SMI-S and Web Services Programming Guide, Cisco DCNM for SAN, Release 11(x)

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.

New and Changed Information

Table 1 summarizes the new and changed features for the *SMI-S and Web Services Programming Guide, Cisco DCNM for SAN, Release 11(x)* and indicates where they are documented. The table includes a brief description of each new feature and the release in which the change occurred.

Table 1 New and Changed Features

Feature	Description	Changed in Release	Where Documented
Command Line Storage Dumps Tool	Added information on getting dumps of storage arrays using command line.	7.1(1)	Appendix B, "Command Line Storage Dumps Tool"
Sample SMI-S Java Client	Added information on the SMI-S java client and their services.	6.1(1)	Chapter 5, "Sample SMI-S Java Client."
SMI-S Notifications	Added information on the SMI-S notifications.	6.1(1)	Chapter 4, "SMI-S Notifications."
Discovery Automation Tool	Added information on how to discover DCNM-SAN, DCNM-LAN, and vCenter using the tool.	6.1.(1)	Chapter 8, "Discovery Automation Tool."
All	The *Cisco MDS 9000 Family SMI-S Programming Reference* and the *Cisco Fabric Manager Web Services Programming Guide* were combined to create this new guide.	5.2(1)	SMI-S and Web Services Programming guide, Cisco DCNM for SAN

For a complete list of Cisco DCNM documentation, see the "Related Documentation" in the Preface.

Preface

This preface describes the audience, organization, and conventions of the *SMI-S and Web Services Programming Guide, Cisco DCNM for SAN, Release 11(x)*. It also provides information on how to obtain related documentation.

Audience

This guide is for users who are familiar with general object-oriented programming techniques and with 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	Introduction to Cisco DCNM for SAN SMI-S	Provides an overview of the support provided for CIM and other standards.
Chapter 2	Configuring and Using Cisco DCNM SMI-S Server	Provides CLI commands to configure the CIM server, and sample scenarios for using CIM to manage your SAN.
Chapter 3	Cisco DCNM SMI-S Server Support	Describes the supported profiles, indications, and Cisco-specific extensions.
Chapter 4	SMI-S Notifications	Describes the SMI-S notifications.
Chapter 5	Sample SMI-S Java Client	Provides the sample SMI-S java client and services.
Chapter 6	Managed Object Format Files for Cisco DCNM SMI-S Server	Provides the text from the MOF files for the Cisco MDS 9000 Family CIM server extensions.

Chapter	Title	Description
Chapter 7	DCNM for SAN Web Services API	Provides application programming interfaces (APIs) that expose DCNM-SAN core software functionalities as remote procedure calls to third-party vendors.
Chapter 8	DCNM for SAN Web Services API	Describes the Cisco DCNM for SAN (DCNM-SAN) Web Services (FMWS) application program interface (API).
Chapter 9	Discovery Automation Tool	This is a tool to discover DCNM-SAN, DCNM-LAN, and vCenter.
Appendix A	Sample Client Programs to Use Web Services	Provides a sample client program using API.
Appendix B	Perl Client	Provides the Perl client installation information and some sample perl client programs.
Appendix C	Command Line Storage Dumps Tool	Provides installing and usage information of the Command Line Storage Dumps tool.

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 Cisco MDS 9000 EPLD Images*
- *Cisco DCNM Release Notes*

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 9000 Family Switch-to-Switch Interoperability Configuration Guide*

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 Software Upgrade and Downgrade 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 MDS 9000 Family Cookbook for Cisco MDS SAN-OS*

Cisco DCNM for SAN

- *System Management Configuration Guide, Cisco DCNM for SAN*
- *Interfaces Configuration Guide, Cisco DCNM for SAN*
- *Fabric Configuration Guide, Cisco DCNM for SAN*
- *Quality of Service Configuration Guide, Cisco DCNM for SAN*
- *Security Configuration Guide, Cisco DCNM for SAN*
- *IP Services Configuration Guide, Cisco DCNM for SAN*
- *Intelligent Storage Services Configuration Guide, Cisco DCNM for SAN*
- *High Availability and Redundancy Configuration Guide, Cisco DCNM for SAN*
- *Inter-VSAN Routing Configuration Guide, Cisco DCNM for SAN*
- *SMI-S and Web Services Programming Guide, Cisco DCNM for SAN*

Cisco DCNM

The following publications support both Cisco DCNM for LAN and DCNM for SAN, and address the new licensing model, the new installation process, and the new features of Cisco DCNM:

- *Cisco DCNM Fundamentals Guide, Release 6.x*
- *Cisco DCNM Installation Guide, Release 6.x*

Command-Line Interface

- *Cisco MDS 9000 Family Command Reference*

Intelligent Storage Networking Services Configuration Guides

- *Cisco MDS 9000 Family 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*

Troubleshooting and Reference

- *Cisco DCNM Troubleshooting Guide*
- *Cisco MDS 9000 Family and Nexus 7000 Series System Messages Reference*
- *Cisco MDS 9000 Family SAN-OS Troubleshooting Guide*
- *Cisco MDS 9000 Family NX-OS MIB Quick Reference*
- *Cisco DCNM for SAN Database Schema Reference*

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

Introduction to Cisco DCNM for SAN SMI-S

Cisco Data Center Network Manager (DCNM) provides an industry standard application programming interface (API) using the Storage Management Initiative Specification (SMI-S). The 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.

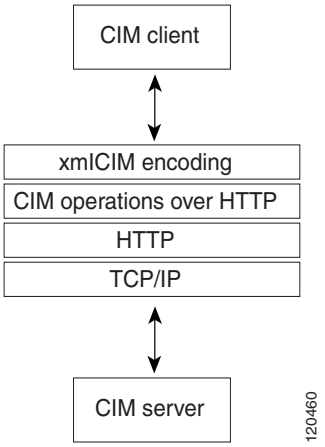
Note CIM is not supported in Cisco MDS NX-OS Release 5.2(1), but is supported in Cisco DCNM Release 5.2(1).

Note Cisco DCNM SMI-S Server is installed as part of the Cisco DCNM-SAN installation.

CIM uses a client/server model. The Cisco DCNM SMI-S Server can be embedded into the vendor product or can be implemented by a proxy server that provides the Cisco DCNM SMI-S Server functionality for the legacy vendor product. The CIM client is the management application that communicates to multiple Cisco DCNM SMI-S Servers to manage the SAN. The CIM client discovers Cisco DCNM SMI-S 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

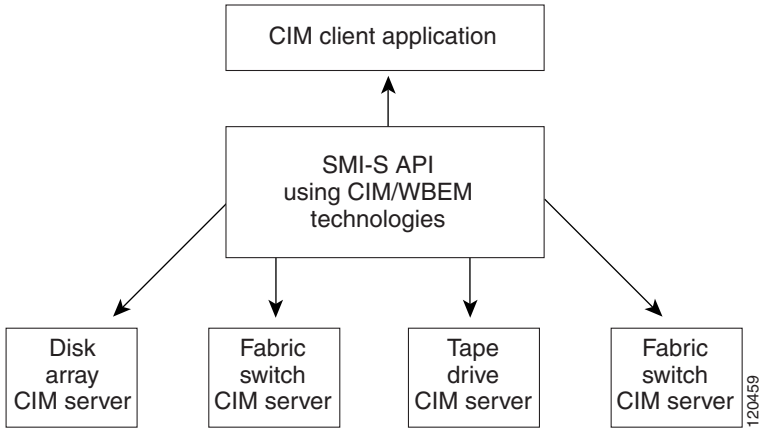


For more information about CIM, refer to the specification available through the Distributed Management Task Force (DMTF) website at http://www.dmtf.org.

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 ensure 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.

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 can define the `NetworkDevice` class to describe this object. The `NetworkDevice` class contains properties or attributes of a network device. Some properties for this `NetworkDevice` class are `IpAddress` and `DeviceType`. The NetworkDevice class controls the network device. Methods and routines trigger actions on the network device. Example of methods are enablePort() and rebootDevice().

After defining 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 the `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 that are 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.

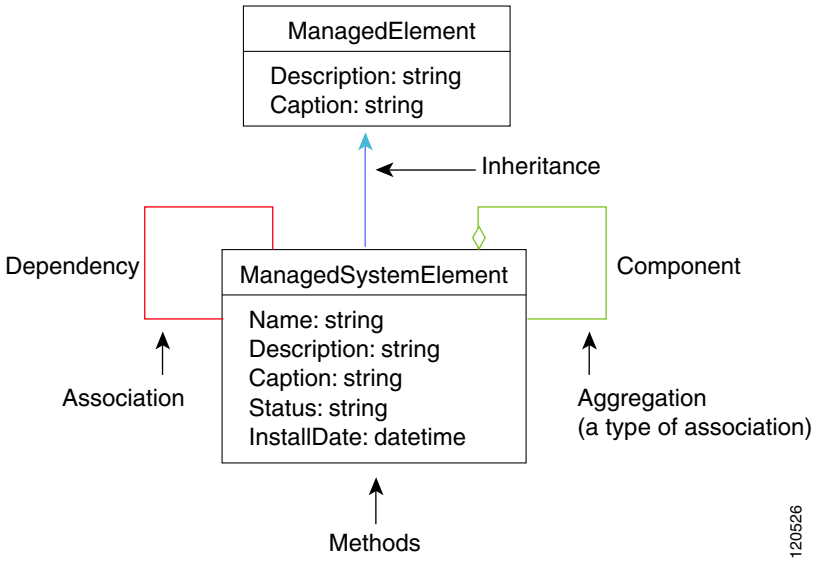
Understanding UML

UML provides 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 Cisco DCNM for SAN

SMI-S defines a number of profiles that specify the managed objects used to control and monitor elements of a SAN. Each switch or director in Cisco DCNM for SAN includes an embedded Cisco DCNM SMI-S Server. The Cisco DCNM SMI-S Server communicates with any CIM client to provide SAN management compatible with SMI-S. The Cisco DCNM SMI-S 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 Cisco DCNM SMI-S Server and the CIM client.

CHAPTER 2

Cisco DCNM SMI-S Server Support

This chapter describes the standard profiles supported by Cisco DCNM SMI-S. The Cisco DCNM SMI-S 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 Cisco DCNM SMI-S Servers either embedded on the SAN elements or supported by a proxy Cisco DCNM SMI-S 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 Cisco DCNM SMI-S Servers. Once the Cisco DCNM SMI-S Servers are identified, the CIM client determines which profiles are supported on Cisco DCNM SMI-S Servers through the Server profile. This profile is mandatory on all Cisco DCNM SMI-S Servers.

Besides the control and monitoring support provided by profiles, the Cisco DCNM SMI-S 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 Cisco DCNM SMI-S Server is discovering the location and support available on Cisco DCNM SMI-S Servers. The SLPv2 provides this discovery mechanism. A CIM client uses SLPv2 to discover Cisco DCNM SMI-S Servers, gathering generic information about what services Cisco DCNM SMI-S Servers provides and the URL where these services are located.

Cisco DCNM SMI-S Server supports SLPv2 as defined in RFC 2608.

Server Profile

Once the CIM client discovers the Cisco DCNM SMI-S Servers within the SAN, the CIM client must determine the level of support each Cisco DCNM SMI-S Server provides. The Server profile defines the capabilities of the Cisco DCNM SMI-S Server. This includes providing the namespace and all profiles and subprofiles supported by Cisco DCNM SMI-S 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 Cisco DCNM SMI-S Server. Similarly, Cisco DCNM SMI-S Server lists all supported optional subprofiles, using the `RegisteredSubProfile` class and the `SubprofileRequiresProfile` association class to associate the subprofile with the profile.

Switch Profile

The Switch profile models the logical and physical aspects of switches. The Computer System class constitutes the center of the switch model. The switch includes discovery components including ports, port statistics, product information, software, and chassis information. It also includes configuration of the switch including switch and port state change, port speed, switch name, symbolic names, and DomainID. Figure 2-1 shows the switch profile from a communication perspective.

The Switch profile also supports the optional Blade subprofile (see the "Blade Subprofile" section on page 2-7) and the optional Access Point Subprofile (see the "Access Point Subprofile" section on page 2-9). Figure 2-2 shows the switch profile from a communication perspective and switch access point subprofile.

