to locate the correct LocalCredentialManagementService instance and to snap the required IKESecretIsNamed, SharedSecretIsShared or LocallyManagedPublicKey associations. Additionally certain remaining properties of each credential shall be filled in as described below.

- Expires: Set to the datetime after which this credential will not be valid. Use a value of “99991231235959.999999+999” if this field is to be ignored.

50.1.2 SharedSecret Credential

- RemoteID: Set to the User ID or other value by which the client is known on the remote system. Typically this will correspond to Account.Userid or Person.UserID as stored on the remote system.
- Secret: Set to the password or other value by which the client is authenticated on the remote system. The value is provided in clear text. There is an underlying assumption that there is a secure communication path being used between the administrator and the CIM Service on the client system. This property is writable, but shall not be readable. Typically this will correspond to Account.Userid or Person.UserID as stored on the remote system.

50.1.3 NamedSharedIKE Credential

- PeerIdentityType: This describes the type of identity used to locate the remote peer. It is an enumerated type that shall correspond to one of the following values: “IPV4_ADDR”, “FQDN”, “USER_FQDN”,

“IPV4_ADDR_SUBNET”,”IPV6_ADDR”, “IPV6_ADDR_SUBNET”, “IPV4_ADDR_RANGE”, “IPV6_ADDR_RANGE”, “DER_ASN1_DN”, “DER_ASN1_GN”, or “KEY_ID”.

- PeerIdentity: An identity value conforming to the PeerIdentityType and naming the remote peer with whom a direct trust relation exists.
- LocalIdentityType: This describes the type of identity used to name the local peer. It is an enumerated type that shall correspond to one of the following values: “IPV4_ADDR”, “FQDN”, “USER_FQDN”, “IPV4_ADDR_SUBNET”,”IPV6_ADDR”, “IPV6_ADDR_SUBNET”, “IPV4_ADDR_RANGE”, “IPV6_ADDR_RANGE”, “DER_ASN1_DN”, “DER_ASN1_GN”, or “KEY_ID”.
- LocalIdentity: An identity value conforming to the LocalIdentityType and naming the local peer with whom a direct trust relation exists.
- SharedSecretName: On creation, this is set to the password or other shared value used to authenticate the client. When read, this is an indirect reference to a shared secret. The SecretService does not expose the actual secret.

50.1.4 UnsignedPublicKey Credential

- PeerIdentityType: This describes the type of identity used to locate the remote peer. It is an enumerated type that shall correspond to one of the following values: “IPV4_ADDR”, “FQDN”, “USER_FQDN”, “IPV4_ADDR_SUBNET”,”IPV6_ADDR”, “IPV6_ADDR_SUBNET”, “IPV4_ADDR_RANGE”, “IPV6_ADDR_RANGE”, “DER_ASN1_DN”, “DER_ASN1_GN”, or “KEY_ID”.
- PeerIdentity: An identity value conforming to the PeerIdentityType and naming the remote peer with whom a direct trust relation exists.
- PublicKey: The DER-encoded raw public key

.

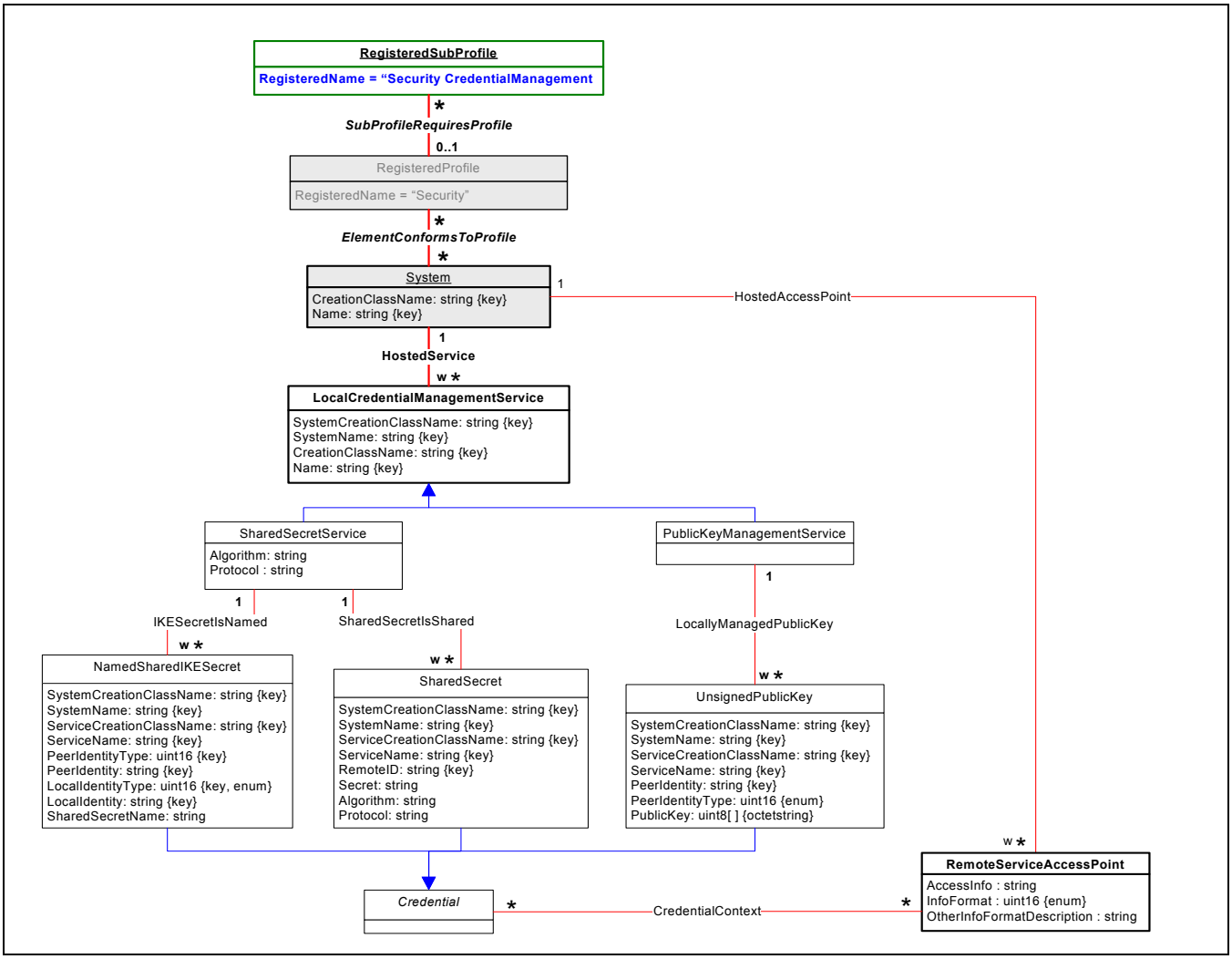

Figure 73 - Credential Management

50.1.5 Credential Use

Once set up, a Credential may be enabled or disabled for use by using CreateInstance or DeleteInstance to add or remove CredentialContext associations between a Credential and the RemoteServiceAccessPoint used to access a remote system.

See Clause 47:, "Device Credentials Subprofile" for a complete discussion of the SMI-S requirements for modeling device credentials.

The SMI-S Server shall securely store the device credentials local to the SMI-S Server. A proxy SMI-S Server may need to store the credentials on disk so that they are available upon reboot. In this case the credentials shall be encrypted for confidentiality.

The device credentials shall be transmitted securely from the SMI-S Server to the device. The mechanism of communicating the credentials to the device is outside the scope of this specification, but it should be over a secure channel if possible.

A SMI-S Server may be configured with the device credentials necessary to talk to the device. If a SMI-S Server supports SSL 3.0 or TLS, the HTTP Client shall use SSL 3.0 or TLS to pass device credentials to the SMI-S Server.

When new device credentials are passed to an SMI-S Server, the device credential information in the device shall be updated immediately.

Only the SMI-S Server responsible for communicating with the device has access to the properties of the SharedSecret object. No other SMI-S Client may read the Secret property of this object as it shall be implemented Write-Only.

50.2 Health and Fault Management Considerations

Not defined in this standard.

50.3 Cascading Considerations

Not defined in this standard.

50.4 Supported Subprofiles and Packages

None.

50.5 Methods of the Profile

None.

50.6 Client Considerations and Recipes

None.

50.7 Registered Name and Version

Security Credential Management version 1.1.0

50.8 CIM Elements

Table 529 describes the CIM elements for Security Credential Management.

Table 529 - CIM Elements for Security Credential Management

Element Name	Requirement	Description
50.8.1 CIM_CredentialContext	Optional	
50.8.2 CIM_HostedAccessPoint	Mandatory	
50.8.3 CIM_HostedService	Mandatory	
50.8.4 CIM_IKESecretIsNamed	Optional	
50.8.5 CIM_LocallyManagedPublicKey	Optional	
50.8.6 CIM_NamedSharedIKESecret	Mandatory	
50.8.7 CIM_PublicKeyManagementService	Mandatory	
50.8.8 CIM_RemoteServiceAccessPoint	Mandatory	

Table 529 - CIM Elements for Security Credential Management

Element Name	Requirement	Description
50.8.9 CIM_SharedSecret	Mandatory	
50.8.10 CIM_SharedSecretIsShared	Optional	
50.8.11 CIM_SharedSecretService	Mandatory	
50.8.12 CIM_System	Mandatory	
50.8.13 CIM_UnsignedPublicKey	Mandatory	

50.8.1 CIM_CredentialContext

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

Table 530 describes class CIM_CredentialContext.

Table 530 - SMI Referenced Properties/Methods for CIM_CredentialContext

Properties	Flags	Requirement	Description & Notes
ElementProvidingContext		Mandatory	
ElementInContext		Mandatory	

50.8.2 CIM_HostedAccessPoint

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

Table 531 describes class CIM_HostedAccessPoint.

Table 531 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

50.8.3 CIM_HostedService

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

Table 532 describes class CIM_HostedService.

Table 532 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

50.8.4 CIM_IKESecretIsNamed

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

Table 533 describes class CIM_IKESecretIsNamed.

Table 533 - SMI Referenced Properties/Methods for CIM_IKESecretIsNamed

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

50.8.5 CIM_LocallyManagedPublicKey

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

Table 534 describes class CIM_LocallyManagedPublicKey.

Table 534 - SMI Referenced Properties/Methods for CIM_LocallyManagedPublicKey

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

50.8.6 CIM_NamedSharedIKESecret

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

Table 535 describes class CIM_NamedSharedIKESecret.

Table 535 - SMI Referenced Properties/Methods for CIM_NamedSharedIKESecret

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
ServiceCreationClassName		Mandatory	Key
ServiceName		Mandatory	Key
PeerIdentity		Mandatory	Key, The identity of the remote peer trusted entity.
PeerIdentityType		Mandatory	The type of the remote PeerIdentity.
LocalIdentity		Mandatory	Key, The identity of the local peer trusted entity.
LocalIdentityType		Mandatory	The type of the LocalIdentity.
SharedSecretName	M	Mandatory	The name of the shared secret,

50.8.7 CIM_PublicKeyManagementService

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

Table 536 describes class CIM_PublicKeyManagementService.

Table 536 - SMI Referenced Properties/Methods for CIM_PublicKeyManagementService

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

50.8.8 CIM_RemoteServiceAccessPoint

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

Table 537 describes class CIM_RemoteServiceAccessPoint.

Table 537 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint

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

50.8.9 CIM_SharedSecret

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

Table 538 describes class CIM_SharedSecret.

Table 538 - SMI Referenced Properties/Methods for CIM_SharedSecret

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
ServiceCreationClassName		Mandatory	Key
ServiceName		Mandatory	Key
RemoteID		Mandatory	Key, The identity of the client as known on the remote system.
Secret		Mandatory	A secret

50.8.10 CIM_SharedSecretIsShared

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

Table 539 describes class CIM_SharedSecretIsShared.

Table 539 - SMI Referenced Properties/Methods for CIM_SharedSecretIsShared

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

50.8.11 CIM_SharedSecretService

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

Table 540 describes class CIM_SharedSecretService.

Table 540 - SMI Referenced Properties/Methods for CIM_SharedSecretService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
CreationClassName		Mandatory	Key
Name		Mandatory	Key
Protocol	M	Mandatory	Select 'IKE' for Shared IKE secrets and 'SharedSecret' for Shared secrets.

50.8.12 CIM_System

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

Table 541 describes class CIM_System.

Table 541 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

50.8.13 CIM_UnsignedPublicKey

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

Table 542 describes class CIM_UnsignedPublicKey.

Table 542 - SMI Referenced Properties/Methods for CIM_UnsignedPublicKey

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
ServiceCreationClassName		Mandatory	Key
ServiceName		Mandatory	Key
PeerIdentity		Mandatory	Key, The identity of the peer trusted entity.
PeerIdentityType		Mandatory	The type of the PeerIdentity.
PublicKey	M	Mandatory	Key, The identity of the peer trusted entity.

EXPERIMENTAL

EXPERIMENTAL

Clause 51: Security Resource Ownership Subprofile

51.1 Description

This subprofile[3] provides the means to model restrictions on CIM operations associated with exclusive use of a resource, For instance, a storage volume in an array. It is intended for environments in which multiple CIM clients may not be completely aware of each other's activities, making it important that use of the resource not be disrupted by a client that is unaware of its use. Specific examples include use of a volume by storage virtualizers and NAS gateways, where attempts to manage the volume by clients not associated with this use could be seriously disruptive. An intended configuration is that a CIM client exists in the cascading device that has exclusive use of the volume, although this is not strictly necessary. The Security Resource Ownership Subprofile is optional.

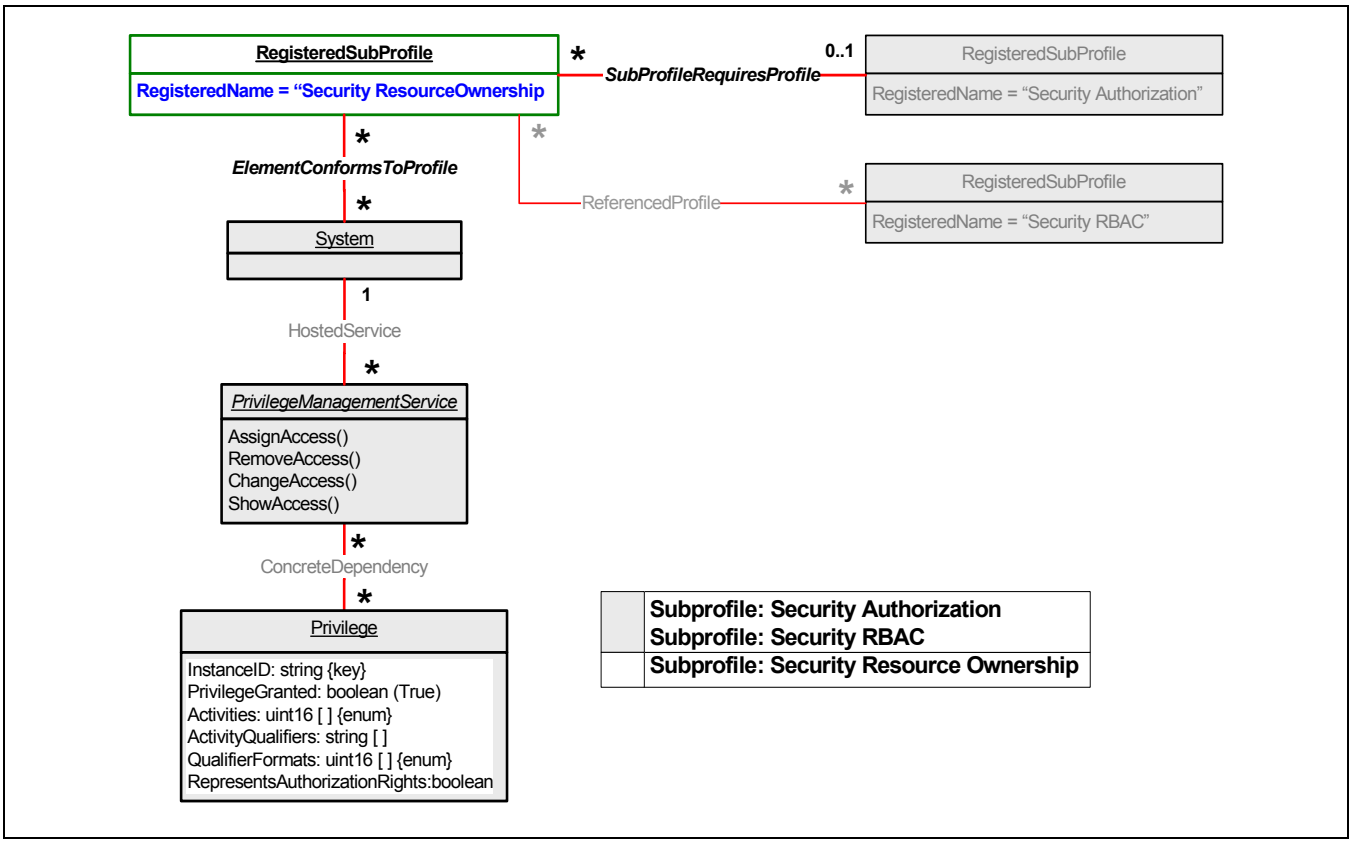

Figure 74 - Security Resource Ownership

.The model is permission-based (i.e., represents allowed operations, as opposed to forbidden ones). Where used, the policy is to deny all rights except those explicitly granted. Specific details of how the Security Resource Ownership Subprofile is applied are specified in the Resource Ownership Considerations subsection of the Cascading Considerations section of the including profile; this includes definition of the contents of the Privilege instances and definition of any propagation rules. The key class in Security Resource Ownership is the Privilege class that is used to grant rights to subjects (for instance, the identity of an embedded CIM client) to act on targets (resources that can be manipulated.)

3. The Security Resource Ownership subprofile was formerly known as Ownership. It has been renamed to avoid confusion with the notion of file owner commonly found in filesystems.

Support for the ShowAccess method is mandatory. It is used to extract which rights have been granted to a subject entity for a particular target resource. The implementation may also make this explicit by instantiating AuthorizedPrivilege instances with appropriate AuthorizedSubject and AuthorizedTarget associations.

An important aspect of this class is the RepresentsAuthorizationRights property:

- A Privilege with RepresentsAuthorizationRights = FALSE is an access privilege that controls invocation of CIM operations. The basic operation of an access privilege is that only the authorized subject identities can perform the Activities (including qualifiers) in the privilege on the authorized target(s).
- A Privilege with RepresentsAuthorizationRights = TRUE is a resource ownership privilege that controls the ability to associate access privileges with objects. The basic operation of an ownership privilege is to control the association of access privileges to target resource; for the Activities (including qualifiers) listed in the ownership privilege. Only authorized subjects of the ownership privilege are permitted to associate an access privilege containing any of those Activities with any target of the ownership privilege. An object that is an authorized target of an ownership privilege is called an owned resource.

