Send documentation comments to mdsfeedback-doc@cisco.com

Cisco MDS 9000 Family SMI-S Programming Reference

August, 2009

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-19544-01

Cisco MDS 9000 Family SMI-S Programming Reference

CONTENTS

New and Changed Information

Table 1 summarizes the new and changed features for the *Cisco MDS 9000 Family SMI-S Programming Reference* and tells you where they are documented. The table includes a brief description of each new feature and the release in which the change occurred.

Note As of NX-OS Release 4.1(1), SAN-OS has been changed to NX-OS. References to SAN-OS releases before 4.1(1) still apply.

Table 1 Documented Features for the Cisco MDS 9000 Family SMI-S Programming Reference

Feature	Description	Changed in Release	Where Documented
Cisco Zone.MOF	Added the Zone.MOF	4.2(1)	Appendix A, "Managed Object Format Files"
Cisco Indications MOF	Added the class Cisco_ZoneAlert to the Cisco Indications MOF	4.2(1)	Appendix A, "Managed Object Format Files"
Zoning Subprofile	Added the class DeviceAlias() to the Zoning Subprofile	4.2(1)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
Zoning Methods	Added the Zoning methods (CreateDeviceAlias() and RemoveDeviceAlias()	4.2(1)	Chapter 3, "Configuring and Using the CIM Server"
Cisco Fabric MOF	Added the Fabric MOF	4.1(3)	Appendix A, "Managed Object Format Files"
Copy running MOF	Added the Copy running MOF	4.1(3)	Appendix A, "Managed Object Format Files"
Copy running config	Added Copy running config to startup configuration	4.1(3)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
Cisco physical computer system	Added a note	4.1(3)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"

Table 1 Documented Features for the Cisco MDS 9000 Family SMI-S Programming Reference

Feature	Description	Changed in Release	Where Documented
Zoneset distribution	Added the section Zone Control Extension	4.1(3)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
	Added Zoneset distribute functionality to the Extrinsic Method table.	4.1(3)	Chapter 3, "Configuring and Using the CIM Server"
Name Change	Changed the name SAN-OS to NX-OS	4.1(1b)	Throughout the book.
FCIP MOF	Added the FCIP MOF	4.1(1b)	Appendix A, "Managed Object Format Files"
FCIP Subprofile	Added the FCIP Subprofile	4.1(1b)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
Zone member addition	Updated the Zone MOF	3.3(1a)	Appendix A, "Managed Object Format Files"
FDMI MOF	Added the FDMI MOF	3.3(1a)	Appendix A, "Managed Object Format Files"
SMI-S 1.2.0 compliance	Added support for compliance with SMI-S 1.2.0 with caveats.	3.3(1a)	Chapter 1, "Overview"
New Indications	Added new indications	3.3(1a)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
FDMI Subprofile	Added FDMI Subprofile	3.3(1a)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
FDMI Subprofile Extensions	Added FDMI Subprofile extensions	3.3(1a)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
SMI-S 1.1.0 compliance	Added support for full compliance with SMI-S 1.1.0.	3.0(1)	Chapter 1, "Overview"
Access point subprofile	Added support for supplying the URL to the switch.	3.0(1)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
Server profile	Added support for announcing services and capabilities using the Server profile.	2.0(1b)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
SLP	Added support for services discovery using Service Location Protocol (SLP).	2.0(1b)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"
CIM indications	Added support for CIM asynchronous event notifications.	2.0(1b)	Chapter 2, "Cisco MDS 9000 Family CIM Server Support"

Table 1 Documented Features for the Cisco MDS 9000 Family SMI-S Programming Reference

Feature	Description	Changed in Release	Where Documented
CIM configuration	Added steps to configure the CIM server.	1.3(1)	Chapter 3, "Configuring and Using the CIM Server"
CIM support	Introduction of CIM support for the Cisco MDS 9000 Family.	1.3(1)	This guide

Preface

This preface describes the audience, organization, and conventions of the *Cisco MDS 9000 Family SMI-S Programming Reference Guide*. It also provides information on how to obtain related documentation.

Audience

To use this programming guide, you must be familiar with general object-oriented programming techniques and the following items:

- Storage Management Initiative Specification (SMI-S)
- Common Information Model (CIM)
- Managed Object Format (MOF) files
- Unified Modeling Language (UML)
- Secure Socket Layer (SSL), if increased security is desired when accessing the CIM server

Organization

This guide is organized as follows:

Chapter	Title	Description
Chapter 1	Overview	Provides an overview of the support provided for CIM and other standards.
Chapter 2	Cisco MDS 9000 Family CIM Server Support	Describes the supported profiles, indications, and Cisco-specific extensions.
Chapter 3	Configuring and Using the CIM Server	Provides CLI commands to configure the CIM server, and sample scenarios for using CIM to manage your SAN.
Appendix A	Managed Object Format Files	Provides the text from the MOF files for the Cisco MDS 9000 Family CIM server extensions.

Document Conventions

Command descriptions use these conventions:

boldface font	Commands and keywords are in boldface.
italic font	Arguments for which you supply values are in italics.
[ ]	Elements in square brackets are optional.
[**x** | **y** | **z**]	Optional alternative keywords are grouped in brackets and separated by vertical bars.

Screen examples use these conventions:

`screen font`	Terminal sessions and information the switch displays are in screen font.
`boldface screen font`	Information you must enter is in boldface screen font.
`italic screen font`	Arguments for which you supply values are in italic screen font.
`< >`	Nonprinting characters, such as passwords, are in angle brackets.
`[ ]`	Default responses to system prompts are in square brackets.
`!, #`	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

This document uses the following conventions:

Note Means reader *take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Related Documentation

The documentation set for the Cisco MDS 9000 Family includes the following documents. To find a document online, use the Cisco MDS NX-OS Documentation Locator at:

http://www.cisco.com/en/US/docs/storage/san_switches/mds9000/roadmaps/doclocater.htm

Release Notes

- *Cisco MDS 9000 Family Release Notes for Cisco MDS NX-OS Releases*
- *Cisco MDS 9000 Family Release Notes for MDS SAN-OS Releases*
- *Cisco MDS 9000 Family Release Notes for Storage Services Interface Images*
- *Cisco MDS 9000 Family Release Notes for Cisco MDS 9000 EPLD Images*

- *Release Notes for Cisco MDS 9000 Family Fabric Manager*

Regulatory Compliance and Safety Information

- *Regulatory Compliance and Safety Information for the Cisco MDS 9000 Family*

Compatibility Information

- *Cisco Data Center Interoperability Support Matrix*
- *Cisco MDS 9000 NX-OS Hardware and Software Compatibility Information and Feature Lists*
- *Cisco MDS NX-OS Release Compatibility Matrix for Storage Service Interface Images*
- *Cisco MDS 9000 Family Switch-to-Switch Interoperability Configuration Guide*
- *Cisco MDS NX-OS Release Compatibility Matrix for IBM SAN Volume Controller Software for Cisco MDS 9000*
- *Cisco MDS SAN-OS Release Compatibility Matrix for VERITAS Storage Foundation for Networks Software*

Hardware Installation

- *Cisco MDS 9500 Series Hardware Installation Guide*
- *Cisco MDS 9200 Series Hardware Installation Guide*
- *Cisco MDS 9100 Series Hardware Installation Guide*
- *Cisco MDS 9124 and Cisco MDS 9134 Multilayer Fabric Switch Quick Start Guide*

Software Installation and Upgrade

- *Cisco MDS 9000 NX-OS Release 4.1(x) and SAN-OS 3(x) Software Upgrade and Downgrade Guide*
- *Cisco MDS 9000 Family Storage Services Interface Image Install and Upgrade Guide*
- *Cisco MDS 9000 Family Storage Services Module Software Installation and Upgrade Guide*

Cisco NX-OS

- *Cisco MDS 9000 Family NX-OS Licensing Guide*
- *Cisco MDS 9000 Family NX-OS Fundamentals Configuration Guide*
- *Cisco MDS 9000 Family NX-OS System Management Configuration Guide*
- *Cisco MDS 9000 Family NX-OS Interfaces Configuration Guide*
- *Cisco MDS 9000 Family NX-OS Fabric Configuration Guide*
- *Cisco MDS 9000 Family NX-OS Quality of Service Configuration Guide*
- *Cisco MDS 9000 Family NX-OS Security Configuration Guide*
- *Cisco MDS 9000 Family NX-OS IP Services Configuration Guide*

- *Cisco MDS 9000 Family NX-OS Intelligent Storage Services Configuration Guide*
- *Cisco MDS 9000 Family NX-OS High Availability and Redundancy Configuration Guide*
- *Cisco MDS 9000 Family NX-OS Inter-VSAN Routing Configuration Guide*

Cisco Fabric Manager

- *Cisco Fabric Manager Fundamentals Configuration Guide*
- *Cisco Fabric Manager System Management Configuration Guide*
- *Cisco Fabric Manager Interfaces Configuration Guide*
- *Cisco Fabric Manager Fabric Configuration Guide*
- *Cisco Fabric Manager Quality of Service Configuration Guide*
- *Cisco Fabric Manager Security Configuration Guide*
- *Cisco Fabric Manager IP Services Configuration Guide*
- *Cisco Fabric Manager Intelligent Storage Services Configuration Guide*
- *Cisco Fabric Manager High Availability and Redundancy Configuration Guide*
- *Cisco Fabric Manager Inter-VSAN Routing Configuration Guide*
- *Cisco Fabric Manager Online Help*
- *Cisco Fabric Manager Web Services Online Help*

Command-Line Interface

- *Cisco MDS 9000 Family Command Reference*

Intelligent Storage Networking Services Configuration Guides

- *Cisco MDS 9000 I/O Acceleration Configuration Guide*
- *Cisco MDS 9000 Family SANTap Deployment Guide*
- *Cisco MDS 9000 Family Data Mobility Manager Configuration Guide*
- *Cisco MDS 9000 Family Storage Media Encryption Configuration Guide*
- *Cisco MDS 9000 Family Secure Erase Configuration Guide*
- *Cisco MDS 9000 Family Cookbook for Cisco MDS SAN-OS*

Troubleshooting and Reference

- *Cisco NX-OS System Messages Reference*
- *Cisco MDS 9000 Family NX-OS Troubleshooting Guide*
- *Cisco MDS 9000 Family NX-OS MIB Quick Reference*
- *Cisco MDS 9000 Family NX-OS SMI-S Programming Reference*
- *Cisco MDS 9000 Family Fabric Manager Server Database Schema*

CHAPTER 1

Overview

The Cisco MDS 9000 Family of multilayer directors and fabric switches provide an industry standard application programming interface (API) using the Storage Management Initiative Specification (SMI-S). SMI-S facilitates managing storage area networks (SANs) in a multivendor environment.

This chapter includes the following sections:

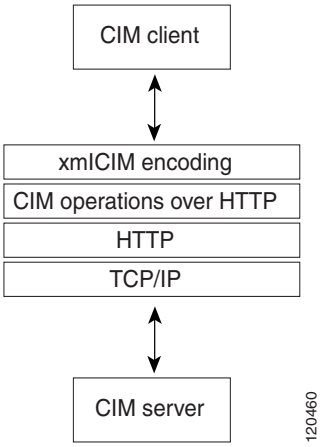
About the Common Information Model

The Common Information Model (CIM) is an object-oriented information model that describes management information in a network or enterprise environment. Because it is object-oriented, CIM provides abstraction, inheritance, and dependency or association relationships between objects within the model. CIM is based on XML and is platform-independent and technology neutral. The management application developer does not need any information about how CIM was implemented on a vendor product; only the API is required to interact with a vendor product.

CIM uses a client/server model. The CIM server can be embedded into the vendor product or can be implemented by a proxy server that provides the CIM server functionality for the legacy vendor product. The CIM client is the management application that communicates to multiple CIM servers to manage the SAN. The CIM client discovers CIM servers through the Service Location Protocol, version 2 (SLPv2) as defined in RFC 2608. SLPv2 uses UDP port 427 for communication and is a discovery protocol that is separate from the CIM client/server communication path.

CIM defines the communications between the client and server in terms of technologies defined in the WEBM Initiative. Figure 1-1 shows the full CIM client/server communications path.

Figure 1-1 CIM Client/Server Communications

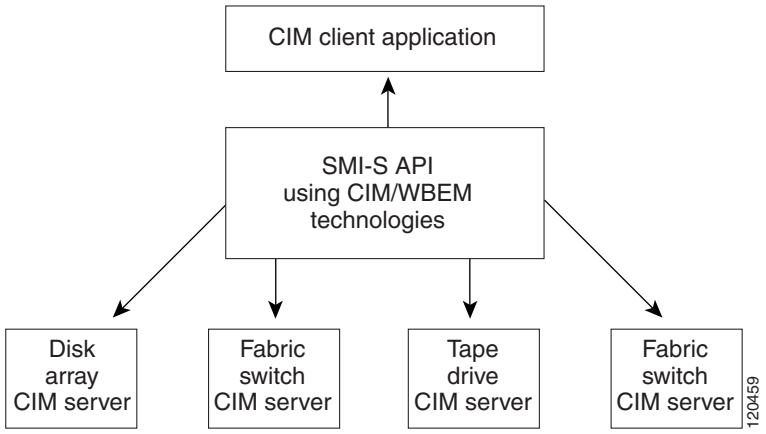


For more information about CIM, refer to the specification available through the Distributed Management Task Force (DMTF) website at http://www.dmtf.org.

About the Storage Management Initiative Specification

The Storage Management Initiative Specification (SMI-S) uses an object-oriented model based on CIM to define a set of objects and services that can manage elements of a SAN. By using a standardized architecture, SMI-S helps management application developers create common and extensible applications that work across multiple SAN vendor products. Figure 1-2 exemplifies SMI-S in a multivendor SAN.

Figure 1-2 SMI-S in a Multivendor SAN



SMI-S provides a set of standard management objects collected in a *profile*. Several profiles are defined in SMI-S that cover common SAN elements, including switches, fabrics, and zoning. These standardized profiles insure interoperability across products within the SAN. SMI-S also defines an automated discovery process, using SLPv2. SMI-S uses CIM defined by the DMTF as part of the WBEM.

For more information about SMI-S, refer to the Storage Networking Industry Association (SNIA) website at http://www.snia.org.

Note Cisco SAN-OS Release 3.0(1) or later, and Cisco NX-OS 4.x are compliant with SMI-S 1.1.0.

About the WBEM Initiative

The WBEM initiative is a set of management and Internet standards developed to unify the management of enterprise computing environments.

The WBEM initiative includes:

- CIM, which provides a common format, language and methodology for collecting and describing management data.
- The CIM-XML Encoding Specification, a standards-based method for exchanging CIM information. CIM-XML uses an xmlCIM encoded payload and HTTP as the transport mechanism. CIM-XML consists of the following specifications:
  - xmlCIM encoding, a standard way to represent CIM information in XML format.
  - CIM operations over HTTP, a transportation method that describes how to pass xmlCIM encoded messages over HTTP.

For more information about the WBEM initiative, refer to the DMTF website at http://www.dmtf.org

Understanding CIM and Unified Modeling Language Notation

SMI-S relies on object-oriented classes as defined in CIM. These classes are frequently defined using Unified Modeling Language (UML). To understand the SMI-S and the Cisco extensions present in this document, you must have a basic understanding of CIM classes and UML.

Understanding CIM Classes

A class is a collection of properties and methods that define a type of object. As an example, a generic network device is a type of object. We could define the NetworkDevice class to describe this object. The NetworkDevice class contains properties or attributes of a network device. Some properties for our NetworkDevice class are IpAddress and DeviceType. Further, we want to control our network device through the NetworkDevice class. So we add methods or routines we can use to trigger actions on our network device. Some methods are enablePort() and rebootDevice().

Now that we have a NetworkDevice class, we can define a class for just switches. Because a switch is a special type of NetworkDevice, we use the object-oriented concept of *inheritance* to define our Switch class. We define the Switch class as a child of the NetworkDevice class. This means the Switch class automatically has the properties and methods of its parent class. From there, we add properties and methods unique to a switch.

CIM defines a special type of class called an *association class*. An association class represents relationships between two or more classes. As an example, we define an association class to show the relationship between a NetworkDevice class and an OperatingSystem class. If there is a many-to-one or many-to-many relationship, the association class is considered an *aggregation*.

Refer to http://www.dmtf.org for a full explanation of CIM.