Figure 2-1 Switch Profile in Communication Perspective

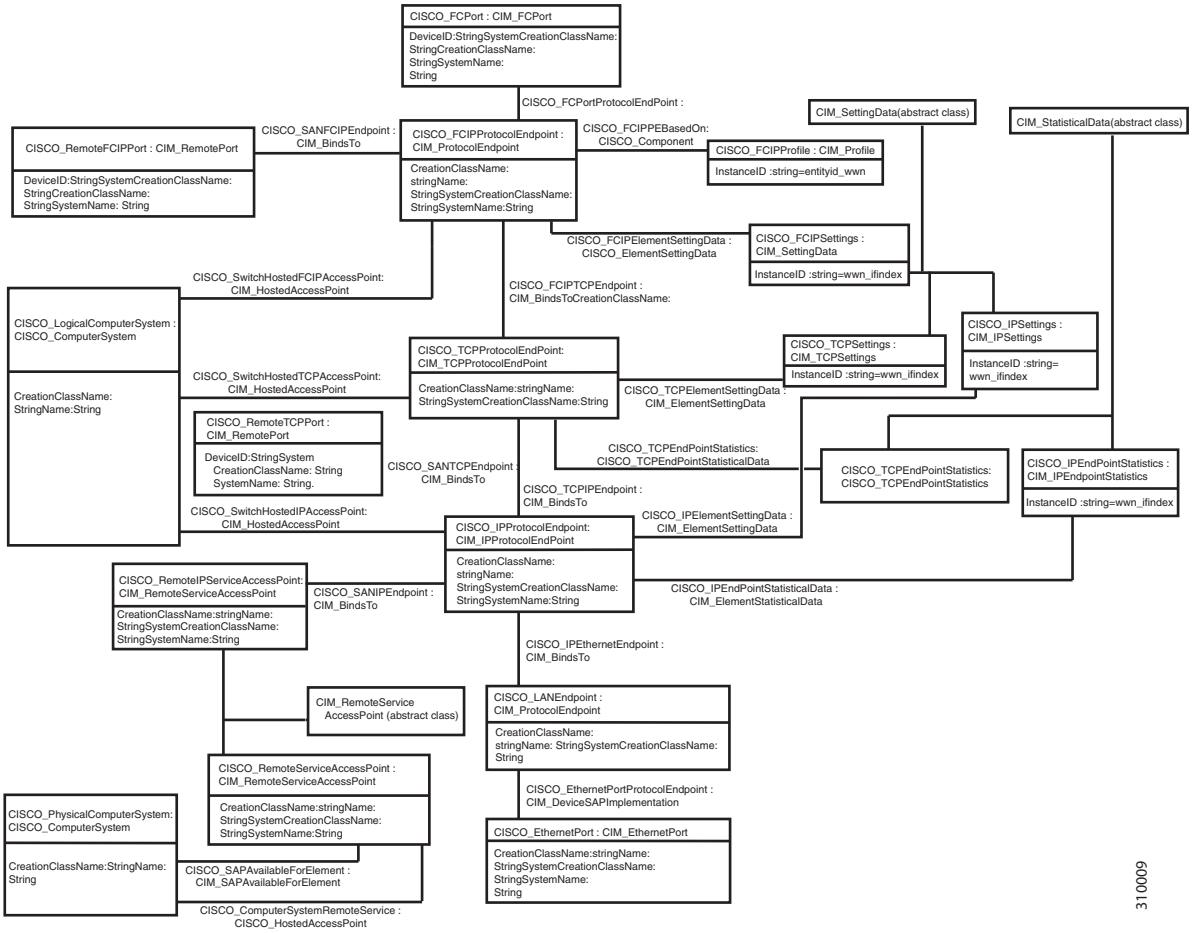


Figure 2-2 Switch Profile in Configuration Perspective and Switch Access Point Subprofile

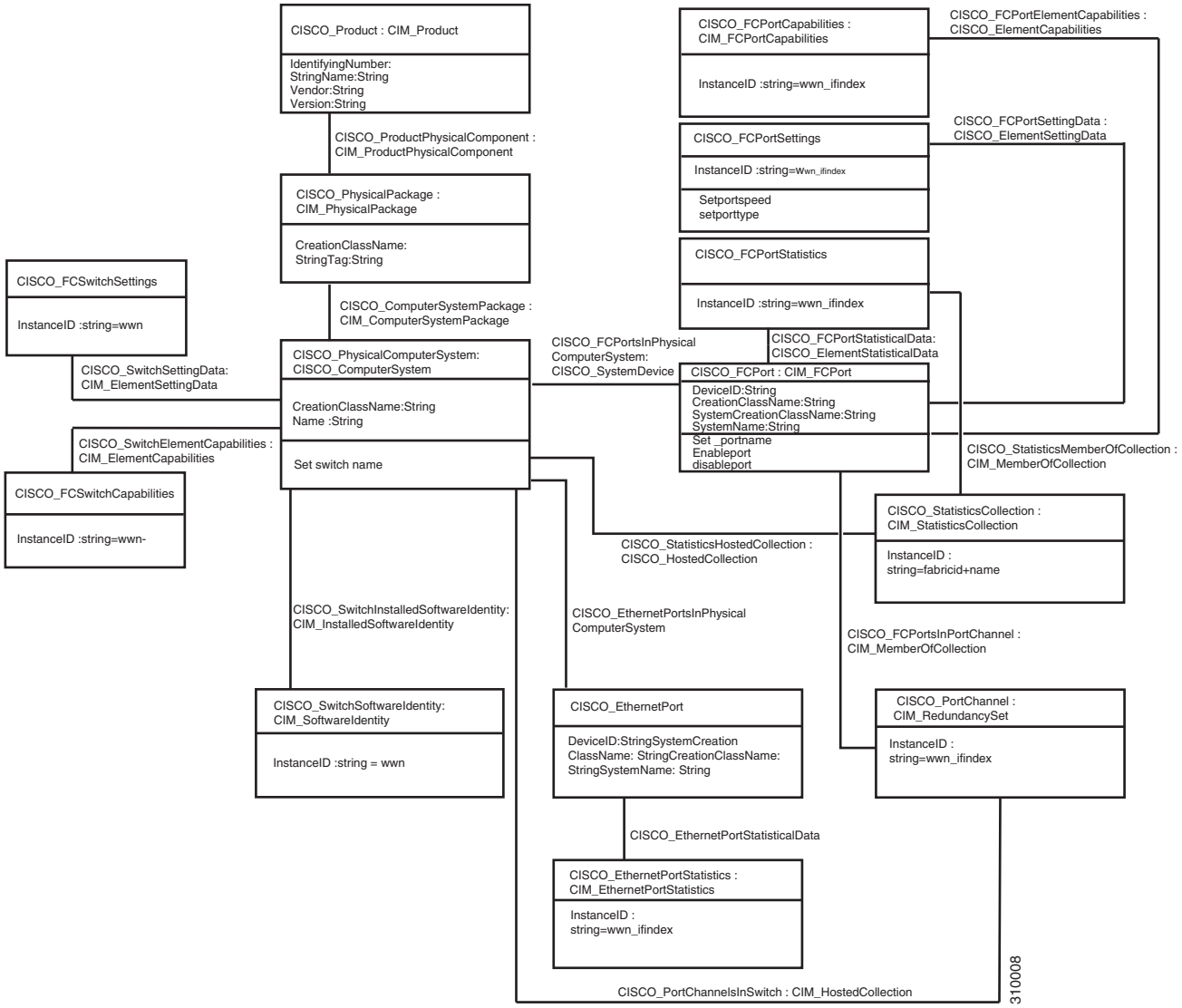


Table 2-1 shows how to use the classes and association classes of Switch profile.

Table 2-1 CIM Elements for Switch Profile

Class	How Used
CISCO_PhysicalComputerSystem:CISCO_ComputerSystem	Identifies the switch, with the Dedicated property set to Switch.
CISCO_ComputerSystemPackage:CIM_ComputerSystemPackage	Associates `PhysicalPackage` to the `ComputerSystem` (Switch).
CISCO_SwitchElementCapabilities:CIM_ElementCapabilities	Associates `FCSwitchCapabilities` to the `ComputerSystem` (Switch).

Table 2-1 CIM Elements for Switch Profile (continued)

Class	How Used
CISCO_SwitchInstalledSoftwareIdentity:CIM_InstalledSoftwareIdentity	Associates the switch and its software identity.
CISCO_SwitchSettingData: CIM_ElementSettingData	Associates `FCSwitchSettings` to `ComputerSystem`.
CISCO_FCPortSettingData: CISCO_ElementSettingData	Associates `FCPortSettings` to `FCPort`.
CISCO_FCPortElementCapabilities:CISCO_ElementCapabilities	Associates the `CISCO_FCPort` and `CISCO_FCPortCapabilities`.
CISCO_FCPortStatisticalData:CISCO_ElementStatisticalData	Associates the `FCPortStatistics` to the `FCPort`.
CISCO_StatisticsMemberOfCollection:CIM_MemberOfCollection	Associates the `NetworkPortStatistics` (fcportstatistics) to the `StatisticsCollection`.
CISCO_TCPProtocolEndPoint:CIM_TCPProtocolEndPoint	A protocol endpoint that is dedicated to running TCP.
CISCO_FCIPTCPEndpoint:CIM_BindsTo	Associates the `CISCO_FCIPProtocolEndpoint` and `CISCO_TCPProtocolEndPoint`.
CISCO_SANTCPEndpoint:CIM_BindsTo	Associates the `CISCO_IPProtocolEndpoint` and `CISCO_TCPProtocolEndPoint`.
CISCO_TCPElementSettingData:CIM_ElementSettingData	Associates the `CISCO_IPProtocolEndpoint` and `CISCO_IPSettings`.
CISCO_FCIPPEBasedOn:CISCO_Component	Associates the membership relationships between a fcipprofile and the fcip protocol endpoints within that switch.
CISCO_FCIPProtocolEndpoint :CIM_ProtocolEndpoint	A protocol endpoint that is dedicated to running fcipport.
CISCO_FCIPElementSettingData:CISCO_ElementSettingData	Associates the `CISCO_FCIPProtocolEndpoint` and `CISCO_FCIPSettings`.
CISCO_FCPortProtocolEndPoint:CIM_DeviceSAPImplementation	Associates the `CISCO_FcPort` with `CISCO_LANEndpoint`.
CISCO_EthernetPortProtocolEndpoint:CIM_DeviceSAPImplementation	Associates `CISCO_EthernetPort` with `CISCO_LANEndpoint`.
CISCO_PortChannel:CIM_RedundancySet	Displays port aggregation for Fibre Channel trunking.
CISCO_PortChannelsInSwitch :CIM_HostedCollection	Aggregates the PortChannels in switch.
CISCO_FCPortsInPortChannel :CIM_MemberOfCollection	Aggregates the `fcports` for port channeling (Trunking).
CISCO_IPProtocolEndpoint: CIM_IPProtocolEndPoint	A protocol end point that is dedicated to running IP.
CISCO_LANEndpoint:CIM_ProtocolEndpoint	A communication endpoint which, when its associated interface device is connected to a LAN, may send and receive data frames. LAN Endpoints include Ethernet, Token Ring and FDDI interfaces.
`CISCO_EthernetPortProtocolEndpoint:CIM_DeviceSAPImplementation`	Associates a `CISCO_EthernetPort` with `CISCO_LANEndpoint`.

Table 2-1 CIM Elements for Switch Profile (continued)

Class	How Used
CISCO_SANFCIPEndpoint:CIM_BindsTo	Associates between CISCO_FCIPProtocolEndpoint and CISCO_ProtocolEndPoint.
CISCO_SANIPEndpoint : CIM_BindsTo	Associates between CISCO_IPProtocolEndpoint and CISCO_RemoteFCIPServiceAccessPoint.
CISCO_IPEthernetEndpoint:CIM_BindsTo	Associates between CISCO_IPProtocollEndpoint and CISCO_LANEndPoint.
CISCO_TCPIPEndpoint:CIM_BindsTo	Associates between CISCO_IPProtocolEndpoint and CISCO_TCPProtocolEndPoint.
CISCO_IPElementSettingData:CIM_ElementSettingData	Associates between CISCO_IPProtocolEndpoint and CISCO_IPSettings.
CISCO_Product:CIM_Product	CISCO_Product is a concrete class that aggregates PhysicalElements, software (SoftwareIdentity and SoftwareFeatures), services and/or other products, and is acquired as a unit.
CISCO_FCPort:CIM_FCPort	Identifies Fibre Channel switch port.
CISCO_logicalfcport:CIM_FCPort	Identifies logical aspects of the port link and the data layers.
CISCO_FCSwitchSettings:CIM_FCSwitchSettings	Identifies Fibre Channel switch settings.
CISCO_FCSwitchCapabilities:CIM_FCSwitchCapabilities	Identifies Fibre Channel switch capabilities.
CISCO_FCPortSettings:CIM_FCPortSettings	Identifies Fibre Channel port settings.
CISCO_FCPortCapabilities:CIM_FCPortCapabilities	Defines configuration options supported by the ports.
CISCO_EthernetPort:CIM_EthernetPort	Identifies Ethernet port.
CISCO_EthernetPortStatistics:CIM_EthernetPortStatistics	Identifies Ethernet port statistics.
CISCO_FCPortStatistics:CIM_FCPortStatistics	Identifies Fibre Channel port statistics.
CISCO_StatisticsCollection :CIM_StatisticsCollection	Collection to aggregate Fibre Channel port statistics to Fibre Channel switch.
CISCO_SwitchSoftwareIdentity:CIM_SoftwareIdentity	Associates switch and its software identity.
CISCO_StatisticsHostedCollection:CISCO_HostedCollection	Associates the statistics collection to the computersystem representing the switch.
CISCO_EndPortsInHostComputerSystem:CISCO_SystemDevice	Identifies end ports in host device.
CISCO_EthernetPortsInPhysicalComputerSystem:CISCO_SystemDevice	Identifies Ethernet port in switch.
CISCO_FCPortsInLogicalComputerSystem:CISCO_SystemDevice	Identifies logical Fibre Channel ports in logical computer system in VSAN.
CISCO_TCPEndPointStatisticalData: CIM_ElementStatisticalData	Aggregates the statistics data for TCP end point.
CISCO_IPEndPointStatisticalData: CIM_ElementStatisticalData	Aggregates the statistics data for IP end point.
CISCO_EthernetPortStatisticalData: CISCO_ElementStatisticalData	Aggregates the statistics data for Ethernet end point.

Table 2-1 CIM Elements for Switch Profile (continued)

Class	How Used
CISCO_TCPSettings:CIM_TCPSettings	Defines TCP transport layer global settings.
CISCO_TCPEndPointStatistics:CIM_TCPEndpointStatistics	Defines the statistics for the TCP end point.
CISCO_IPSettings:CIM_IPSettings	Defines the operational settings for an IP implementation that are configured on a system-wide basis.
CISCO_IPEndPointStatistics:CIM_IPEndpointStatistics	Records the statistics for an CIM_IPProtocolEndpoint.

Table 2-2 shows the services supported by the switch profile.

Table 2-2 Switch Profile Services

Device Name	Description
Enable Fcport	Describes how to enable a port on a Fibre Channel switch.
Disable fcport	Describes how to disable a port on a Fibre Channel switch.
Setportspeed	Describes how to modify the speed of a port on a Fibre Channel switch.
Setporttype	Describes how to modify the port type on a Fibre Channel switch.
Setswitchname	Describes how to modify the name of a Fibre Channel switch.
Setportname	Describes how to modify the name of a port on a Fibre Channel switch.

Blade Subprofile

This subprofile describes how blades in a director class switch can be discovered and managed.

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 the Blade subprofile with the Switch profile, the CIM client gains a chassis-level view into the switch, associating ports to modules and modules to a switch. Figure 2-3 shows the switch blade subprofile.