An object can be subject to operation restrictions imposed by this subprofile only when it is an owned resource (i.e., the target of a resource ownership privilege). The algorithm is:

1) In the absence of an ownership privilege on a resource, any client may assign access privileges to that resource.
2) If an object is an owned resource (the target of a resource ownership privilege) then only subjects represented by owning Identities may assign access rights covered by the ownership Privilege instance to that resource.
3) In the absence of an access privilege on a resource, all clients are granted Read and Detect access (see the CIM Authorization model for information on the intrinsic operations covered by Read and Detect). All other access is denied.
4) All object reference parameters of each extrinsic method shall be checked; it is not sufficient to check only the first object reference parameter on the theory that the extrinsic is invoked on that object.
5) When Security Resource Ownership is in use, the CIM Client shall authenticate to the CIMOM to prevent misuse of Identity; an unauthenticated CIM Client will not be able to invoke any operation that is restricted by an access privilege.

For an object to be both owned and manageable via the controlling CIM Client, that object needs to be the target of a resource ownership privilege (for the ownership rights) and an access privilege (to allow management operations).

To enable future flexibility and (hopefully) minimize the opportunity for client programming errors, a resource supporting the Security Resource Ownership Subprofile shall either:

1) Instantiate one or more ownership Privilege instances containing allowable sets of rights to be granted. These are associated to the PrivilegeManagementService via ConcreteDependency associations. To assign ownership, the RepresentsAuthorizationRights property shall be set to TRUE in a copy of a Privilege instance passed in the ChangeAccess method. Otherwise, access rights are defined.
2) Instantiate one or more Role instances having ownership Privilege instances associated via MemberOfCollection. As above, these Privilege instances contain allowable sets of rights to be granted. Unless the Role applies to all resources in the System, the Role instances shall be associated to applicable resources via RoleLimitedToTarget. The infrastructure may restrict the ability of the client to modify Role instances, including associations and associated Privileges. To assign ownership, a Role with Privileges, associated by MemberOfCollection, that have RepresentsAuthorizationRights set to TRUE, shall be associated via MemberOfCollection to one or more Identity instances. Each selected Identity instance shall be associated via ServiceAffectsElement to PrivilegeManagementService that is also associated to the resource via ServiceAffectsElement.

Privilege propagation rules, as defined by an instance of PrivilegePropagationRule, is a means of specifying how rights are propagated by a ChangeAccess call. The infrastructure may publish available propagation strategies via instances of PrivilegePropagationRule associated to a resource via PolicySetAppliesToElement associations. Alternatively, a Profile may define a set of “well-known” PrivilegePropagationRules that apply to particular types of resources and which may be discovered via enumeration. In either case, these available rules may be referenced in a ChangeAccess method.

51.1.1 Design Considerations

ServiceAffectsElement associations are assumed between Services and affected elements. (See Figure 75.) This subprofile does not require an implementation to present these associations unless there is more than one PrivilegeManagementService in the profiled Namespace.

ServiceAvailableToElement associations are assumed between Services and using elements (See Figure 75.) This subprofile does not require an implementation to present these associations unless there is more than one System in the profiled Namespace.

AuthorizedPrivilege instances are assumed when a Privilege is granted to a subject or assigned to a target. (See Figure 76.) AuthorizedTarget and AuthorizedSubject associations are assumed between the AuthorizedPrivilege and the target and subject entities respectively. This subprofile does not require the implementation to make these instances explicit. Instead this profile relies on the ChangeAccess method to grant or deny rights and on the ShowAccess method to display rights.

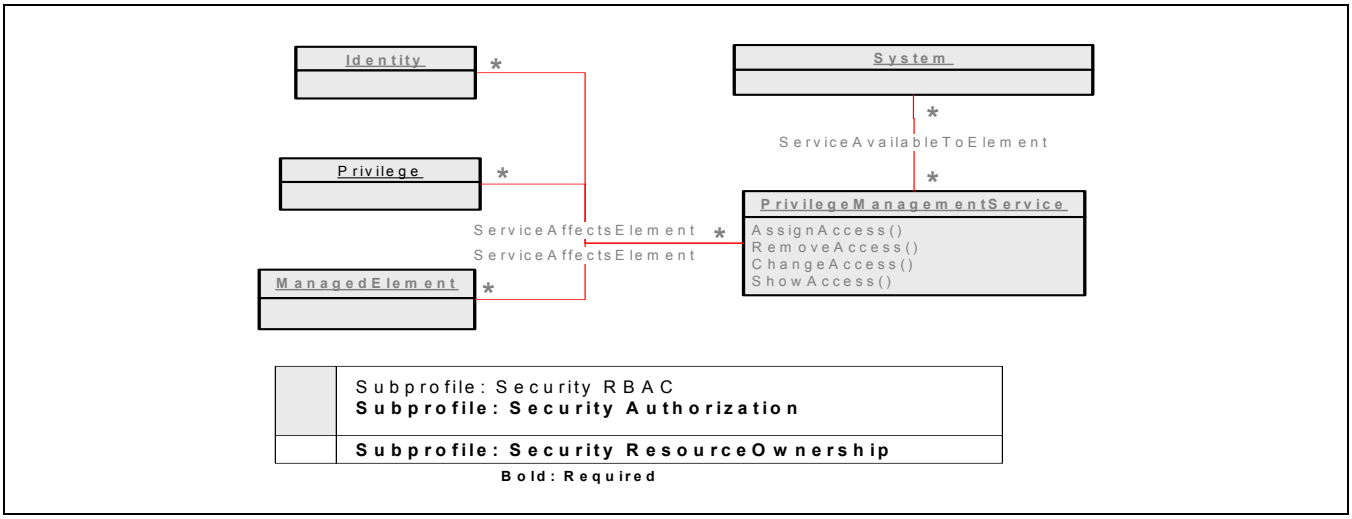

Figure 75 - Service Associations

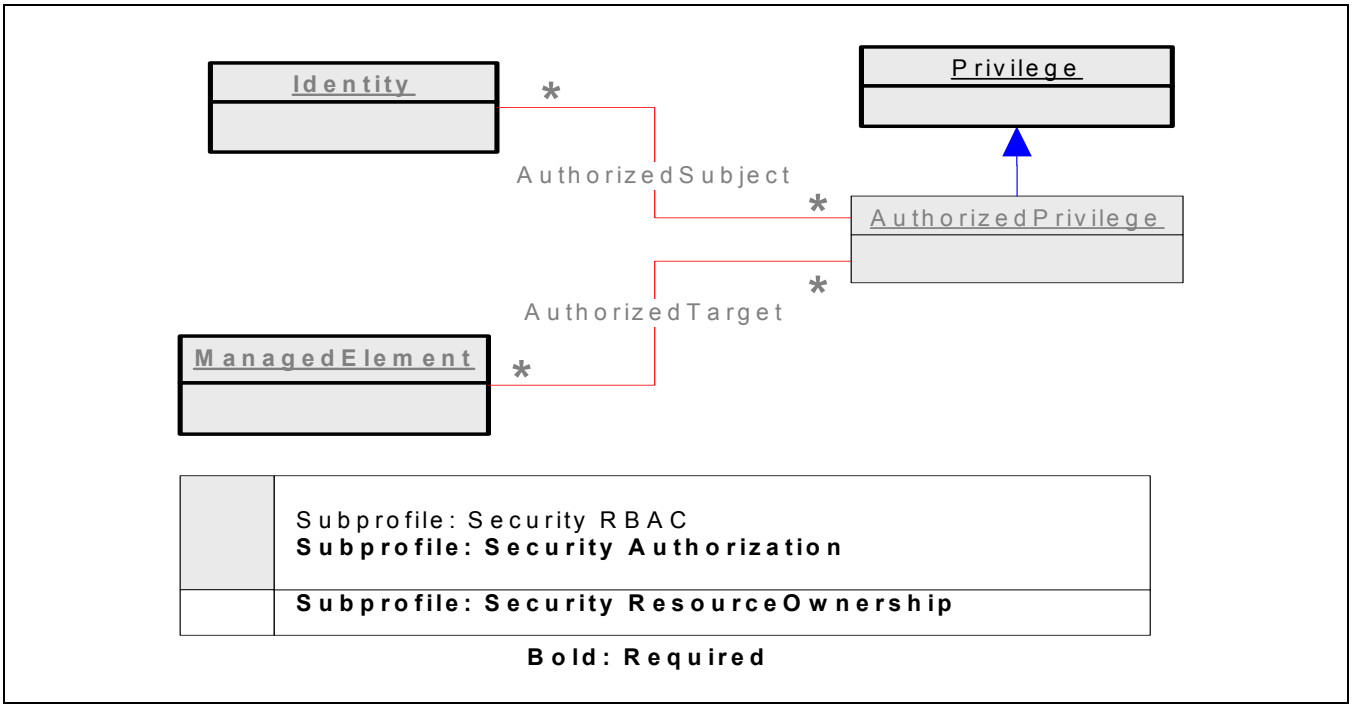

Figure 76 - AuthorizedPrivilege

51.2 Health and Fault Management Considerations

Not defined in this standard.

51.3 Cascading Considerations

Not defined in this standard.

51.4 Supported Subprofiles and Packages

None.

51.5 Methods of the Profile

None.

51.6 Client Considerations and Recipes

51.6.1 Show Ownership Rights

```
// DESCRIPTION
// List the Subjects that have authorization rights to a resource.
// These subjects have ownership for the associated privileges.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// $Resource-> contains a reference to a resource (Any Managed Element)
// $PMS-> contains a reference to the PrivilegeManagementService
```

```
//

// Get Privileges for resource
//
#result = $PMS->ShowAccess(,$Resource->, $OutSubject->[], null, $OutPrivilege[])

// Verify that the operation performed successfully.
if (#Result != 0) {
    <EXIT! Show access for the specified resource failed>
}

// Filter out the non authorization rights
//
#k = 0
for #j in $OutPrivilege[] {
    if ($OutPrivilege[#j].RepresentsAuthorizationRights = True) {
     #k++
     $Subject->[#k] = $OutSubject->[#j]
     $Privilege->[#k] = $OutPrivilege->[#j]
    }
}



//
// $Resource-> contains resource
// $Subject->[] contains array of references to Identities (or other subjects),
//   with Authorization rights to a resource
// $Privilege[] contains array of Privileges, corresponding to the subject
                           entries.
//
```

51.6.2 Deny ownership rights

51.6.3 // DESCRIPTION

// Remove a set of authorization rights, (represented by a Privilege), from a named

// Subject for a resource.

//

// PRE-EXISTING CONDITIONS AND ASSUMPTIONS

// The calling subject MUST be an owner for the named set of rights.

// Note: A resource is typically represented by an instance of some type of

// CIM_ManagedElement. Conceptually, a resource could also be an association instance.

// It is up to referencing Profiles to apply any additional constraints on the types of

// instances that are considered to be resource.

```
//
// $Identity-> contains a reference to a subject Identity
// $Resource-> contains a reference to a resource
// $Privilege contains a Privilege
// $PMS-> contains a reference to the PrivilegeManagementService
//
// This recipe is NOT dealing with Privilege Propagation.
//

// Set the Privilege to eliminate all rights
//
$Privilege[1] = $Privilege
$Privilege[1].PrivilegeGranted = False
$Privilege[1].RepresentsAuthorizationRights = True

// Eliminate all rights to the resource.
// Note that we don't care whether someone else did it already.
//
#result = $PMS->ChangeAccess($Identity->,$Resource->,null,$Privilege[])

// Verify that the operation performed successfully.
if (#Result != 0) {
   <EXIT! Changing access for the specified resource failed>
}

// $Privilege[] contains the result array of Privileges between the subject and target
//
Grant ownership rights
            // DESCRIPTION
            // Give a named Subject a set of authorization rights,
            //    (represented by a Privilege) for a resource.
            //
            // PRE-EXISTING CONDITIONS AND ASSUMPTIONS
```

```
// The calling subject MUST be an owner.
//  This call also makes the named subject an owner.
//  The assumption is that the calling subject trusts the named subject.
//
// $Identity-> contains a reference to a subject Identity
// $Resource-> contains a reference to a resource
// $Privilege contains a Privilege to be granted
// $PMS-> contains a reference to the PrivilegeManagementService
//
// This recipe is NOT dealing with Privilege Propagation.
//

// Set the Privilege
//
$Privilege[1] = $Privilege
$Privilege[1].PrivilegeGranted = True
$Privilege[1].RepresentsAuthorizationRights = True

#result = $PMS->ChangeAccess($Identity->,$Resource->,null,$Privilege[])

// Verify that the operation performed successfully.
if (#Result != 0) {
    <EXIT! Changing access for the specified resource failed>
}

// $Privilege[] contains the result array of Privileges between the subject and
                            target
//
```

51.7 Registered Name and Version

Security Resource Ownership version 1.3.0

51.8 CIM Elements

Table 543 describes the CIM elements for Security Resource Ownership.

Table 543 - CIM Elements for Security Resource Ownership

Element Name	Requirement	Description
51.8.1 CIM_AuthorizationRule	Optional	
51.8.2 CIM_AuthorizationRuleAppliesToIdentity	Optional	
51.8.3 CIM_AuthorizationRuleAppliesToPrivilege	Optional	
51.8.4 CIM_AuthorizationRuleAppliesToRole	Optional	

Table 543 - CIM Elements for Security Resource Ownership

Element Name	Requirement	Description
51.8.5 CIM_AuthorizationRuleAppliesToTarget	Optional	
51.8.6 CIM_AuthorizedPrivilege	Optional	
51.8.7 CIM_AuthorizedSubject	Optional	
51.8.8 CIM_AuthorizedTarget	Optional	
51.8.9 CIM_ConcreteDependency (Service to AuthorizedPrivilege)	Optional	
51.8.10 CIM_ConcreteDependency (Service to Privilege)	Optional	
51.8.11 CIM_HostedService	Mandatory	
51.8.12 CIM_Identity	Optional	
51.8.13 CIM_ManagedElement	Optional	
51.8.14 CIM_MemberOfCollection (AuthorizedPrivilege to Role)	Optional	
51.8.15 CIM_MemberOfCollection (Identity to Role)	Optional	
51.8.16 CIM_MemberOfCollection (Privilege to Role)	Optional	
51.8.17 CIM_MemberOfCollection (Role to Role)	Optional	
51.8.18 CIM_OwningCollectionElement	Optional	
51.8.19 CIM_PolicyRuleInSystem (System to AuthorizationRule)	Optional	
51.8.20 CIM_PolicyRuleInSystem (System to PrivilegePropogationRule)	Optional	
51.8.21 CIM_PolicySetAppliesToElement	Optional	
51.8.22 CIM_Privilege	Optional	
51.8.23 CIM_PrivilegeManagementService	Mandatory	
51.8.24 CIM_PrivilegePropagationRule	Optional	
51.8.25 CIM_Role	Optional	
51.8.26 CIM_RoleLimitedToTarget	Optional	
51.8.27 CIM_ServiceAffectsElement (Service to AuthorizedPrivilege)	Optional	
51.8.28 CIM_ServiceAffectsElement (Service to Identity)	Optional	

Table 543 - CIM Elements for Security Resource Ownership

Element Name	Requirement	Description
51.8.29 CIM_ServiceAffectsElement (Service to ManagedElement)	Optional	
51.8.30 CIM_ServiceAffectsElement (Service to Privilege)	Optional	
51.8.31 CIM_ServiceAvailableToElement	Mandatory	

51.8.1 CIM_AuthorizationRule

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

Table 544 describes class CIM_AuthorizationRule.

Table 544 - SMI Referenced Properties/Methods for CIM_AuthorizationRule

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

51.8.2 CIM_AuthorizationRuleAppliesToIdentity

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

Table 545 describes class CIM_AuthorizationRuleAppliesToIdentity.

Table 545 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToIdentity

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

51.8.3 CIM_AuthorizationRuleAppliesToPrivilege

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

Table 546 describes class CIM_AuthorizationRuleAppliesToPrivilege.

Table 546 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToPrivilege

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

51.8.4 CIM_AuthorizationRuleAppliesToRole

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

Table 547 describes class CIM_AuthorizationRuleAppliesToRole.

Table 547 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToRole

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

51.8.5 CIM_AuthorizationRuleAppliesToTarget

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

Table 548 describes class CIM_AuthorizationRuleAppliesToTarget.

Table 548 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToTarget

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

51.8.6 CIM_AuthorizedPrivilege

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

Table 549 describes class CIM_AuthorizedPrivilege.

Table 549 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
RepresentsAuthorizationRights		Mandatory	Must be an Access right for this subprofile.
PrivilegeGranted		Mandatory	Only Grant type privileges are allowed.

51.8.7 CIM_AuthorizedSubject

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

Table 550 describes class CIM_AuthorizedSubject.

Table 550 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject

Properties	Flags	Requirement	Description & Notes
Privilege		Mandatory	
PrivilegedElement		Mandatory	

51.8.8 CIM_AuthorizedTarget

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 551 describes class CIM_AuthorizedTarget.

Table 551 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget

Properties	Flags	Requirement	Description & Notes
Privilege		Mandatory	
TargetElement		Mandatory	

51.8.9 CIM_ConcreteDependency (Service to AuthorizedPrivilege)

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

Table 552 describes class CIM_ConcreteDependency (Service to AuthorizedPrivilege).

Table 552 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Service to AuthorizedPrivilege)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

51.8.10 CIM_ConcreteDependency (Service to Privilege)

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

Table 553 describes class CIM_ConcreteDependency (Service to Privilege).

Table 553 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Service to Privilege)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

51.8.11 CIM_HostedService

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

Table 554 describes class CIM_HostedService.

Table 554 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

51.8.12 CIM_Identity

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

Table 555 describes class CIM_Identity.

Table 555 - SMI Referenced Properties/Methods for CIM_Identity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	

51.8.13 CIM_ManagedElement

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

51.8.14 CIM_MemberOfCollection (AuthorizedPrivilege to Role)

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Optional

Table 556 describes class CIM_MemberOfCollection (AuthorizedPrivilege to Role).

Table 556 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (AuthorizedPrivilege to Role)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

51.8.15 CIM_MemberOfCollection (Identity to Role)

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

Table 557 describes class CIM_MemberOfCollection (Identity to Role).

Table 557 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Identity to Role)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

51.8.16 CIM_MemberOfCollection (Privilege to Role)

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

Table 558 describes class CIM_MemberOfCollection (Privilege to Role).

Table 558 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Privilege to Role)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

51.8.17 CIM_MemberOfCollection (Role to Role)

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

Table 559 describes class CIM_MemberOfCollection (Role to Role).

Table 559 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Role to Role)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

51.8.18 CIM_OwningCollectionElement

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

Table 560 describes class CIM_OwningCollectionElement.

Table 560 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

51.8.19 CIM_PolicyRuleInSystem (System to AuthorizationRule)

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

Table 561 describes class CIM_PolicyRuleInSystem (System to AuthorizationRule).