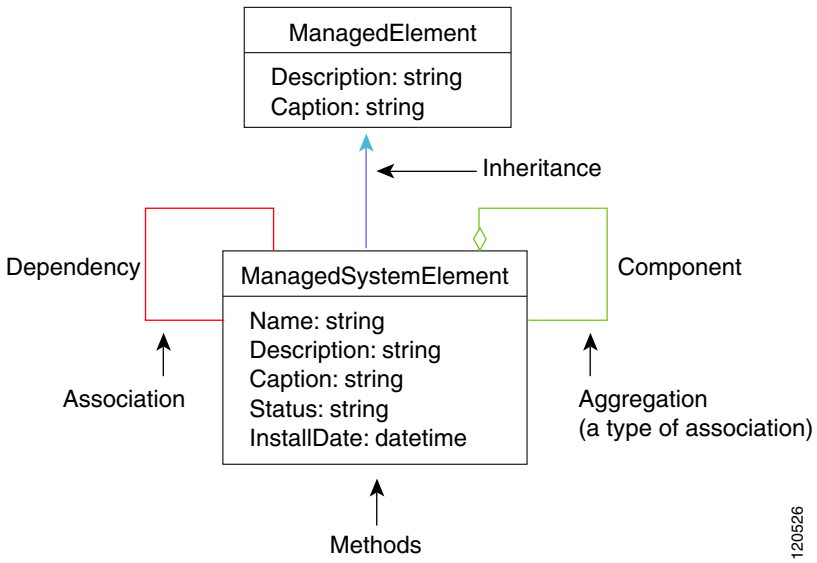
Understanding UML

UML draws a visual representation of the classes that describe a product or technology. UML contains many visual elements, but only a subset are described here. Refer to http://www.uml.org for a full explanation of UML.

Figure 1-3 shows an example section from a UML diagram for CIM classes. This diagram shows:

- blue lines for inheritance between classes
- green lines for aggregation between classes
- red lines for associations between classes

Figure 1-3 UML Example Diagram



About SMI-S and CIM in the Cisco MDS 9000 Family

Each switch or director in the Cisco MDS 9000 Family includes an embedded CIM server. The CIM server communicates with any CIM client to provide SAN management compatible with SMI-S. The CIM server includes the following standard profiles, subprofiles, and features as defined in SMI-S:

- Service Location Protocol version 2 (SLPv2).
- Server profile.

- CIM indications.
- Fabric profile.
  - Zoning Control subprofile.
  - Enhanced Zoning and Enhanced Zoning Control subprofile.
  - FDMI subprofile.
- Switch profile, including the Blade subprofile and Access Point subprofile.
- xmlCIM encoding and CIM operations over HTTP as specified by the WBEM initiative.
- HTTPS, which uses Secure Socket Layer (SSL). HTTPS is optional but provides enhanced security by encrypting communications between the CIM server and the CIM client.

Table 1-1 shows the Cisco SAN-OS release that supports different versions of SMI-S.

Table 1-1 Cisco SAN-OS Support for SMI-S

Cisco SAN-OS Release	SMI-S Support	Description
3.3(1a)	SMI-S 1.2.0 compliant with caveats	• All required indications are not supported • Limited WQL/CQL support
3.0(1)	SMI-S 1.1.0 compliant	Additional support for Server profile and Access Port subprofile.
2.0(1b)	Supports SMI-S 1.0.2	Supports SLPv2, CIM indications, and the Server profile.

CHAPTER 2

Cisco MDS 9000 Family CIM Server Support

SMI-S defines a number of profiles that specify the managed objects used to control and monitor elements of a SAN. The Cisco MDS 9000 Family CIM server supports the standard profiles listed in this chapter. The CIM server also supports extensions to these profiles to support features in Cisco MDS NX-OS that are not available from the standard profiles.

This chapter includes the following sections:

Managing SANs Through SMI-S

SANs are created in a multivendor environment. Hosts, fabric elements (switches, directors), and data storage devices are integrated from different vendors to create an interoperable storage network. Managing these elements from different vendors is problematic to the network administrator. Each element has its own management interface that may be proprietary. A network administrator must work with these disparate management APIs to build a cohesive management application that controls and monitors the SAN.

The SMI-S addresses this management problem by creating a suite of flexible, open management API standards based on the vendor- and technology-independent CIM. Using the SMI-S APIs, collected in *profiles* of common management classes, a network administrator can create a simplified management application CIM client to control and monitor the disparate SAN elements that support SMI-S and CIM. With CIM servers either embedded on the SAN elements or supported by a proxy CIM server, these elements are accessible to the network administrator's CIM client application.

SMI-S uses the Service Location Protocol version 2 (SLPv2) to discover CIM servers. Once the CIM servers are identified, the CIM client determines the profiles supported on the CIM servers through the Server profile. This profile is mandatory on all SMI-S based CIM servers.

Besides the control and monitoring support provided by profiles, the CIM server also supports asynchronous delivery of events through CIM *indications*. Indications provide immediate notification of important occurrences such as when an interface goes down.

Service Location Protocol

The first step in managing a network of SAN elements with CIM servers is discovering the location and support available on the CIM servers. The SLPv2 provides this discovery mechanism. A CIM client uses SLPv2 to discover CIM servers, gathering generic information about what services the CIM servers provide and the URL where these services are located.

The Cisco MDS 9000 Family CIM server supports SLPv2 as defined in RFC 2608.

Note Cisco MDS SAN-OS Release 2.0(1b) and later, and Cisco NX-OS 4.x support SLPv2 for the Cisco MDS 9000 Family CIM server.

Server Profile

Once the CIM client discovers the CIM servers within the SAN, the CIM client must determine the level of support each CIM server provides. The Server profile defines the capabilities of the CIM server. This includes providing the namespace and all profiles and subprofiles supported by the CIM server.

For each supported profile, the Server profile instantiates the `RegisteredProfile` class. Each instance of this class gives the CIM client the profile name and unique ID that is supported by the CIM server. Similarly, the CIM server lists all supported optional subprofiles, using the `RegisteredSubProfile` class and the `SubprofileRequiresProfile` association class to associate the subprofile with the profile.

Note Cisco MDS SAN-OS Release 2.0(1b) and later, and Cisco NX-OS 4.x support the Server profile for the Cisco MDS 9000 Family CIM server.

Note For a Server profile instance diagram, refer to SMI-S at http://www.snia.org.

Switch Profile

The Switch profile models the physical and logical aspects of switches. The CIM client uses the Switch profile to identify that the CIM server is on a switch and uses classes in the Switch profile to identify and manage Fibre Channel ports on the switch.

The Switch profile also supports the optional Blade subprofile (see the "Blade Subprofile" section on page 2-3) and the optional Access Point Subprofile (see the "Access Point Subprofile" section on page 2-4).

Note For a Switch profile instance diagram, refer to the SMI-S at http://www.snia.org.

Table 2-1 shows how to use the Switch profile classes and association classes to model the switch and ports.

Table 2-1 Using the Switch Profile

Class	How Used
ComputerSystem	Identifies the switch, with the Dedicated property set to Switch.
PhysicalElement	Identifies the physical aspects of a device.
FCPort	Identifies logical aspects of the port link and the data layers.
FCPortCapabilities	Defines configuration options supported by the ports.
FCPortStatistics	Identifies port statistics, showing real-time port traffic information for each instance of FCPort class.
FCSwitchCapabilities	Defines configuration options supported by the switch.
FCSwtichSettings	Requests configuration changes on the switch.
FCPortSettings	Requests configuration changes on the ports.

Note A new Property has been added to CISCO_PhysicalComputerSystem to display the IP address of the switch.

Copy running config command

The **copy running config** command copies the current running configuration of the switch to the startup configuration (similar to **copy running-config startup-config** command in the switch). When HPSE device makes changes to an inactive and power cycles the MDS switch, the changes made to the inactive zoneset are lost.

Changes made to an inactive zone set by default are not persisted on the MDS side. You need to commit the changes by using the **copy running-config startup-config** command and enters this same command in CLI and Fabric Manager to commit the changes. The management applications can provide an additional step to help you persist the changes, by providing additional an check box to copy the running-config to the startup-config. This functionality is available through SNMP, but not currently available through the SMI-S provider which would be an enhancement request to the provider API.

Blade Subprofile

The CIM client uses the optional Blade Subprofile to model the physical and logical aspects of a supervisor module, switching module, or services module in a switch. Combining this with the Switch profile, the CIM client gains a chassis-level view into the switch, associating ports to modules and modules to a switch.

Table 2-2 shows how to use the classes and association classes to model a module.

Table 2-2 Using the Blade Subprofile

Class	How Used
LogicalModule	Identifies a supervisor module, switching module, or services module as an aggregation point for the switch ports.
ModulePort	Associates the ports to a module.

Note For a Blade Subprofile instance diagram, refer to the SMI-S at http://www.snia.org.

Access Point Subprofile

The CIM client uses the Access Point subprofile to return the URL to access the switch and install or launch Fabric Manager or Device Manager. If Fabric Manager or Device Manager have not been installed, then the URL gives the option to install them. If Fabric Manager or Device Manager have been installed, then the URL gives the option to launch either of them.

Table 2-3 shows how to use the classes and association classes to model a module.

Table 2-3 Using the Access Point Subprofile

Class	How Used
HostedAccessPoint	Associates the RemoteServiceAccessPoint to the system on which it is hosted.
RemoteServiceAccessPoint	A ServiceAccessPoint for management tools. Returns the URL for the switch that can be used to install or launch Fabric Manager or Device Manager.
SAPAvailableForElement	Identifies the subset of devices in the system that are serviced by RemoteServiceAccessPoint.

Note For an Access Point subprofile instance diagram, refer to the SMI-S at http://www.snia.org.

Fabric Profile

A fabric is composed of one or more switches and network elements interconnected in a SAN. The Fabric profile models the physical and logical aspects of the fabric containing the SAN switches listed by the Switch profile.

Fabrics can contain one or more virtual SANs, or VSANs. See the "Cisco MDS Extensions to the Switch and Fabric Profiles" section on page 2-9 for more information on the Cisco VSAN extension. Because routing in the Cisco MDS 9000 Family is based on the VSAN, the ConnectivityCollection and ProtocolEndpoint classes must be associated with the VSAN, not the fabric.

Table 2-4 shows how to use the classes and association classes of the Fabric profile to model the fabric.

Table 2-4 Using the Fabric Profile

Class	How Used
AdminDomain	Identifies fabrics and VSANs.
ContainedDomain	Associates a VSAN to a fabric.
ConnectivityCollection	Groups a set of ProtocolEndpoint classes together that can communicate with each other directly and represents the foundation necessary for routing. Associates to a VSAN using the Component association class.
ComputerSystem	Represents the fabric elements that contain ports, such as switches, hosts, and storage systems. The Dedicated property is set to Switch. Associates to a VSAN using the Component association class.
FCPort	Represents the logical aspects of the link and data layers. Associates to the ProtocolEndpoint class by the DeviceSAPImplementation association class and associates to the ComputerSystem class by the SystemDevice association class.
ProtocolEndpoint	Represents the higher network layers for routing. Associates to the ConnectivityCollection class by the ConnectivityMemberOfCollection association class.
ActiveConnection	Represents a link that associates two ProtocolEndpoint classes as a connection that is currently carrying traffic.

Note The Cisco MDS 9000 Family CIM server only provides the LogicalPortGroup class for the fabric, not for hosts or storage systems.

The CIM server requires that the name of the fabric or VSAN be unique within the same CIM namespace. Names are identified by the Name class property with an associated optional NameFormat property. A VSAN identifier is the VSAN identification followed by the world-wide name (WWN) of the principal switch, for example,"1_2001000530000A0A" (the NameFormat indicates that it is a WWN). For VSANs, the fabric identifier is a string because there is no principal switch per fabric.

Note For a Fabric profile instance diagram, refer to the SMI-S at http://www.snia.org.

The CIM server supports the following optional subprofiles from the Fabric profile:

- Zone Control subprofile (see the "Zone Control Subprofile" section on page 2-6)
- Enhanced Zoning and Enhanced Zoning Control subprofile (see the "Enhanced Zoning and Enhanced Zoning Control Subprofile" section on page 2-6)
- FDMI subprofile (see the "FDMI Subprofile" section on page 2-7)

The Fabric profile also supports a number of extensions specific to the Cisco MDS 9000 Family. See the "Cisco MDS Extensions to the Switch and Fabric Profiles" section on page 2-9.

Zone Control Subprofile

Zoning enables the CIM client to set up access control between storage devices or user groups. The Zone Control subprofile is a subprofile of the Fabric profile and models zoning information for the fabric. It incorporates read and write functionality including the following operations:

- Creating and deleting zones and zone sets
- Creating and deleting zone members (using `ZoneMembershipSettingData`)
- Adding and removing zone members to zones
- Adding and removing zones to zone sets
- Activating and deactivating a zone set

The CIM server supports all the CIM classes and association classes described by the SMI-S zoning model.

Zone Control Extension

The zone set distribute functionality accross a VSAN has been added.

Enhanced Zoning and Enhanced Zoning Control Subprofile

The Enhanced Zoning and Enhanced Zoning Control subprofile is a subprofile of the Fabric profile and provides additional modeling of Cisco zoning information for management purposes. This includes support for the following:

- Creating and deleting zone aliases
- Adding and removing zone members to zone aliases

This subprofile supports all CIM classes and association classes described by the SMI-S zoning model except the concept of sessions for zoning.

When a zone is in enhanced mode, the SMI-S operation of retrieving zone instances is supported; however, the following SMI-S operations are not supported:

- Creating zones, zone sets, or zone alias
- Deleting zones, zone sets, or zone alias
- Activating a zoneset
- Adding a member to a zone in enhanced mode

Using the Zoning Subprofile

In the Cisco MDS CIM implementation, zoning occurs under the VSAN, not the fabric.

Note For Zoning subprofile instance diagrams, refer to the SMI-S at http://www.snia.org.

Table 2-5 shows how to use the classes and association classes of the Zoning subprofiles to model zoning.

Table 2-5 Using the Zoning Subprofile

Class	How Used
ZoneMembershipSettingData	Identifies zone members and indicates the member ID (defined in the CIM schema) and how the device was zoned.
ZoneAlias	Identifies zone aliases. Contains zone members (ZoneMembershipSettingData class) associated by the ElementSettingData association class.
ZoneSets	Identifies zone sets. Contains zones associated by the MemberOfCollection association class.
AdminDomain	Identifies VSANs. Only contains zone sets that are associated by the HostedCollection association class.
ZoneControl	Provides operations to control zone objects, such as creating, removing, and activating both zones and zone sets.
ZoneService	Manages the creation of zone sets, zones, zone aliases, and zone members, as well as activation of the zone set. The ZoneService class is hosted on the CISCO_Vsan class, which is a subclass of AdminDomain.
ActiveConnection	Represents a link that associates two ProtocolEndpoint classes as a connection that is currently carrying traffic.
DeviceAlias	Identifies device aliases (contains PWWN members) accross a fabric.

Zones and zone sets that are active have the Active property set to True by the CIM server. Zones can only contain the following types of objects:

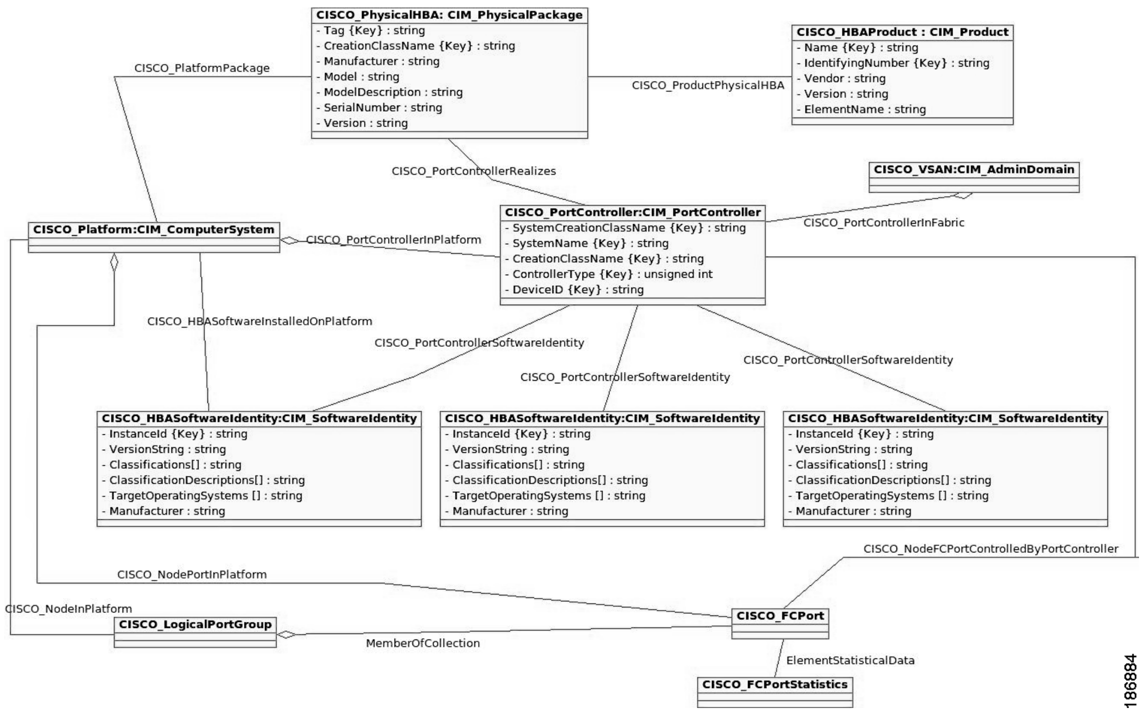
- Zone members (ZoneMembershipSettingData class) associated by the ElementSettingData association class.
- Zone aliases (ZoneAlias class; defined by SMI-S as NamedAddressCollections class) associated by the MemberOfCollection association class.