Figure 2-3 Switch Blade Subprofile

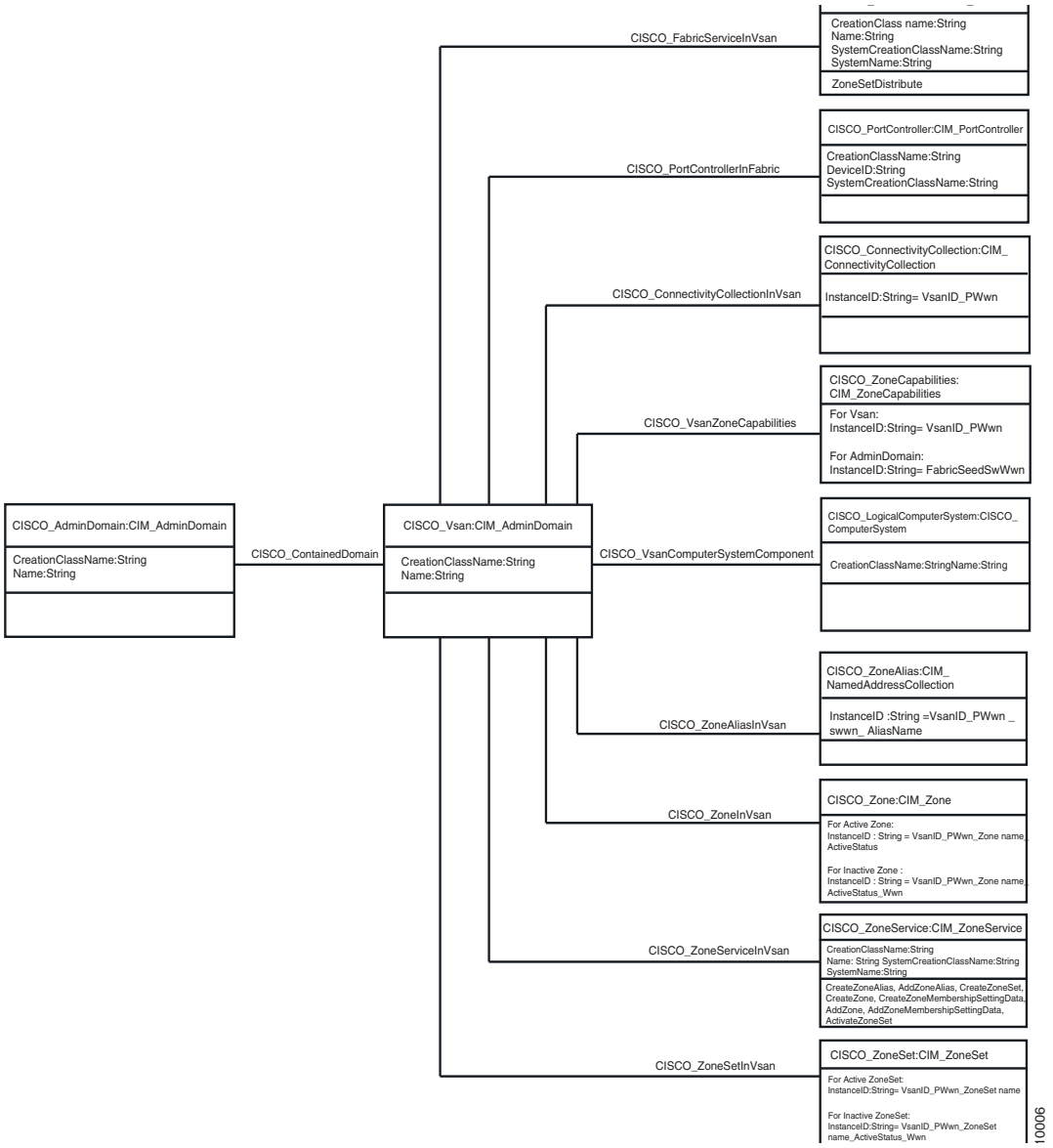


Table 2-3 shows how to use the classes and association classes of Blade subprofile.

Table 2-3 CIM Elements for Blade Subprofile

Class	How Used
`CISCO_LogicalModule:CIM_LogicalModule`	Identifies a blade, supervisor module, switching module, or services module as an aggregation point for the switch ports.
CISCO_ModuleFcPort: CISCO_ModulePort	Associates the logical module to the Fibre Channel port.
CISCO_ModuleEthernet Port:CISCO_ModulePort	Associates the logical module to the Ethernet port.

Table 2-3 CIM Elements for Blade Subprofile (continued)

Class	How Used
CISCO_ProductPhysicalComponent:CIM_ProductPhysicalComponent	Associates the physical element with product.
CISCO_LogicalModulesInPhysicalComputerSystem:CISCO_SystemDevice	Associates `CISCO_PhysicalComputerSystem` and `blade`.
CISCO_PhysicalPackageLogicalModuleRealizes:CISCO_Realizes	Associates `CISCO_PhysicalPackage` and `CISCO_LogicalModule`.
CISCO_PhysicalPackage:CIM_PhysicalPackage	Associates the physical package within which the logical module is stored as rack.
CISCO_LogicalModulesInPhysicalComputerSystem:CISCO_SystemDevice	Associates the logical module to the switch.
CISCO_EthernetPortsInPhysicalComputerSystem:CISCO_SystemDevice	Associates Ethernet ports in `PhysicalComputerSystem`.
CISCO_FCPortsInPhysicalComputerSystem:CISCO_SystemDevice	Associates `fcports` in `PhysicalComputerSystem`.
CISCO_PhysicalElementEthernetPortRealizes:CISCO_Realizes	Associates `Ethernetports` in `PhysicalElement`.
CISCO_PhysicalElementFcPortRealizes:CISCO_Realizes	Associates `Fcports` in `PhysicalElement`.

Access Point Subprofile

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

For Access Point subprofile, see Figure 2-2.

Table 2-4 shows how to use the classes and association classes of Access Point subprofile.

Table 2-4 CIM Elements for Access Point Subprofile

Class	How Used
CISCO_RemoteServiceAccessPoint:CIM_RemoteServiceAccessPoint	A ServiceAccessPoint for management tools. Returns the URL for the switch that can be used to install or launch Cisco DCNM-SAN or Device Manager.
CISCO_RemoteIPServiceAccessPoint:CIM_RemoteServiceAccessPoint	A ServiceAccessPoint for management tools. Returns the URL for the switch that can be used to install or launch Cisco DCNM-SAN or Device Manager.
CISCO_SANIPEndpoint:CIM_BindsTo	Associates CISCO_IPProtocolEndpoint and CISCO_RemoteFCIPServiceAccessPoint.
CISCO_ComputerSystemRemoteService:CISCO_HostedAccessPoint	Associates CISCO_PhysicalComputerSystem and CISCO_RemoteServiceAccessPoint.
CISCO_SAPAvailableForElement:CIM_SAPAvailableForElement	Associates between CISCO_PhysicalComputerSystem and CISCO_RemoteServiceAccessPoint. CISCO_SAPAvailableForElement conveys the semantics of a Service Access Point that is available for a ManagedElement.
CISCO_RemoteFCIPPort:CIM_RemotePort	Adds port information to the access data (such as IP address) that is specified in and inherited from RemoteServiceAccessPoint.
CISCO_SANFCIPEndpoint:CIM_BindsTo	Associates between CISCO_FCIPProtocolEndpoint and CISCO_RemoteFCIPPort.
CISCO_RemoteTCPPort:CIM_RemotePort	Adds port information to the access data (such as IP address) that is specified in and inherited from RemoteServiceAccessPoint.

Switch Partitioning Subprofile

The Switch Partitioning subprofile is used when a switch is implemented for multiple instances of a profile. The instances of the profile can be a mix of Switch profile and a different profile or a Switch Profile and a Extender Profile. The switch representing the entire set of systems is called the Partitioning System and the system that it is hosting is called the Partitioned System. For virtual fabrics, ANSI T11 calls the partitioning system the Core Switch and the partitioned system the Virtual Switch. Figure 2-4 shows the switch partitioning subprofile.

Figure 2-4 Switch Partitioning Subprofile

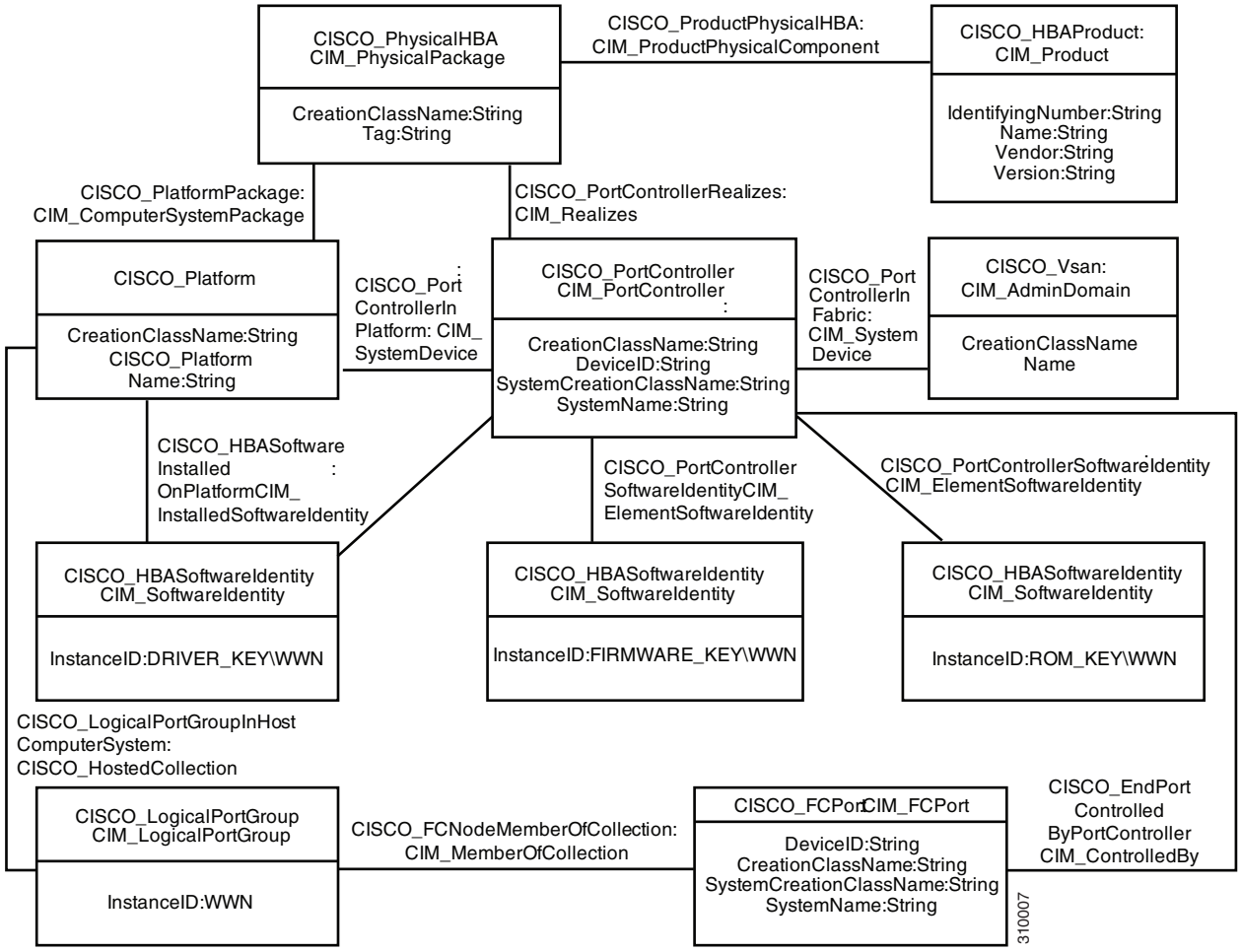


Table 2-5 shows how to use the classes and association classes of Switch Partitioning subprofile.

Table 2-5 CIM Elements for FabricSwitchPartitioning

Class	How Used
CISCO_PhysicalComputerSystem:CISCO_ComputerSystem	The partitioning computer system (core switch).
CISCO_LogicalComputerSystem:CISCO_ComputerSystem	The partitioned computer system that acts like switch (virtual switch).

Table 2-5 CIM Elements for FabricSwitchPartitioning (continued)

Class	How Used
CISCO_LogicalComputerSystemsInAdminDomain:CISCO_Component	Associates `CISCO_AdminDomain` and `CISCO_LogicalComputerSystem`.
CISCO_LogicalForPhysicalComputerSystem:CISCO_HostedDependency	Associates the partitioning computer system and partitioned computer system.
CISCO_LogicalSwitchElementCapabilities:CISCO_ElementCapabilities	Represents the association between managed elements and their capabilities.
CISCO_LogicalSwitchInstalledSoftwareIdentity : CISCO_InstalledSoftwareIdentity	Associates `CISCO_LogicalComputerSystem` and `CISCO_LogicalSwitchSoftwareIdentity`.
CISCO_LogicalSwitchSoftwareIdentity:CISCO_SoftwareIdentity	Software details of the logical switch.
CISCO_LogicalSwitchSettingData:CISCO_ElementSettingData	Associates `CISCO_LogicalComputerSystem` and `CISCO_FCLogicalSwitchSettings`.
CISCO_FCLogicalSwitchSettings:CIM_FCSwitchSettings	Depicts the switch settings class.
CISCO_ProtocolEndPointLogicalComputerSystem	Associates `CISCO_LogicalComputerSystem` and `CISCO_ProtocolEndPoint`.
CISCO_SwitchHostedFCIPAccessPoint:CIM_HostedAccessPoint	Associates `CISCO_LogicalComputerSystem` and `CISCO_FCIPProtocolEndPoint`.
CISCO_SwitchHostedIPAccessPoint: CIM_HostedAccessPoint	Associates `CISCO_LogicalComputerSystem` and `CISCO_IPProtocolEndPoint`.
CISCO_SwitchHostedTCPAccessPoint: CIM_HostedAccessPoint	Associates `CISCO_LogicalComputerSystem` and `CISCO_TCPProtocolEndPoint`.
CISCO_VsanComputerSystemComponent : CISCO_Component	Associates `CISCO_LogicalComputerSystem` and `CISCO_Vsan`.
CISCO_ZoneInLogicalComputerSystem :CISCO_HostedCollection	Associates `CISCO_LogicalComputerSystem` and `CISCO_Zone`.
CISCO_ZoneSetInLogicalComputerSystem : CISCO_HostedCollection	Associates `CISCO_LogicalComputerSystem` and `CISCO_ZoneSet`.
CISCO_FCPortsInLogicalComputerSystem:CISCO_SystemDevice	Associates `CISCO_LogicalComputerSystem` and `CISCO_LogicalFCPort`.
CISCO_HostComputerSystem : CISCO_ComputerSystem	The partitioned computer system which acts like the host (the host computer).
CISCO_EndPortsInHostComputerSystem:CISCO_SystemDevice	Associates `CISCO_HostComputerSystem` and `CISCO_EndPort`.
CISCO_HostComputerSystemsInAdminDomain:CISCO_Component	Associates `CISCO_HostComputerSystem` and `CISCO_admindomain`.
CISCO_LogicalPortGroupInHostComputerSystem:CISCO_HostedCollection	Associates `CISCO_HostComputerSystem` and `CISCO_LogicalPortGroup`.
CISCO_ProtocolEndPointHostComputerSystem:CISCO_HostedAccessPoint	Associates `CISCO_HostComputerSystem` and `CISCO_ProtocolEndPoint`.
CISCO_EndPort:CIM_FCPort	Identifies the switch port that connects to the host.
CISCO_EndPortControlledByPortController:CIM_ControlledBy	This association represents the relationship between an end port and `CIM_PortControllerclass`.