Table 561 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem (System to AuthorizationRule)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

51.8.20 CIM_PolicyRuleInSystem (System to PrivilegePropogationRule)

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

Table 562 describes class CIM_PolicyRuleInSystem (System to PrivilegePropogationRule).

Table 562 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem (System to PrivilegePropogationRule)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

51.8.21 CIM_PolicySetAppliesToElement

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

Table 563 describes class CIM_PolicySetAppliesToElement.

Table 563 - SMI Referenced Properties/Methods for CIM_PolicySetAppliesToElement

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

51.8.22 CIM_Privilege

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 564 describes class CIM_Privilege.

Table 564 - SMI Referenced Properties/Methods for CIM_Privilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
RepresentsAuthorizationRights		Mandatory	Must be an Access right for this subprofile.
PrivilegeGranted		Mandatory	Only Grant type privileges are allowed.

51.8.23 CIM_PrivilegeManagementService

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

Table 565 describes class CIM_PrivilegeManagementService.

Table 565 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
CreationClassName		Mandatory	Key
Name		Mandatory	Key
ChangeAccess()		Optional	

51.8.24 CIM_PrivilegePropagationRule

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

Table 566 describes class CIM_PrivilegePropagationRule.

Table 566 - SMI Referenced Properties/Methods for CIM_PrivilegePropagationRule

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

51.8.25 CIM_Role

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

Table 567 describes class CIM_Role.

Table 567 - SMI Referenced Properties/Methods for CIM_Role

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

51.8.26 CIM_RoleLimitedToTarget

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

Table 568 describes class CIM_RoleLimitedToTarget.

Table 568 - SMI Referenced Properties/Methods for CIM_RoleLimitedToTarget

Properties	Flags	Requirement	Description & Notes
TargetElement		Mandatory	
DefiningRole		Mandatory	

51.8.27 CIM_ServiceAffectsElement (Service to AuthorizedPrivilege)

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

Table 569 describes class CIM_ServiceAffectsElement (Service to AuthorizedPrivilege).

Table 569 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to AuthorizedPrivilege)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

51.8.28 CIM_ServiceAffectsElement (Service to Identity)

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

Table 570 describes class CIM_ServiceAffectsElement (Service to Identity).

Table 570 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to Identity)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

51.8.29 CIM_ServiceAffectsElement (Service to ManagedElement)

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

Table 571 describes class CIM_ServiceAffectsElement (Service to ManagedElement).

Table 571 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to ManagedElement)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

51.8.30 CIM_ServiceAffectsElement (Service to Privilege)

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

Table 572 describes class CIM_ServiceAffectsElement (Service to Privilege).

Table 572 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to Privilege)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

51.8.31 CIM_ServiceAvailableToElement

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

Table 573 describes class CIM_ServiceAvailableToElement.

Table 573 - SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement

Properties	Flags	Requirement	Description & Notes
ServiceProvided		Mandatory	
UserOfService		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

Clause 52: Security Role Based Access Control Subprofile

52.1 Description

52.1.1 Overview

The Role Based Access Control (RBAC.) subprofile enables management of authorization using RBAC Roles, (see Figure 77). The Security RBAC subprofile is a subprofile of the Security Authorization subprofile.

If this subprofile is supported, the CIM Server may publish some number of Roles via OwningCollectionElement associations to the top level System. Rights are granted to a Role by Privilege instances associated via MemberOfCollection. Target resources are associated to a Role via RoleLimitedToTarget associations.

If a subject Identity is associated to a Role via MemberOfCollection and if CurrentlyAuthenticated is true, then the entity named by the Identity is authorized to exercise all rights granted by the Role to target resources.

If there are no RoleLimitedToTarget associations, then the Role applies to all resources in the System. If there are RoleLimitedToTarget association, then those associations identify the target resources of the role.

A Role may collect other Roles via MemberOfCollection. Privileges of the included Role are granted to Identities of the including Role for those resources that are scoped to both Roles.

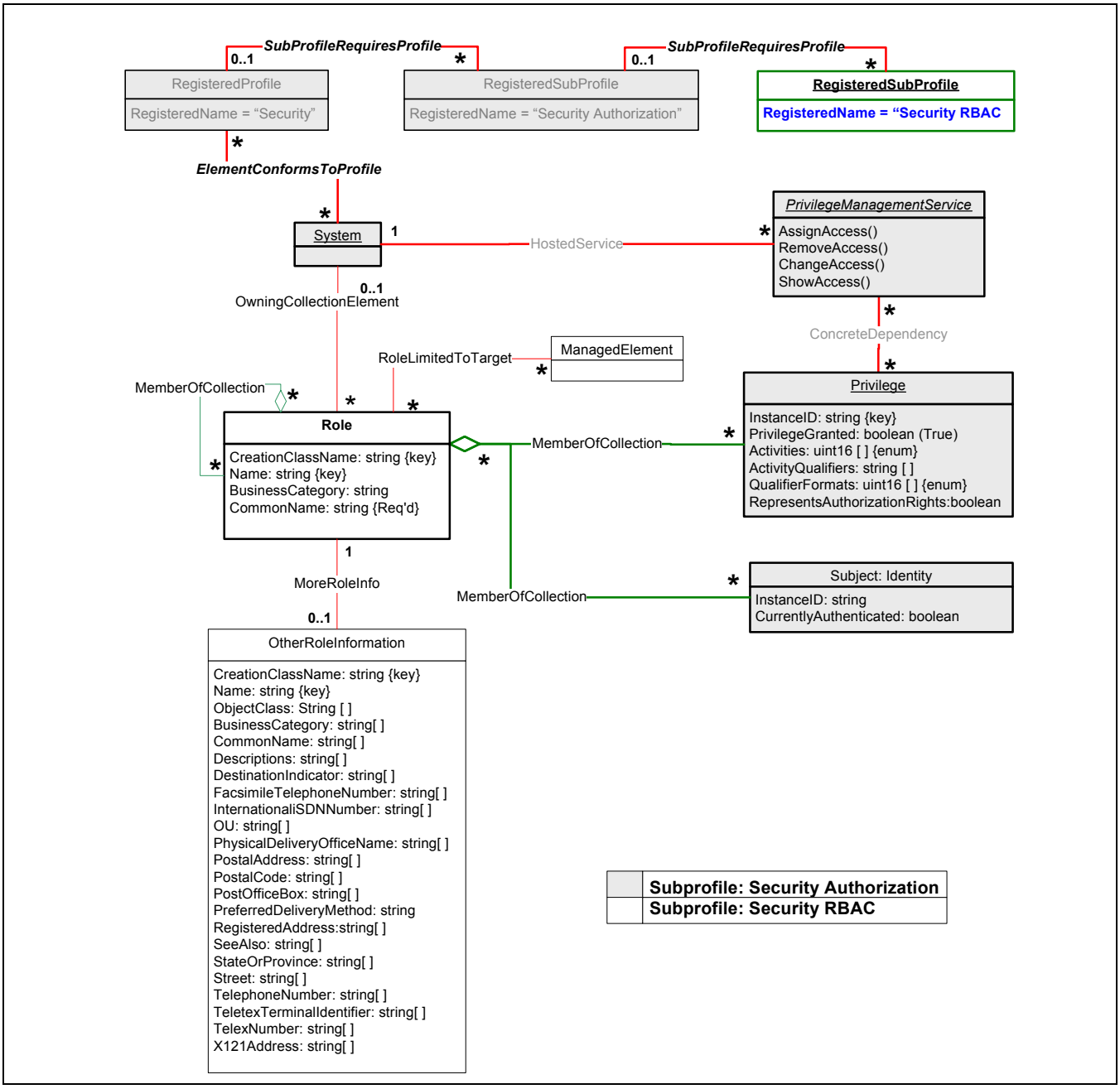

Figure 77 - Role-Based Access Control

52.1.2 Default Authorization

The ChangeAccess method is not used to grant or deny authorization via Roles. Rather, this subprofile uses CIM Intrinsic methods CreateInstance and DeleteInstance on appropriate associations and on the Role class itself. The following describes the

- All resources of a system conforming to this subprofile are scoped to any Role with no RoleLimitedToTarget associations.

- Only resources associated by RoleLimitedToTarget are scoped to a Role with RoleLimitedToTarget associations. CreateInstance and DeleteInstance are used to add or delete RoleLimitedToTarget associations.
- MemberOfCollection associations are used to grant Privileges to a Role. CreateInstance and DeleteInstance are used to add or delete MemberOfCollection associations between Privilege and Role instances.
- MemberOfCollection associations are used to place Identities into a Role. CreateInstance and DeleteInstance are used to add or delete MemberOfCollection associations between Identity and Role instances.
- Every Identity in a Role is authorized with all rights defined by all Privileges granted to the Role for all resources scoped to the Role. The set of authorized rights is adjusted dynamically as a result of CreateInstance and DeleteInstance operations on the MemberOfCollection and RoleLimitedToTarget associations described above.
- MemberOfCollection associations are used to incorporate one Role into another Role. CreateInstance and DeleteInstance are used to add or delete MemberOfCollection associations between Role instances. The following additional rules apply:
  - Identities of the incorporating Role are authorized with all rights defined by all Privileges granted to the incorporated Role for all resources that are scoped to the intersection of the set of resources scoped to each Role.
  - This process is recursive through the MemberOfCollection association between Roles with the added conditions that:
    - At each level, the intersecting set of resources found at level *n* is intersected with the set of resources scoped to level *n+1*.
    - This intersection forms the set of resources to which the Identities of level *1* are authorized with the Privileges of level *n+1*.

52.1.3 Authorization Policy

This subprofile extends the Authorization Policy defined in the Security Authorization subprofile.

In addition associations specified in the Security Authorization subprofile. AuthenticationRuleAppliesToRole may be used to incorporate a Role into an AuthenticationRule. This is shown in Figure 78.

The details of the specification of AuthorizationRules are left to the profiles and subprofiles that reference this subprofile.

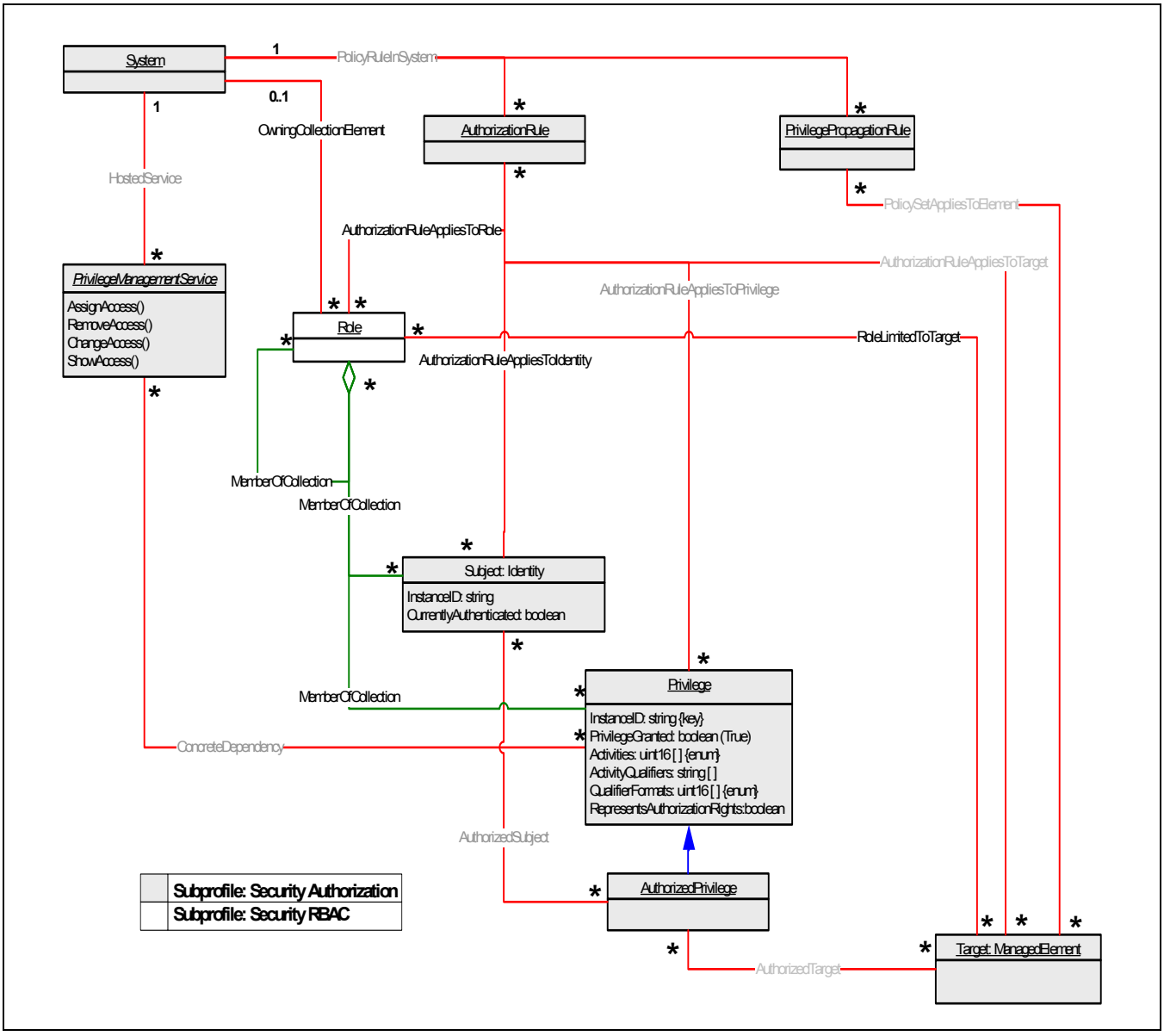

Figure 78 - Policy Rules

52.1.4 Design Considerations

ConcreteDependency associations are assumed between Services and the elements that they directly manage (See Figure 79.) This subprofile does not REQUIRE an implementation to present these associations unless there is more than one PrivilegeManagementService in the profiled Namespace.

ServiceAffectsElement associations are assumed between Services and affected elements. (See Figure 79.) This

subprofile does not REQUIRE an implementation to present these associations unless there is more than one PrivilegeManagementService in the profiled Namespace.

ServiceAvailableToElement associations are assumed between Services and using elements (See Figure 79.) This subprofile does not REQUIRE an implementation to present these associations unless there is more than one System in the profiled Namespace.

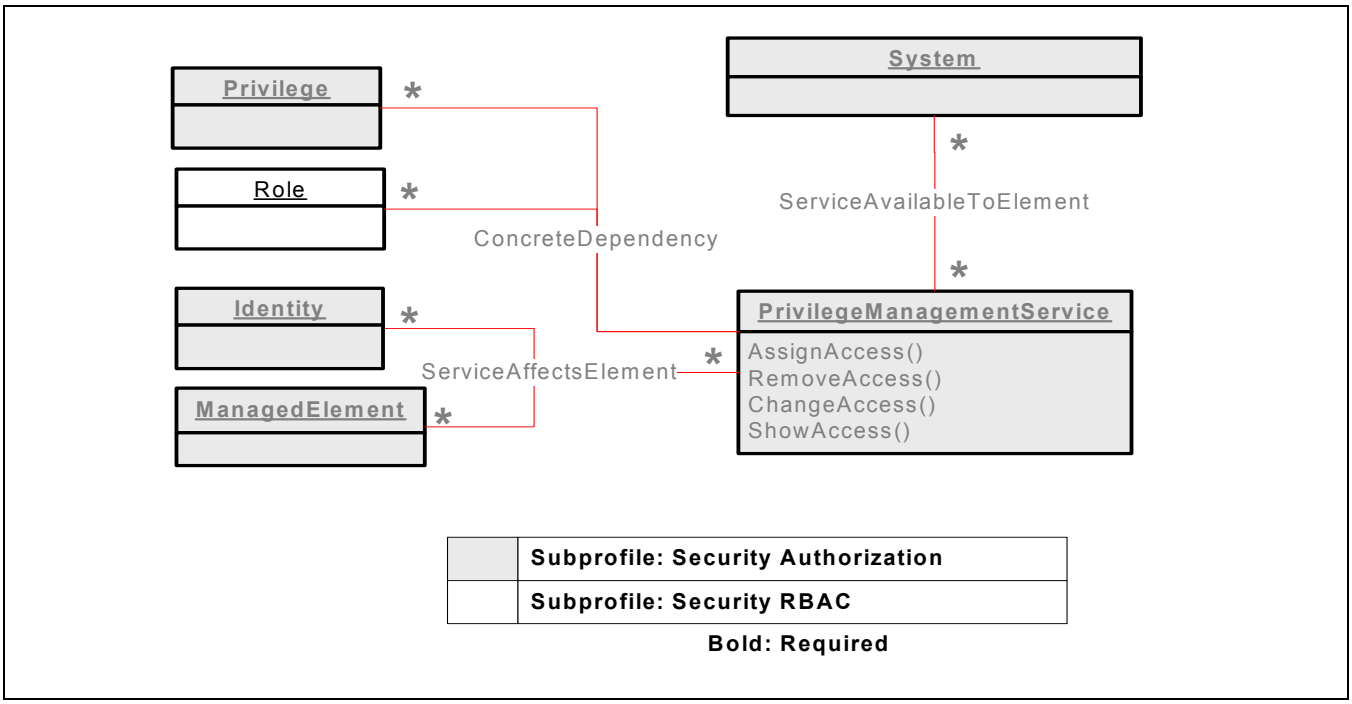

Figure 79 - Service Associations

This subprofile does not require the implementation to make AuthorizedPrivilege instances explicit. However, there existence is assumed whenever a Role containing one or more Privileges is associated by MemberOfCollection to an Identity.

- In the case where there is no RoleLimitedToTarget association, then all ManagedElements are implicitly authorized to the collected Identity instances.
- If RoleLimitedToTarget associations are used, then only those ManagedElements are authorized. Figure 80 shows this case.
- Additionally Figure 80 shows the case where a Role is collected into another role. Only the intersection of target resources between the included and including Roles are granted permission for Identities of the including Role. For example, in Figure 80, note that Identity B does not become authorized to

ManagedElement A. However, Identity A does become authorized to ManagedElement AB.This subprofile relies on the ShowAccess method to display rights the rights granted by membership in a Role.