FDMI Subprofile

The Fabric Device Management Interface (FDMI) manages host bus adapters (HBA) through the Fabric and complements data in the fabric profile. It allows any entity in the fabric to expose the HBA information through the SMI without having an agent resident on the host containing the HBA. The fabric profile only addresses HBA type devices. The HBA Management Interface defined by FDMI is a subset of the interface defined by the Fibre Channel HBA API specification.

Figure 2-1 shows the FDMI subprofile instance diagram. The classes are defined in CISCO_HBA.mof. If the FDMI- enabled HBA supports the host name, then CISCO_PortController associates to a platform through CISCO_PortControllerInPlatform.If the FDMI-enabled HBA does not support the host name, then CISCO_PortController associates to a fabric, through CISCO_PortControllerInFabric.

Figure 2-1 UML Diagram for FDMI Subprofile



Using the FDMI Subprofile

In the Cisco MDS CIM implementation, the FDMI subprofile occurs under the fabric.

Note For FDMI subprofile instance diagrams, refer to the SMI-S at http://www.snia.org.

Table 2-6 shows how to use the classes and association classes of the FDMI subprofile.

Table 2-6 Using the FDMI Subprofile

Class	How Used
CISCO_PhysicalHBA	Represents FDMI enabled physical HBA card attached to a switch.
CISCO_HBAProduct	Represents product information of FDMI enabled physical HBA card attached to a switch.
CISCO_Platform	Represents a fabric-connected entity, containing one or more Node objects, that has registered with a fabric's Management Server service.
PortController	Represents the Port Controller of a FDMI enabled HBA.

Cisco MDS Extensions to the Switch and Fabric Profiles

The Cisco MDS 9000 Family CIM server supports additional classes that provide management for SAN features not covered by the standard SMI-S profiles. These extensions include:

VSAN Extensions

A VSAN is a virtual SAN that is created by partitioning the physical fabric into one or more logical fabrics. The Cisco MDS switches base routing on VSANs. The CIM client uses these VSAN extensions to identify VSANs and their associations to physical fabrics and switches.

The VSAN model in the CIM server uses the DMTF partition model. Partitioning, as defined by DMTF, is the virtual division of a single entity into multiple entities. It applies to any resource and can span namespaces and CIM object managers. Each partitioning entity manages its underlying partitions. A partitioned entity may be unaware that it is partitioned, and users may be unaware that a resource is shared. Refer to the standard partitioning model described in the CIM 2.8 schema, available from the DMTF website at http://www.dmtf.org.

Note For more information about VSANs, refer to the *Cisco MDS 9000 Family Fabric Manager Configuration Guide* or the *Cisco MDS 9000 Family CLI Configuration Guide*.

The VSAN extension provided by the Cisco MDS 9000 Family CIM server is both compatible with, and an extension of, the standard partition model. It models VSANs as a partitioned physical fabric. The E ports, F ports, PortChannels, and ports supporting FC IP and iSCSI on Cisco MDS switches all support the partitioning model.

The `HostedDependency` association class can describe the following relationships:

- Partitioning (fan in)
  - Antecedent is the partitioning entity
  - Dependent is the partitioned entity
- Clustering (fan out)

Figure 2-2 shows a UML diagram of a fabric partitioned into two VSANs. The physical switch is partitioned into two logical switches, Partitioned Switch 1 and Partitioned Switch 2. The partitions are identified as belonging to the physical switch using the `HostedDependency` association class. The VSANs are identified as belonging to the corresponding switch partitions using the `Component` association class.

Figure 2-2 UML Diagram of Fabric Partitioning

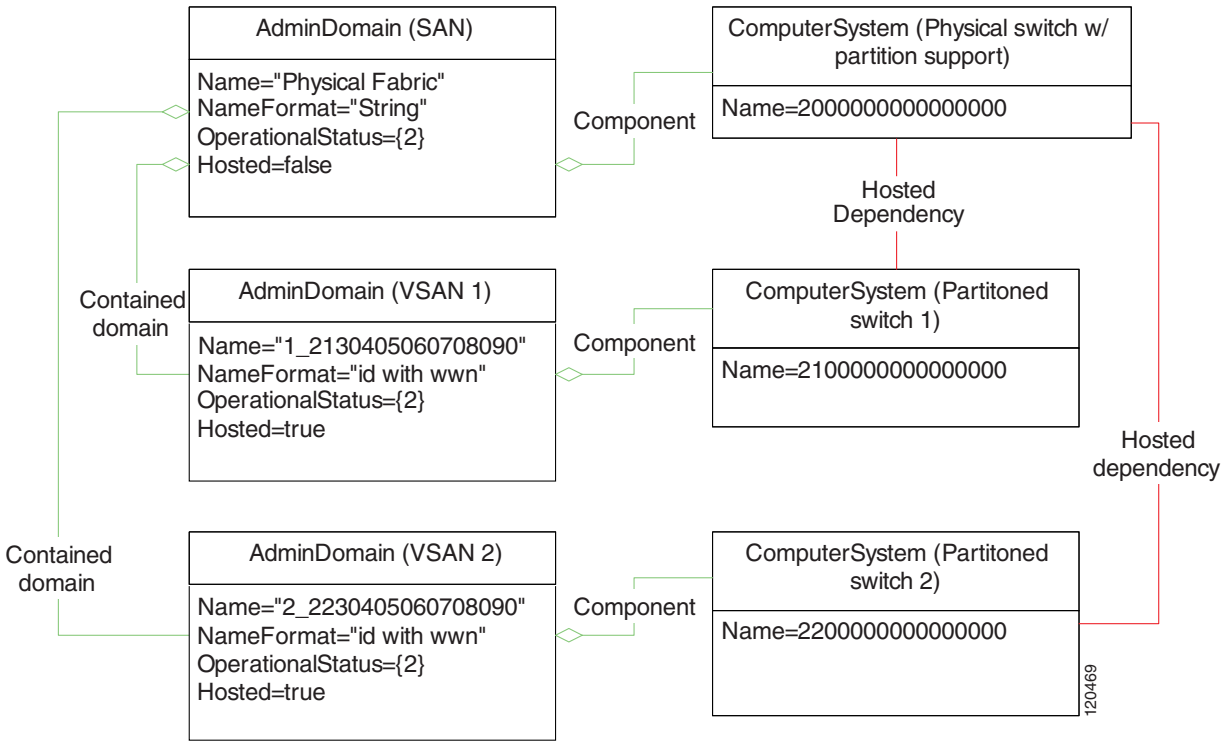
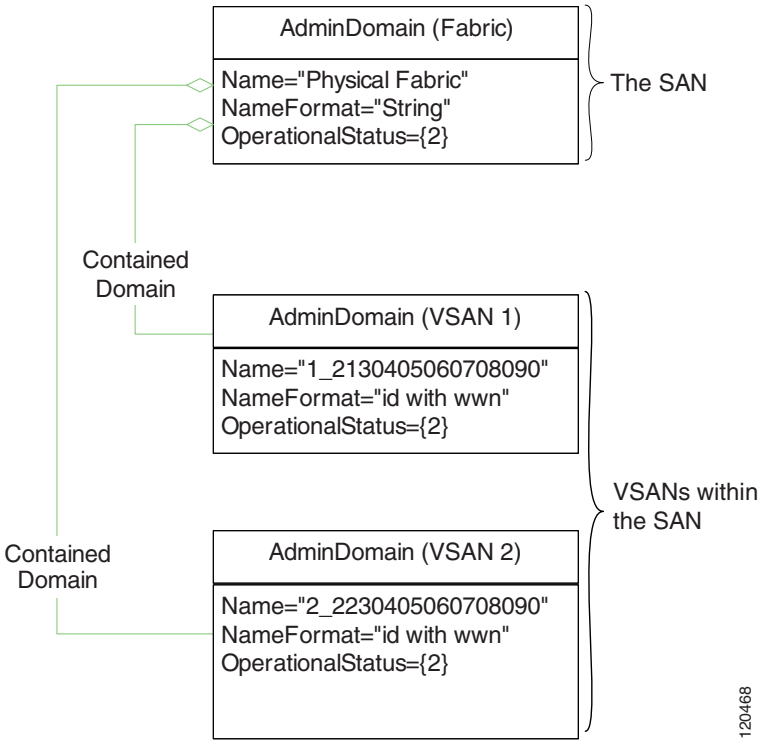


Figure 2-3 isolates the VSAN component from Figure 2-2. The physical fabric is partitioned into two VSANs, VSAN 1 and VSAN 2. Each VSAN is identified by the `AdminDomain` class. The VSANs can be identified as belonging to the physical fabric using the `ContainedDomain` association class.

Figure 2-3 VSAN Partitioning Example

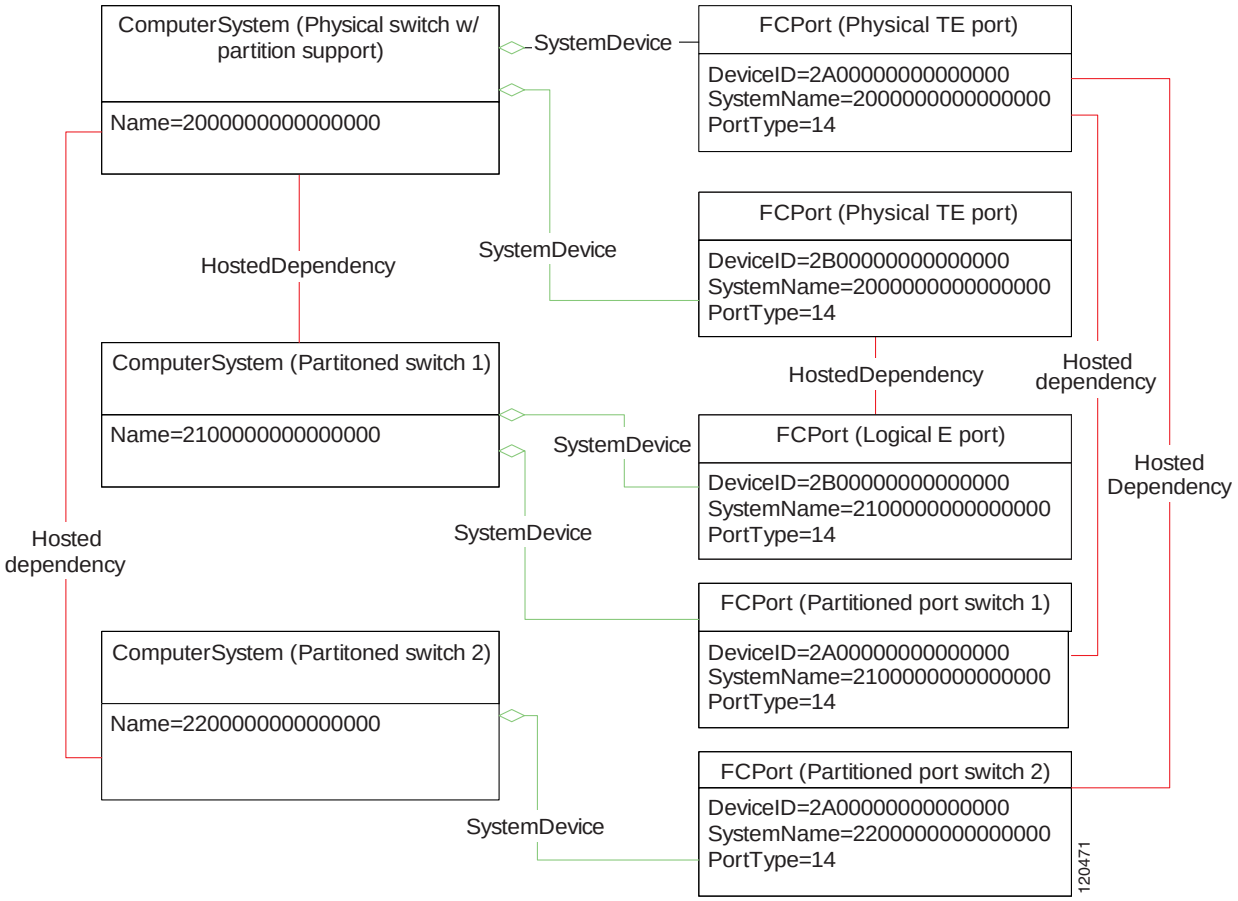


TE Port Extensions

TE ports are E ports that can carry traffic for multiple VSANs. The CIM server uses the existing fabric-to-FC port association classes to model membership of TE ports in multiple VSANs. Figure 2-4 shows the physical and logical port relationship to the switch. The two illustrated physical ports are partitioned into logical ports, and the logical ports are identified as belonging to the physical ports using the `HostedDependency` association class. A physical TE port is partitioned into two logical ports, one for Partitioned switch 1 (associated to VSAN 1 in Figure 2-2) and one for Partitioned switch 2 (associated to VSAN 2 in Figure 2-2).

The physical ports are identified as components of the physical switch using the `SystemDevice` association class, and the partitioned ports are identified as components of the corresponding partitioned switch using the `SystemDevice` association class.

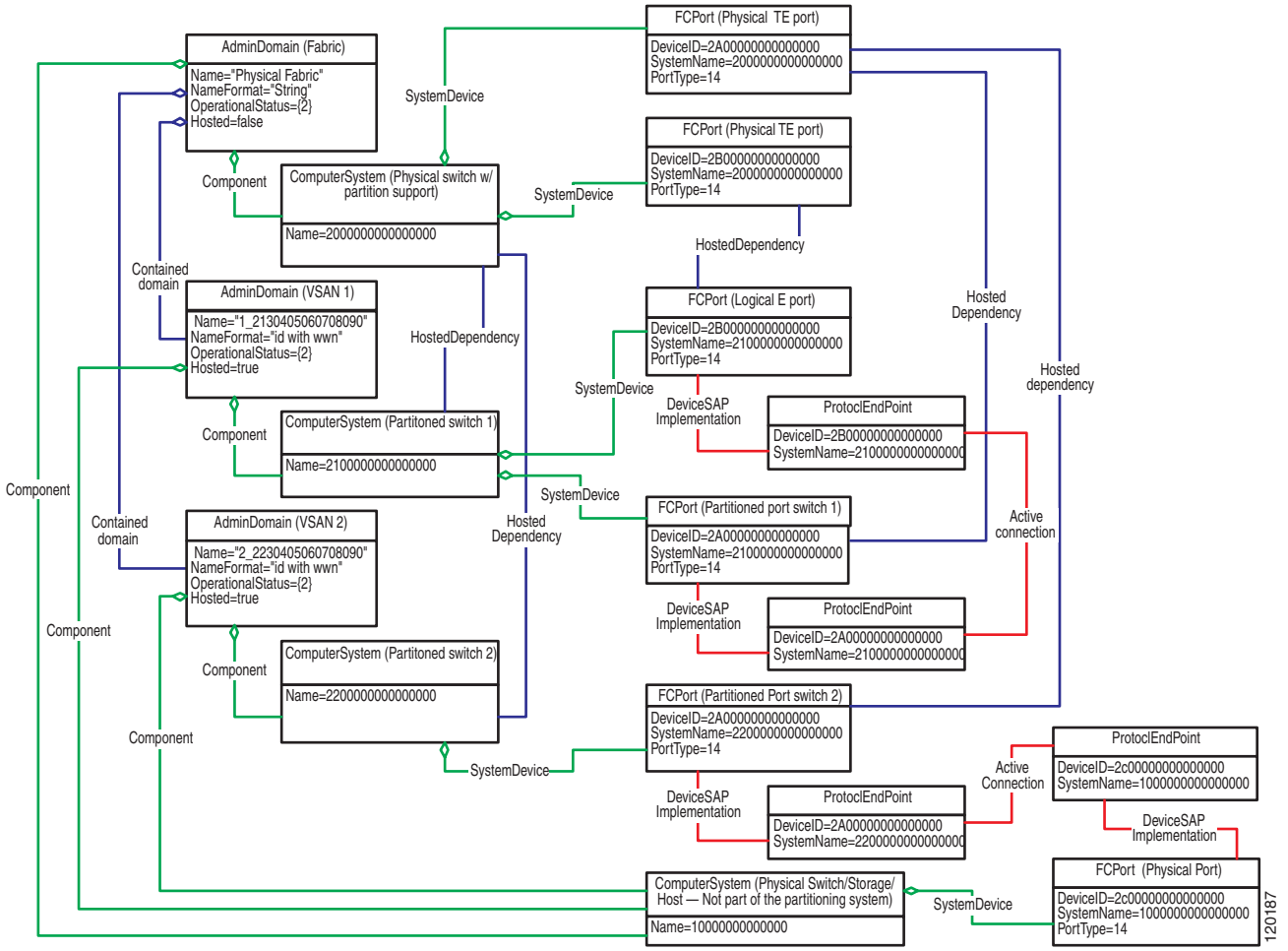
Figure 2-4 TE Port Partitioning Example



Note For more information about trunking, refer to the *Cisco MDS 9000 Family Fabric Manager Configuration Guide* or the *Cisco MDS 9000 Family CLI Configuration Guide*.