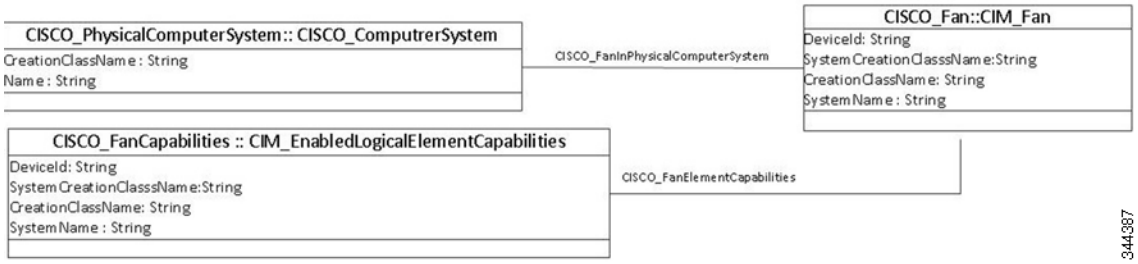
Table 2-5 CIM Elements for FabricSwitchPartitioning (continued)

Class	How Used
CISCO_EndPortSAPImplementation:CISCO_DeviceSAPImplementation	This association represents the relationship between an end port and `CISCO_ProtocolEndPoint`.
CISCO_FCNodeMemberOfCollection:CIM_MemberOfCollection	This association represents the relationship between an end port and `CISCO_LogicalPortGroup`.
CISCO_LogicalFCPort:CIM_FCPort	Fibre Channel port in the logical computer system.
CISCO_FCPortSAPImplementation:CISCO_DeviceSAPImplementation	Associates `Cisco_Logicalfcport` and `CISCO_ProtocolEndPoint`.
CISCO_FCPortsInLogicalComputerSystem:CISCO_SystemDevice	Associates `Cisco_logicalfcport` and `CISCO_LogicalComputerSystem`.
CISCO_LogicalFCPortForFCPort:CISCO_HostedDependency	Associates `Cisco_logicalfcport` and `CISCO_FCPort`.

Fan Profile

The fan profile describes the fan management in the switch.This profile includes classes which model for fan capabilities, fan relationship with switches, and its status. Figure 2-5 displays the fan profile.

Figure 2-5 Fan Profile



Power Supply Profile

The power profile describes the power supply management in the switch.This profile includes classes which model for power supply capabilities, power supply relationship with switches, and its status. Figure 2-6 displays the power supply profile.

Figure 2-6 Power Supply Profile

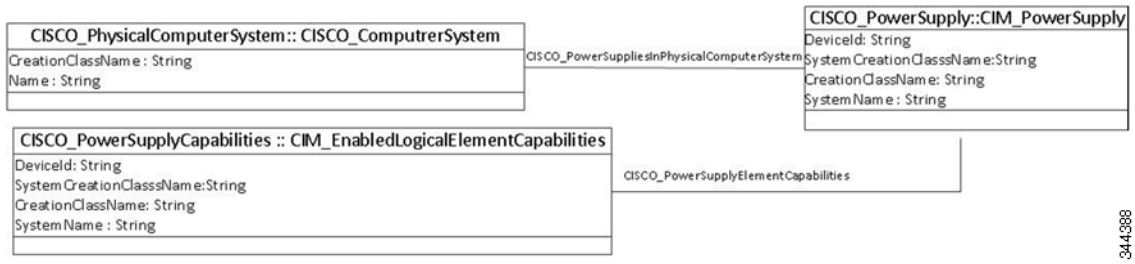


Table 2-6 shows how to use the classes and association classes of the Fan and Power Supply profile.

Table 2-6 CIM Elements for Fan and Power Supply Profile

Class	How Used
CISCO_Fan::CIM_Fan	Represents the fan in the physical switch.
CISCO_FanCapabilities::CIM_Enabled LogicalElementCapabilities	Represents the capabilities of the fan.
CISCO_FanElementCapabilities::CISCO_ElementCapabilties	Associates between the fan and its capabilities.
CISCO_FansInPhysicalComputerSystem::CISCO_SystemDevice	Associates between the fan and PhysicalComputerSystem.
CISCO_PowerSupply::CIM_PowerSupply	Represents the Power Supply in the physical switch.
CISCO_ PowerSupplyCapabilities::CIM_Enabled LogicalElementCapabilities	Represents the capabilities of PowerSupply.
CISCO_ PowerSupplyElementCapabilities:: CISCO_ElementCapabilties	Associates between the PowerSupply and its capabilities.
CISCO_PowerSuppliesInPhysicalComputerSystem::CISCO_SystemDevice	Associates between the PowerSupply andthe PhysicalComputerSystem.

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.

The SAN and fabrics are represented in CIM by the AdminDomain. SAN contains one or more fabrics, which are modeled as AdminDomains. For Fibre Channel fabrics, the identifier (AdminDomain.Name) is the fabric WWN which is the switch name of the principal switch. The AdminDomain for the Fibre Channel fabric has a NameFormat of WWN. Fabrics can contain one or more virtual SANs (VSANs). Figure 2-7 shows the fabric profile for zone sets. Figure 2-8 shows the fabric profile for the host computer system. Figure 2-9 shows the fabric profile for storage computer system. Figure 2-10 shows the fabric profile for port.

Figure 2-7 Fabric Profile for Zonest

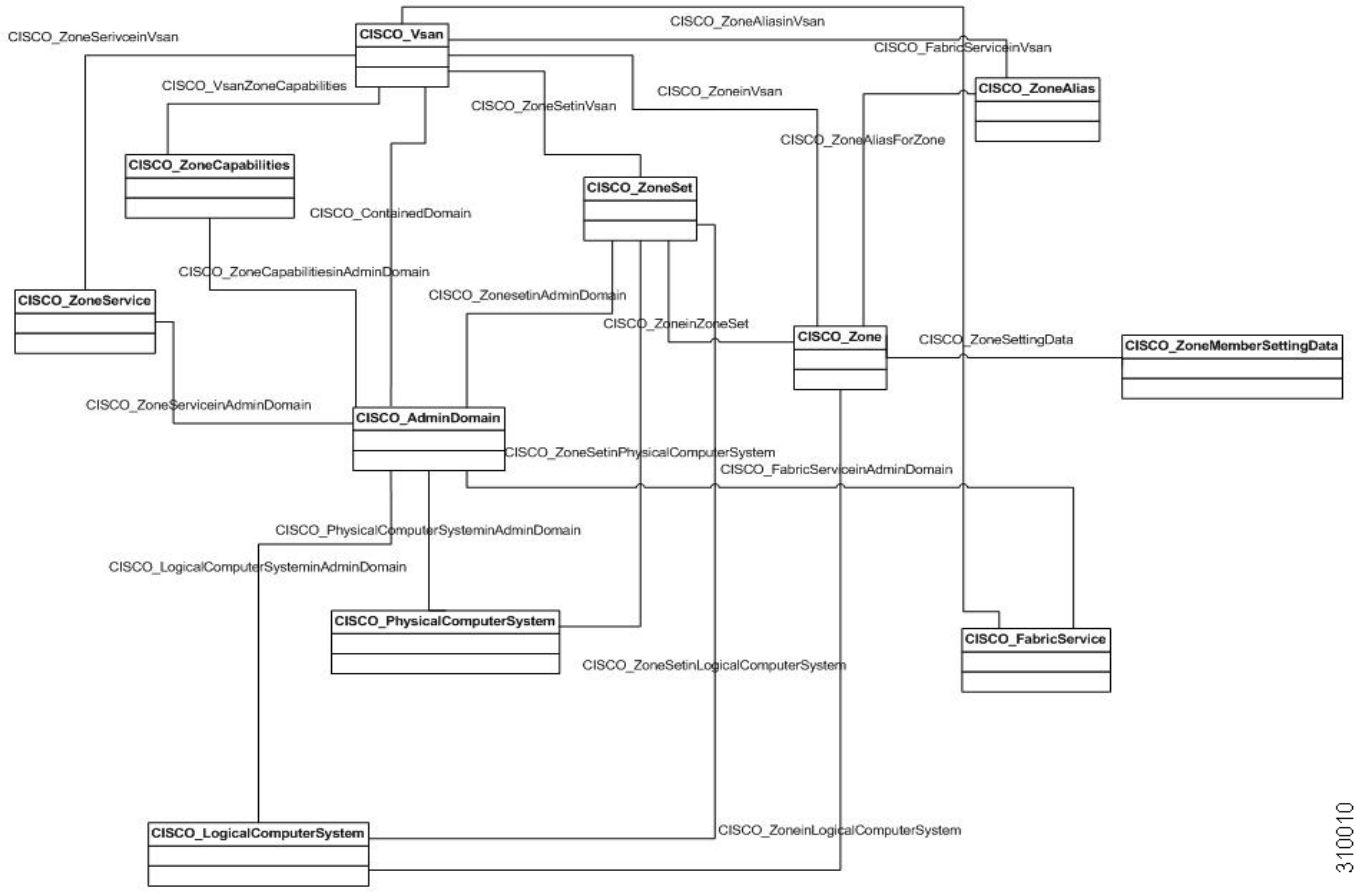


Figure 2-8 Fabric Profile for Host Computer System

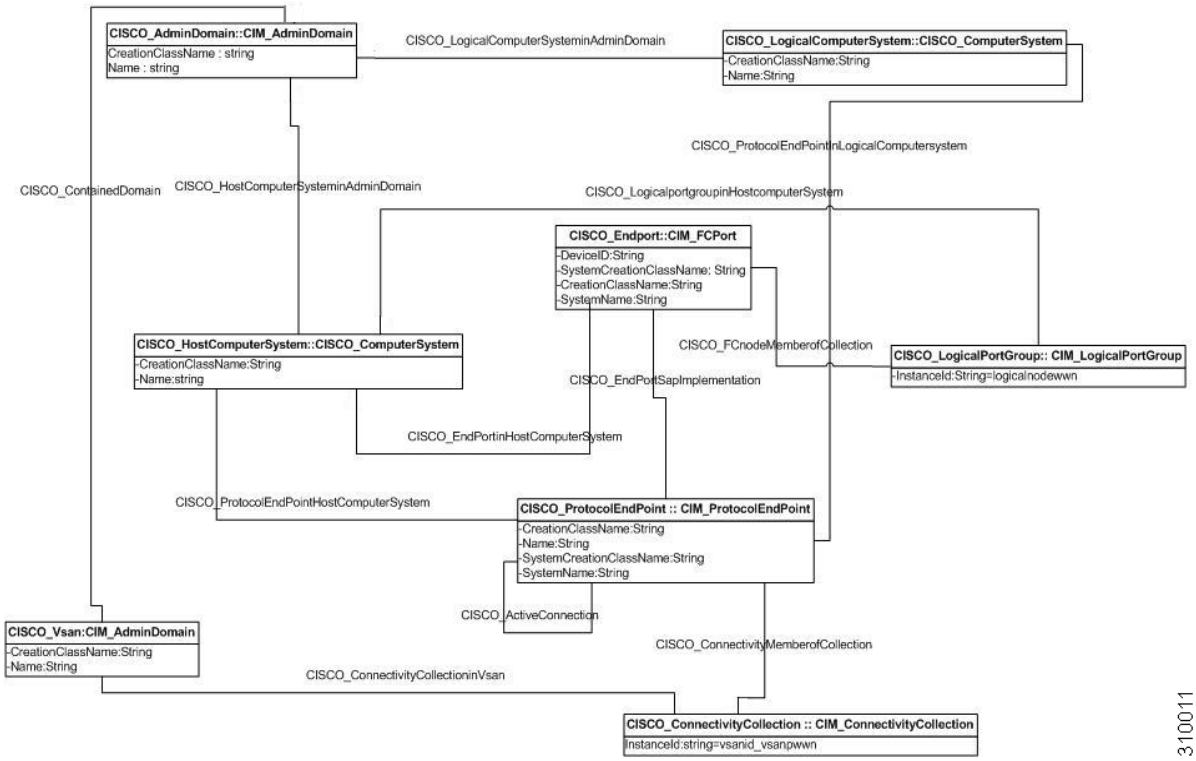


Figure 2-9 Fabric Profile for Storage Computer System

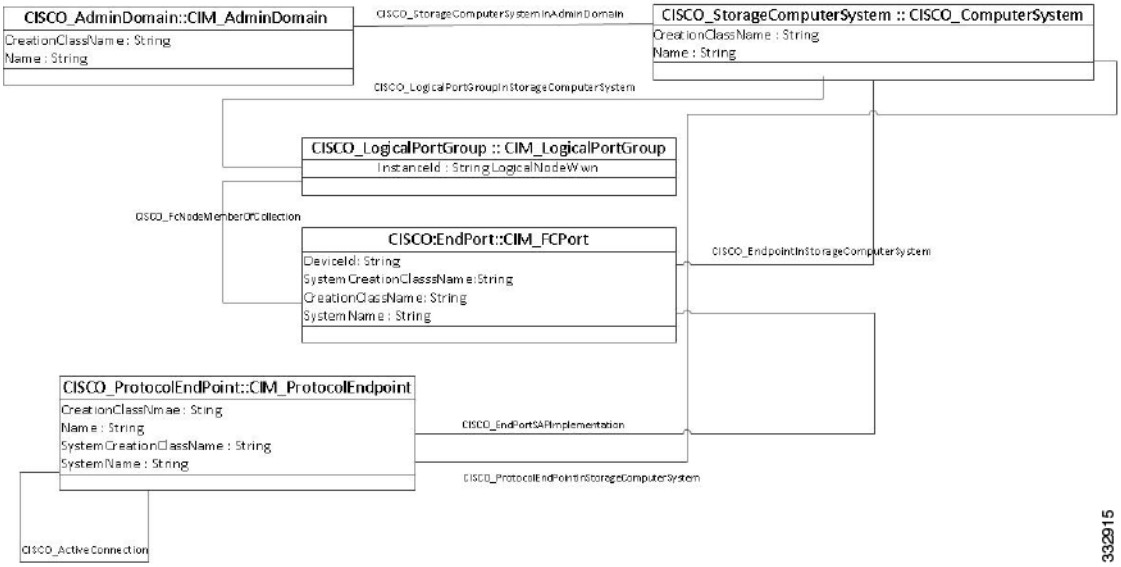


Figure 2-10 Fabric Profile for Port

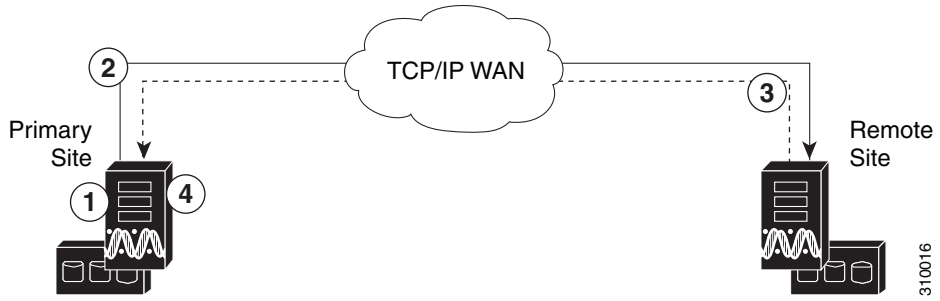


Table 2-7 shows how to use the classes and association classes of the Fabric profile.