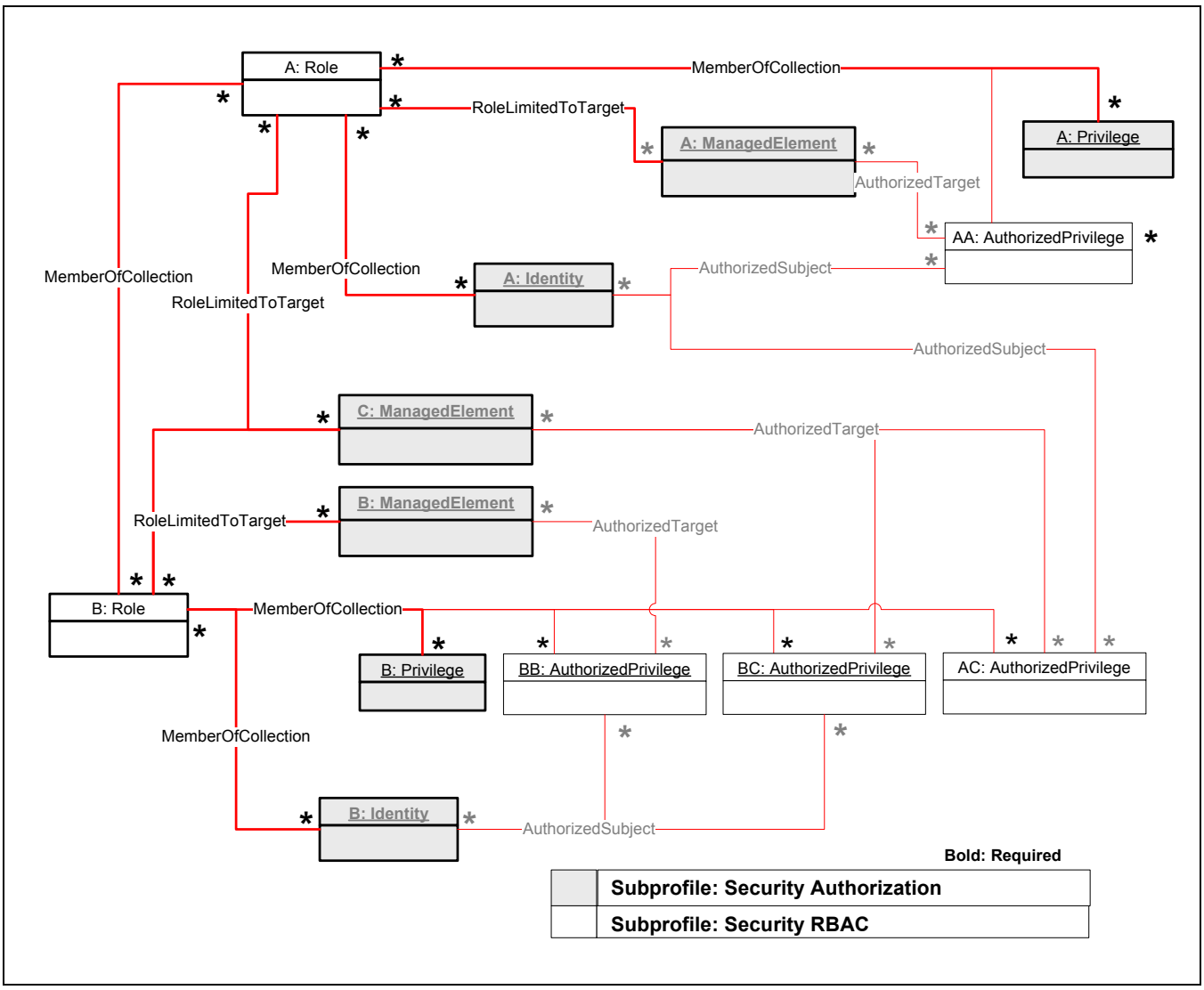

Figure 80 - AuthorizedPrivilege

52.2 Health and Fault Management Consideration

Not defined in this standard.

52.3 Cascading Considerations

Not defined in this standard.

52.4 Supported Subprofiles and Packages

None.

52.5 Methods of the Profile

None.

52.6 Client Considerations and Recipes

52.6.1 List the Roles associated with an Identity

```
// DESCRIPTION
// For a specific Identity, this recipe lists all associated Roles
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// $Identity-> contains a reference to an Identity
//

//===================================================================
// Subroutines of SecurityRBAC 1
//===================================================================
Sub GetMemberRoles($StartRoles->[], $Roles->[])
{
    // Get Member Roles
    for #i in $StartRoles->[]
    {
        $MemberRoles->[] = AssociatorNames($StartRoles->[#i],

                           “CIM_MemberOfCollection”,”CIM_Role”,Collection,)

        // Append Member Roles to Roles output.
     // Note that on the first iteration size of Roles is 0.
     //       On the next interation Roles.size is now size of previous
                           MemberRoles.
        //
        #i = $Roles->[].size
        for #j in $MemberRoles->[]
        {
            $Roles->[(#i+#j] = $MemberRoles->[#j]
        }
     //  Get Members of Members
     //
        &GetMemberRoles($MemberRoles->[], Roles->[])
}



//===================================================================
//SecurityRBAC 1 Recipe starts here
//===================================================================

// Find the first-level Roles of an Identity.
//
$Roles->[] = AssociatorNames($Identity>,
                           “CIM_MemberOfCollection”,”CIM_Role”,Member,)

//Append Member Roles
```

```
&GetMemberRoles($Roles->[], $Roles->[])

// ON OUTPUT
//
// $Roles->[] contains a list of pointers to Roles
//
```

52.6.2 List the Privileges of a Role

```
// DESCRIPTION
// For a specific Role, this recipe lists all associated Privileges obtained
// via membership in various Roles.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// $Role-> contains a reference to a Role
//

//===============================================================
// Subroutines of SecurityRBAC 2
//===============================================================
Sub GetMemberPrivileges($StartRoles->[], $Roles->[], $Privileges->[])
{
    // Get Member Roles
    for #i in $StartRoles->[]
    {
        $MemberRoles->[] = AssociatorNames($StartRoles->[#i],

                           "CIM_MemberOfCollection","CIM_Role",Collection,)

        // Append Member Roles to Roles output.
     // Note that on the first iteration size of Roles is 0.
     //      On the next interation Roles.size is now size of previous
     // MemberRoles.
        //
        #i = $Roles->[].size
        for #j in $MemberRoles->[]
        {
            $Roles->[#i+#j] = $MemberRoles->[#j]

            // Now append the Privileges for each member
         //
            $MemberPrivs->[] = AssociatorNames(&MemberRoles-[#j],
        "CIM_MemberOfCollection","CIM_Privilege",Collection,)
        #k = $Privileges->[].size
            for #l in $MemberPrivs->[]
            {
                $Privileges->[#k+#l] = $MemberPrivs->[#l]
            }
        }
```

```
        //  Get Members of Members
        //
           &GetMemberRoles($MemberRoles->[], Roles->[], $Privileges->[])
}


//===============================================================
//SecurityRBAC 2 Recipe starts here
//===============================================================

// Find the first-level Privileges
//
    $Privileges->[] = AssociatorNames($Role->,

                            "CIM_MemberOfCollection","CIM_Privilege",Collection,)

//Append Member Privileges
$Roles->[1] = $Role->&GetMemberPrivileges($Roles->[], $Roles->[], $Privileges->[])

// ON OUTPUT
//
// $Roles->[] contains a list of pointers to Roles in the Role hierarchy
// $Privileges->[] contains a list of pointers to Privileges from the Role
                            hierachy
//
```

52.7 Registered Name and Version

Security RBAC version 1.1.0

52.8 CIM Elements

Table 574 describes the CIM elements for Security RBAC.

Table 574 - CIM Elements for Security RBAC

Element Name	Requirement	Description
52.8.1 CIM_AuthorizationRule	Optional	
52.8.2 CIM_AuthorizationRuleAppliesToRole	Optional	
52.8.3 CIM_ConcreteDependency	Mandatory	
52.8.4 CIM_HostedService	Mandatory	
52.8.5 CIM_Identity	Optional	
52.8.6 CIM_ManagedElement	Optional	
52.8.7 CIM_MemberOfCollection	Optional	
52.8.8 CIM_MoreRoleInfo	Optional	

Table 574 - CIM Elements for Security RBAC

Element Name	Requirement	Description
52.8.9 CIM_OtherRoleInformation	Optional	
52.8.10 CIM_OwningCollectionElement	Optional	
52.8.11 CIM_PolicyRuleInSystem	Optional	
52.8.12 CIM_Privilege	Optional	
52.8.13 CIM_PrivilegeManagementService	Optional	
52.8.14 CIM_Role	Optional	
52.8.15 CIM_RoleLimitedToTarget	Optional	
52.8.16 CIM_System	Mandatory	

52.8.1 CIM_AuthorizationRule

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

Table 575 describes class CIM_AuthorizationRule.

Table 575 - SMI Referenced Properties/Methods for CIM_AuthorizationRule

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

52.8.2 CIM_AuthorizationRuleAppliesToRole

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

Table 576 describes class CIM_AuthorizationRuleAppliesToRole.

Table 576 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToRole

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

52.8.3 CIM_ConcreteDependency

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

Table 577 describes class CIM_ConcreteDependency.

Table 577 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

52.8.4 CIM_HostedService

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

Table 578 describes class CIM_HostedService.

Table 578 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

52.8.5 CIM_Identity

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

Requirement: Optional

Table 579 describes class CIM_Identity.

Table 579 - SMI Referenced Properties/Methods for CIM_Identity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	Entity is authenticated to use this Identity.

52.8.6 CIM_ManagedElement

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

52.8.7 CIM_MemberOfCollection

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

Table 580 describes class CIM_MemberOfCollection.

Table 580 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

52.8.8 CIM_MoreRoleInfo

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

Table 581 describes class CIM_MoreRoleInfo.

Table 581 - SMI Referenced Properties/Methods for CIM_MoreRoleInfo

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

52.8.9 CIM_OtherRoleInformation

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

Table 582 describes class CIM_OtherRoleInformation.

Table 582 - SMI Referenced Properties/Methods for CIM_OtherRoleInformation

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key, Must match that of Role

52.8.10 CIM_OwningCollectionElement

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

Table 583 describes class CIM_OwningCollectionElement.

Table 583 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

52.8.11 CIM_PolicyRuleInSystem

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

Requirement: Optional

Table 584 describes class CIM_PolicyRuleInSystem.

Table 584 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

52.8.12 CIM_Privilege

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

Table 585 describes class CIM_Privilege.

Table 585 - SMI Referenced Properties/Methods for CIM_Privilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
RepresentsAuthorizationRights		Mandatory	Rights are to assign rights.
PrivilegeGranted		Mandatory	Instantiated Privileges will only be granted.

52.8.13 CIM_PrivilegeManagementService

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

Table 586 describes class CIM_PrivilegeManagementService.

Table 586 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

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

Table 586 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

52.8.14 CIM_Role

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

Table 587 describes class CIM_Role.

Table 587 - SMI Referenced Properties/Methods for CIM_Role

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

52.8.15 CIM_RoleLimitedToTarget

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

Table 588 describes class CIM_RoleLimitedToTarget.

Table 588 - SMI Referenced Properties/Methods for CIM_RoleLimitedToTarget

Properties	Flags	Requirement	Description & Notes
TargetElement		Mandatory	
DefiningRole		Mandatory	

52.8.16 CIM_System

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

Table 589 describes class CIM_System.

Table 589 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

EXPERIMENTAL

EXPERIMENTAL

Clause 53: IdentityManagement Subprofile

53.1 Description

This subprofile of the Security profile provides support for adding and managing users of a system and for mapping those users to accounts, people and organizations.

Users are assumed to have Identity instances to represent their ability to be authenticated. Identity instances may stand alone or may be linked to Accounts, Organizations, OrgUnits, UserContacts, Persons, Groups or Roles.

All Identity instances shall be unique within the namespace of the conformant System.

53.1.1 Identities

Identities represent a user of a system and when authenticated, represent a security principal.'

Authentication is performed by an authentication service which may be represented as an AuthenticationService. If represented, this specification relies on the implementation to instantiate appropriate ServiceAffectsElement associations between the AuthenticationService and an Identities.

If there are multiple Systems in the namespace, and the Identity is scoped to a particular System, then IdentityContext associations shall be instantiated between the Identity and the scoping System. CreateInstance and DeleteInstance may be used to instantiate IdentityContext associations. IdentityContext instances shall be deleted by the infrastructure as a side-affect of deleting an Identity.

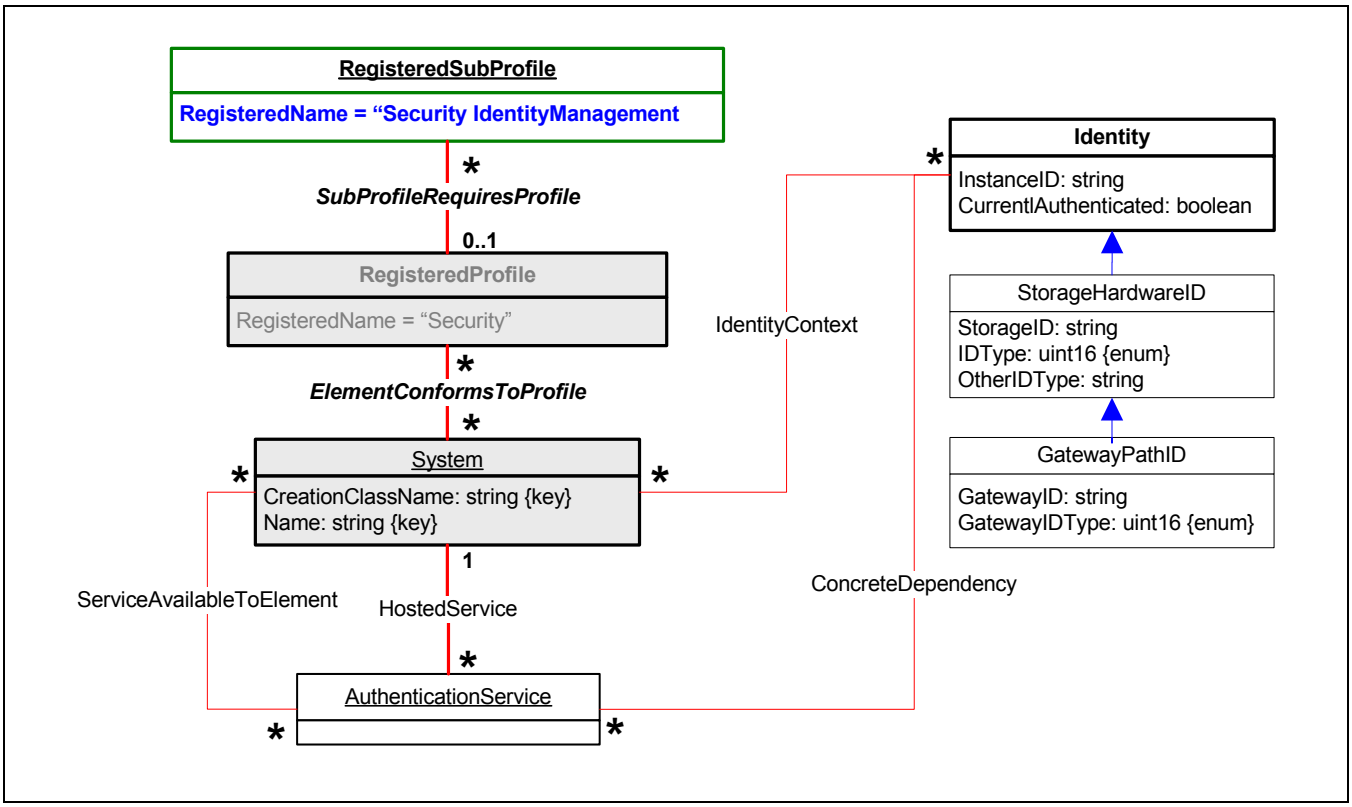

Figure 81 - Identities

53.1.1.1 Standalone Identities

Two types of stand-alone Identities may be instantiated, StorageHardwareID and GatewayPathID. Use CreateInstance and DeleteInstance to instantiate stand-alone Identities. The detailed specification for use of StorageHardwareID and GatewayPathID instances is deferred to profiles or subprofiles that reference this subprofile.

53.1.1.2 Network Identities

NetworkIdentities represent a particular IPProtocolEndpoint or a collection of IPProtocolEndpoints.

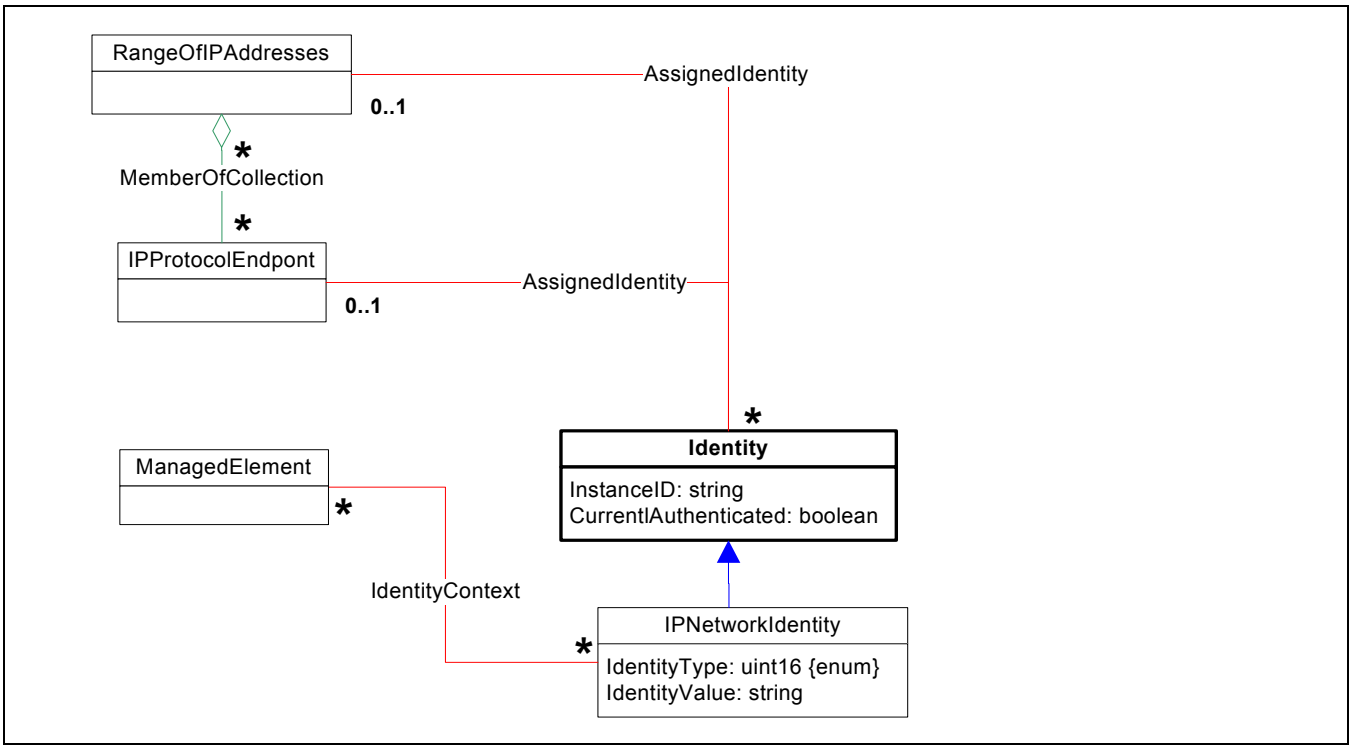

Figure 82 - IPNetworkIdentity

53.1.2 Accounts

Accounts are used for the purpose of authenticating Identities and may additionally be used to as a basis for tracking other information about the use of a system by a particular Identity. Account is essentially another aspect of Identity and is associated via ConcreteIdentity.