Figure 2-5 shows the full UML diagram for VSAN fabric and port partitioning in a SAN switch.

Figure 2-5 UML Diagram for VSAN Partitioning



PortChannel Extensions

A PortChannel is the aggregation of multiple physical Fibre Channel ports into one logical port to provide aggregated bandwidth, load balancing, and link redundancy. The CIM server supports a PortChannel port type in the `Cisco_FCPort` class. The `Component` association class can be used to associate individual ports with a PortChannel.

PortChannels are supported by the CIM server only for the local switch on which the CIM server is running. The CIM server also exports active connections for remote PortChannels, with two limitations:

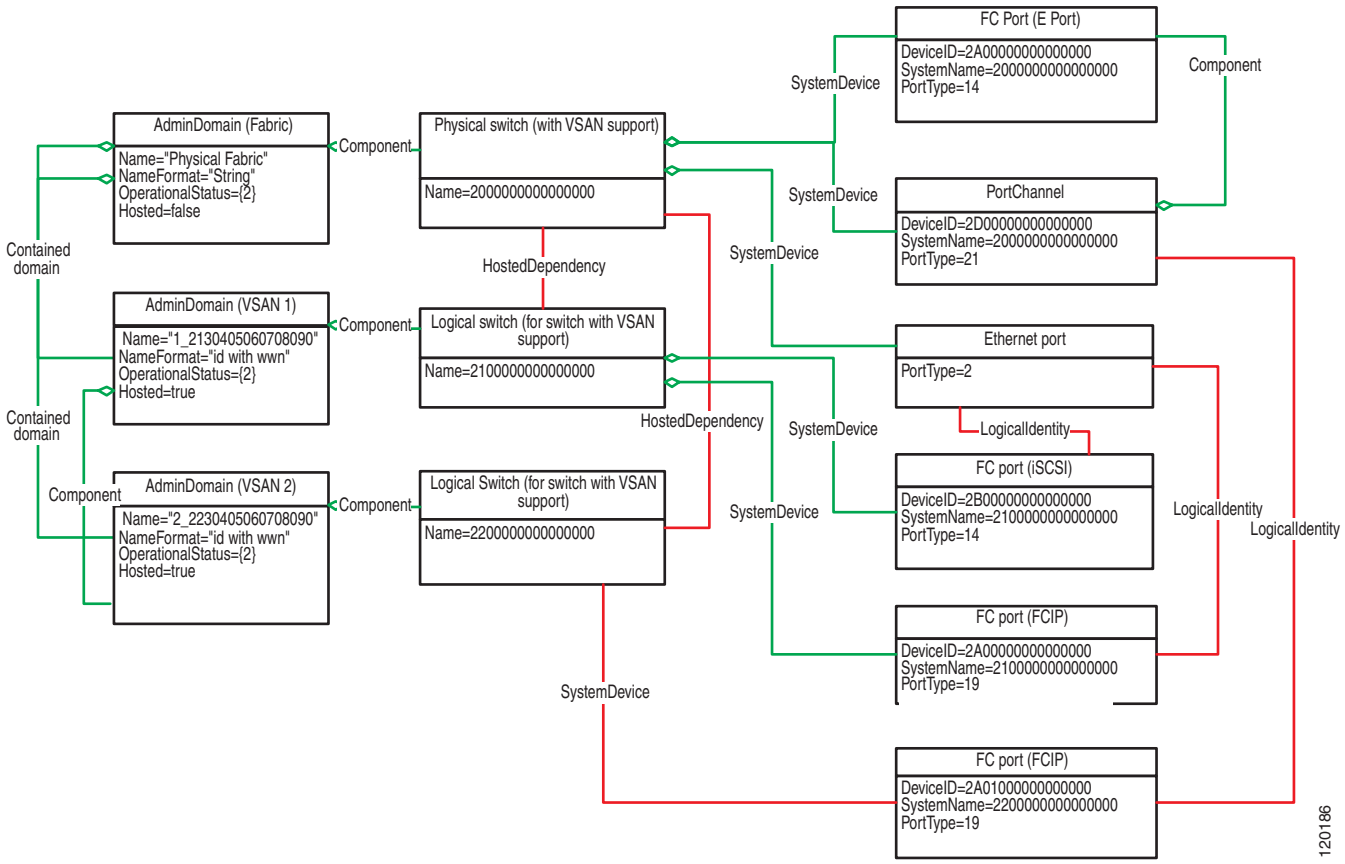
- The remote PortChannel WWN is not available; the remote switch WWN and port index are provided.
- The `Component` and `LogicalIdentity` association classes of the remote PortChannel are not available.

Note For more information about PortChannels, refer to the *Cisco MDS 9000 Family Fabric Manager Configuration Guide* or the *Cisco MDS 9000 Family CLI Configuration Guide*.

Figure 2-6 shows the relationships among ports and PortChannels on the switch that is running the CIM server. In this example:

- The PortChannels and ports are identified as belonging to the physical switch using the `SystemDevice` association class.
- The individual ports are identified as belonging to the PortChannels using the `Component` association class.

Figure 2-6 UML Instance Diagram of the Relationships Among Ports Using FCIP, PortChannels, and Ethernet Ports



FCIP Extensions

The CIM server uses the current `FCPort` class to discover information about ports supporting FCIP. For the local switch (the switch on which the CIM server is running), the CIM server uses the `LogicalIdentity` association class to link ports supporting FCIP that are on the same module.

The CIM server exports active connections for remote ports running FCIP, with two limitations:

- The WWN of the port running FCIP is not available; the remote switch WWN and port index are provided.
- The `LogicalIdentity` association class of the port running FCIP is not available.

Note For more information about FCIP, refer to the *Cisco MDS 9000 Family Fabric Manager Configuration Guide* or the *Cisco MDS 9000 Family CLI Configuration Guide*.

Figure 2-6 shows the relationships among ports running FCIP and other entities. In this example:

- The ports running FCIP are associated with other entities using the `LogicalIdentity` association class. One port running FCIP is a logical entity of an individual Ethernet port, and the other is a logical entity of a PortChannel that is comprised of Ethernet ports.
- The port running FCIP, Ethernet port, and PortChannel are identified as belonging to the physical switch using the `SystemDevice` association class.

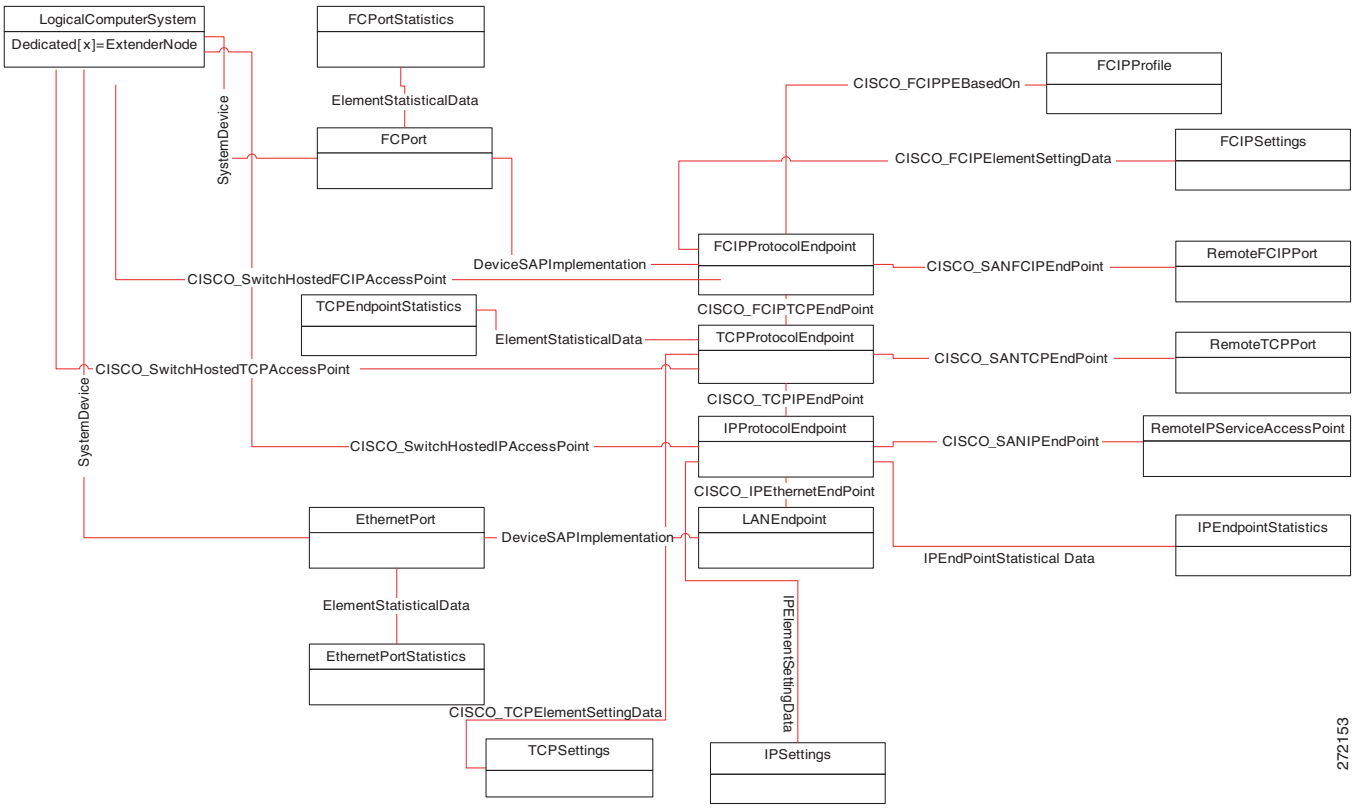
FCIP Subprofile

FCIP enables the transmission of Fibre Channel (FC) information by tunneling data between SAN facilities over IP networks which facilitates data sharing over a geographically distributed enterprise. FCIP Extender Subprofile helps in the discovery of the FCIP topology. The `ComputerSystem` class is the core of the model (which is CISCO_LogicalComputerSystem) that the FCIP interfaces are created for and to which they belong. The `TCPSettings` and `IPSettings` classes represent the global configuration of the FC Extender transport layer. The `Port` group of classes contains the following classes: `FCPort` and `EthernetPort`. The `FCPort` class represents the connection of a FC Extender to a SAN. This class connects to other `FCPort` classes to represent Fibre Channel connections. This class could be replaced with other port types to represent SANs based on other interconnect technology. The `EthernetPort` class represents an Ethernet link between FC Extender nodes.

The CIM FCIP settings defines the FCIP settings for a group of ProtocolEndpoints (ProtocolIFType - Fcip) which belongs to the `ComputerSystem` (Extender Node)

Figure 2-7 shows the relationship between various classes in the FCIP Subprofile.

Figure 2-7 FCIP Subprofile Instance Diagram



iSCSI Extensions

You can use the current `EthernetPort` class to discover information about the port and use the `LogicalIdentity` association class to associate Gigabit Ethernet ports with iSCSI. This association class is only available for ports local to the CIM server.

Figure 2-6 shows the relationships among ports running iSCSI and other entities. In this example:

- The port running iSCSI is identified as belonging to the Ethernet port using the `LogicalIdentity` association class.
- The port running iSCSI is identified as belonging to the physical switch using the `SystemDevice` association class.

Note For more information about iSCSI, refer to the *Cisco MDS 9000 Family Fabric Manager Configuration Guide* or the *Cisco MDS 9000 Family CLI Configuration Guide*.

Fabric Profile Extensions

In addition to the standard Fabric profile, the following classes and association classes that are specific to the Cisco MDS 9000 Family are supported:

```
CISCO_ActiveConnection
CISCO_AdminDomain
```

```
CISCO_FCPort
CISCO_FCPortCapabilities
CISCO_FCPortSettings
CISCO_Vsan
CISCO_Component
CISCO_ComputerSystem
CISCO_ConnectivityCollection
CISCO_ConnectivityMemberOfCollection
CISCO_ContainedDomain
CISCO_DeviceSAPImplementation
CISCO_FCPortStatistics
CISCO_HostedAccessPoint
CISCO_HostedCollection
CISCO_ProtocolEndPoint
CISCO_PhysicalPackage
CISCO_PhysicalElement
CISCO_Product
CISCO_Realizes
CISCO_SystemDevice
CISCO_ComputerSystemPackage
CISCO_ElementStatisticalData
CISCO_LogicalPortGroup
CISCO_LogicalModule
CISCO_ModulePort
CISCO_HostedDependency
CISCO_LogicalIdentity
CISCO_PhysicalComputerSystem
CISCO_LogicalComputerSystem
CISCO_FCNodeMemberOfCollection
```

The port identifiers for the CISCO_FCPort class that are supported by the CIM server are described in Table 2-7. Port identifiers 16004 through 16012 are Cisco extensions.

Table 2-7 Port Identifiers Supported by the Cisco MDS 9000 Family CIM Server

Port Identifier	Port Type
0	Unknown
1	Other
10	N
11	NL
12	F/NL
13	Nx
14	E
15	F
16	FL
17	B
18	G
16004(cisco specific) etc.	PortChannel
16010	FCIP
16011	iSCSI-F

Table 2-7 Port Identifiers Supported by the Cisco MDS 9000 Family CIM Server (continued)

Port Identifier	Port Type
16012	iSCSI-N
16000...65535	Vendor reserved

See the "Cisco Fabric MOF" section on page A-12 for the full definition of the Cisco fabric extensions.

Zoning Subprofile Extensions

In addition to the standard zoning subprofiles, the following classes and association classes that are specific to Cisco are supported:

```
CISCO_HostedService
CISCO_ZoneMemberOfCollection
CISCO_ZoneMembershipSettingData
CISCO_ZoneSet
CISCO_Zone
CISCO_ZoneCapabilities
CISCO_ZoneAlias
CISCO_ElementSettingData
CISCO_ZoneService
CISCO_SystemSpecificCollection
```

See the "Cisco Zone MOF" section on page A-14 for the full definition of the Cisco zoning extensions.

FDMI Subprofile Extensions

In addition to the standard FDMI subprofile, the following classes and association classes that are specific to the Cisco MDS 9000 Family are supported:

```
PortControllerRealizes
PlatformPackage
PortControllerSoftwareIdentity
HBASoftwareInstalledOnPlatform
NodeFCPortControlledByPortController
ProductPhysicalHBA
PlatformInFabric
NodePortInPlatform
NodeInPlatform
PortControllerInPlatform
PortControllerInFabric
```

See the "Cisco FDMI MOF" section on page A-16 for the full definition of the Cisco FDMI extensions.

CIM Indications

SMI-S provides asynchronous *indications* for changes in the CIM server or the managed elements controlled by the CIM server. These indications can inform a CIM client that:

- The Zone database has changed. (Zone indications are sent in the enhanced zoning mode.)

- The SAN configuration has changed.
- The SAN switch health has degraded.
- The SAN fabric performance has degraded.
- The name server database has changed.
- A VSAN has been added or deleted or modified.
- The fan status has changed.
- The temperature status has changed.
- A power supply status has changed.
- A field replaceable unit (FRU) has been inserted or removed or changed.
- Users have been added or deleted, or a password changed for a user or a login failed for a user.