Table 2-7 CIM Elements for Fabric

Class	How Used
`CISCO_ActiveConnection :CIM_ActiveConnection`	Associates a connection that is currently communicating or is configured to communicate between two ServiceAccessPoints.
`CISCO_AdminDomain:CIM_AdminDomain`	AdminDomain representing the SAN.
`CISCO_FCIPPEBasedOn:CISCO_Component`	Establishes membership relationships between a FCIP profile and the FCIP protocol end points within a switch.
`CISCO_HostComputerSystemsInAdminDomain:CISCO_Component`	Associates `CISCO_AdminDomain` and `CISCO_HostComputerSystem`.
CISCO_PhysicalComputerSystemsInAdminDomain:CISCO_Component	Associates `CISCO_AdminDomain` and `CISCO_PhysicalComputerSystem`.
CISCO_LogicalComputerSystemsInAdminDomain:CISCO_Component	Associates `Cisco_LogicalComputerSystem` and `Cisco_AdminDomain`.
CISCO_ConnectivityCollection:CIM_ConnectivityCollection	Collects the protocol endpoints of the fabric.
CISCO_ContainedDomain :CIM_ContainedDomain	Associates a fabric (Cisco_Vsan) to a SAN (CISCO_AdminDomain).
CISCO_VsanZoneCapabilities:CISCO_ElementCapabilities	Represents zone capabilities in the VSAN.
CISCO_EthernetPortProtocolEndpoint:CIM_DeviceSAPImplementation	Associates the Ethernet port to the LAN end point.
CISCO_FCPortProtocolEndPoint:CIM_DeviceSAPImplementation	Associates the Fibre Channel port to the FCIP end point.
CISCO_ZoneInPhysicalComputerSystem:CISCO_HostedCollection	Represents active and local zones of the switch. Active zones in all VSAN in which the switch is participating are considered. Local zones in the particular switch is represented.
CISCO_ZoneInLogicalComputerSystem:CISCO_HostedCollection	Associates zone (active and inactive) and `CISCO_logicalcomputersystem`.
CISCO_ZoneAliasInVsan :CISCO_ZoneHostedCollection	Represents the zone alias in the VSAN.
CISCO_ZoneInVsan : CISCO_ZoneHostedCollection	Displays the active zones in VSAN and the local zones in switches.

Table 2-7 CIM Elements for Fabric (continued)

Class	How Used
CISCO_LogicalPortGroup :CIM_LogicalPortGroup	Collection of one or more ports that are logically grouped for administrative and discovery or topology purposes. Logical port groups define port collections for access control or for use in routing policy or other management tasks.
CISCO_FCNodeMemberOfCollection:CIM_MemberOfCollection	Associates `CISCO_LogicalPortGroup` with endport.
CISCO_ZoneAliasForZone :CISCO_ZoneMemberOfCollection	Associates `Cisco_zonealias` and `Cisco_zone`.
CISCO_ZonesInZoneSet:CISCO_ZoneMemberOfCollection	Represents the zones present in the zone set.
CISCO_ZoneSetInPhysicalComputerSystem:CISCO_HostedCollection	Displays the active and local zone sets in all the VSANs in which the switch is participating.
CISCO_ZoneSetInVsan:CISCO_ZoneHostedCollection	Displays the active zone set in the VSAN. Displays the local zone set in all the switches present in the VSAN.
CISCO_ZoneSetInAdminDomain:CISCO_ZoneHostedCollection	Displays the zone set in admin domain which is the zone set present in the default VSAN (vsan 1).
CISCO_ZoneSetInLogicalComputerSystem:CISCO_HostedCollection	Displays the active and local zone sets in the VSAN (logical switch) in which the switch is participating.
CISCO_Zone:CIM_Zone	A zone is a group of ports, end points, nodes, zones, and namedAddressCollections that are managed collectively by the fabric. A zone indicates a set of members that are participating together in the fabric.
CISCO_ZoneSet:CIM_ZoneSet	ZoneSet is a group of zones that are managed collectively by the fabric. The zones are under enforcement by the fabric, only if the zone set is set to active. It displays all the active and local zone sets throughout the fabric. At any time, only one zone set is active in a VSAN.
CISCO_ZoneCapabilities :CIM_ZoneCapabilities	Exposes the capabilities for zoning of an AdminDomain.
CISCO_ZoneMemberSettingData:CIM_ZoneMembershipSettingData	Provides the identification criteria for possible zone and zone alias members. Thirteen different zone members are supported.
CISCO_ZoneSettingData :CISCO_ElementSettingData	Displays the zone member setting data of the selected zone instance, depending on whether it is an active or local instance.
CISCO_FabricServiceInVsan:CISCO_HostedService	Associates `Cisco_FabricService` and `Cisco_VSAN`.
CISCO_FabricService:CIM_Servic	Allows all of the fabric configuration changes.
CISCO_FabricServiceInAdminDomain:CISCO_HostedService	Association between `Cisco_FabricService` and `Cisco_AdminDomain`.
CISCO_StorageComputerSystem:CISCO_ComputerSystem	Represents target in the fabric.
CISCO_EndPort:CIM_FcPort	Identifies the switch port that connects to the host.
CISCO_ProtocolEndPoint:CIMProtocolEndPoint	A communication point from which data can be sent or received. ProtocolEndpoints link system or computer interfaces to LogicalNetworks.

Table 2-7 CIM Elements for Fabric (continued)

Class	How Used
CISCO_StorageComputerSystemsInAdminDomain :CISCO_Component	Associates CISCO_AdminDomain and `CISCO_StorageComputerSystem`.
CISCO_EndPortsInStorageComputerSystem:CISCO_SystemDevice	Associates `CISCO_StorageComputerSystem` and `CISCO_EndPort`.
CISCO_ProtocolEndPointStorageComputerSystem:CISCO_HostedAccessPoint	Associates `CISCO_StorageComputerSystem` and `CISCO_ProtocolEndpoint`.
CISCO_FcNodeMemberOfCollection:CIM_MemberOfCollection	Associates `CISCO_LogicalPortGroup` and `CISCO_EndPort`.
CISCO_EndPortSAPImplementation:CISCO_DeviceSAPImplementation	Asscoiates `CISCO_EndPort` to the protocol end point.

N Port Virtualizer Profile

N port virtualization (NPV) reduces the number of Fibre Channel domain IDs in DCNM-SAN. Switches operating in NPV mode are not part of the fabric and pass traffic between NPV core switch links and end devices and eliminates the domain IDs for these edge switches.

NPV integrates all locally connected N ports into one or more external NP links, which shares the domain ID of the NPV core switch among multiple NPV switches. NPV also allows multiple devices to add to the same port on the NPV core switch, which reduces the need for more ports on the core.

Figure 2-11 N Port Virtualizer Profile

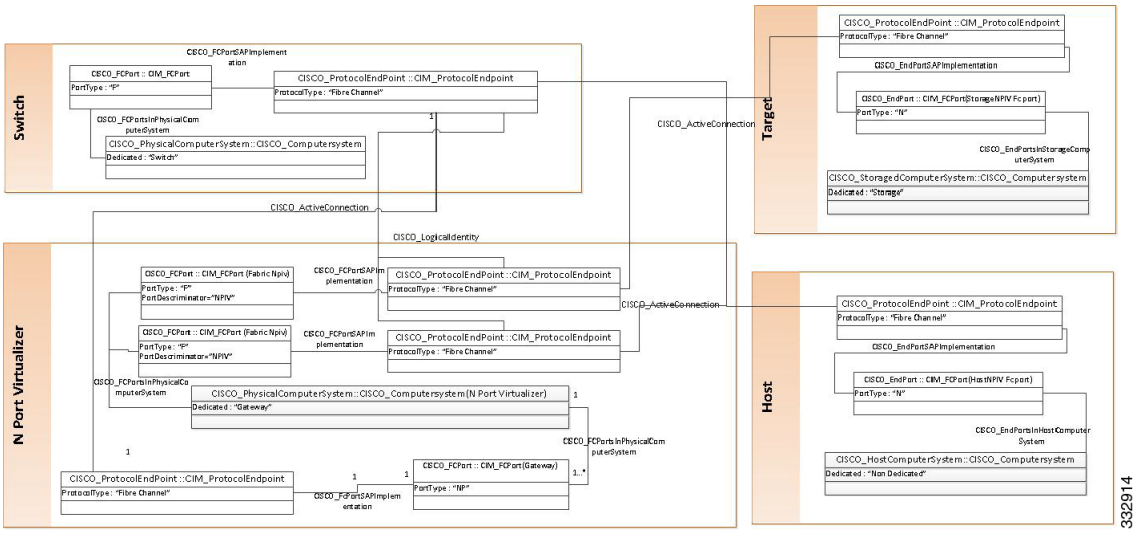


Table 2-8 shows how to use the classes and association classes of the N Port Virtualizer profile.

Table 2-8 CIM Elements for N Port Virtualizer Profile

Class	How Used
`CISCO_PhysicalComputerSystemsInAdminDomain : CISCO_Component` (N Port Virtualizer to Fabric)	Aggregates N Port Virtualizers in the `AdminDomain` that represents the fabric.
`CISCO_PhysicalComputerSystem:CISCO_ComputerSystem` (N PortVirtualizer)	The computer system representing the N Port Virtualizer.
`CISCO_PhysicalComputerSystem:CISCO_ComputerSystemPackage (N Port Virtualizer to Physical Package)`	This class is required if the Switch profile is implemented. Associates `PhysicalPackage` to the `ComputerSystem` (N Port Virtualizer).
`CISCO_FCPortSAPImplementation:CISCO_DeviceSAPImplementation` (ProtocolEndpoint to Gateway FCPort)	Associates the N Port Virtualizer Gateway FC port to its `ProtocolEndpoint`.
CISCO_FCPortSAPImplementation:CISCO_DeviceSAPImplementation (ProtocolEndpoint to NPIV FCPort)	Associates the N Port Virtualizer NPIV FC port to its `ProtocolEndpoint`.
CISCO_ActiveConnection: CIM_ActiveConnection(Gateway)	The association between `ProtocolEndpoints` representing the links between Fibre Channel switch ports and N Port Virtualizer gateway ports that are used to create active connections between platform and switch ports.
CISCO_ActiveConnection: CIM_ActiveConnection(N Port Virtualiztion)	The association between `ProtocolEndpoints` representing the links between Fibre Channel platform ports and switch ports that are created through an N Port Virtualizer.
CISCO_FCPort: CIM_FCPort (Fabric NPIV)	NPIV Fibre Channel ports of the N Port Virtualizer.
CISCO_FCPort: CIM_FCPort (Gateway)	A Fibre Channel port of the N Port Virtualizer that is used to connect to the switch.
Class CISCO_ProtocolEndPointLogicalComputerSystem :CISCO_HostedAccessPoint (N Port Virtualizer System to protocolEndpoint)	Associates the `ProtocolEndpoint` to the N Port Virtualizer `ComputerSystem`.
CISCO_LogicalIdentity: CIM_LogicalIdentity	Associates `ProtocolEndpoints` of N Port Virtualizer NPIV FC ports to `ProtocolEnpoints` of Switch FC ports.
CISCO_ProtocolEndPoint:CIMProtocolEndPoint(N Port Virtualizer)	The endpoint of a link (`ActiveConnection`) on the N Port Virtualizer.

Table 2-8 CIM Elements for N Port Virtualizer Profile (continued)

Class	How Used
CISCO_FCPortsInPhysicalComputerSystemextends CISCO_SystemDevice (Gateway FCPort to Gateway System)	Associates N Port Virtualizer Gateway FC ports to the `ComputerSystem` (N Port Virtualizer).
CISCO_FCPortsInPhysicalComputerSystemextends CISCO_SystemDevice(N Port Virtualizer NPIV FCPort to Gateway System	Associates N Port Virtualizer NPIV FC ports to the `ComputerSystem` (N Port Virtualizer).