When creation of Accounts is supported, the implementation shall present an AccountManagementService instance together with HostedService and ServiceAvailableToElement associations.

If an AccountManagementService is present, instances of Account may be added or deleted using the CreateInstance and DeleteInstance intrinsic methods. The key properties: SystemCreationClassName, SystemName, CreationClassName, and Name of each Account shall be fully specified at creation time. The implementation shall add or delete the AccountOnSystem associations automatically.

Modeling one or more AccountManagementService instances is optional for this subprofile. If there is only one AccountManagementService with a ServiceAvailableToElement association to the named System, then a ManagesAccount association may be implied or the implementation may automatically instantiate one. However if there is more than one AccountManagementService with a ServiceAvailableToElement association to the named System, an instance of ManagesAccount shall be added by a CreateInstance of an Account. The choice of which AccountManagementService to associate to is made intrinsically by the implementation. ManagesAccount instances are deleted automatically when an Account is deleted.

For each Account instance, this subprofile recommends a corresponding Identity instance, associated by ConcreteIdentity. When the Account is created, UserID is set and the UserPassword is specified in clear-text. The creation request is expected to be performed over a secure channel. This subprofile REQUIRES that the UserPassword property shall be write only.

UserContact and Person instances that are associated to an Account via a common Identity instance may have the same, non-null UserID. Setting UserID, UserCertificate or UserPassword properties on such related Account, UserContact or Person instances shall also set the corresponding entries in matching instances.

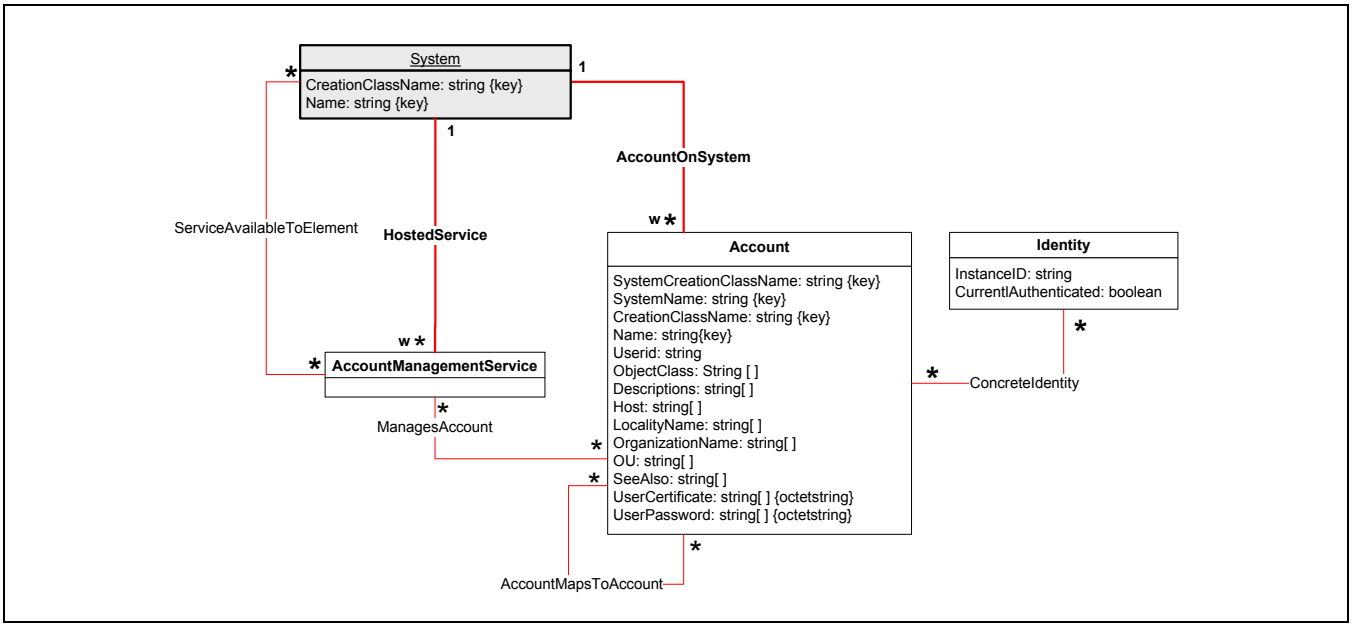

Figure 83 - Account Management

53.1.3 Organizational Directories

There are three basic types of OrganizationalEntities that may be stored in a namespace:

- Organization instances describe top-level entities, like organizations. (See 53.1.3.1.)
- OrgUnit instances describe sub-units of organizations. (See 53.1.3.1.)
- UserEntity instances describe people. (See 53.1.3.2.)

Any OrganizationalEntity may aggregate any number of Collections, such as Groups or Roles. This is managed by CreateInstance or DeleteInstance of CollectionInOrganization associations. This association may be used to associate a Collection to at most one OrganizationalEntity.

Any System may aggregate any number of Collections. This is managed by CreateInstance or DeleteInstance of OwningCollectionElement associations. This association may be used to associate a Collection to at most one System.

A single Collection shall not have both OwningCollectionElement and CollectionInOrganization associations.

Any OrganizationalEntity may aggregate any number of other OrganizationalUnits. For example, a Company may have Business Units and Business Units may have Departments. This is managed by CreateInstance or DeleteInstance of OrgStructure associations. An OrganizationalUnit may belong to at most one OrganizationalUnit.

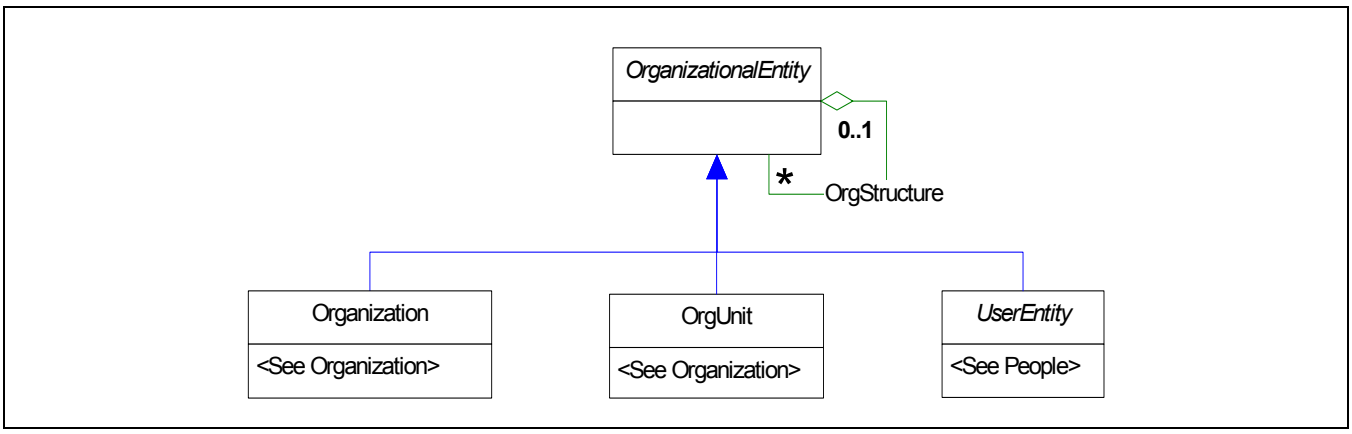

Figure 84 - OrganizationalEntities

53.1.3.1 Organizations

There are two types of OrganizationalEntities. (See Figure 85.) The difference between the two is largely subjective, however this subprofile RECOMMENDS that Organization instances be used to describe businesses, clubs, families, or governments and that OrgUnit instances be used to describe sub-units within Organizations. To make the amount of information provided in these classes more manageable, much of the less commonly used information is defined by properties of the OtherOrganizationInfo and OtherOrgUnitInfo classes respectively,

The key properties: CreationClassName and Name of each shall be fully specified at creation time. Name defines a namespace unique name for the instance of the class. Additionally, the OganizationName or OU properties are also required and name the OrganizationalEntity.

Either or these classes may be added or deleted by the CreateInstance or DeleteInstance intrinsic methods.

At most one OtherOrganizationalInfo or OtherOrgUnitInfo instance per respective Organization or OrgUnit may be instantiated using CreateInstance or DeleteInstance. When instantiated a MoreOrganizationInfo or

MoreOrgUnitInfo association is instantiated to the corresponding Organization or OrgUnit with the same Name. It is an error if either there is no matching instance or there is already an instance of this type with the same Name.

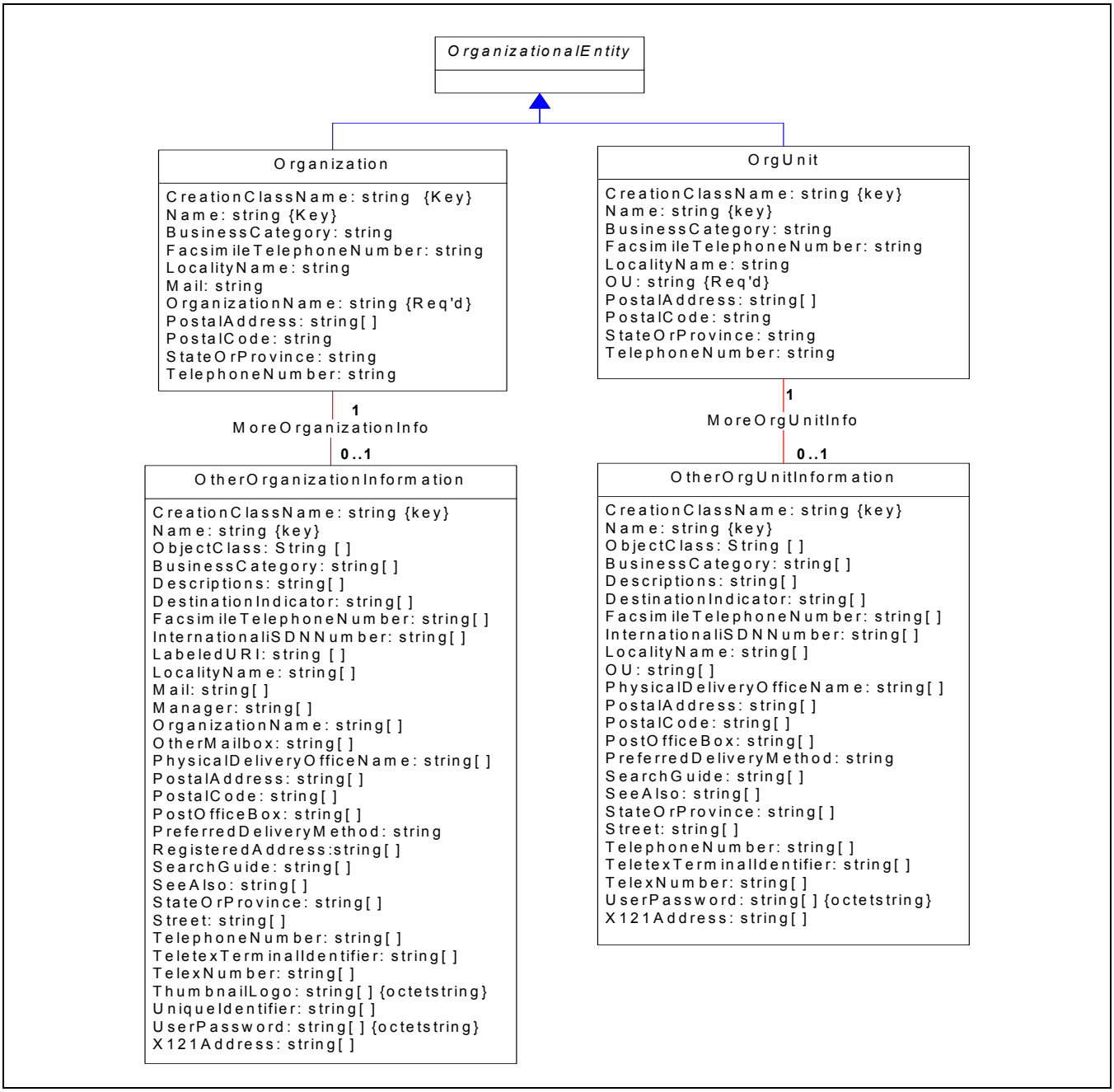

Figure 85 - Organizations and OrgUnits

53.1.3.2 People

A person may be represented by either an Account instance, (see 53.1.2), or by a UserContact instance, (see Figure 86.) Subjectively, Accounts are used to authenticate and track user of a system, where UserContacts are used to represent a person to clients of a system.

The Person class subclasses from UserContact and provides additional information about a person. This is further enhanced by OtherPersonInformation.

A Person instance together with an OtherPersonInformation instance provides UserID and Password. As such, the pair could be used for authentication, in place of Account. However this subprofile RECOMMENDS that Account instances be used for authentication and that UserContact instances be used to describe directory information about a person.

Instances of UserContact, Person, OtherPersonInformation, and MorePersonInfo may be added or deleted using CreateInstance and DeleteInstance intrinsic methods. The key properties of each shall be fully specified at creation time. Additionally the Surname property is required for UserContact or Person instances.

There shall be exactly one Person instance with the same Name property for each instantiated OtherPersonInfo instance.

UserContact and Person instances associated to the same Identity match an Account instance with the same, non-null UserID. Setting UserID, UserCertificate or UserPassword properties on Account, UserContact or Person instances shall also set the corresponding entries in matching instances.

For this subprofile, when a UserContact or Person instance is created, it is mandatory to create an Identity associated via AssignedIdentity.

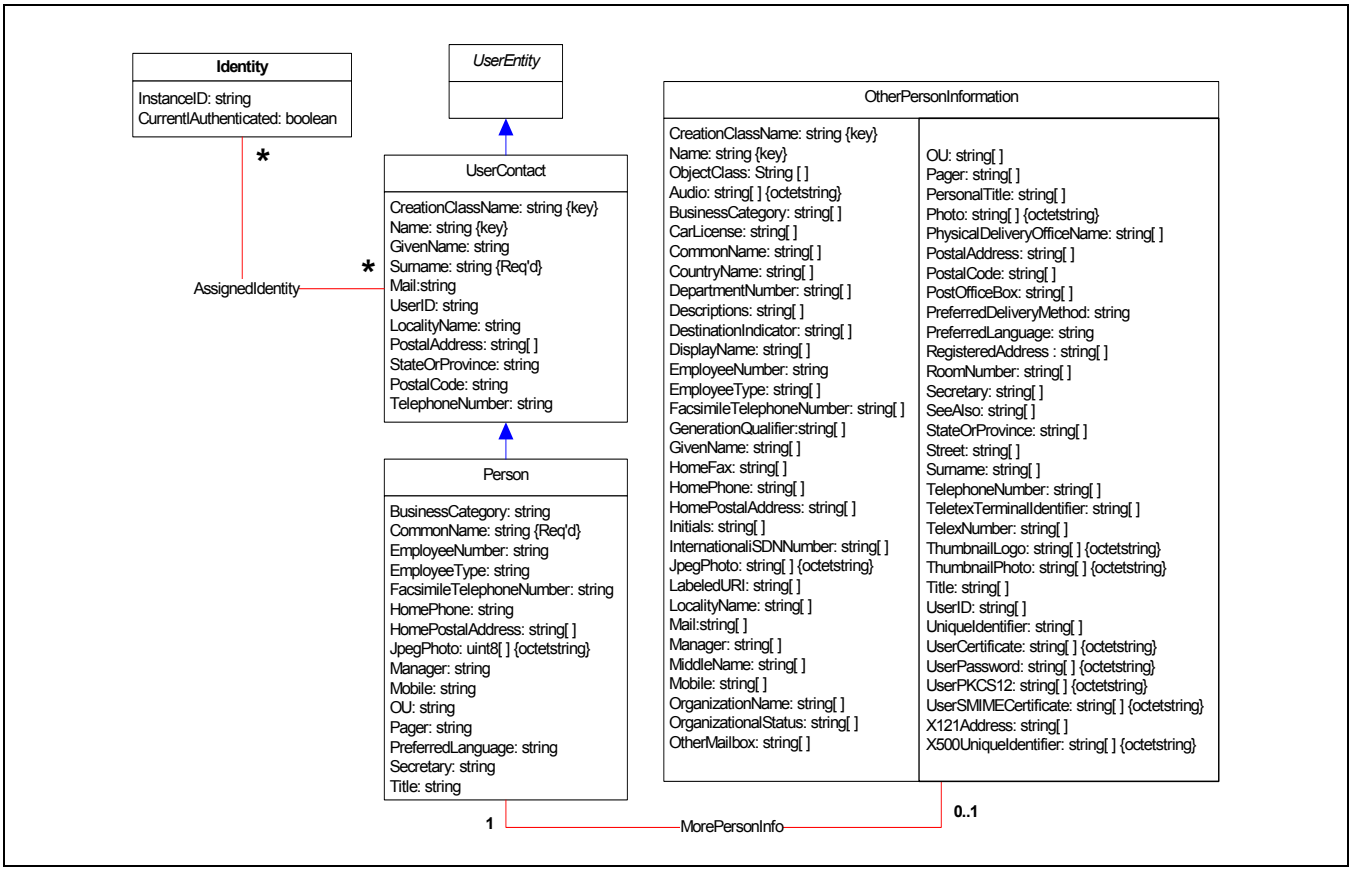

Figure 86 - People

53.1.4 Groups

A Group is an aggregation of ManagedElements. These shall be Identities. An Identity is assigned to a Group via AssignedIdentity in order to assign privileges to a Group or to incorporate a Group into a Role. Unless otherwise specified, the Authentication policy for the Group Identity is that a successful authentication of a MemberOfCollection Identity also authenticates the Group Identity for that user.

Both Groups and Roles may be aggregated via OwningCollectionElement into an OrganizationalEntity instance.

Member information defined by an OtherGroupInformation instance may be associated to a Group via the MoreGroupInfo association.

All of these associations, OwningCollectionElement, MemberOfCollection, AssignedIdentity, and MoreGroupInformation, may be added or deleted via CreateInstance or DeleteInstance intrinsic methods: The key properties of each shall be fully specified at creation time.

All may be added or deleted using CreateInstance and DeleteInstance intrinsic methods. The key properties of each shall be fully specified at creation time. In addition to their keys, both Roles and Groups require that the CommonName property shall be specified at creation time.

There shall be exactly one Group instance with the same Name property for each instantiated OtherGroupInformation instance.