Indications can also be used when a CIM class method is invoked that will take a long time to finish. Rather than tie up the CIM server (block) until the operation completes, the CIM server responds that the operation started, and the CIM server continues handling other requests (non-blocking). When the original, long operation completes, the CIM server sends a CIM indication asynchronously to the CIM client, showing the result of the operation. A CIM client must subscribe to indications it wants to receive from the CIM server.

The Cisco MDS 9000 Family CIM server supports the following Cisco-specific indications:

```
CISCO_LinkStateChange
CISCO_LinkUp
CISCO_Linkdown
CISCO_MediaFRUInserted
CISCO_MediaFRURemoved
CISCO_VSANChanged
CISCO_ZoneSetAlert
CISCO_ZoneAlert
CISCO_EnvironmentalAlert
CISCO_FanAlert
CISCO_PowerAlert
CISCO_TempAlert
CISCO_NameServerDatabaseChanged
```

See the "Cisco Indications MOF" section on page A-25 for the Cisco Indications MOF file.

CHAPTER 3

Configuring and Using the CIM Server

This chapter provides the steps to configure the CIM server in Cisco MDS 9000 Family products and provides some sample scenarios for using CIM objects to manage your SAN. This chapter includes the following sections:

Note For information about CLI commands, refer to the *Cisco MDS 9000 Family Command Reference*.

Configuring the CIM Server

The CIM server can be configured through the CLI. To configure the CIM server, you must first enable it. For added security, you can install an SSL certificate to encrypt the login information and enable HTTPS before enabling the CIM server. The CIM server requires HTTP or HTTPS or both to be enabled. By default, HTTP is enabled and secure HTTPS is disabled. Using HTTPS encrypts all management traffic between the CIM client and the CIM server and is the recommended configuration.

Creating a Certificate Using OpenSSL

You need a valid certificate to configure the CIM server. You can use OpenSSL to create the private key and certificate needed by the CIM server. Refer to http://www.openssl.org.

To create a self-signed certificate and private key using OpenSSL, follow these steps:

Step 1 Create a file called ssl.conf on your workstation. This is used to specify the distinguished name. Sample contents of this file are:

```
[ req ]
distinguished_name = req_distinguished_name
prompt = no
[ req_distinguished_name ]
CN = Common Name
emailAddress = test@email.address
```

Step 2 Use the **openssl** command to create the private key and the certificate by typing the following:

```
/usr/bin/openssl req -x509 -days 365 -newkey rsa:2048 -nodes -config ./ssl.conf -keyout
./key.pem -out ./cert.pem
```

Step 3 Concatenate the private key and the certificate into a single file.

```
cat key.pem cert.pem > cimserver1.pem
```

Step 4 Copy cimserver1.pem to bootflash: on your switch. You use this as the certificate when configuring the CIM server.

Installing the Certificate and Enabling the CIM Server

To configure a CIM server using the HTTPS protocol, follow these steps:

	Command	Purpose
Step 1	`switch# config t`	Enters configuration mode.
Step 2	`switch(config)# cimserver certificate bootflash:cimserver1.pem`	Installs a Secure Socket Layer (SSL) certificate specified in the file named with a .pem extension.
	`switch(config)# cimserver clearcertificate Certificate1`	Optional. Clears the specified SSL certificate (Certificate1).
Step 3	`switch(config)# cimserver enableHttps`	Enables HTTPS (secure protocol).
	`switch(config)# no cimserver enableHttps`	Optional. Disables HTTPS (default).
Step 4	`switch(config)# feature cimserver`	Enables the CIM server.
	`switch(config)# no feature cimserver`	Optional. Disables the CIM server (default).

To configure a CIM server using the HTTP protocol, follow these steps:

	Command	Purpose
Step 1	`switch# config t`	Enters configuration mode.
Step 2	`switch(config)# cimserver enable`	Enables the CIM server using the default HTTP (nonsecure) protocol.
	`switch(config)# no cimserver enable`	Optional. Disables the CIM server (default).
	`switch(config)# no cimserver enableHttp`	Optional. Disables HTTP.
	`switch(config)# cimserver enableHttp`	Optional. Enables HTTP and reverts to the switch default.

Setting a CIM Server Log Level

To configure a CIM server log level, follow these steps:

	Command	Purpose
Step 1	`switch# show cimserver logs`	Displays CIM server logs.

	Command	Purpose
Step 2	`switch# conf t`	Configures CIM server log levels.
Step 3	`switch(config)# cimserver logLevel`	Sets the CIM server log level. The log levels range from 1 to 5 (1-trace, 2-information, 3-warning, 4-severe, 5-fatal).

Performing Discovery and Performance Monitoring

You can use the Fabric and Switch profiles to implement discovery and performance monitoring. See the "Fabric Profile" section on page 2-4 and the "FDMI Subprofile" section on page 2-7 for more information on these profiles.

Discovery provides information about the physical and logical entities within the SAN. This information changes dynamically as SAN entities are added, moved, or removed. Discovery also includes the discovery of object classes as well as related association classes, properties, and return status codes that are provided by servers in the managed environment.

Table 3-1 shows how to perform discovery, using the intrinsic methods defined by CIM. Use these methods to retrieve information about the switch and fabric.

Table 3-1 Performing Discovery

Method	How Used
enumerateInstances()	Enumerates instances of a CIM class.
enumerateInstanceNames()	Enumerates names of instances of a CIM class.
getInstance()	Gets a CIM instance.
associators()	Enumerates associators of a CIM object.
associatorName()	Enumerates names of associators of a CIM object.
references()	Enumerates references to a CIM object.
referenceName()	Enumerates names of references to a CIM object.

The target of these methods is the location of the CIM server, which is identified by the switch IP address.

Performance monitoring provides the status and statistics for the local ports. Only ports on the local switch can be monitored. You can retrieve statistics from the properties of the `FCPortStatistics` class for `FCPort` class instances on the CIM server.

Note The namespace of the CIM server is root/cimv2.

Modeling a Module Using the Blade Subprofile

You can use the Blade Subprofile to model a supervisor module, switching module, or services module within a switch. Table 3-2 shows how to use the association classes in this subprofile to map ports to modules and modules to switches.

Table 3-2 Using the Blade Subprofile Association Classes

Class	How Used
Realizes	Associates the `LogicalModule` class to the `PhysicalPackage` class. Use this class to map modules to the switch.
ModulePort	Associates the `FCPort` class to the `LogicalModule` class. Use this class to map individual ports to modules within the switch.

See the "Blade Subprofile" section on page 2-3 for more information about the Blade subprofile.

Configuring Zoning

The zoning model in the SMI-S uses extrinsic and intrinsic methods to manage zoning within the SAN fabric. Extrinsic methods are methods specific to a particular class. Intrinsic methods are methods inherited from the CIM and present in every applicable class.

To create a zone member (referred to as `ZoneMembershipSettingData`), zone, zone alias, or zone set, use `invokeMethod(`*`operand`*`)`. The operand can be one of the extrinsic methods from the zoning subprofiles as shown in Table 3-3.

Table 3-3 Zoning Extrinsic Methods

Extrinsic Method	How Used
`CreateZoneMembershipSettingData()`	Creates a `ZoneMembershipSettingData` and adds it to the specified `Zone` or `NamedAddressCollection`. The `ConnectivityMemberID` is dependent upon the `ConnectivityMemberType`.
`CreateZone()`	Creates a `Zone` and associates it to `AdminDomain` where the `ZoneService` is hosted.
`CreateZoneAlias()`	Creates a `ZoneAlias` and associates it to `AdminDomain` where the `ZoneService` is hosted.
`CreateZoneSet()`	Creates a `ZoneSet` and associates it to the `AdminDomain` where the `ZoneService` is hosted.
`AddZone()`	Adds the `Zone` to the specified `ZoneSet`. Adding a `Zone` to a `ZoneSet` extends the zone enforcement definition of the `ZoneSet` to include the members of that `Zone`. If adding the `Zone` is successful, the `Zone` should be associated to the `ZoneSet` by `MemberOfCollection`.
`AddZoneMembershipSettingData()`	Adds `ZoneMembershipSettingData` to the `Zone` or `NamedAddessCollection`.
`AddZoneAlias()`	Adds the `Zone Alias to` the `Zone`.

Table 3-3 Zoning Extrinsic Methods (continued)

Extrinsic Method	How Used
`ActivateZoneSet ()`	Sets the ZoneSet to active.
`ZoneSetDistribute()`	Distributes the full ZoneSet along with active zoneset per vsan in the Fabric
`CreatDeviceAlias()`	Creates a device alias with the given device alias name and PWWN.

Use the `DeleteInstance(`*`instance_name`*`)` intrinsic method to remove a zoning item from a collection or to delete the zoning item entirely. The `DeleteInstance()` method requires a reference to one of the instances shown in Table 3-4.

Table 3-4 Deleting Zoning Entities

Class	How Used
`CIM_ElementSettingData`	Removes a zone member from a zone or zone alias. Use `deleteInstance()` to delete the instance of `ElementSettingData` that associates the zone member to the zone.
`CIM_MemberOfCollection`	Removes a zone or zone alias from a zone set. Use `deleteInstance()` to delete the instance of `MemberOfCollection` that associates the zone or zone alias to the zone set.
`CIM_ZoneMembershipSettingData`	Deletes a zone member. This automatically removes it from any zone or zone alias.
`CIM_Zone`	Deletes a zone.
`CIM_ZoneAlias`	Deletes a zone alias.
`CIM_ZoneSet`	Deletes a zone set.
`RemoveDeviceAlias()`	Removes the device alias with the given device alias name.

See the "Zone Control Subprofile" section on page 2-6 and the "Enhanced Zoning and Enhanced Zoning Control Subprofile" section on page 2-6 for information about the zoning subprofiles.

Note These methods are supported for the CIM protocol only and cannot be entered as commands at the CLI. For more information about SMI-S, refer to the SNIA website at http://www.snia.org. For more information about CIM, refer to the DMTF website at http://www.dmtf.org.

APPENDIX A

Managed Object Format Files

This appendix provides the text from the Managed Object Format (MOF) files for the Cisco MDS 9000 Family CIM server extensions. These MOF files are an extension to the standard MOF files and provide management for VSANs, PortChannels, FCIP, and iSCSI.

For information about the standard MOF files, refer to the DMTF website at the following URL: http://www.dmtf.org.

This appendix includes the following sections:

Cisco MOF Files for Cisco NX-OS Release 4.2(1)

This section includes the MOF files supported by Cisco NX-OS Release 4.2(1). It includes the following topics:

Cisco Zone.MOF

The Cisco Zone.MOF for Cisco NX-OS Release 4.2(1) provides extensions to the Zone profile to manage Zoneset, Zones, Zonealias and devicealias.

```
[Version ("1.0.0"), Description (
    "cisco zoneset class")]
class CISCO_ZoneSet : CIM_ZoneSet
{
};

class CISCO_Zone : CIM_Zone
{};

class CISCO_ZoneAlias : CIM_NamedAddressCollection
{};
```

```
class CISCO_DeviceAlias : CIM_NamedAddressCollection
{
    string name;
    string pwwn;
};
class CISCO_ZoneMemberSettingData : CIM_ZoneMembershipSettingData
{};

class CISCO_ZoneService : CIM_ZoneService
{};

class CISCO_SystemSpecificCollection : CIM_SystemSpecificCollection
{};

class CISCO_ZoneMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_ElementSettingData : CIM_ElementSettingData
{};

class CISCO_HostedService : CIM_HostedService
{};

class CISCO_ZoneHostedCollection : CIM_HostedCollection
{};
```

Cisco MOF Files for Cisco NX-OS Release 4.1(3)

This section includes the MOF files supported by Cisco NX-OS Release 4.1(3). It includes the following topics:

Cisco Fabric MOF

The Cisco Fabric MOF for Cisco NX-OS Release 4.x provides extensions to the fabric profile to manage VSANs, PortChannels, and other Cisco-specific entities within the fabric.

```
[Version ("1.0.0"), Description (
    "cisco faric and switch profile classes")]
class CISCO_ActiveConnection : CIM_ActiveConnection
{};

class CISCO_AdminDomain : CIM_AdminDomain
{};

   [Version ( "2.7.1"), Description (
       "Capabilities and management of a Fibre Channel Port Device.") ]
class CISCO_FCPort : CIM_FCPort {

      [Override ( "PortType"), Description (
          "The specific mode currently enabled for the Port.  The "
          "values: \"N\" = Node Port, \"NL\" = Node Port supporting FC "
          "arbitrated loop, \"E\" = Expansion Port connecting fabric "
          "elements (for example, FC switches), \"F\" = Fabric "
          "(element) Port, \"FL\" = Fabric (element) Port supporting "
```

```
            "FC arbitrated loop, \"B\" = Bridge and \"G\" = Generic "
            "Port.  PortTypes are defined in the ANSI X3 standards.  "
            "When set to 1 (\"Other\"), the related property "
            "OtherPortType contains a string description of the port's "
            "type."),
         ValueMap { "0", "1", "10", "11", "12", "13", "14", "15", "16",
            "17", "18", "16004", "16010", "16011", "16012", "16000..65535"},
         Values { "Unknown", "Other", "N", "NL", "F/NL", "Nx", "E", "F",
            "FL", "B", "G", "PortChannel", "FCIP", "ISCSI-F", "ISCSI-N", "Vendor Reserved"}
]
   uint16 PortType;

   uint16 PortAvailability = 2;
};

class CISCO_Vsan : CIM_AdminDomain {
};
class CISCO_Component : CIM_Component
{};

class CISCO_ComputerSystem : CIM_ComputerSystem
{
 boolean PrimarySwitch;
};

class CISCO_ConnectivityCollection : CIM_ConnectivityCollection
{};

class CISCO_ConnectivityMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_ContainedDomain : CIM_ContainedDomain
{};

class CISCO_DeviceSAPImplementation : CIM_DeviceSAPImplementation
{};


class CISCO_FCPortStatistics : CIM_FCPortStatistics
{};

class CISCO_HostedAccessPoint : CIM_HostedAccessPoint
{};

class CISCO_HostedCollection : CIM_HostedCollection
{};

class CISCO_ProtocolEndPoint : CIM_ProtocolEndPoint
{};

class CISCO_PhysicalPackage : CIM_PhysicalPackage
{};

class CISCO_PhysicalElement : CIM_PhysicalElement
{};

class CISCO_Product : CIM_Product
{};

class CISCO_Realizes : CIM_Realizes
{};

class CISCO_SystemDevice : CIM_SystemDevice
{};
```

```
class CISCO_ComputerSystemPackage : CIM_ComputerSystemPackage
{};
class CISCO_ProductPhysicalComponent : CIM_ProductPhysicalComponent
{};
class CISCO_ElementStatisticalData : CIM_ElementStatisticalData
{};
class CISCO_LogicalPortGroup : CIM_LogicalPortGroup
{};


class CISCO_LogicalModule : CIM_LogicalModule
{};

class CISCO_ModulePort : CIM_ModulePort
{};

class CISCO_EthernetPort : CIM_EthernetPort
{};

class CISCO_HostedDependency : CIM_HostedDependency
{};

class CISCO_LogicalIdentity : CIM_LogicalIdentity
{};

class CISCO_PhysicalComputerSystem : CISCO_ComputerSystem
{
string IpAddress;
};

class CISCO_LogicalComputerSystem : CISCO_ComputerSystem
{};

class CISCO_FCNodeMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_FabricHostedService : CIM_HostedService
{};


class CISCO_FabricService : CIM_Service [Description ("The method creates a VSAN.")]
uint32 CreateVSAN(
[Required, IN, Description (
"A user-friendly name for the ZoneSet that is unique "
"within the AdminDomain."),
ModelCorrespondence { "CIM_ZoneSet.ElementName" }]
string VSANName,
[Required, IN, Description (
"A user-friendly name for the ZoneSet that is unique "
"within the AdminDomain."),
ModelCorrespondence { "CIM_ZoneSet.ElementName" }]
uint16 VSANIdentifier,
[IN ( false ), OUT, Description ("A reference to the newly created VSAN .")]
CISCO_VSAN ref VSAN );

uint32 ZoneSetDistribute (
[Required, IN, Description ("VsanID accross which the zoneset needs to be distributed")]
CISCO_VSAN ref VSAN);
};
```

Cisco Copy Running MOF

The Cisco Copy Running MOF copies the current running configuration of the switch to the startup configuration.

```
for copy running startup
class Copy_Running:CIM_Service
{
    uint32 Execute();
};
```