FDMI Profile

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 2-12 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 2-12 FDMI Subprofile

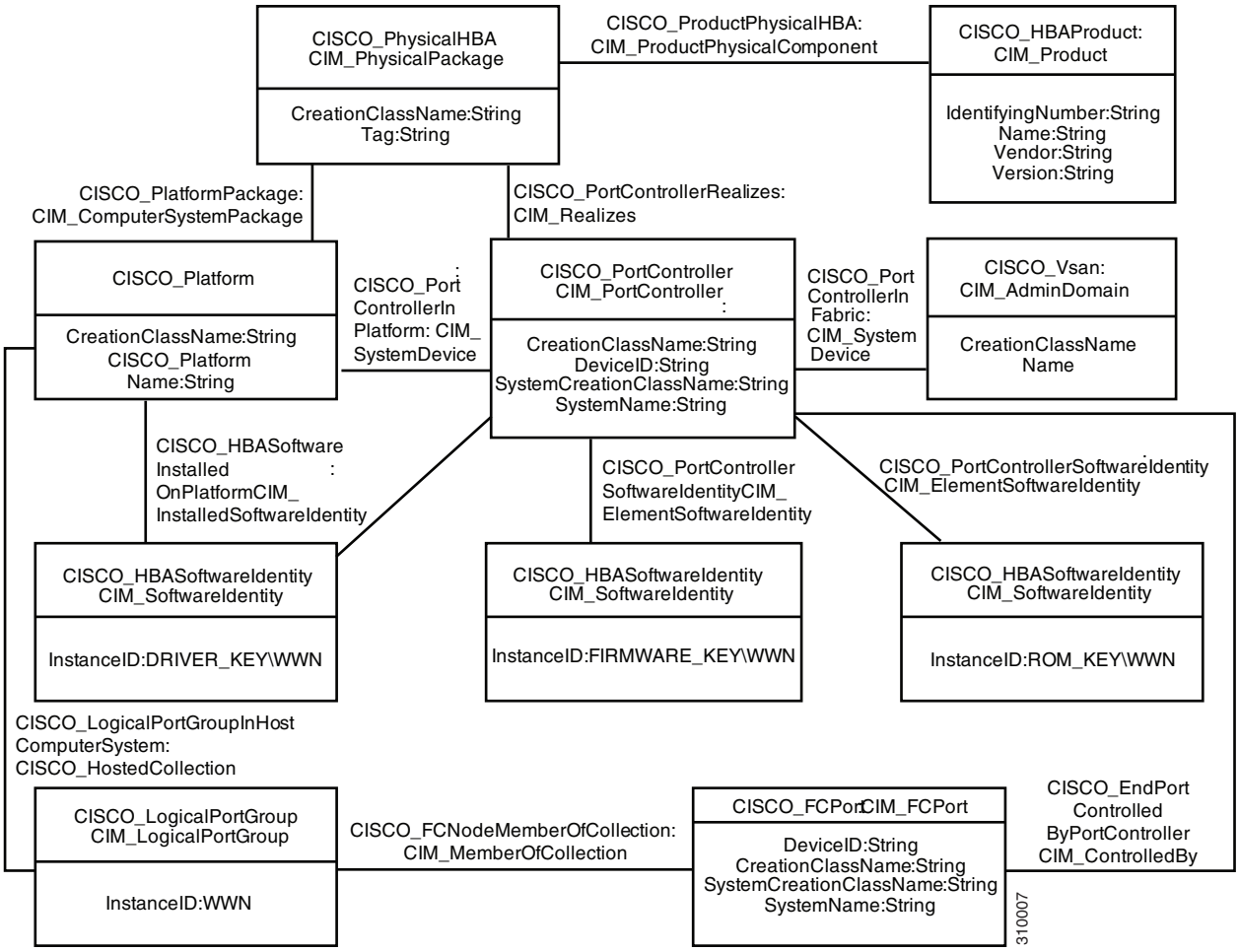


Table 2-9 shows how to use the classes and association classes of the FDMI subprofile.

Table 2-9 CIM Elements for FDMI

Class	How Used
`CISCO_Platform:CIM_ComputerSystem`	Represents a fabric-connected entity, containing one or more Node objects, that has registered with a fabric's Management Server service. This class also represents the HBA host.
`CISCO_HBASoftwareInstalledOnPlatform:CIM_InstalledSoftwareIdentity`	Allows identification of the platform on which HBA driver is installed.
`CISCO_PlatformHostedSANAccessPoint:CIM_HostedAccessPoint`	Associates a `ProtocolEndPoint` and the platform provided. Each platform can host many `ProtocolEndPoints`.
`CISCO_PlatformPackage:CIM_ComputerSystemPackage`	Denotes one or more physical HBAs that recognizes a platform.
CISCO_PortController:CIM_PortController	Represents the port controller of an FDMI-enabled HBA. PortController is a logical device corresponding to a hardware network port controller. Port controllers provide various features depending on their types and versions.

Table 2-9 CIM Elements for FDMI (continued)

Class	How Used
CISCO_PortControllerInPlatform:CIM_SystemDevice	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.
CISCO_EndPortControlledByPortController:CIM_ControlledBy	Represents the relationship between a `Cim_Portcontroller`, which depicts the control services of the port to `CIM_EndPort`.
CISCO_PortControllerSoftwareIdentity:CIM_ElementSoftwareIdentity	Associates any software that is associated with the port controller.
CISCO_EndPort:CIM_FCPort	Models the switch FC port that connects to the host.
CISCO_EndPortsInHostComputerSystem:CISCO_SystemDevice	Identifies end ports in host device.
CISCO_FCNodeMemberOfCollection:CIM_MemberOfCollection	Associates FCPort (end port) to the `LogicalPortGroup`.
CISCO_LogicalPortGroup:CIM_LogicalPortGroup	A collection of one or more ports that are logically grouped for administrative and discovery or topology purposes. LogicalPortGroups define port collections for access control, or for use in routing policy or other management tasks.
CISCO_LogicalPortGroupInHostComputerSystem:CISCO_HostedCollection	Associates the `LogicalPortGroup` (Fibre Channel node) to the hosting system.
CISCO_PhysicalHBA:CIM_PhysicalPackage	Represents an FDMI-enabled physical HBA card attached to a switch.
CISCO_PortControllerRealizes:CIM_Realizes	Defines the mapping between devices and the physical elements that implement them.
CISCO_ProductPhysicalHBA: CIM_ProductPhysicalComponent	Associates `HBAproduct` with `physicalHBA`.
CISCO_PortControllerInFabric:CIM_SystemDevice	Defines a SystemSpecificCollection in the context of a scoping system. This association is created if `CISCO_PortController` cannot be scoped within `CISCO_Platform`.
CISCO_HBAProduct:CIM_Product	Represents product information of an FDMI-enabled physical HBA card attached to a switch.

Virtual Fabrics Subprofile

Fibre Channel SANs can logically separate the hardware into multiple fabrics and keep them physically interconnected. The term for this technology is defined by ANSI T11 as virtual fabrics. ANSI T11 identifies the hardware as core switches.

To be consistent with more DMTF schematics, the Virtual Fabrics subprofile names the partitioning systems. ANSI T11 identifies the switching construct that resides in the partitioning system as the virtual switch. The Fabric profile provides the option to discover virtual fabrics and virtual switches. The Virtual Fabrics subprofile provides the option to discover the underlying partitioning system. The Switch Partitioning subprofile provides the method to configure the partitioning system. Figure 2-13 shows the virtual fabrics subprofile.

Figure 2-13 Virtual Fabrics Subprofile

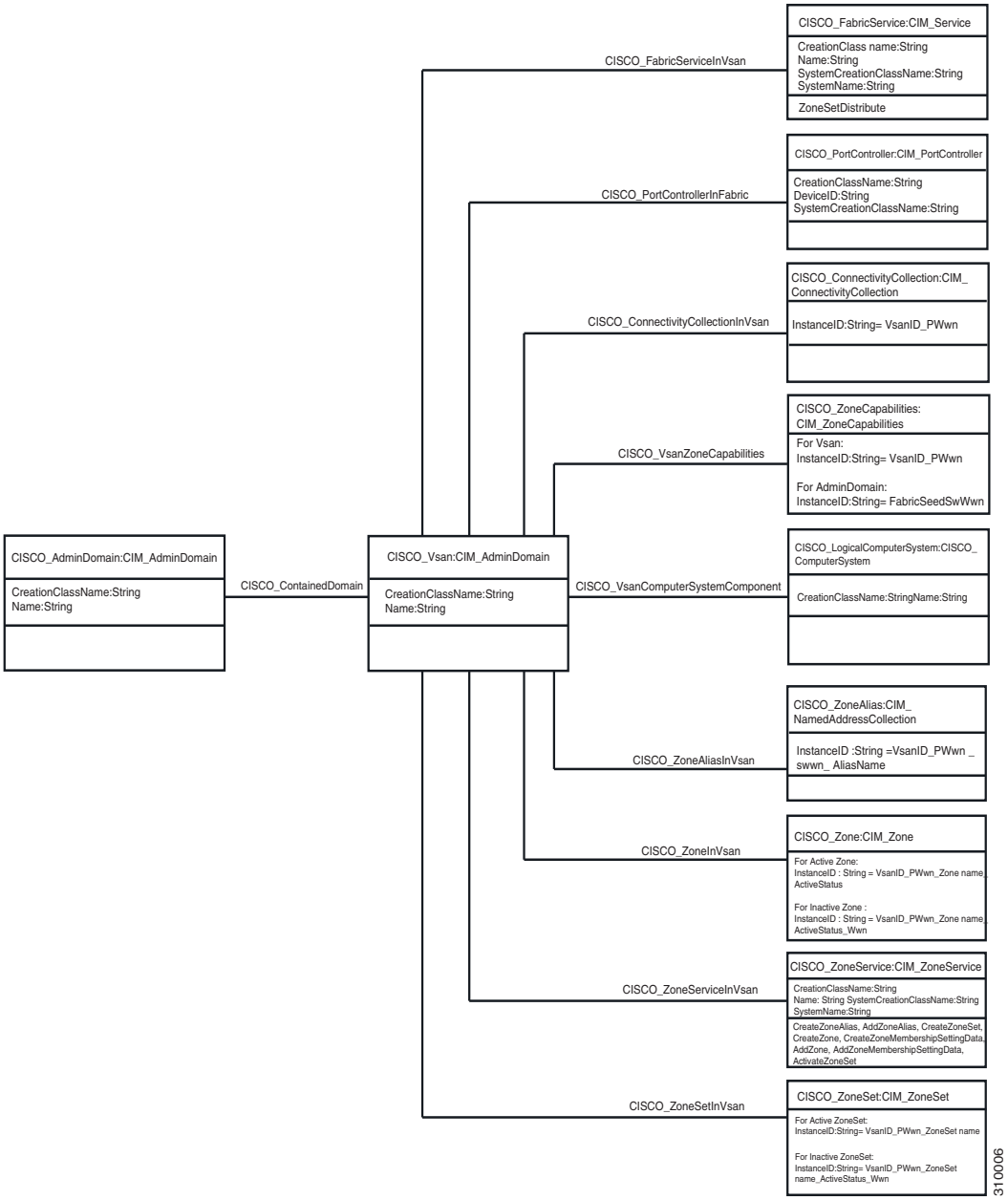


Table 2-10 shows how to use the classes and association classes of theVirtual Fabrics subprofile.

Table 2-10 CIM Elements for Virtual Fabrics

Class	How Used
`CISCO_Vsan:CIM_AdminDomain`	AdminDomain representing the SAN.
`CISCO_ConnectivityCollectionInVsan:CISCO_HostedCollection`	Associates `Cisco_vsan` and `CISCO_ConnectivityCollection`.

Table 2-10 CIM Elements for Virtual Fabrics (continued)

Class	How Used
CISCO_ContainedDomain:CISCO_HostedCollection	Associates Cisco_vsan and Cisco_admindomain.
CISCO_FabricServiceInVsan :CISCO_HostedService	Associates Cisco_vsan and CISCO_FabricService.
CISCO_PortControllerInFabric:CIM_SystemDevice	Associates Cisco_vsan and CISCO_PortController.
CISCO_VsanComputerSystemComponent:CISCO_Component	Associates Cisco_vsan and CISCO_LogicalComputerSystem.
CISCO_VsanZoneCapabilities:CISCO_ElementCapabilities	Represents the association between ManagedElements and their capabilities.
CISCO_ZoneAliasInVsan:CISCO_ZoneHostedCollection	Associates Cisco_vsan and CISCO_ZoneAlias.
CISCO_ZoneInVsan:CISCO_ZoneHostedCollection	Associates Cisco_vsan and CISCO_Zone.
CISCO_ZoneServiceInVsan:CISCO_HostedService	Associates Cisco_vsan and CISCO_ZoneService.
CISCO_ZoneSetInVsan:CISCO_ZoneHostedCollection	Associates Cisco_vsan and CISCO_ZoneSet.

Enhanced Zoning and Enhanced Zoning Control Subprofile

This profile describes the additional zoning functions for enhanced zoning. Sessions are normally part of enhanced zoning, but are included in the base fabric profile to address the various types of zoning operations into a single object model. Figure 2-14 shows a enhanced zoning and enhanced zoning control.

Table 2-11 shows how to use the classes and association classes of Enhanced Zoning and Enhanced Zoning Control subprofile.

Table 2-11 CIM Elements for Enhanced Zoning and Enhanced Zoning Control

Class	How Used
CISCO_ZoneAliasSettingData:CISCO_ElementSettingData	Associates ZoneMembershipSettingData to ZoneAlias.
CISCO_ZoneAliasInVsan:CISCO_ZoneHostedCollection	Associates the zone alias to the AdminDomain.
CISCO_ZoneAliasForZone:CISCO_ZoneMemberOfCollection	Associates the zone alias with zone.
CISCO_ZoneAlias:CIM_NamedAddressCollection	Depicts zone alias.
CISCO_ZoneService:CIM_ZoneService	Allows all of the zoning configuration changes.