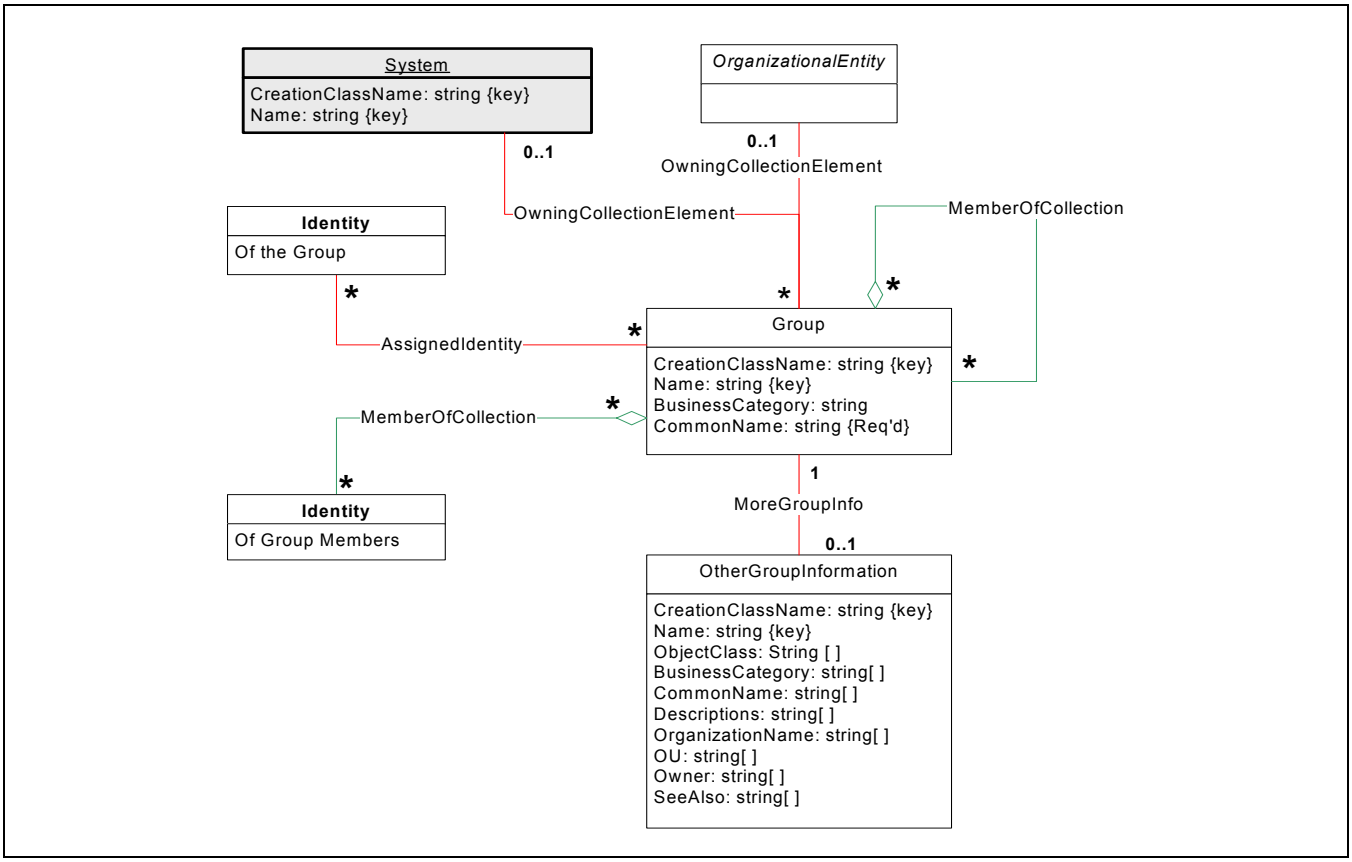

Figure 87 - Groups and Roles

53.2 Health and Fault Management Considerations

TBD

53.3 Cascading Considerations

TBD

53.4 Supported Profiles and Packages

TBD

53.5 Methods of the Profile

TBD

53.6 Client Considerations and Recipes

53.6.1 Create a new User instance with an associated Identity

```
// DESCRIPTION
// This recipe will create a UserContact and an associated Identity.

// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// $User is a template supplied by the application for a new
//  instance of the class CIM_UserContact or one of its subclasses.
//  It is up to the incorporating profile to define exactly what subclass of
//  UserContact and any constraints on properties that must be filled in and what
                          values are permissible.

// Create a new Identity for the UserContact
//
$Identity = NewInstance("CIM_Identity")
$Identity-> = CreateInstance($Identity)

// Create the UserContact instance
//
$User-> = CreateInstance($User);

// Create AssignedIdentity between UserContact and Identity
//
$AssignedIdentity = NewInstance("CIM_AssignedIdentity")
$AssignedIdentity.IdentityInfo = $Identity->
$AssignedIdentity.ManagedElement= $User->
$AssignedIdentity-> = CreateInstance($AssignedIdentity)

// ON OUTPUT
//
// $User-> References the User
// $Identity-> References the Identity of the Account
// $AssignedIdentity-> References the AssignedIdentity association
```

53.6.2 Create an Account for an Identity

```
// DESCRIPTION
// This recipe creates an Account and attaches it to an existing Identity.

// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// $Identity-> points to an Identity.
//
// $Account contains a new Account.
//   Account.UserID MUST be set.  It is synonomous with User Name.
```

```
//          If the named Identity has an AssignedIdentity association to a
                        UserContact instance, then
//          theAccount.UserID MUST match that of UserContact.
//      Account.Password must be set to the encrypted value that it will compare
                        to.

// Create the Account
//
$Account-> = CreateInstance($Account);

// Create ConcreteIdentity between Account and Identity
//
$ConcreteIdentity = NewInstance("CIM_ConcreteIdentity")
$ConcreteIdentity.SameElement = $Identity->
$ConcreteIdentity.SystemElement = $Account->
$ConcreteIdentity-> = CreateInstance($ConcreteIdentity)

// ON OUTPUT
//
// $Account-> References the Account
// $Identity-> References the Identity of the Account
// $ConcreteIdentity-> References the ConcreteIdentity association
```

53.6.3 Create an Account and attach it to an existing User

```
// DESCRIPTION
// This recipe creates an Account and attach it to an existing User.

// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
// $User-> points to an UserContact.
//      This recipe assumes that each UserContact instance has at least one
                        Identity
//      A user may have zero or more accounts.  Each Account/User pair has exactly
                        one Identity.
//      Account and Identity correlate on UserID
//
// $Account contains a new Account.
//   Account.UserID MUST be set.  It is synonomous with User Name.
//          If the named Identity has an AssignedIdentity association to a
                        UserContact instance, then
//          theAccount.UserID MUST match that of UserContact.
//      Account.Password must be set to the encrypted value that it will compare
                        to.


//  Get Identities currrently assigned to the User.
//
$Identity->[] = AssociatorNames ($User->, "CIM_AssignedIdentity",null,null)
```

```
// Case 1: Account.UserID matches User.UserID
//
if ($Account.UserID = $User->UserID)
{
    // This is the principal Account.
    // To simplify, this recipe assumes this is the first Account added.
    //
    if ($Identity->[]size() != 1)
    {
     <ERROR! Expecting exactly one Identity when adding principal account>
    }

    $Account2->[] = AssociatorNames ($Identity[1]->,
                          "CIM_ConcreteIdentity","CIM_Account",null,null)
    if ($Account2->[]size() != 0)
    {
     <ERROR! Principal account is already added.>
    }

    // Create the Account
    //
    $Account-> = CreateInstance($Account);

    // Create ConcreteIdentity between Account and Identity
    //
    $ConcreteIdentity = NewInstance("CIM_ConcreteIdentity")
    $ConcreteIdentity.SameElement = $Identity->[1]
    $ConcreteIdentity.SystemElement = $Account->
    $ConcreteIdentity-> = CreateInstance($ConcreteIdentity)
    <EXIT: "Principal Account Added">
}

// If we are here, we are adding a secondary account.  We assume the account does
                          not already exist.
//  But don't take it for granted.

for #i in $Identity->[]
{
    $Account2[] = AssociatorNames ($Identity->[#i],
                          "CIM_ConcreteIdentity","CIM_Account",null,null)
    for #j in $Account2->[]
    {
        if (Account.UserID = Account2->[#j].UserID)
        {
         <ERROR! Specified secondary account is already added.>
        }
    }
}
```

```
// Create the Account and create a new Identity instance together with
                            associations.
//
$Account-> = CreateInstance($Account);
$Identity = NewInstance($Identity);
$Identity-> = CreateInstance($Identity);

// Create ConcreteIdentity between Account and Identity
//
$ConcreteIdentity = NewInstance("CIM_ConcreteIdentity")
$ConcreteIdentity.SameElement = $Identity->
$ConcreteIdentity.SystemElement = $Account->
$ConcreteIdentity-> = CreateInstance($ConcreteIdentity)

// Create AssignedIdentity between User and Identity
//
$AssignedIdentity = NewInstance("CIM_AssignedIdentity")
$AssignedIdentity.IdentityInfo = $Identity->
$AssignedIdentity.ManagedElement= $User->
$AssignedIdentity-> = CreateInstance($AssignedIdentity)

// Check that all these instances are created
//
try {
     $Account = GetInstance($Account->)
     $Identity = GetInstance($Identity->)
     $ConcreteIdentity = GetInstance($ConcreteIdentity->)
     $AssignedIdentity = GetInstance($AssignedIdentity->)
}
catch (CIM Exception $Exception) {
     throw $Exception
}




<EXIT: "Secondary Account Added">
```

53.7 Registered Name and Version

Security Identity Management version 1.1.0

53.8 CIM Elements

Table 590 describes the CIM elements for Security Identity Management.

Table 590 - CIM Elements for Security Identity Management

Element Name	Requirement	Description
53.8.1 CIM_Account	Optional	
53.8.2 CIM_AccountManagementService	Optional	
53.8.3 CIM_AccountMapsToAccount	Mandatory	
53.8.4 CIM_AccountOnSystem	Mandatory	
53.8.5 CIM_AssignedIdentity	Mandatory	
53.8.6 CIM_AuthenticationService	Optional	
53.8.7 CIM_ConcreteDependency	Mandatory	
53.8.8 CIM_ConcreteIdentity	Mandatory	
53.8.9 CIM_GatewayPathID	Optional	
53.8.10 CIM_Group	Optional	
53.8.11 CIM_HostedService	Mandatory	
53.8.12 CIM_IPNetworkIdentity	Optional	
53.8.13 CIM_Identity	Optional	
53.8.14 CIM_IdentityContext	Mandatory	
53.8.15 CIM_ManagedElement	Optional	
53.8.16 CIM_ManagesAccount	Mandatory	
53.8.17 CIM_MemberOfCollection	Optional	
53.8.18 CIM_MoreGroupInfo	Optional	
53.8.19 CIM_MoreOrgUnitInfo	Optional	
53.8.20 CIM_MoreOrganizationInfo	Optional	
53.8.21 CIM_MorePersonInfo	Optional	
53.8.22 CIM_OrgStructure	Optional	
53.8.23 CIM_OrgUnit	Optional	
53.8.24 CIM_Organization	Optional	
53.8.25 CIM_OrganizationalEntity	Optional	
53.8.26 CIM_OtherGroupInformation	Optional	
53.8.27 CIM_OtherOrgUnitInformation	Optional	
53.8.28 CIM_OtherOrganizationInformation	Optional	
53.8.29 CIM_OtherPersonInformation	Optional	

Table 590 - CIM Elements for Security Identity Management

Element Name	Requirement	Description
53.8.30 CIM_OwningCollectionElement	Optional	shall not be present if CollectionInOrganization is present.
53.8.31 CIM_Person	Optional	
53.8.32 CIM_ServiceAvailableToElement	Mandatory	
53.8.33 CIM_StorageHardwareID	Optional	
53.8.34 CIM_System	Mandatory	
53.8.35 CIM_UserContact	Optional	

53.8.1 CIM_Account

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

Table 591 describes class CIM_Account.

Table 591 - SMI Referenced Properties/Methods for CIM_Account

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
CreationClassName		Mandatory	Key
Name		Mandatory	Key
UserID		Mandatory	The name the user is known by in the System. Matches any UserContact or Person with the same value. Changing here changes corresponding values on matching UserContact or Person instances.
UserCertificate		Mandatory	The Public Key Certificate of this user. Changing here changes corresponding values on matching UserContact or Person instances.
UserPassword		Mandatory	The password used with the UserID. Changing here changes corresponding values on matching UserContact or Person instances.

53.8.2 CIM_AccountManagementService

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 592 describes class CIM_AccountManagementService.

Table 592 - SMI Referenced Properties/Methods for CIM_AccountManagementService

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

53.8.3 CIM_AccountMapsToAccount

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

Table 593 describes class CIM_AccountMapsToAccount.

Table 593 - SMI Referenced Properties/Methods for CIM_AccountMapsToAccount

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

53.8.4 CIM_AccountOnSystem

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

Table 594 describes class CIM_AccountOnSystem.

Table 594 - SMI Referenced Properties/Methods for CIM_AccountOnSystem

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

53.8.5 CIM_AssignedIdentity

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

Table 595 describes class CIM_AssignedIdentity.

Table 595 - SMI Referenced Properties/Methods for CIM_AssignedIdentity

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
IdentityInfo		Mandatory	

53.8.6 CIM_AuthenticationService

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

Table 596 describes class CIM_AuthenticationService.

Table 596 - SMI Referenced Properties/Methods for CIM_AuthenticationService

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

53.8.7 CIM_ConcreteDependency

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

Table 597 describes class CIM_ConcreteDependency.

Table 597 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

53.8.8 CIM_ConcreteIdentity

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

Table 598 describes class CIM_ConcreteIdentity.

Table 598 - SMI Referenced Properties/Methods for CIM_ConcreteIdentity

Properties	Flags	Requirement	Description & Notes
SameElement		Mandatory	
SystemElement		Mandatory	

53.8.9 CIM_GatewayPathID

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

Table 599 describes class CIM_GatewayPathID.

Table 599 - SMI Referenced Properties/Methods for CIM_GatewayPathID

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	True if currently authenticated

53.8.10 CIM_Group

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

Table 600 describes class CIM_Group.

Table 600 - SMI Referenced Properties/Methods for CIM_Group

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key
CommonName		Mandatory	The Name by which the Group is known

53.8.11 CIM_HostedService

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

Table 601 describes class CIM_HostedService.

Table 601 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

53.8.12 CIM_IPNetworkIdentity

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

Table 602 describes class CIM_IPNetworkIdentity.

Table 602 - SMI Referenced Properties/Methods for CIM_IPNetworkIdentity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	True if currently authenticated

53.8.13 CIM_Identity

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

Table 603 describes class CIM_Identity.

Table 603 - SMI Referenced Properties/Methods for CIM_Identity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	True if currently authenticated

53.8.14 CIM_IdentityContext

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

Table 604 describes class CIM_IdentityContext.

Table 604 - SMI Referenced Properties/Methods for CIM_IdentityContext

Properties	Flags	Requirement	Description & Notes
ElementInContext		Mandatory	
ElementProvidingContext		Mandatory	

53.8.15 CIM_ManagedElement

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

53.8.16 CIM_ManagesAccount

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

Table 605 describes class CIM_ManagesAccount.

Table 605 - SMI Referenced Properties/Methods for CIM_ManagesAccount

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

53.8.17 CIM_MemberOfCollection

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

Table 606 describes class CIM_MemberOfCollection.

Table 606 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

53.8.18 CIM_MoreGroupInfo

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

Table 607 describes class CIM_MoreGroupInfo.

Table 607 - SMI Referenced Properties/Methods for CIM_MoreGroupInfo

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

53.8.19 CIM_MoreOrgUnitInfo

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

Table 608 describes class CIM_MoreOrgUnitInfo.

Table 608 - SMI Referenced Properties/Methods for CIM_MoreOrgUnitInfo

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

53.8.20 CIM_MoreOrganizationInfo

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

Table 609 describes class CIM_MoreOrganizationInfo.

Table 609 - SMI Referenced Properties/Methods for CIM_MoreOrganizationInfo

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

53.8.21 CIM_MorePersonInfo

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

Requirement: Optional

Table 610 describes class CIM_MorePersonInfo.

Table 610 - SMI Referenced Properties/Methods for CIM_MorePersonInfo

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

53.8.22 CIM_OrgStructure

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

Table 611 describes class CIM_OrgStructure.

Table 611 - SMI Referenced Properties/Methods for CIM_OrgStructure

Properties	Flags	Requirement	Description & Notes
Child		Mandatory	
Parent		Mandatory	

53.8.23 CIM_OrgUnit

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

Table 612 describes class CIM_OrgUnit.

Table 612 - SMI Referenced Properties/Methods for CIM_OrgUnit

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key
OU		Mandatory	The Name by which the Organizational Unit is known

53.8.24 CIM_Organization

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

Table 613 describes class CIM_Organization.

Table 613 - SMI Referenced Properties/Methods for CIM_Organization

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key
OrganizationName		Mandatory	The Name by which the Organization is known

53.8.25 CIM_OrganizationalEntity

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

53.8.26 CIM_OtherGroupInformation

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

Table 614 describes class CIM_OtherGroupInformation.

Table 614 - SMI Referenced Properties/Methods for CIM_OtherGroupInformation

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key, Must match that of Group

53.8.27 CIM_OtherOrgUnitInformation

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

Table 615 describes class CIM_OtherOrgUnitInformation.

Table 615 - SMI Referenced Properties/Methods for CIM_OtherOrgUnitInformation

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key, Must match that of OrgUnit.

53.8.28 CIM_OtherOrganizationInformation

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

Table 616 describes class CIM_OtherOrganizationInformation.

Table 616 - SMI Referenced Properties/Methods for CIM_OtherOrganizationInformation

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key, Must match that of Organization.

53.8.29 CIM_OtherPersonInformation

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

Table 617 describes class CIM_OtherPersonInformation.

Table 617 - SMI Referenced Properties/Methods for CIM_OtherPersonInformation

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key, Must match that of Person.
UserID		Mandatory	The Name by which the User is known to the System. Matches all Account or Person instances in the namespace with the same UserID. Changing here changes corresponding values on matching UserContact or Account instances.

Table 617 - SMI Referenced Properties/Methods for CIM_OtherPersonInformation

Properties	Flags	Requirement	Description & Notes
UserCertificate		Mandatory	The Public Key Certificate of this user. Changing here changes corresponding values on matching UserContact or Account instances.
UserPassword		Mandatory	The password used with the UserID. Changing here changes the corresponding values on matching UserContact or Account instances.

53.8.30 CIM_OwningCollectionElement

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

Table 618 describes class CIM_OwningCollectionElement.

Table 618 - SMI Referenced Properties/Methods for CIM_OwningCollectionElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

53.8.31 CIM_Person

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

Table 619 describes class CIM_Person.

Table 619 - SMI Referenced Properties/Methods for CIM_Person

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

Table 619 - SMI Referenced Properties/Methods for CIM_Person

Properties	Flags	Requirement	Description & Notes
Surname		Mandatory	The Name by which the User is known to other Persons
UserID		Mandatory	The Name by which the User is known to the System. Matches all Account or Person instances in the namespace with the same UserID. Changing here changes corresponding values on matching UserContact or Account instances.