Cisco MOF Files for Cisco NX-OS Release 4.1(1b)

This section includes the MOF files supported by Cisco NX-OS Release 4.1(1b). It includes the following topic:

Cisco FCIP MOF

```
N[Provider("FCIPProvider"),
    Description (
        "A communication endpoint which, when its associated interface "
        "device is connected to a LAN, may send and receive data "
        "frames. LANEndpoints include Ethernet, Token Ring and FDDI "
        "interfaces.")]
class CISCO_LANEndpoint : CIM_ProtocolEndpoint {

    [Override("SystemCreationClassName"), Key,
        Propagated("CIM_System.CreationClassName"),
        Description ("The scoping System's CreationClassName."),
        MaxLen ( 256 )]
    string SystemCreationClassName;

    [Override("SystemName"), Key, Propagated("CIM_System.Name"),
        Description ("The scoping System's Name."),
        MaxLen ( 256 )]
    string SystemName;

    [Override("CreationClassName"), Key, Description (
        "CreationClassName indicates the name of the class or the "
        "subclass used in the creation of an instance. When used "
        "with the other key properties of this class, this property "
        "allows all instances of this class and its subclasses to be "
        "uniquely identified."),
        MaxLen ( 256 )]
    string CreationClassName;

    [Override("Name"), Key, Description (
        "A string that identifies this ProtocolEndpoint with either "
        "a port or an interface on a device. To ensure uniqueness, "
        "the Name property should be prepended or appended with "
        "information from the Type or OtherTypeDescription "
        "properties. The method selected is described in the "
        "NameFormat property of this class."),
        MaxLen ( 256 )]
    string Name;
```

```
    [Override ( "NameFormat" ), Description (
        "NameFormat contains the naming heuristic that is chosen to "
        "ensure that the value of the Name property is unique. For "
        "example, one might choose to prepend the name of the port "
        "or interface with the Type of ProtocolEndpoint that this "
        "instance is (e.g., IPv4) followed by an underscore."),
        MaxLen ( 256 )]
    string NameFormat;

    [Override("ProtocolIFType"), Description (
        "ProtocolIFType's enumeration is limited to Layer 2-related and "
        "reserved values for this subclass of ProtocolEndpoint."),
        ValueMap { "1", "6", "9", "15", "222..4095", "4116..32767", "32768.." },
        Values { "Other", "Ethernet CSMA/CD", "ISO 802.5 Token Ring",
            "FDDI", "IANA Reserved", "DMTF Reserved", "Vendor Reserved" }]
    uint16 ProtocolIFType = 6;
};
/[Provider("FCIPProvider"),
    Description (
        "A communication point from which data may be "
        "sent or received. ")]
class CISCO_FCIPProtocolEndpoint : CIM_ProtocolEndpoint {

    [Override("SystemCreationClassName"), Key,
        Propagated("CIM_System.CreationClassName"),
        Description ("The scoping System's CreationClassName."),
        MaxLen ( 256 )]
    string SystemCreationClassName;

    [Override("SystemName"), Key, Propagated("CIM_System.Name"),
        Description ("The scoping System's Name."),
        MaxLen ( 256 )]
    string SystemName;

    [Override("CreationClassName"), Key, Description (
        "CreationClassName indicates the name of the class or the "
        "subclass used in the creation of an instance. When used "
        "with the other key properties of this class, this property "
        "allows all instances of this class and its subclasses to be "
        "uniquely identified."),
        MaxLen ( 256 )]
    string CreationClassName;

    [Override("Name"), Key, Description (
        "A string that identifies this ProtocolEndpoint with either "
        "a port or an interface on a device. To ensure uniqueness, "
        "the Name property should be prepended or appended with "
        "information from the Type or OtherTypeDescription "
        "properties. The method selected is described in the "
        "NameFormat property of this class."),
        MaxLen ( 256 )]
    string Name;

    [Override("NameFormat"), Description (
        "NameFormat contains the naming heuristic that is chosen to "
        "ensure that the value of the Name property is unique. For "
        "example, one might choose to prepend the name of the port "
        "or interface with the Type of ProtocolEndpoint that this "
        "instance is (e.g., IPv4) followed by an underscore."),
        MaxLen ( 256 )]
    string NameFormat;

    [Description (
        "Identifies the FCIP Tunnel on the GigE port."
```

```
        "The value ranges from 0 to 7." )]
    uint16 TunnelID;

    [Description (
        "IP address of the remote end of the "
        "FCIP connection.")]
    string RemoteIPAddress;

    [Description (
        "IP address for the given port.")]
    string LocalIPAddress ;

    [Description (
        "WWN of remote switch.")]
    string RemoteWWN;

    [Description (
        "WWN of the local FC switch.")]
    string LocalWWN;

    [Description (
        "Committed traffic rate on this FCIP channel.")]
    uint32 CommittedRate;

    [Description (
        "Flag to indicate if compression will be used.")]
    boolean Compression;

    [Description (
        "Flag to indicate if SACK will be used.")]
    boolean SelectiveACK;

    [Description (
        "Flag to indicate if path MTU discovery will be used.")]
    boolean PathMTU = false;

    [Description (
        "This indicates the Retransmit time in milliseconds.")]
    uint32 RetransmitTime;

    [Description (
        "This indicates the maximum number of retransmissions that "
        "will be attempted.")]
    uint16 MaxRetransmissions;

    [Description (
        "The Keep alive time in TCP.")]
    uint32 KeepAliveTimeout;

class CISCO_FCIPSettings : CIM_SettingData
{

    [Override("InstanceID"), Key, Description (
        "A string that identifies this ProtocolEndpoint with either "
        "a port or an interface on a device. To ensure uniqueness, "
        "the Name property should be prepended or appended with "
        "information from the Type or OtherTypeDescription "
        "properties. The method selected is described in the "
        "NameFormat property of this class."),
        MaxLen ( 256 )]
    string InstanceID;

    [Description (
        "Identifies the FCIP Tunnel on the GigE port."
```

```
        "The value ranges from 0 to 7." )]
    uint16 TunnelID;

    [Description (
         "If the value is set to 'false' this link endpoint actively"
        " tries to connect to the peer. If it is set to 'true' the link"
         "endpoint waits for the peer to connect to it. ")]
        boolean PassiveMode;

    [Description (
         "The maximum number of TCP connections allowed on this"
         "link. ")]
    uint32 NumTcpConnections;

        boolean CheckTimestamp;

         [Description(
         "The accepted time difference between the local time"
         "and the timestamp value received in the FCIP header."
         "By default this value will be EDTOV/2. EDTOV is the"
     "Error_Detect_Timeout Value used for Fibre channel Ports"
         "as the timeout value for detecting an error condition.")]
    uint32 TimeStampTolerance;

        [Description(
         "The remote TCP port to which the local FCIP entity will"
         "connect if and when it initiates a TCP connection setup"
         "for this link. ")]
        uint32 TcpRemPort;

//Wrong Description need to set it right
    [Description(
         "The remote TCP port to which the local FCIP entity will"
         "connect if and when it initiates a TCP connection setup"
         "for this link. ")]
    boolean LocalPortEnable;

    [Description(
         "If the value is set to 'true', the TCP active opener"
          "initiates FCIP special frames and the TCP passive"
          "opener responds to the FCIP special frames."
          "If it is set to 'false', the FCIP special frames are"
          "neither generated nor responded to. ")]
    boolean SpecialFrameEnable;

    [Description(
         "If the value is set to 'true', a message is"
          "sent in response to a (Fibre Channel) ELS Echo"
          "frame received from the peer. Some B Port"
          "implementations use ELS Echo request/response frames"
          "as Link Keep Alive."
          "If it is set to 'false', this response is not"
          "generated."
          "This object is valid only if the"
          "cfmFcipLinkExtLocalBPortEnable is  'true'. ")]
    boolean BPortKAEnable;

    [Description(
         "The value to be set for the TOS field in IP header"
         "for the TCP control connection."
         "The cfmFcipLinkExtCntrlQOSField,cfmFcipLinkExtDataQOSField"
         "must be set in the same SNMP set request. SET operation would"
         "fail if this object is set individually. ")]
    uint32 CntrlQOSField;
```

```
 [Description(
     "The value to be set for the TOS field in IP header"
     "for the TCP Data connection."
     "The cfmFcipLinkExtCntrlQOSField,cfmFcipLinkExtDataQOSField"
     "must be set in the same SNMP set request. SET operation would"
     "fail if this object is set individually. ")]
uint32 DataQOSField;

[Description(
   "The ifIndex of the interface on which this FCIP link was"
   "initiated. ")]

uint32 EthIfIndex;

 [Description(
     "The Write accelerator allows for enhancing SCSI write"
     "performance."
     "If 'true', the FCIP Write accelerator is enabled on this link"
     "If 'false' it is disabled.")]
boolean WriteAccelerator;

 [Description(
     "The configuration for the IP compression."
      "'none'                  - ip compression is disabled."

     " 'highCompressionRatio' - indicates better compression"
     "                          performance at the cost of lower"
      "                         throughput."

     " 'highThroughput'       - indicates better throughput at"
    "                           the cost of lower compression"
     "                          performance."

      "'auto'                  - indicates that an appropriate"
      "                                    mode wll be picked based on"
       "                        the bandwidth and data."

      " 'mode1'                 - fast compression mode for high"
                               " bandwidth WAN links with bandwidth"
                                "greater than 30 Mbps."

     " 'mode2'                 - high compression mode for"
      "                                   moderately low bandwidth WAN links,"
       "                       i.e. bandwidth between 15 and 30 Mbps."

     " 'mode3'                 - high compression mode for"
      "                          low bandwidth WAN links,"
      "                                 i.e. banwidth less than 15 Mbps."

     )]
uint32 IPComp;

[Description(
     "The Tape accelerator allows for enhancing Tape write"
     "performance."
     "If 'true', the FCIP Tape accelerator is enabled on this link"
     "If 'false' it is disabled.")]
boolean TapeAccelerator;

[Description(
          "The flow control buffer size.")]
uint32 FlowCtrlBufSize;
```

```
    [Description(
         "Indicates whether the IP Security has been turned on or"
         " off on this link.")]
    boolean IPSec;

    [Description(
      "The physical ifIndex of the interface on which this FCIP link"
      "is currently bound. ")]
    uint32 PhyIfIndex;

    [Description(
         "When Write Acceleration is operationally off for the FCIP"
        " link, the value of this object will be set to 'false'."
         "When Write Acceleration is operationally on for the FCIP"
         "link, the value of this object will be set to 'true'. ")]
    boolean WriteAccOper;

     [Description(
         "When Tape Acceleration is operationally off for the FCIP"
         " link, the value of this object will be set to 'false'."
         "When Tape Acceleration is operationally on for the FCIP"
         "link, the value of this object will be set to 'true'. ")]
    boolean TapeAccOper;

    [Description(
         "This object represents the state of the Tape Read"
         "Acceleration for an FCIP link.  Tape Read Acceleration"
         "is automatically operational when Tape Acceleration is"
         "operational (cfmFcipLinkExtTapeAccOper) and both sides"
         "of the FCIP link support Tape Read Acceleration."
         " When Tape Read Acceleration is operationally off for"
         "the FCIP link, the value of this object is 'false'."
         "When Tape Read Acceleration is operationally on for"
         "the FCIP link, the value of this object is 'true'. ")]
    boolean TapeAccReadOper;

//  [Description(
//          "When Tape Acceleration is operationally off for the FCIP
//          link, the value of this object will be set to 'false'.
//          When Tape Acceleration is operationally on for the FCIP
//          link, the value of this object will be set to 'true'. "
//  )]
};

[Provider("FCIPProvider"),
    Description ("A protocol endpoint that is dedicated to running IP.")]

class CISCO_IPProtocolEndpoint: CIM_IPProtocolEndPoint {

    [Override("SystemCreationClassName"), Key, Propagated("CIM_System.CreationClassName"),
        Description ("The scoping System's CreationClassName."),
        MaxLen ( 256 )]
    string SystemCreationClassName;

    [Override("SystemName"), Key, Propagated("CIM_System.Name"),
        Description ("The scoping System's Name."),
        MaxLen ( 256 )]
    string SystemName;

    [Override("CreationClassName"), Key, Description (
        "CreationClassName indicates the name of the class or the "
        "subclass used in the creation of an instance. When used "
        "with the other key properties of this class, this property "
        "allows all instances of this class and its subclasses to be "
```

```
        "uniquely identified."),
        MaxLen ( 256 )]
    string CreationClassName;

    [Override("Name"), Key, Description (
        "A string that identifies this ProtocolEndpoint with either "
        "a port or an interface on a device. To ensure uniqueness, "
        "the Name property should be prepended or appended with "
        "information from the Type or OtherTypeDescription "
        "properties. The method selected is described in the "
        "NameFormat property of this class."),
        MaxLen ( 256 )]
    string Name;

    [Override("NameFormat"), Description (
        "NameFormat contains the naming heuristic that is chosen to "
        "ensure that the value of the Name property is unique. For "
        "example, one might choose to prepend the name of the port "
        "or interface with the Type of ProtocolEndpoint that this "
        "instance is (e.g., IPv4) followed by an underscore."),
        MaxLen ( 256 )]
    string NameFormat;

    [Override("IPv4Address"), Description (
        "The IPv4 address that this ProtocolEndpoint represents.")]
    string IPv4Address;

    [Override("SubnetMask"), Description (
        "The mask for the IPv4 address of this ProtocolEndpoint, if "
        "one is defined.")]
    string SubnetMask;

    [Override("ProtocolIFType"), Description (
        "ProtocolIFType's enumeration is limited to IP-related and "
        "reserved values for this subclass of ProtocolEndpoint."),
        ValueMap { "1", "222..4095", "4096", "4097", "4098",
            "4116..32767", "32768.." },
        Values { "Other", "IANA Reserved", "IPv4", "IPv6", "IPv4/v6",
            "DMTF Reserved", "Vendor Reserved" }]
    uint16 ProtocolIFType = 4096;

    [Override("AddressOrigin"), Experimental, Description (
        "AddressOrigin identifies the method by which the IP "
        "Address, Subnet Mask, and Gateway were assigned to the "
        "IPProtocolEndpoint."),
        ValueMap { "0", "1", "2", "3", "4", "5", "6..32767", "32768.." },
        Values { "Unknown", "Other", "Not Applicable", "Static", "DHCP",
            "BOOTP", "DMTF Reserved", "Vendor Reserved" }]
    uint16 AddressOrigin = 3;

};
```

Cisco MOF Files for Cisco SAN-OS Release 3.x and NX-OS Release 4.x

This section includes the MOF files supported by Cisco SAN-OS Release 3.x and NX-OS Release 4.x. It includes the following topics:

Cisco Fabric MOF

The Cisco Fabric MOF for Cisco SAN-OS Release 3.x and NX-OS Release 4.x provides extensions to the Fabric profile to manage VSANs, PortChannels, and other Cisco-specific entities within the fabric. See the "FDMI Subprofile" section on page 2-7.

```
[Version ("1.0.0"), Description (
    "cisco fabric and switch profile classes")]
class CISCO_ActiveConnection : CIM_ActiveConnection
{};

class CISCO_AdminDomain : CIM_AdminDomain
{};

   [Version ( "2.7.1"), Description (
       "Capabilities and management of a Fibre Channel Port Device.") ]
class CISCO_FCPort : CIM_FCPort {

      [Override ( "PortType"), Description (
          "The specific mode currently enabled for the Port.  The "
          "values: \"N\" = Node Port, \"NL\" = Node Port supporting FC "
          "arbitrated loop, \"E\" = Expansion Port connecting fabric "
          "elements (for example, FC switches), \"F\" = Fabric "
          "(element) Port, \"FL\" = Fabric (element) Port supporting "
          "FC arbitrated loop, \"B\" = Bridge and \"G\" = Generic "
          "Port.  PortTypes are defined in the ANSI X3 standards.  "
          "When set to 1 (\"Other\"), the related property "
          "OtherPortType contains a string description of the port's "
          "type."),
       ValueMap { "0", "1", "10", "11", "12", "13", "14", "15", "16",
          "17", "18", "16004", "16010", "16011", "16012", "16000..65535"},
       Values { "Unknown", "Other", "N", "NL", "F/NL", "Nx", "E", "F",
          "FL", "B", "G", "PortChannel", "FCIP", "ISCSI-F", "ISCSI-N", "Vendor Reserved"}
]
   uint16 PortType;
uint16 PortAvailability = 2;
};

class CISCO_Vsan : CIM_AdminDomain {
};
class CISCO_Component : CIM_Component
{};

class CISCO_ComputerSystem : CIM_ComputerSystem
{};

class CISCO_ConnectivityCollection : CIM_ConnectivityCollection
{};

class CISCO_ConnectivityMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_ContainedDomain : CIM_ContainedDomain
{};

class CISCO_DeviceSAPImplementation : CIM_DeviceSAPImplementation
{};
```

```
class CISCO_FCPortStatistics : CIM_FCPortStatistics
{};

class CISCO_HostedAccessPoint : CIM_HostedAccessPoint
{};

class CISCO_HostedCollection : CIM_HostedCollection
{};

class CISCO_ProtocolEndPoint : CIM_ProtocolEndPoint
{};

class CISCO_PhysicalPackage : CIM_PhysicalPackage
{};

class CISCO_PhysicalElement : CIM_PhysicalElement
{};

class CISCO_Product : CIM_Product
{};

class CISCO_Realizes : CIM_Realizes
{};

class CISCO_SystemDevice : CIM_SystemDevice
{};

class CISCO_ComputerSystemPackage : CIM_ComputerSystemPackage
{};

class CISCO_ProductPhysicalComponent : CIM_ProductPhysicalComponent
{};

class CISCO_ElementStatisticalData : CIM_ElementStatisticalData
{};

class CISCO_LogicalPortGroup : CIM_LogicalPortGroup
{};

class CISCO_LogicalModule : CIM_LogicalModule
{};

class CISCO_ModulePort : CIM_ModulePort
{};

class CISCO_EthernetPort : CIM_EthernetPort
{};

class CISCO_HostedDependency : CIM_HostedDependency
{};

class CISCO_LogicalIdentity : CIM_LogicalIdentity
{};

class CISCO_PhysicalComputerSystem : CISCO_ComputerSystem
{};

class CISCO_LogicalComputerSystem : CISCO_ComputerSystem
{};

class CISCO_FCNodeMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_FabricHostedService : CIM_HostedService
```

```
{};

class CISCO_ObjectManagerHost : CIM_System
{};

class CISCO_FCPortCapabilities : CIM_FCPortCapabilities
{};

class CISCO_FCSwitchCapabilities : CIM_FCSwitchCapabilities
{};

class CISCO_FCPortSettings : CIM_FCPortSettings
{};

class CISCO_FCSwitchSettings : CIM_FCSwitchSettings
{};

class CISCO_ElementCapabilities : CIM_ElementCapabilities
{};

class CISCO_ElementSettingDataSys : CIM_ElementSettingData
{};

class CISCO_SoftwareIdentity : CIM_SoftwareIdentity
{};

class CISCO_ElementSoftwareIdentity : CIM_ElementSoftwareIdentity
{};

class CISCO_SAPAvailableForElement : CIM_SAPAvailableForElement
{};

class CISCO_RemoteServiceAccessPoint : CIM_RemoteServiceAccessPoint
{};
```

Cisco Zone MOF

The Cisco Zone MOF for Cisco SAN-OS Release 3.x and NX-OS Release 4.x provides extensions to the zoning subprofiles. See the "Zoning Subprofile Extensions" section on page 2-18.

```
[Version ("1.0.0"), Description (
    "cisco zoneset class")]
class CISCO_ZoneSet : CIM_ZoneSet
{};

class CISCO_Zone : CIM_Zone
{};

class CISCO_ZoneAlias : CIM_NamedAddressCollection
{};

class CISCO_ZoneMemberSettingData : CIM_ZoneMembershipSettingData{

[Override ( "ConnectivityMemberType" ), Description (
          "ConnectivityMemberType specifies the type of identification "
          "used in the ConnectivityMemberID field. For Fibre Channel, "
          "several of the enumerated values require additional "
          "explanation: \n"
          "* A ConnectivityMemberType equal to 2 (Permanent Address) "
          "indicates that an NxPort WWN (pWWN)value should be specified in "
          "the related ConnectivityMemberID property. \n"
```

```
                "* A ConnectivityMemberType of 3 (FCID) indicates "
                "that an NxPort Address ID(FCID) value should be specified in the "
                "related ConnectivityMemberID property. \n"
                "* A ConnectivityMemberType of 4 (Switch Port ID) indicates "
                "that a Domain or Port Number(DomainID) value should be specified in "
                "the related ConnectivityMemberID property.(eg. 06:40) \n"
                "* A ConnectivityMemberType of 5 (fcalias) "
                "indicates that alias name which denotes a port ID or WWN shoud be "
                "specified in the related ConnectivityMemberID property."
                "* A ConnectivityMemberType of 6 (Interface) "
                "indicates that a interface of local switch. The fc interface should"
                "be specified in the related ConnectivityMemberID property(eg. fc1/9)"
                "* A ConnectivityMemberType of 7 (fWWN) "
                "indicates that Fabric port WWN.The WWN of the fabric "
                "port value should be specified in the "
                "related ConnectivityMemberID property."
                "* A ConnectivityMemberType of 8 (Network Address IpV4) "
                "indicates that IPv4 address of an attached device in 32 bits"
                "in dotted decimal format should be specified in the "
                "related ConnectivityMemberID property."
                "* A ConnectivityMemberType of 9 (Network Address IpV6) "
                "indicates that IPv6 address—The IPv6 address of an attached device "
                "in 128 bits in colon(:)-separated hexadecimal format should be specified"
                " in related ConnectivityMemberID property."
                "* A ConnectivityMemberType of 10 (Interface with Remote SWWN) "
                "indicates that a interface of remote switch. The fc interface should"
                "be specified along with Switch WWN in the related ConnectivityMemberID"
                "property(eg. fc1/9:20000005300084DF)"
                "* A ConnectivityMemberType of 11 (Interface with DomainID) "
                "indicates that a interface of local switch. The fc interface should"
                "be specified along with the Domain Id in the related "
                "ConnectivityMemberID property(eg.fc1/9:25)" )]
                "* A ConnectivityMemberType of 12 (Symbolic-node name) "
                "indicates that a symbolic-node name"
                "should be specified in the "
                "related ConnectivityMemberID property."

   uint16 ConnectivityMemberType;
};

class CISCO_ZoneService : CIM_ZoneService
{};

class CISCO_SystemSpecificCollection : CIM_SystemSpecificCollection
{};

class CISCO_ZoneMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_ElementSettingData : CIM_ElementSettingData
{};

class CISCO_HostedService : CIM_HostedService
{};

class CISCO_ZoneHostedCollection : CIM_HostedCollection
{};

class CISCO_ZoneCapabilities : CIM_ZoneCapabilities
{};
```

Cisco FDMI MOF

The Cisco FDMI MOF for Cisco SAN-OS Release 3.x and NX-OS Release 4.x provides extensions to the Fabric profile to manage VSANs, PortChannels, and other Cisco-specific entities within the fabric. See the "FDMI Subprofile" section on page 2-7.

```
[Provider("FDMI_Provider"),Description (
        "This class represents FDMI enabled physical HBA card attached "
        "to a switch" )]
class CISCO_PhysicalHBA: CIM_PhysicalPackage {

    [Override("Tag"), Key, MaxLen (256), Description (
        "A unique physical identifier that serves as the key for "
        "the HBA.  The HBA serial number could be used as a tag.\n" )]
    string Tag;

    [Override("CreationClassName"), Key, MaxLen (256), Description (
        "CreationClassName indicates the name of the class or the "
        "subclass used in the creation of an instance.")]
    string CreationClassName= "CISCO_PhysicalHBA";

    [Override("Manufacturer"), MaxLen (256), Description (
        "The name of the organization responsible for "
        "manufacturing the HBA.")]
    string Manufacturer;

    [Override("Model"), MaxLen (64), Description (
        "The name by which the HBA is generally known.")]
    string Model;

    [Description (
        "The detailed description of the model of the HBA. The "
        "value might provide a more detailed identification of the "
        "HBA than the Model property does."),
        MaxLen (256)]
    string ModelDescription;

    [Override("SerialNumber"), MaxLen (64), Description (
        "A manufacturer-allocated number used to identify the HBA. "
        "This value SHOULD match a serial number engraved or "
        "printed in the HBA.")]
    string SerialNumber;

    [Override("Version"), MaxLen (64), Description (
        "A string indicating the version of the HBA card.")]
    string Version;
} ;

/// CISCO_HBAProduct
[Provider("FDMIProvider"),
        Description ("This class represents product information of FDMI enabled physical
            HBA card attached to a switch."
)]

class CISCO_HBAProduct: CIM_Product {

    [Override("Name"),Key, Description (
        "Commonly used Product name."),
        MaxLen ( 256 )]
    string Name;

    [Override("IdentifyingNumber"),Key, Description (
        "A manufacturer-allocated number used to identify the HBA. "
```

```
        "This value SHOULD match a serial number engraved or "
        "printed in the HBA."),
        MaxLen ( 64 )]
    string IdentifyingNumber;

    [Override("Vendor"),Key, Description (
        "The name of the Product's supplier, or entity selling the "
        "Product (the manufacturer, reseller, OEM, etc.). "
        "Corresponds to the Vendor property in the Product object in "
        "the DMTF Solution Exchange Standard."),
        MaxLen ( 256 )
        ]
    string Vendor;

    [Override("Version"),Key, Description (
        "A string indicating the version of the HBA card."),
        MaxLen ( 64 )]
    string Version;

    [Override("ElementName"), Description(
        "The detailed description of the model of the HBA. The "
        "value might provide a more detailed identification of the "
        "HBA than the Model property does ")]
    string ElementName;
};

// CISCO_Platform
[Provider("FDMIProvider"),
    Description (
        "CISCO_Platform represents a fabric-connected entity, "
        "containing one or more Node objects, that has registered "
        "with a fabric's Management Server service.")]

class CISCO_Platform: CIM_ComputerSystem {

    [Override ("CreationClassName"), Key, MaxLen (256),
     Description (
        "CreationClassName indicates the name of the class or the "
        "subclass used in the creation of an instance.")]
    string CreationClassName= "CISCO_Platform";

    [Override ("Name"), Key, MaxLen (256), Description (
        "The inherited Name serves as key of the platform in an "
        "enterprise environment.  This value has the following "
        "format:\n"
        "\"WWN\":\"Platform Name\".")]
    string Name;

    [Override ("ElementName"), Required, Description (
        "A user-friendly name for the object. This property allows "
        "each instance to define a user-friendly name IN ADDITION TO "
        "its key properties/identity data, and description "
        "information.")]
    string ElementName;

    [Override ( "NameFormat" ),Required, Description (
        "The ComputerSystem object and its derivatives are Top Level "
        "Objects of CIM. They provide the scope for numerous "
        "components. Having unique System keys is required. The "
        "NameFormat property identifies how the ComputerSystem Name "
        "is generated. The NameFormat ValueMap qualifier defines the "
        "various mechanisms for assigning the name. Note that "
        "another name can be assigned and used for the "
        "ComputerSystem that better suit a business, using the "
```

```
        "inherited ElementName property."),
        ValueMap { "Other", "IP", "Dial", "HID", "NWA", "HWA", "X25",
            "ISDN", "IPX", "DCC", "ICD", "E.164", "SNA", "OID/OSI",
            "WWN", "NAA" }]
    string NameFormat = "Other";

    [Write, Override ("Dedicated"), Description(
        "Platform type.  Although this is represented as an array, "
        "only one type is specified at any given time (array size is "
        "always 1).  When writing this property, users should "
        "specify only a single type in an array size of exactly 1.  "
        "Specifying more or less than 1 type results in an exception "
        "with an invalid argument error code."),
        Values{"Unknown", "Others", "Gateway", "dummy3", "dummy4",
            "Converter", "HBA", "Swproxy", "StorageDev", "Host",
            "Storsubsys", "Module", "Driver", "StorAccess"},
        ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
            "11", "12", "13"}]
    uint16 Dedicated[];

    [Override ("OtherIdentifyingInfo"), Description(
        "Platform name: for example, host name.")]
    string OtherIdentifyingInfo[];

    [Write, Description(
        "The set of management IP Addresses used to access this "
        "platform.")]
    string MgmtAddressList[];
};
// CISCO_PortController
[Description("CISCO_PortController represents the port controller of an FDMI enabled
HBA.")]
class CISCO_PortController: CIM_PortController {

    [Override("SystemCreationClassName"), Key, MaxLen (256), Description (
        "The scoping system's creation class name.  The "
        "scoping system is the CISCO_Platform or "
        "CISCO_Fabric of which this device is part.")]
    string SystemCreationClassName;

    [Override("SystemName"), Key, MaxLen (256), Description (
        "The scoping system's Name property.  The value "
        "is equivalent to the platform name if the scoping system is an "
        "instance of CISCO_Platform or the Proxy Switch WWN if the "
        "scoping system is an instance of CISCO_Fabric.")]
    string SystemName;

    [Override("CreationClassName"), Key, MaxLen (256),
     Description (
        "CreationClassName indicates the name of the CISCO_PortController "
        "class that, when used with the other key properties of this "
        "class, uniquely identifies an instance of the "
        "CISCO_PortController class.")]
    string CreationClassName= "CISCO_PortController";

    [Override("DeviceID"), Key, MaxLen (64), Description (
        "This is the Serial Number of the HBA")]
    string DeviceID;

    [Override("ControllerType"),Required,Description (
        "The type or model of the port controller. Specific values "
        "will be enumerated in a later release of this schema. When "
        "set to 1 (\"Other\"), the related property "
        "OtherControllerType contains a string description of the "
```

```
        "controller's type."),
        ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8" },
        Values { "Unknown", "Other", "Ethernet", "IB", "FC", "FDDI",
            "ATM", "Token Ring", "Frame Relay" }]
    uint16 ControllerType = 4;
};


class CISCO_HBASoftwareIdentity : CIM_SoftwareIdentity
{};

class CISCO_ElementSoftwareIdentity : CIM_ElementSoftwareIdentity
{};

// Associations

// CISCO_PortControllerRealizes

[Association,
    Provider("FDMIProvider"),
    Description (
        "CISCO_PortControllerRealizes is the association that defines "
        "the mapping between devices and the physical elements "
        "that implement them.")]
class CISCO_PortControllerRealizes: CIM_Realizes {

    [Override ("Antecedent"), Description (
        "The physical HBA that implements the Device.")]
    CISCO_PhysicalHBA REF Antecedent;

    [Override ("Dependent"), Description (
        "The Device.")]
    CISCO_PortController REF Dependent;
};
// CISCO_PlatformPackage

[Association,
    Description (
        "This association denotes one or more physical HBAs that "
        "realize a Platform.")]
class CISCO_PlatformPackage: CIM_ComputerSystemPackage {

    [Override ("Antecedent"), Description (
        "The physical HBA that realizes a Platform.")]
    CISCO_PhysicalHBA REF Antecedent;

    [Override ("Dependent"), Description (
        "The Platform.")]
    CISCO_Platform REF Dependent;
};
// CISCO_PortControllerSoftwareIdentity

[Association,
    Description (
        "The PortControllerSoftwareIdentity relationship identifies any "
        "software that is associated with the device and this association "
        "can return multiple instances.")]
class CISCO_PortControllerSoftwareIdentity: CIM_ElementSoftwareIdentity {

    [Override ("Antecedent"), Description (
        "The SoftwareIdentity on the device.")]
    CISCO_HBASoftwareIdentity REF Antecedent;

    [Override ("Dependent"), Description (
```

```
        "The logical device that requires or uses the software.")]
    CISCO_PortController REF Dependent;
};
// CISCO_HBASoftwareInstalledOnPlatform

[Association,
    Description (
        "The SofwareInstalledOnPlatform relationship allows the "
        "identification of the platform on which HBA driver "
        "is installed and this association can return multiple instances.")]
class CISCO_HBASoftwareInstalledOnPlatform: CIM_InstalledSoftwareIdentity {

    [Key, Override("System"), Max (1), Description (
        "Reference to the platform hosting a particular "
        "SoftwareIdentity.")]
    CISCO_Platform REF System;

    [Key, Override("InstalledSoftware"), Description (
        "Reference to the driver that is installed on the "
        "platform.")]
    CISCO_HBASoftwareIdentity REF InstalledSoftware;
};
// CISCO_NodeFCPortControlledByPortController

[Association,
    Description (
        "This association represents the relationship between a "
        "device and ports.")]
class CISCO_NodeFCPortControlledByPortController: CIM_ControlledBy {
    [Override ("Antecedent"), Description (
        "The device that controls the port.")]
    CISCO_PortController REF Antecedent;

    [Override ("Dependent"), Description (
        "The port being controlled.")]
    CISCO_FCPort REF Dependent;

    [Override("DeviceNumber"), MaxLen(255), Description (
        "Address of associated port in context of the antecedent "
        "device. This may be a comma-separated list in case there "
        "are multiple addresses.")]
    string DeviceNumber;
};
// CISCO_ProductPhysicalHBA

[Association,
    Description (
        "The HBA is shipped to the customer by a third party "
        "(OEM/reseller) to the customer.  This class associates "
        "the HBA with the product.")]
class CISCO_ProductPhysicalHBA: CIM_ProductPhysicalComponent {

    [Override ("GroupComponent"), Description (
        "The product.")]
    CISCO_HBAProduct REF GroupComponent;

    [Override ("PartComponent"), Description (
        "The HBA that is shipped as a product.")]
    CISCO_PhysicalHBA REF PartComponent;
};

CISCO_PlatformInFabric

[Association, Aggregation,
```

```
    Description (
        "CISCO_PlatformInFabric is a generic association used to "
        "establish membership relationships between the fabric and "
        "platforms connected to the fabric.")]
class CISCO_PlatformInFabric: CIM_Component {

    [Override("GroupComponent"), Aggregate, Key, Description (
        "The fabric that has connected platforms.")]
    CISCO_VSAN REF GroupComponent;

    [Override("PartComponent"), Key, Description (
        "The platforms connected to this fabric.")]
    CISCO_Platform REF PartComponent;
};
// CISCO_NodePortInPlatform

[Association, Aggregation,
    Description (
        "CISCO_NodePortInPlatform is a generic association used to "
        "establish membership relationships between a platform and the "
        "node ports contained within that platform.")]
class CISCO_NodePortInPlatform: CIM_SystemDevice {

    [Override("GroupComponent"), Description (
        "The platform that has contained node ports.")]
    CISCO_Platform REF GroupComponent;

    [Override("PartComponent"), Description (
        "The node ports contained in this platform.")]
    CISCO_FCPort REF PartComponent;
};

// CISCO_NodeInPlatform

[Association,
        Description (
                "CISCO_NodeInPlatform defines a SystemSpecificCollection "
                "in the context of a scoping system.  Only nodes that are "
                "present in the platform database and also present in the "
                "Name Server are considered.")]

class CISCO_NodeInPlatform: CIM_HostedCollection {

    [Override ("Antecedent"), Description (
        "A platform hosts a collection of nodes.")]
    CISCO_Platform REF Antecedent;

    [Override ("Dependent"), Description (
        "The nodes that are hosted on a platform.")]
    CISCO_LogicalPortGroup REF Dependent;
};

// CISCO_PortControllerInPlatform

[Association,
    Description (
        "CISCO_PortControllerInPlatform defines a SystemSpecificCollection "
        "in the context of a scoping system.  The node registered "
        "in the platform database must also be registered in the "
        "Name Server.")]
class CISCO_PortControllerInPlatform: CIM_SystemDevice {

    [Override ("GroupComponent"), Description (
        "A platform hosts a collection of devices.")]
```

```
    CISCO_Platform REF GroupComponent;

    [Override ("PartComponent"), Description (
        "The devices hosted on a platform.")]
    CISCO_PortController REF PartComponent;
};

// CISCO_PortControllerInFabric

[Association,
    Provider("FDMIProvider"),
    Description (
        "CISCO_PortControllerInFabric defines a SystemSpecificCollection "
        "in the context of a scoping system.")]
class CISCO_PortControllerInFabric: CIM_SystemDevice {

    [Override ("GroupComponent"), Description (
        "A platform hosts a collection of devices.")]
    CISCO_VSAN REF GroupComponent;

    [Override ("PartComponent"), Description (
        "The devices hosted on a platform.")]
    CISCO_PortController REF PartComponent;
};
```