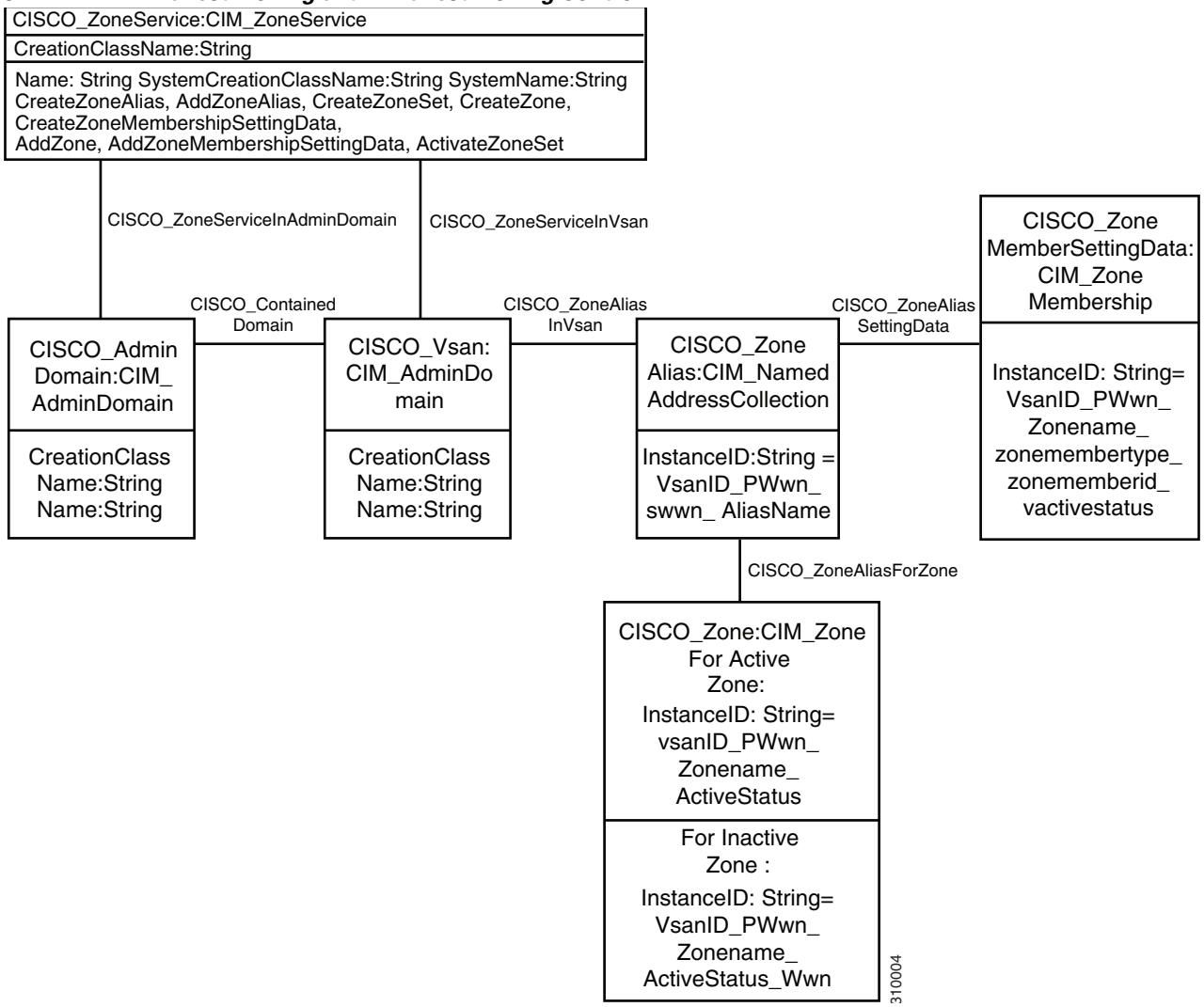
Extrinsic methods for this subprofile are as follows:

- CreateZoneAlias—Creates a ZoneAlias in the principal switch of the selected VSAN.
- AddZoneAlias—Adds the ZoneAlias to the zone.

Intrinsic methods for this subprofile are as follows:

- Delete zonealias—Deletes a zone alias.

Figure 2-14 Enhanced Zoning and Enhanced Zoning Control



Zone Control Subprofile

This profile includes extrinsic methods for creating zone sets, zones, and zone members (see Figure 2-14) and adding zones to zone sets and zone members to zones. SMI-S also defines intrinsic methods for the removing of zone members from zones and zone aliases, zones from zone sets, and deleting zone members, zones, and zone sets.

Table 2-12 shows how to use the classes and association classes of the Zone Control subprofile.

Table 2-12 CIM Elements for Zone Control

Class	How Used
`CISCO_ZoneServiceInAdminDomain:CISCO_HostedService`	Associates the `ZoneService` to the `AdminDomain` representing the fabric.

Table 2-12 CIM Elements for Zone Control (continued)

Class	How Used
`CISCO_ZoneServiceInVsan:CISCO_HostedService`	Associates the `ZoneService` and the associated VSAN.
`CISCO_ZoneService:CIM_ZoneService`	Allows for all of the zoning configuration changes.

Extrinsic methods for this subprofile are as follows:

- CreateZoneSet—The method creates a zone set in the principal switch of the selected VSAN.
- CreateZone—The method creates a zone in the principal switch of the selected VSAN.
- CreateZoneMembershipSettingData—The method creates a zone member and adds it to the specified zone or zone alias depending on the value of the input parameter systemSpecificCollection.
- AddZone—This method adds a zone to a zone set on the principal switch of the selected VSAN.
- AddZoneMembershipSettingData—The method adds the zone member to the specified zone or zone alias depending on the value of the input parameter systemSpecificCollection.
- ActivateZoneSet—This method enables the activation of a zone set.

Intrinsic methods for this subprofile are as follows:

- Delete zoneset—Deletes a zone set.
- Delete zone—Deletes a zone.
- Delete Zonemember—Deletes a zone member.

Configuring and Using Cisco DCNM SMI-S Server

This chapter provides the steps to configure Cisco DCNM SMI-S Server in Cisco DCNM 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*.

Installing Cisco DCNM SMI-S Server

Cisco DCNM SMI-S Server is installed as part of the Cisco DCNM installation. You can use Cisco DCNM-SAN installed locally to discover the SAN fabric. For more information on discovering the fabric using Cisco DCNM-SAN client, see the *Cisco DCNM Fundamentals Guide*.

Note All the platforms supported by Cisco DCNM are supported by SMI-S Server. SMI-S Server is configured as a startup service.

Changing the Default SMI-S Port

To change the default SMI-S port, follow these steps:

Step 1 Stop the Cisco SMI-S agent service.

Step 2 Rename **<installdir>/dcm/smis/server/jserver/bin/tcppe.properties.0** file to **<installdir>/dcm/smis/server/jserver/bin/tcppe.properties.**

Step 3 Edit **<installdir>/dcm/smis/server/jserver/bin/tcppe.properties** file to change the port by updating the property **cim-xml.http.portnumber** or **cim-xml.https.portnumber,** depending on whether DCNM is installed with http or https. Only one of this property is present in the file.

Step 4 Save the file and close it.

Step 5 Restart the Cisco SMI-S Agent service.

Step 6 SMI-S provider will use the port number assigned to the property above.

Performing Discovery and Performance Monitoring

You can use the Fabric and Switch profiles to implement discovery and performance monitoring. See the "Fan Profile" section on page 2-13 and the "FDMI Profile" section on page 2-21 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 by 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 Cisco DCNM SMI-S 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 Cisco DCNM SMI-S Server.

Note The namespace of Cisco DCNM SMI-S Server is 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 2-7 for more information about the Blade subprofile.

Configuring Zoning

[doc team: need info to use SCVMM/SMI-S/DCNM to configure Cisco zoning]

In the 10.1x installation guide I found -

"Cisco Prime Network Services Controller (NSC) provides the orchestration and automation of network services in Cisco Programmable Fabric. The Prime NSC supports integration with virtual computer and storage managers such as vCenter and System Center Virtual Machine Manager (SCVMM) and provides end-to-end orchestration and automation for services in Cisco Programmable Fabric."

Is NSC needed.

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.

Table 3-3 Zoning Extrinsic Methods (continued)

Extrinsic Method	How Used
`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`.
`ActivateZoneSet ()`	Sets the `ZoneSet` to active.
`ZoneSetDistribute()`	Distributes the full `ZoneSet` along with active zone set 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 2-26 and the "Enhanced Zoning and Enhanced Zoning Control Subprofile" section on page 2-25 for information about the zoning subprofiles.

Note 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.

SMI-S Notifications

This chapter includes the following sections:

WBEM Server

WBEM server provides the interfaces to allow operators to specify the faults that needs to be notified. The CIM classes used to establish this monitoring are included in Event Common model.

Event Common Model

The following are the components of the event common model:

- Indication Provider—Detects the traps and passes to the WBEM (CIMOM) server s indication. An indication is an instance of a class derived from the CIM_Indication.
- Filters—A predefined pattern checked by CIMOM against indications. Filters are instances of classes derived from CIM_Indication filter.
- Subscriptio—WBEM server consults the subscription to see if any operator is interested in being notified about indication. Subscriptions are instances of associations derived from CIM_IndicationSubscriptions.
- Handlers— WBEM server sends notifications to them. Handlers are defined in instances of classes derived from CIM_IndicationHandler.
- Listenersq—Represents the SMI-S clients and receive indications.

Figure 4-1 Architecture of the Event Common Model

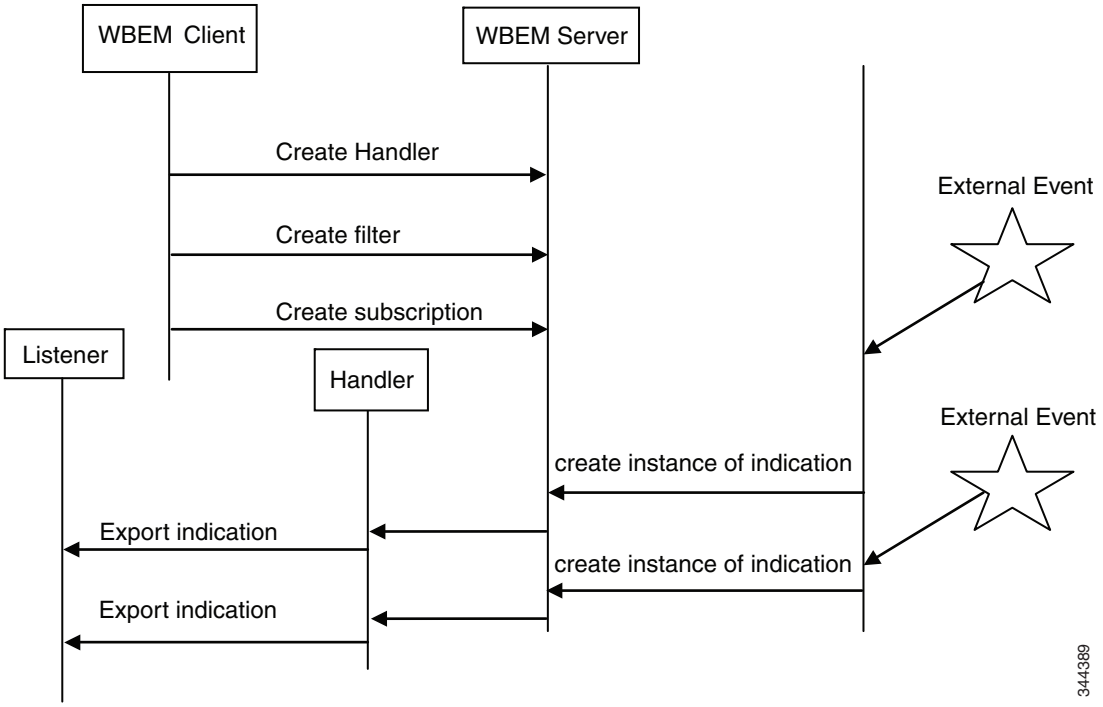
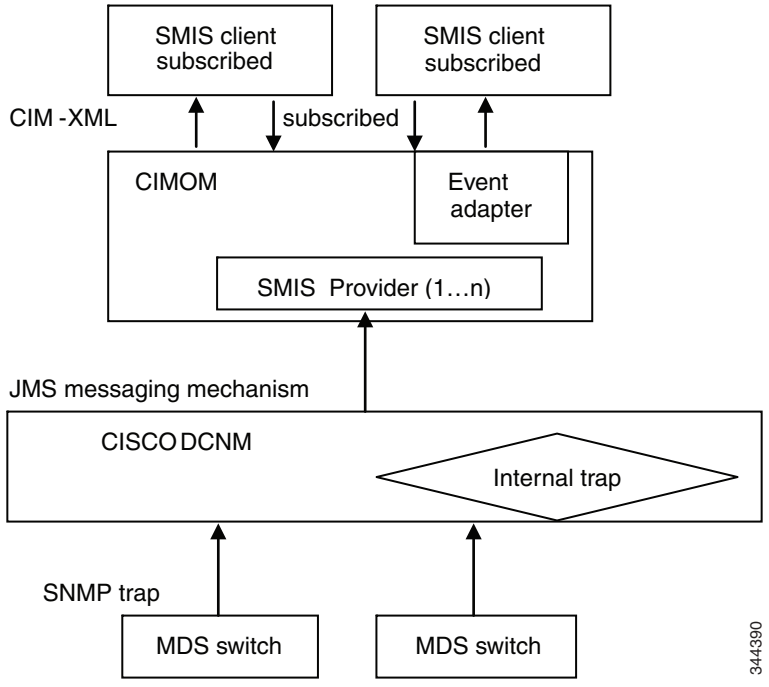


Figure 4-2 Event Handling Mechanism in WBEM Server



SNMP traps are sent asynchronously from the end devices like switches for changes in the ports, VSAN, and zone which are managed by CISCO DCNM. The SNMP traps are received and processed in the Cisco DCNM module. There can be internal traps generated in Cisco DCNM as a part of the

intercommunication between the FM modules. All the traps are sent to SMI-S module through JMS messaging mechanism. SMI-S provider receives the events and formats it as per indication provider schema and forwards it to the CIMOM. CIMOM sends the indication to the SMI-S client which are subscribed for interested indication.

Supported Indications in SMI-S Server

The supported events are either CIM_InstIndication or CIM_AlertIndication type.

- CIM_InstIndication—Describes events for creation, deletion, and modification of instances . These are also known as Life cycle events.

  The Subscription format of CIM_InstIndication is:

  wbemclient -s https://localhost/cimv2 -u username -p password -f "SELECT * from CIM_InstDeletion WHERE sourceInstance ISA CISCO_endport"

  The supported events are:

  - CISCO_PortAdded
  - CISCO_PortRemoved
  - CISCO_SwitchAdded
  - CISCO_SwitchRemoved

- CIM_AlertIndication—used for all external events.