53.8.32 CIM_ServiceAvailableToElement

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

Table 620 describes class CIM_ServiceAvailableToElement.

Table 620 - SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement

Properties	Flags	Requirement	Description & Notes
ServiceProvided		Mandatory	
UserOfService		Mandatory	

53.8.33 CIM_StorageHardwareID

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

Table 621 describes class CIM_StorageHardwareID.

Table 621 - SMI Referenced Properties/Methods for CIM_StorageHardwareID

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	True if currently authenticated

53.8.34 CIM_System

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

Table 622 describes class CIM_System.

Table 622 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

53.8.35 CIM_UserContact

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

Table 623 describes class CIM_UserContact.

Table 623 - SMI Referenced Properties/Methods for CIM_UserContact

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key
Surname		Mandatory	The Name by which the User is known to other users.
UserID		Mandatory	The Name by which the User is known to the System. Matches all Account or Person instances in the namespace with the same UserID. Changing here changes corresponding values on matching Person or Account instances.

EXPERIMENTAL

EXPERIMENTAL

Clause 54: 3rd Party Authentication Subprofile

54.1 Description

This subprofile extends the Security Identity Management profile by specifying the necessary elements required to manage the relationships between a CIM Server and 3rd party Authentication Servers such as Radius.

The implementation shall use a HostedService association between the System and the AuthenticationService.

In this environment, the local AuthenticationService may delegate authentication requests to a 3rd-party authentication service which is accessed through a RemoteServiceAccessPoint as shown in Figure 88. The implementation shall instantiate a ServiceSAPDependency between the RemoteServiceAccessPoint and the AuthenticationService.

If the 3rd Party Authentication Service requires that the local system authenticate itself, then the required Credential is associated via CredentialContext to the RemoteServiceAccessPoint instance. (See the Security CredentialMangement subprofile.) This may be accomplished using intrinsic operations.

UserContact.Name, Group.Name, and Role.Name are used as a correlatable identifier for users, groups, and roles respectively. Note that the UserID property of UserContact is synonymous with a typical user Name. A user may have multiple Identities. This specification restricts a Group to at most one Identity and does not assign Identities to Roles. An Identity for a UserContact is matched via an AssignedIdentity association and a match on both Name and UserID in the UserContact.

In the event that there is more than one 3rd Party Authentication Service, this profile does not specify the means used by which the local authentication service locates the correct 3rd Party Authentication Service, UserID, and if specified the Realm. See 42.2.1 HTTP Security Background. A sufficient authentication strategy is to pass the requestor's UserID, Realm and credentials to each Authentication service.

The 3rd Party Authentication Service should respond true or false, and if true should also respond with a list of discontinued names which represent at most one authenticated UserContact and a set of Group, and Role elements to which the authenticated user belongs. Each returned distinguished name matches the Name property of at most one such element.

If a UserContact is matched via Name, the UserID shall match that instance of UserContact or that of an associated Account instance. This specification allows a UserContact to be associated via AssignedIdentity to multiple Identities, which in turn may be associated to at most one Account via ConcreteIdentity. An Identity is selected which has matching Name and either a matching UserID or an associated Account with a matching UserID. If no match is found, then this user is not known on this system. A profile that incorporates this subprofile may define an AuthenticationRule that designates some other Identity to authenticate in the case a matching Identity is not found by the above algorithm.

Additionally, the 3rd Party Authentication Service may return the distinguished names of groups or roles to which the user belongs. These names correlated to Group.Name or Role.Name. This specification restricts a Group to at most one Identity associated via AssignedIdentity. If a Group is matched, then the user belongs to the group and the Groups Identity is authenticated. If a Role is matched, then the user is authenticated for the Role. Profiles or subprofiles that rely on this profile may further qualify the types of Identity and AuthenticationRules that may be used.

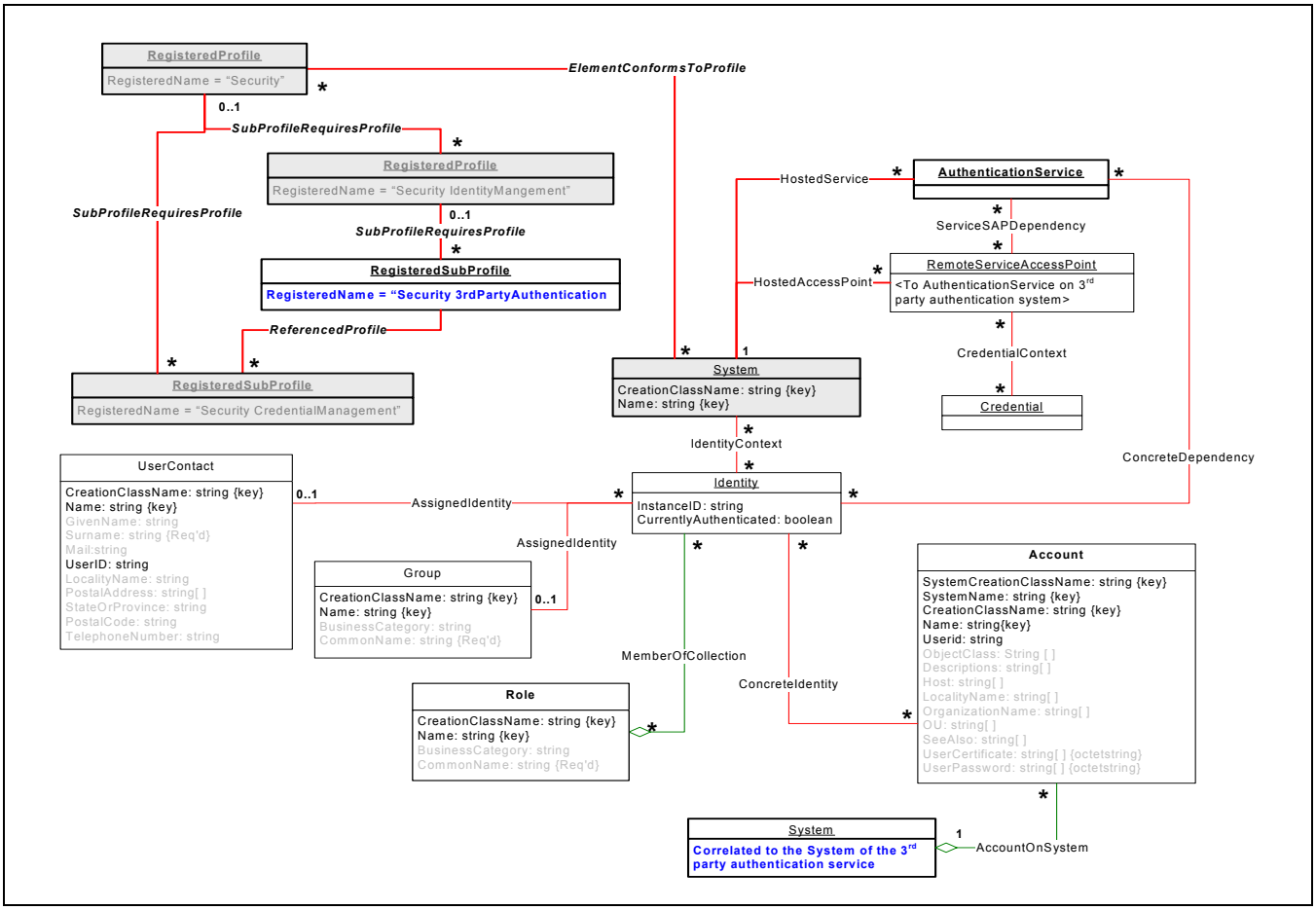

Figure 88 - 3rd Party Authentication for the CIM Service

54.1.1 Durable Names and Correlatable IDs of the Profile

When a UserID is passed from an SMI-S Client to an SMI-S Server and then to a 3rd Party Authentication service, there needs to be some means to assure that each is referring to the same entity. The process specified here is for the client to pass the server a UserID, together with Realm and Credential information. The server passes this through to the authentication service, which maps this to a particular user and zero or more groups and roles to which the User belongs. This subprofile specifies that users, groups, and roles need to have unique distinguished names, (see IETF RFC 4514) These distinguished names are returned to the SMI-S Server by the 3rd Party Authentication service. The SMI-S Server correlates these distinguished names to the Name property of UserContact, Group, or Role instances.

The Identity of a user is determined by a match on both the UserID provided by the SMI-S Client and the distinguished name returned by the 3rd Party Authentication service. (See the algorithm described in the previous section.)

54.2 Client Considerations and Recipes

54.2.1 Create a new User instance with an associated Identity.

The client should use the "Create a new User instance with an associated Identity" recipe defined in the Security Identity Management subprofile. The UserContact (or subclass) instance supplied by the SMI-S Client shall have

the Name property set to match the corresponding information on held on the system supporting the 3rd Party Authentication service. The UserID property shall be that of the principal account for that user.

54.2.2 Add an Account for a User.

If more than one Identity is maintained for a user on the SMI-S server, the client should use the “Create an Account and attach it to an existing User.” recipe defined in the Security Identity Management subprofile. The UserContact (or subclass) instance named by the SMI-S client shall correspond by NAME to the distinguished name of the user as known on the system of the 3rd Party Authentication service. If this is the principal account, the UserID property of the Account shall match that of the named UserContact instance. In all cases the UserID property of the supplied Account shall match the UserID used to authenticate the user. Since the Account is not directly authenticated, the Password property shall not be specified.

54.3 Registered Name and Version

Security Authorization version 1.1.0

54.4 CIM Elements

Table 624 describes the CIM elements for Security Authorization.

Table 624 - CIM Elements for Security Authorization

Element Name	Requirement	Description
54.4.1 CIM_AuthorizationRule	Optional	
54.4.2 CIM_AuthorizationRuleAppliesToIdentity	Optional	
54.4.3 CIM_AuthorizationRuleAppliesToPrivilege	Optional	
54.4.4 CIM_AuthorizationRuleAppliesToTarget	Optional	
54.4.5 CIM_AuthorizedPrivilege	Optional	
54.4.6 CIM_AuthorizedSubject	Optional	
54.4.7 CIM_AuthorizedTarget	Optional	
54.4.8 CIM_ConcreteDependency	Optional	
54.4.9 CIM_ConcreteDependency	Optional	
54.4.10 CIM_HostedService	Mandatory	
54.4.11 CIM_Identity	Optional	
54.4.12 CIM_ManagedElement	Optional	
54.4.13 CIM_PolicyRuleInSystem	Optional	
54.4.14 CIM_PolicySetAppliesToElement	Optional	
54.4.15 CIM_Privilege	Optional	
54.4.16 CIM_PrivilegeManagementService	Mandatory	

Table 624 - CIM Elements for Security Authorization

Element Name	Requirement	Description
54.4.17 CIM_PrivilegePropagationRule	Optional	
54.4.18 CIM_ServiceAffectsElement (Service to AuthorizedProvolege)	Optional	
54.4.19 CIM_ServiceAffectsElement (Service to Iden tity)	Optional	
54.4.20 CIM_ServiceAffectsElement (Service to ManagedElement)	Optional	
54.4.21 CIM_ServiceAffectsElement (Service to Privilege)	Optional	
54.4.22 CIM_ServiceAvailableToElement	Mandatory	
54.4.23 CIM_System	Mandatory	

54.4.1 CIM_AuthorizationRule

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

Table 625 describes class CIM_AuthorizationRule.

Table 625 - SMI Referenced Properties/Methods for CIM_AuthorizationRule

Properties	Flags	Requirement	Description & Notes
SystemCreationClas sName		Mandatory	Key
SystemName		Mandatory	Key
CreationClassName		Mandatory	Key
PolicyRuleName		Mandatory	Key

54.4.2 CIM_AuthorizationRuleAppliesToIdentity

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

Table 626 describes class CIM_AuthorizationRuleAppliesToIdentity.

Table 626 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToIdentity

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

54.4.3 CIM_AuthorizationRuleAppliesToPrivilege

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

Table 627 describes class CIM_AuthorizationRuleAppliesToPrivilege.

Table 627 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToPrivilege

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

54.4.4 CIM_AuthorizationRuleAppliesToTarget

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

Table 628 describes class CIM_AuthorizationRuleAppliesToTarget.

Table 628 - SMI Referenced Properties/Methods for CIM_AuthorizationRuleAppliesToTarget

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

54.4.5 CIM_AuthorizedPrivilege

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

Requirement: Optional

Table 629 describes class CIM_AuthorizedPrivilege.

Table 629 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
RepresentsAuthorizationRights		Mandatory	Must be an Access right for this subprofile.
PrivilegeGranted		Mandatory	Only Grant type privileges are allowed.

54.4.6 CIM_AuthorizedSubject

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

Table 630 describes class CIM_AuthorizedSubject.

Table 630 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject

Properties	Flags	Requirement	Description & Notes
Privilege		Mandatory	
PrivilegedElement		Mandatory	

54.4.7 CIM_AuthorizedTarget

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

Table 631 describes class CIM_AuthorizedTarget.

Table 631 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget

Properties	Flags	Requirement	Description & Notes
Privilege		Mandatory	
TargetElement		Mandatory	

54.4.8 CIM_ConcreteDependency

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

Table 632 describes class CIM_ConcreteDependency.

Table 632 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

54.4.9 CIM_ConcreteDependency

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

Table 633 describes class CIM_ConcreteDependency.

Table 633 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

54.4.10 CIM_HostedService

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

Table 634 describes class CIM_HostedService.

Table 634 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Dependent		Mandatory	

54.4.11 CIM_Identity

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

Table 635 describes class CIM_Identity.

Table 635 - SMI Referenced Properties/Methods for CIM_Identity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
CurrentlyAuthenticated		Mandatory	The Identified entity is authenticated or not

54.4.12 CIM_ManagedElement

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

54.4.13 CIM_PolicyRuleInSystem

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

Table 636 describes class CIM_PolicyRuleInSystem.

Table 636 - SMI Referenced Properties/Methods for CIM_PolicyRuleInSystem

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

54.4.14 CIM_PolicySetAppliesToElement

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

Table 637 describes class CIM_PolicySetAppliesToElement.

Table 637 - SMI Referenced Properties/Methods for CIM_PolicySetAppliesToElement

Properties	Flags	Requirement	Description & Notes
PolicySet		Mandatory	
ManagedElement		Mandatory	

54.4.15 CIM_Privilege

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

Table 638 describes class CIM_Privilege.

Table 638 - SMI Referenced Properties/Methods for CIM_Privilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key
RepresentsAuthorizationRights		Mandatory	Indicates the privilege is to assign the named rights to subjects.
PrivilegeGranted		Optional	Only Grant type privileges are allowed.

54.4.16 CIM_PrivilegeManagementService

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 639 describes class CIM_PrivilegeManagementService.

Table 639 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key
SystemName		Mandatory	Key
CreationClassName		Mandatory	Key
Name		Mandatory	Key
ChangeAccess()		Optional	
ShowAccess()		Optional	

54.4.17 CIM_PrivilegePropagationRule

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

Table 640 describes class CIM_PrivilegePropagationRule.

Table 640 - SMI Referenced Properties/Methods for CIM_PrivilegePropagationRule

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

54.4.18 CIM_ServiceAffectsElement (Service to AuthorizedProvolege)

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

Table 641 describes class CIM_ServiceAffectsElement (Service to AuthorizedProvolege).

Table 641 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to AuthorizedProvolege)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	
AffectedElement		Mandatory	

54.4.19 CIM_ServiceAffectsElement (Service to Iden tity)

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

Table 642 describes class CIM_ServiceAffectsElement (Service to Iden tity).

Table 642 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to Iden tity)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

54.4.20 CIM_ServiceAffectsElement (Service to ManagedElement)

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

Table 643 describes class CIM_ServiceAffectsElement (Service to ManagedElement).

Table 643 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to ManagedElement)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	
AffectingElement		Mandatory	

54.4.21 CIM_ServiceAffectsElement (Service to Privilege)

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

Table 644 describes class CIM_ServiceAffectsElement (Service to Privilege).

Table 644 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Service to Privilege)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	
AffectedElement		Mandatory	

54.4.22 CIM_ServiceAvailableToElement

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

Table 645 describes class CIM_ServiceAvailableToElement.

Table 645 - SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement

Properties	Flags	Requirement	Description & Notes
ServiceProvided		Mandatory	
UserOfService		Mandatory	

54.4.23 CIM_System

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

Table 646 describes class CIM_System.

Table 646 - SMI Referenced Properties/Methods for CIM_System

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Key
Name		Mandatory	Key

EXPERIMENTAL

Clause 55: Cross Profile Considerations

55.1 Overview

Many applications access data from multiple profiles to perform operations. 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 90 and Figure 91 are simplified instance diagrams that are used to illustrate the algorithms.

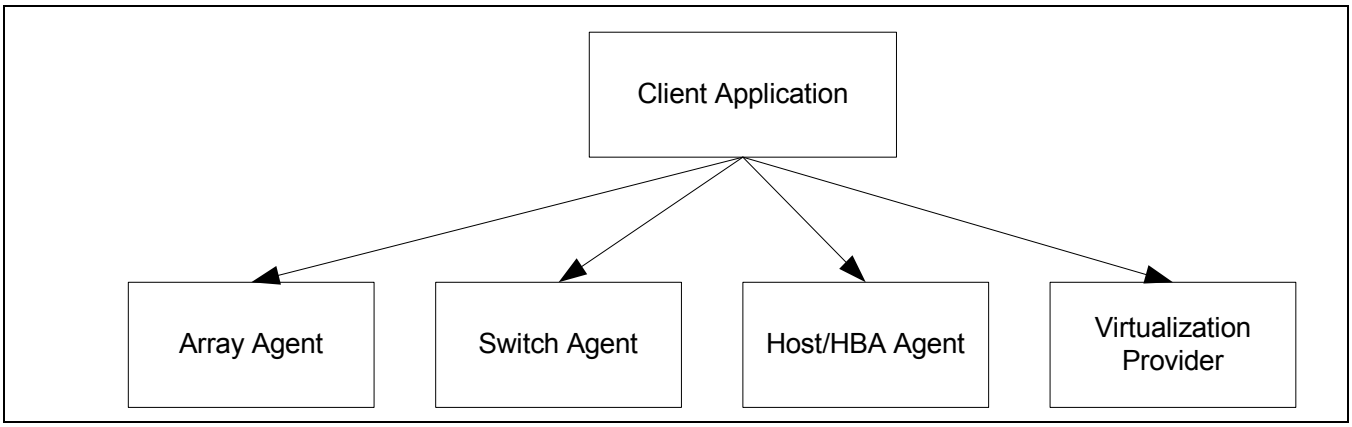

Figure 89 - System Diagram

55.2 HBA model

Figure 90 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.