Cisco MOF Files for Cisco SAN-OS Release 2.x

This section includes the MOF files supported by Cisco SAN-OS Release 2.x. It includes the following topics:

Cisco Fabric MOF

The Cisco Fabric MOF for Cisco SAN-OS Release 2.x provides extensions to the Fabric profile to manage VSANs, PortChannels, and other Cisco-specific entities within the fabric. See the "Cisco MDS Extensions to the Switch and Fabric Profiles" section on page 2-9.

```
[Version ("1.0.0"), Description (
    "cisco fabric and switch profile classes")]
class CISCO_ActiveConnection : CIM_ActiveConnection
{};

class CISCO_AdminDomain : CIM_AdminDomain
{};

   [Version ( "2.7.1"), Description (
       "Capabilities and management of a Fibre Channel Port Device.") ]
class CISCO_FCPort : CIM_FCPort {

      [Override ( "PortType"), Description (
          "The specific mode currently enabled for the Port.  The "
          "values: \"N\" = Node Port, \"NL\" = Node Port supporting FC "
          "arbitrated loop, \"E\" = Expansion Port connecting fabric "
          "elements (for example, FC switches), \"F\" = Fabric "
          "(element) Port, \"FL\" = Fabric (element) Port supporting "
          "FC arbitrated loop, \"B\" = Bridge and \"G\" = Generic "
```

```
            "Port.  PortTypes are defined in the ANSI X3 standards.  "
            "When set to 1 (\"Other\"), the related property "
            "OtherPortType contains a string description of the port's "
            "type."),
         ValueMap { "0", "1", "10", "11", "12", "13", "14", "15", "16",
            "17", "18", "16004", "16010", "16011", "16012", "16000..65535"},
         Values { "Unknown", "Other", "N", "NL", "F/NL", "Nx", "E", "F",
            "FL", "B", "G", "PortChannel", "FCIP", "ISCSI-F", "ISCSI-N", "Vendor Reserved"}
]
   uint16 PortType;
};

class CISCO_Vsan : CIM_AdminDomain {
      [Override ( "NameFormat"), Description (
            "The NameFormat property identifies how the Name of the "
            "AdminDomain is generated, using the heuristic specified in "
            "the CIM V2 System Model spec.  It assumes that the "
            "documented rules are traversed in order, to determine and "
            "assign a Name.  The NameFormat Values list defines the "
            "precedence order for assigning the Name of the "
            "AdminDomain.  \n"
            "\n"
            "\"FC\" has been deprecated and replaced by \"WWN\" to be "
            "consistent with the other ValueMaps."),
         ValueMap { "Other", "AS", "NAP", "NOC", "POP", "RNP", "IP",
            "IPX", "SNA", "Dial", "WAN", "LAN", "ISDN", "Frame Relay",
            "ATM", "E.164", "IB", "FC", "Policy Repository", "WWN", "ID with WWN"},
         Values { "Other", "Autonomous System",
            "Network Access Provider", "Network Operations Center",
            "Point of Presence", "Regional Network Provider", "IP",
            "IPX", "SNA", "Dial", "WAN", "LAN", "ISDN", "Frame Relay",
            "ATM", "E.164", "Infiniband", "Fibre Channel",
            "Policy Repository", "Fibre Channel Worldwide Name", "Virtual SAN ID and
Worldwide Name"},
         ModelCorrespondence { "CIM_AdminDomain.Name"} ]
   string NameFormat;
};

class CISCO_Component : CIM_Component
{};

class CISCO_ComputerSystem : CIM_ComputerSystem
{};

class CISCO_ConnectivityCollection : CIM_ConnectivityCollection
{};

class CISCO_ConnectivityMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_ContainedDomain : CIM_ContainedDomain
{};

class CISCO_DeviceSAPImplementation : CIM_DeviceSAPImplementation
{};

class CISCO_FCPortStatistics : CIM_FCPortStatistics
{};

class CISCO_HostedAccessPoint : CIM_HostedAccessPoint
{};

class CISCO_HostedCollection : CIM_HostedCollection
{};
```

```
class CISCO_ProtocolEndPoint : CIM_ProtocolEndPoint
{};

class CISCO_PhysicalPackage : CIM_PhysicalPackage
{};

class CISCO_PhysicalElement : CIM_PhysicalElement
{};

class CISCO_Product : CIM_Product
{};

class CISCO_Realizes : CIM_Realizes
{};

class CISCO_SystemDevice : CIM_SystemDevice
{};

class CISCO_ComputerSystemPackage : CIM_ComputerSystemPackage
{};
class CISCO_ProductPhysicalComponent : CIM_ProductPhysicalComponent
{};
class CISCO_ElementStatisticalData : CIM_ElementStatisticalData
{};
class CISCO_LogicalPortGroup : CIM_LogicalPortGroup
{};


class CISCO_LogicalModule : CIM_LogicalModule
{};

class CISCO_ModulePort : CIM_ModulePort
{};

class CISCO_EthernetPort : CIM_EthernetPort
{};

class CISCO_HostedDependency : CIM_HostedDependency
{};

class CISCO_LogicalIdentity : CIM_LogicalIdentity
{};

class CISCO_PhysicalComputerSystem : CISCO_ComputerSystem
{};

class CISCO_LogicalComputerSystem : CISCO_ComputerSystem
{};

class CISCO_FCNodeMemberOfCollection : CIM_MemberOfCollection
{};
```

Cisco Zone MOF

The Cisco Zone MOF for Cisco SAN-OS Release 2.x provides extensions to the zoning subprofiles. See the "Cisco MDS Extensions to the Switch and Fabric Profiles" section on page 2-9.

```
[Version ("1.0.0"), Description (
    "cisco zoneset class")]
class CISCO_ZoneSet : CIM_ZoneSet
```

```
{
};

class CISCO_Zone : CIM_Zone
{};

class CISCO_ZoneAlias : CIM_NamedAddressCollection
{};

class CISCO_ZoneMemberSettingData : CIM_ZoneMembershipSettingData
{};

class CISCO_ZoneService : CIM_ZoneService
{};

class CISCO_SystemSpecificCollection : CIM_SystemSpecificCollection
{};

class CISCO_ZoneMemberOfCollection : CIM_MemberOfCollection
{};

class CISCO_ElementSettingData : CIM_ElementSettingData
{};

class CISCO_HostedService : CIM_HostedService
{};
```

Cisco Indications MOF

The Cisco Indications MOF provides extensions to the SMI-S standard indications to provide indications of link state changes. This MOF supports Cisco SAN-OS Release 2.0(1a) or later. See the "FDMI Subprofile Extensions" section on page 2-18.

```
[Version ("2.2.0")]
class CISCO_LinkStateChange : CISCO_AlertIndication
{
      [Description (
        "The desired state of the interface. The testing (3) state"
        "indicates that no operational packets can be passed. When a"
        "managed system initializes, all interfaces start with"
        "ifAdminStatus in the down(2) state. As a result of either"
        "explicit management action or per configuration information"
        "retained by the managed system, ifAdminStatus is then"
        "changed to either the up(1) or testing(3) states (or remains"
        "in the down(2) state)."),
      ValueMap {"1", "2", "3"},
      Values { "up", "down", "testing"}]
        uint32 ifAdminStatus;

      [Description (
        "The current operational state of the interface. "),
      ValueMap {"1", "2", "3", "4", "5", "6", "7"},
      Values { "up", "down", "testing", "unknown", "dormant",
               "notPresent", "lowerLayerDown"}]
        uint32 ifOperStatus;
        uint32 ifIndex;
};

class CISCO_LinkUp : CISCO_LinkStateChange
{};
```

```
class CISCO_LinkDown : CISCO_LinkStateChange
{};

class CISCO_MediaFRU : CISCO_AlertIndication
{
   uint32 PhysicalIndex;
   string PhysicalDescr;
   uint32 PhysicalVendorType_len;
   uint32 PhysicalContainedIn;
   [
    Description ("Entity Physical Class Type "),
    ValueMap {"1", "2", "3","4" , "5", "6", "7", "8", "9","10", "11" } ,
    Values {"ENT_OTHER","UNKNOWN_ENTITY", "CHASSIS", "BACKPLANE","CONTAINER",
"POWERSUPPLY", "FAN", "SENSOR", "MODULE", "PORT", "STACK"}
   ]
   uint32 PhysicalClass;

   uint32 PhysicalParRelPos;
   string PhysicalName;
   string PhysicalHardwareRev;
   string PhysicalFirmwareRev;
   string PhysicalSoftwareRev;
   string PhysicalSerialNum;
   string PhysicalMfgName;
   string PhysicalModelName;
   string PhysicalAlias;
   string PhysicalAssetID;
   boolean PhysicalIsFRU;
   boolean Valid;

   [
    Description ( "Module Admin Status Status"),
        ValueMap {"1", "2", "3","4"},
        Values {"CEFC_PHYS_STATUS_OTHER ","CEFC_PHYS_STATUS_SUPPORTED",
"CEFC_PHYS_STATUS_UNSUPPORTED", "CEFC_PHYS_STATUS_INCOMPATIBLE"}
   ]
   uint16   PhysicalStatus;

   string PhySecondSerialNum;
   string PhyProductNumber;
   string PhyPartRevision;
   string PhyMfgDate;
   string PhysicalCLEICode;
   uint16 PhySramSize;
   string PhysicalNameofSlot;

};

class CISCO_MediaFRUInserted : CISCO_MediaFRU
{};

class CISCO_MediaFRURemoved : CISCO_MediaFRU
{};

class CISCO_MediaFRUChanged: CISCO_AlertIndication
{
  uint32  PhysicalIndex;
    [Description (
        "Module Operational Status"),
     ValueMap {"1", "2","4","5","6","7","8","9","10","11","12",
           "13","14","15","16","17","18","19","20","21"},
     Values {
"MOD_OPER_UNKNOWN","MOD_OPER_OK","MOD_OPER_DISABLED","MOD_OPER_OKBUTDIAGFAILED",
          "MOD_OPER_BOOT","MOD_OPER_SELFTEST", " MOD_OPER_FAILED", "MOD_OPER_MISSING",
```

```
           "MOD_OPER_MISMATCHWITHPARENT", "MOD_OPER_MISMATCHCONFIG",
"MOD_OPER_DIAGFAILED",
           "MOD_OPER_DORMANT" , " MOD_OPER_OUTOFSERVICEADMIN",
"MOD_OPER_OUTOFSERVICEENVTEMP",
           "MOD_OPER_POWEREDDOWN", "MOD_OPER_POWEREDUP", "  MOD_OPER_POWERDENIED",
           "MOD_OPER_POWERCYCLED", "MD_OPER_OKBUTPOWEROVERWARNING","
MOD_OPER_OKBUTPOWEROVERCRITICAL",
           "MOD_OPER_SYNCINPROGRESS" }
     ]
   uint16 ModuleOperStatus;

        [Description (
                "Module Admin Status Status"),
         ValueMap {"1", "2", "3","4"},
         Values {"Admin Enabled","Admin Disabled", "Admin Reset", "Admin Out of Service"}
         ]
   uint16 ModuleAdminStatus;
        [Description (
                "Module Admin Status Status"),
         ValueMap {"1", "2", "3","4","5"},
         Values {"UNKNOWN_RESET ","POWERUP", "PARITYERROR",
"CLEARCONFIGRESET","MANUALRESET"}
         ]
  uint16 ModuleResetReason;
  string ModuleResetReasonDescription;
  uint32 numPorts;
  uint32 boot_mode;
  uint8  isValid;
  uint8  mod_state;
  uint8  mod_type;
  uint8  pad[2];
  uint32 mod_no;
  uint32 ModuleUpTime;
  uint32 numFcPorts;
};

class CISCO_VSANChanged: CISCO_AlertIndication
{};

class CISCO_ZoneSetAlert: CISCO_AlertIndication
{
    string ZoneSetName;
    uint32 VsanId;
};
class CISCO_ZoneAlert : CISCO_AlertIndication
{
    uint32 VsanId;
};

class CISCO_EnvironmentalAlert: CISCO_AlertIndication
{
    string EnvAlertDescription;
        uint32  PhysicalIndex;
    uint32  OperationalStatus;
};

class CISCO_FanAlert: CISCO_EnvironmentalAlert
{};

class CISCO_PowerAlert: CISCO_EnvironmentalAlert
{
      uint32  FRUPowerAdminStatus;
      uint32  FRUCurrent;
};
```

```
class CISCO_TempAlert: CISCO_EnvironmentalAlert
{
       uint32 SensorValue;
       uint32 SensorThresholdValue;
       uint32 SensorThresholdIndex;
};

class CISCO_NameServerDatabaseChanged: CISCO_AlertIndication
{
};
class CISCO_SecurityAlert : CISCO_AlertIndication
{};
class CISCO_UserAddedOnSwitch: CISCO_SecurityAlert
{};
class CISCO_UserRemovedOnSwitch: CISCO_SecurityAlert
{};
class CISCO_UserModifiedOnSwitch: CISCO_SecurityAlert
{};
class CISCO_UserLoginFailed: CISCO_SecurityAlert
{};
```

INDEX

V

W

X

Z