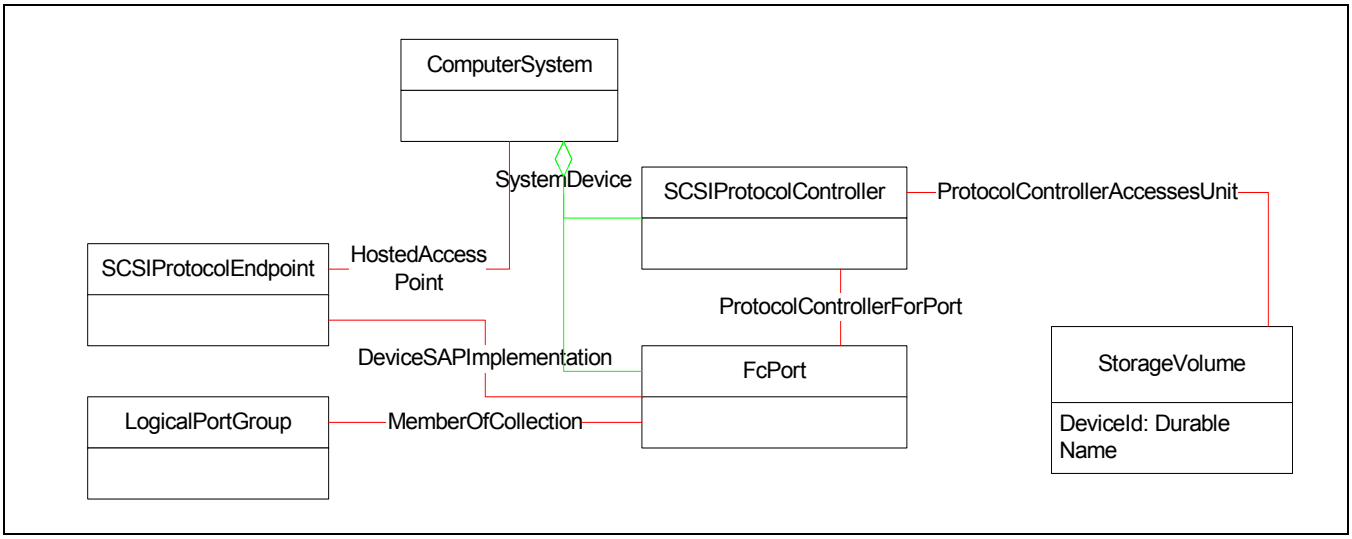

Figure 90 - Host Bus Adapter Model

55.3 Switch Model

Figure 91 represents a two-port Fibre channel switch. The model also includes objects representing links to remote ports the switch agent knows about, and ComputerSystems

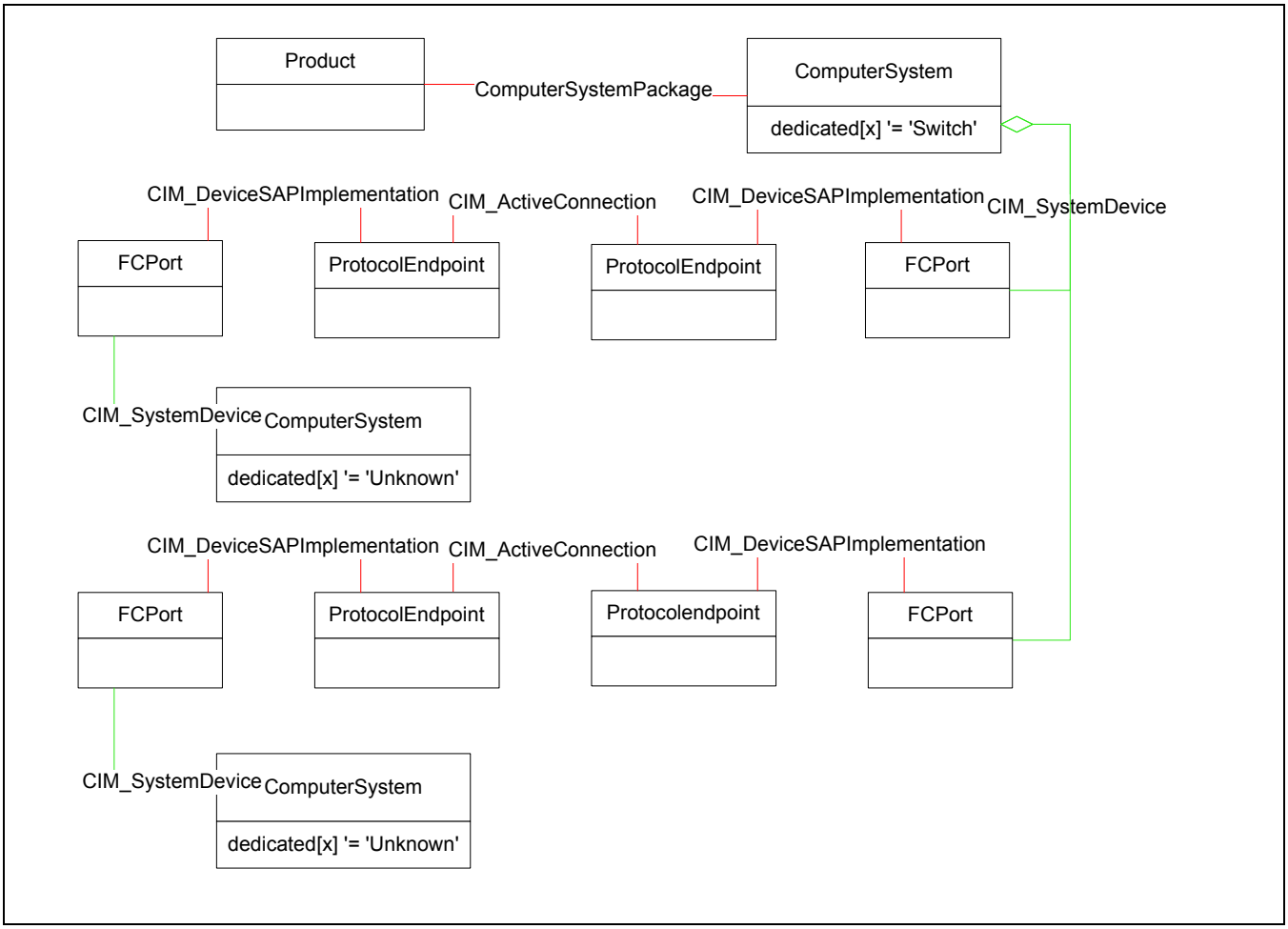

Figure 91 - Switch Model

55.3.1 Recipes

55.3.1.1 Disclaimer

The recipes in this section are included for illustrative purposes only. As of version 1.2.0 of this specification, these recipes are not part of CTP and may not have been validated.

55.3.1.2 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]
        }
    }
}
```

55.3.1.3 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
    }
}
```

55.3.1.4 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
    }
}
```

55.4 Array Model

Figure 92 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.

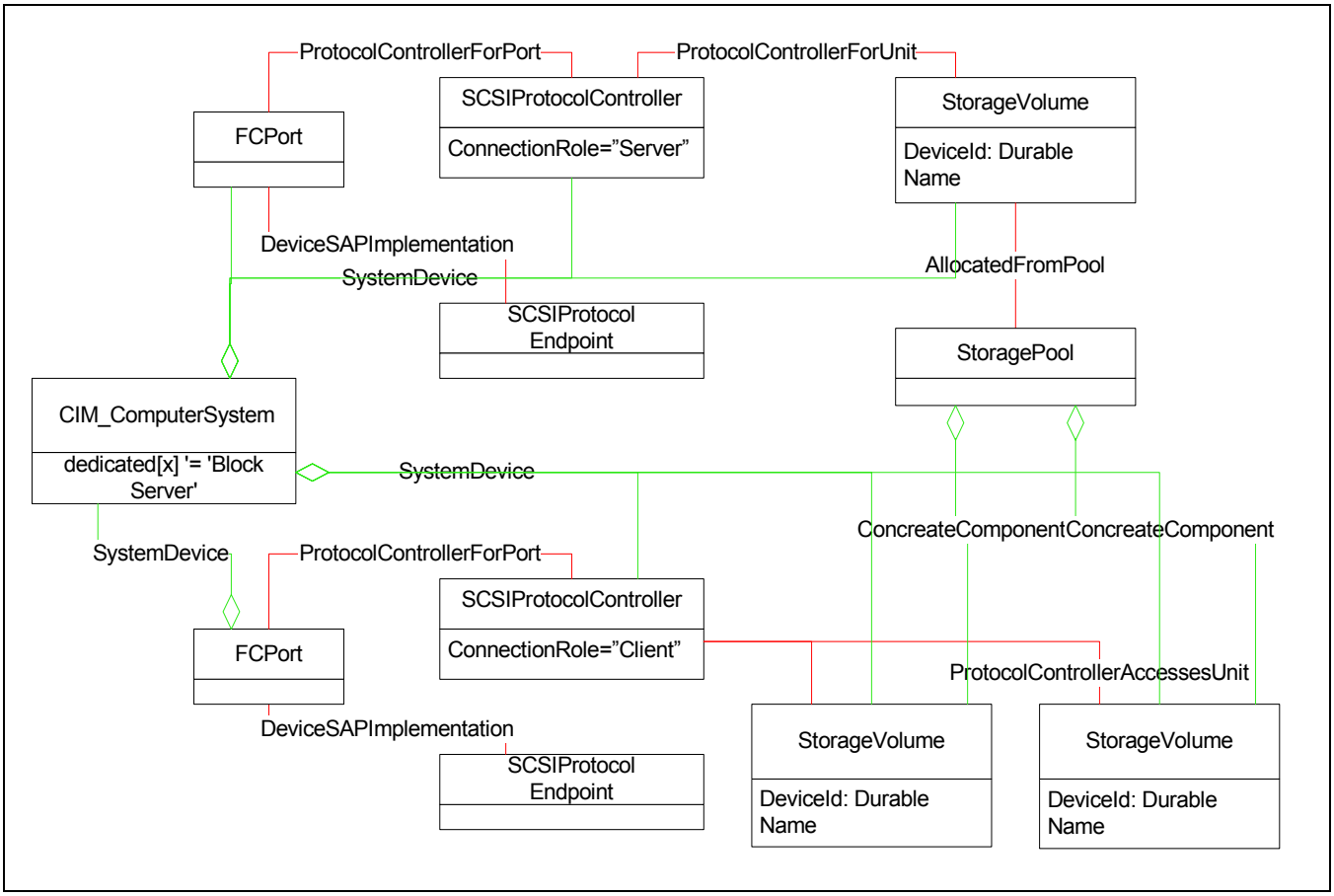

Figure 92 - Array Instance

55.5 Storage Virtualization Model

Figure 93 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.

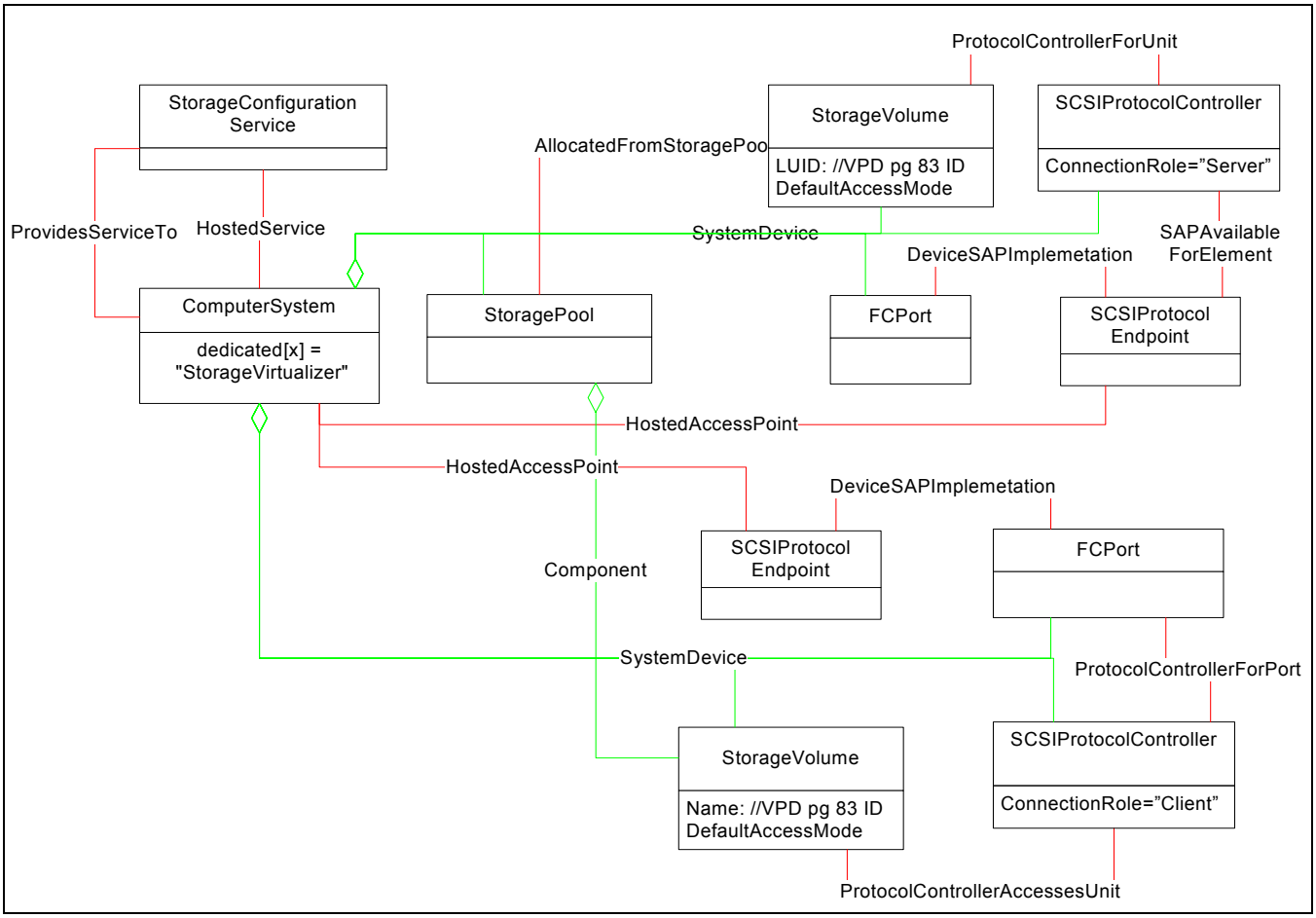

Figure 93 - Virtualization Instance

55.6 Fabric Topology (HBA, Switch, Array)

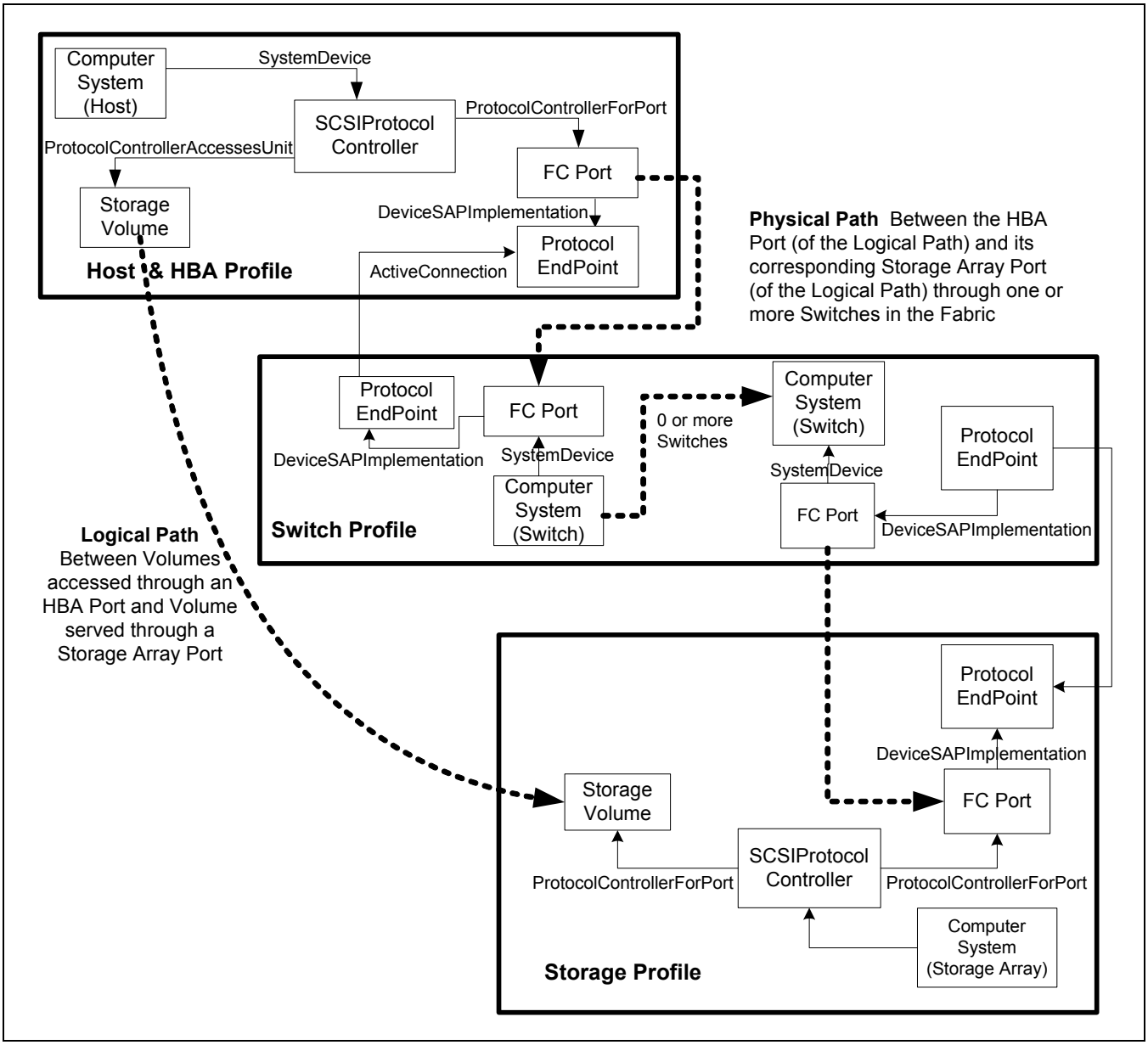

Figure 94 - Fabric Topology

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.

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.

55.6.1 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.

55.6.1.1 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.
```

55.6.1.2 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.
```

55.6.1.3 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.
```

55.6.1.4 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.
```

55.6.1.5 Fabric Layer

```
Fibre Channel Fabric layer piece of the Logical Device 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 ISLÂ’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 the Server Profile clause 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());
}

}
}
}
```

55.6.1.6 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.
```

55.6.1.7 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.
```

55.6.1.8 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.
```

55.6.1.9 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.
```

55.6.1.10 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.
```

55.6.1.11 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.
```

55.6.1.12 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.
```