  For example, Poweralert ,tempalert, link up, and link down.

  The subscription format of CIM_AlertIndication is:

  wbemclient -s https://localhost/cimv2 -u username -p password -f "SELECT * FROM CISCO_LinkDown"

  The supported events are:

  - CISCO_FanAlert
  - CISCO_LinkDown
  - CISCO_LinkUp
  - CISCO_PowerAlert
  - CISCO_TempAlert
  - CISCO_MediaFRUChanged
  - CISCO_MediaFRUInserted
  - CISCO_MediaFRURemoved
  - CISCO_ZoneAlert
  - CISCO_ZoneSetAlert
  - CISCO_NameServerDatabaseChange
  - CISCO_UserAdded
  - CISCO_UserLoginFailed
  - CISCO_UserModified
  - CISCO_UserRemoved
  - CISCO_VsanChanged

Sample SMI-S Java Client

This chapter describes the Sample SMI-S java client developed using JSR 48 specifications. JSR 48 is a set of java WBEM Service APIs and reference implementation for WBEM. WBEM is an initiative from the DMTF that unifies systems management and instrumentation.

The SMI-S java client uses the client and listener packages provided by the JSR 48 specifications, which consists of classes and interfaces for developing WBEM Clients and WBEM Event Listeners. The WBEMClient interface in the client package is used to invoke WBEM operations against a WBEM Server. A WBEMClient implementation can be retrieved from the WBEMClientFactory by specifying the protocol to be used. IndicationListener is implemented by the code that wants to create a listener for indications. The WBEMListener interface is used to add or remove WBEM Indication Listeners.

The SMI-S java client uses the Cisco DCNM SMI-S external agent for managing and monitoring Cisco DCNM for SAN.

You can perform the following tasks from the services provided by the SMI-S client tool:

- List Zonesets, Zones, and Zonemembers in VSAN
  - Obtain VSANs using enumerateInstanceNames API in WBEMClient.
  - Obtain Zonesets, Zones, and Zonemembers using associatorInstances API in WBEMClient on VSAN.
- List Zones in Zoneset
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get zonesets in selected VSAN using associatorInstances.
  - Get zones in selected zoneset.
- List Zonemembers in Zone
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zones in selected VSAN using associatorInstances.
  - Get Zonemembers in selected zone.
- Create new ZoneSet and Zones in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method createzoneset in Zoneservice class with zoneset and zone name.
- Add existing Zone to existing Zoneset in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.

  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method addzone in Zoneservice class with existing zoneset and zone information.
- Add Zonemember to existing Zone in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method CreateZoneMembershipSettingData in Zoneservice class with existing zone and new zonemember information.
- Activate and Deactivate a Zoneset in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method ActivateZoneSet in Zoneservice class with existing zoneset for activation or deactivating the zoneset.

Installing Sample SMI-S Client

Note Cisco SMI-S Client is packaged with DCNM SMI-S Server.

When you choose Cisco DCNM with SMI-S Sever during installation, SMI-S Client is also available. For more information, see the "Installing Cisco DCNM SMI-S Server" section on page 3-1.

The following are the SMI-S Client installation locations:"Installing Cisco DCNM SMI-S Server" section on page 3-1

- On Microsoft Windows, by default, Cisco DCNM is installed at C:\Program Files\Cisco Systems.
- On a UNIX (Solaris or Linux) machine, Cisco DCNM is installed at /usr/local/cisco/dcm or $HOME/dcm.
- The SMI-S Client is available at <*DCNM install dir*>dcm/smis/client.
- The batch files to compile and execute SMI-S Client and SMI-S Indication Client are available at <*DCNM install dir*>*/dcm/smis/client/bin*.
- The wbem jars are available at <*DCNM install dir*>/SmisClient/lib.
- The source files of SMI-S Client and SMI-S Indication Client are available at <*DCNM install dir*>/dcm/smis/client/src and the corresponding class files are available at <*DCNM install dir*>/dcm/smis/client/build.
- The configuration file used to retrieve Cisco DCNM credentials to access DCNM SMI-S agent are available at <*DCNM install dir*>/SmisClient/conf.
- You can choose the type of indication to be subscribed in the *filter.properties config* file.

Services Provided by SMI-S Java Client

The following NMS services are supported:

- List Switches
- List Hosts

- List Targets
- List VSANs in fabric
- List Switches in VSAN
- Enumerate the Instance of Any CIM class
- Enumerate the names of Any CIM class
- Execute Zone Operations

Table 5-1 provides the details of the SMI-S java client services.

Table 5-1 Services Provided by SMI-S Java Client

Services Name	Description
List Switches	This service lists the VSANs present in the fabric. You need to select the VSAN to see the list of switches. The tool uses the enumerateInstances API in Interface WBEMClient.
List Hosts	This service lists the hosts present in the fabric and also lists the endports present in the host. The tool uses the enumerateInstances API in Interface WBEMClient and an associatorInstances to get the host associated to the end ports.
List Target	This service lists the targets present in the fabric and also lists the end ports present in the target. The tool uses the enumerateInstances API in Interface WBEMClient and an associatorInstances to get the target associated to the end ports.
List VSANs in Fabric	This service lists all of the VSANs present in the fabric and their properties. For example, the status, WWN, VSAN ID, and temporary name of the VSAN. The tool uses the enumerateInstances API in the Interface WBEMClient.
List Switches in the VSAN	This service lists all the VSANs that are up and active in the fabric. Once you select it, the logical computer systems (logical switches) present in that VSAN are displayed. It also gives information of the physical switch each logical switch is hosted on.
Enumerate the Names of any CIM Class	This option allows you to select a CIM class of your choice to view CIMObjectPath names for a specified CIM class.
Enumerate the Instance of any CIM Class	This option allows you to select a CIM class of your choice to view instances of a specified CIM class.
Execute Zone Operations	This option lists the bunch of zone services supported by the Cisco SMI-S client tool.
Subscription of the Event Listener	SMI-S Indication client can be used to subscribe to events.

Examples of Developing SMI-S Client Using WBEM Solutions

The following example shows how to obtain WBEMclient with the specified protocol from WBEMClientFactory:

```
/*
            * WBEMClient handle
            */
            WBEMClient clientObj = null;

            /*
            * The host to connect to. In the form of a WBEM URL. Make sure the WBEM
            * Server is running before trying this example.
            */

            String host = "http://localhost:5988/cimv2";


            try {
                  /*
                  * Create an object path using the host variable.
                  */
                  CIMObjectPath cns = new CIMObjectPath(host);

                  /*
                  * Create the principal - used for authentication/authorization
                  */
                  UserPrincipal up = new UserPrincipal(username);

                  /*
                  * Create the credential - used for authentication/authorization
                  */
                  PasswordCredential pc = new PasswordCredential(password);

                  /*
                  * Add the principal and credential to the subject.
                  */
                  Subject s = new Subject();
                  s.getPrincipals().add(up);
                  s.getPrivateCredentials().add(pc);

                  /*
                  * Create a CIM client connection using the either CIM-XML or
                  * WS-Management protocol
                  */
                  clientObj = WBEMClientFactory.getClient("CIM-XML");
                  clientObj.initialize(cns, s, null);
            } catch (WBEMException e)
```

The following example shows using the enumerateInstanceNames API in WBEMClient:

```
// get wbem client obj with specified protocol
    WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

// pass the CIM class & name space

CIMObjectPath cop = new CIMObjectPath(className, namespace);
CloseableIterator<CIMObjectPath> ei = wbclient.enumerateInstanceNames(cop);
```

The following example shows using the enumerateInstances API in WBEMClient:

```
// get wbem client obj with specified protocol
```

```
    WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

// pass the CIM class
CloseableIterator<CIMInstance> ei = wbclient. enumerateInstances(className,null);


}
```

The following example shows how to get association names:

```
// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

CloseableIterator<CIMObjectPath> an =null;
    try{
            an = wbclient.associatorNames(cimop,null, resultClass, null, null);
        } catch (WBEMException e) {
            System.out.println(e);
        }
```

The following example shows how to get assoication Instances:

```
// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");
CloseableIterator<CIMInstance> a =null;
try{
a = wbclient.associatorInstances(cimop,null, resultClass, null, null, false, null);
    } catch (WBEMException e) {
        System.out.println(e);
    }
```

The following example shows how to use the invoke method and use the extrinsic methods of CIM Class:

```
.// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

UnsignedInteger32 uInt32=(UnsignedInteger32) wbclient.invokeMethod(CIM_classname,
methodname,inArgsAlias,outArgs);

Sample code for registering listener for indications:
class Listener implements IndicationListener {
        ……
        public void indicationOccured(String pIndicationURL, CIMInstance pIndication) {
            CIMInstance indicationInstance = pIndication;
            System.out.println("Received indication instance: " + indicationInstance);
        }
…}
```

The following example shows how to register listener indications:

```
//get the WBEMListener from the WBEMListenerFactory
WBEMListener listnr = WBEMListenerFactory.getListener("CIM-XML");
//add the listener to the port. use 0 to specify any available port
Int port = api.addListener(cl,0, "http");
// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");
//create the filter
```

```
String filter  ="SELECT SELECT * from CIM_InstIndication WHERE sourceInstance ISA
CISCO_PhysicalComputerSystem";
CIMObjectPath op = new CIMObjectPath("CIM_IndicationFilter", namespace);
        CIMClass filterClass = client.getClass(op, false, true, false, null);
        CIMInstance filterInstance = filterClass.newInstance();
        CIMProperty<?>[] fcpArray = {
        new CIMProperty<String>("Query", CIMDataType.STRING_T, filter, false, false,
null),
        new CIMProperty<String>("QueryLanguage", CIMDataType.STRING_T, "WQL", false,
false, null) };
        filterInstance = filterInstance.deriveInstance(fcpArray);
        CIMObjectPath filterOP = client.createInstance(filterInstance);

//create the listener
CIMObjectPath op = new CIMObjectPath("CIM_ListenerDestinationCIMXML", namespace);
        CIMProperty<?>[] cpa = { new CIMProperty<String>("Destination",
                CIMDataType.STRING_T, "http://" + systemName + ":"+ port +"/", false,
false, null) };

        // create new instance of listener
        CIMInstance listenerInstance = new CIMInstance(op, cpa);
        CIMObjectPath listenerOP = client.createInstance(listenerInstance);
//create a subscription, an association between the filter and listener.
CIMProperty<?>[] sicpArray = {
                            new CIMProperty<CIMObjectPath>("Filter", new CIMDataType(
                                    filterOP.getObjectName()), filterOP, true, false, null),
                            new CIMProperty<CIMObjectPath>("Handler", new CIMDataType(
                                    listenerOP.getObjectName()), listenerOP, true, false,
null) };
                    CIMInstance subscriptionInstance = new CIMInstance(
                            new CIMObjectPath("CIM_IndicationSubscription", namespace,
sicpArray), sicpArray);
                    try {
                        CIMObjectPath subscriptionOP =
client.createInstance(subscriptionInstance);

                    } catch (WBEMException e) {
                        throw e;
                    }
```

Managed Object Format Files for Cisco DCNM SMI-S Server

This chapter provides the text from the Managed Object Format (MOF) files for the Cisco DCNM SMI-S 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.

CISCO_ActiveConnection.mof

```
CISCO_ActiveConnection
[Association,
       Description (
           "This association defines a connection that is currently "
           "communicating, or is configured to communicate, between two "
           "ServiceAccessPoints i.e. two CISCO_ProtocolEndPoints." ),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ActiveConnectionProviderImpl")]
class CISCO_ActiveConnection : CIM_ActiveConnection
{
      [Override ( "Antecedent" ),
       Description (
          "A ServiceAccessPoint that is configured to communicate "
          "or is actively communicating with the Dependent SAP. In "
          "a unidirectional connection, this SAP is the one that is "
          "transmitting." )]
   CISCO_ProtocolEndPoint REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "A second ServiceAccessPoint that is configured to "
          "communicate or is actively communicating with the "
          "Antecedent SAP. In a unidirectional connection, this SAP "
          "is the one that is receiving the communication." )]
   CISCO_ProtocolEndPoint REF Dependent;

};
```

CISCO_AdminDomain.mof

```
CISCO_AdminDomain
    [Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_AdminDomainProviderImpl")]
class CISCO_AdminDomain : CIM_AdminDomain
```

```
{};
```

CISCO_AdminDomainConformsToFabricProfile.mof

```
[Association, Version ( "3.1.0" ), Description (
       "The SMISConformsToProfile association defines the "
       "RegisteredProfiles that are conformant with a specific "
       "verision of SIM-S. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_AdminDomainConformsToFabricProfile
ProviderImpl")
    ]
class CISCO_AdminDomainConformsToFabricProfile : CIM_ElementConformsToProfile {

      [Key, Override ( "ConformantStandard" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The RegisteredProfile to which the ManagedElement conforms.")]
   CISCO_FabricProfile REF ConformantStandard;

      [Key, Override ( "ManagedElement" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The ManagedElement that conforms to the RegisteredProfile.")]
   CISCO_AdminDomain REF ManagedElement;
};
```

CISCO_AlertIndication.mof

```
CISCO_AlertIndication
class CISCO_AlertIndication: CIM_AlertIndication {

    [Override("IndicationIdentifier"), Description (
        "Unique numeric identifier for indication.")]
    string IndicationIdentifier;

    [Override("IndicationTime"), Description (
        "The time and date of creation of the Indication. "
        "The property may be set to NULL if the entity creating the "
        "Indication is not capable of determining this information. "
        "Note that IndicationTime may be the same for two Indications "
        "that are generated in rapid succession.")]
    datetime IndicationTime;

    [Override("AlertType"), Description (
        "Primary classification of the Indication. "
        "The following values are defined: \n"
        "1 - Other. The Indication's OtherAlertType property conveys "
            "its classification. Use of \"Other\" in an enumeration is a "
            "standard CIM convention. It means that the current Indication "
            "does not fit into the categories described by this enumeration. \n"
        "2 - Communications Alert. An Indication of this type is "
            "principally associated with the procedures and/or processes "
            "required to convey information from one point to another. \n"
        "3 - Quality of Service Alert. An Indication of this type is "
            "principally associated with a degradation or errors in the "
            "performance or function of an entity. \n"
        "4 - Processing Error. An Indication of this type is "
            "principally associated with a software or processing fault. "
            "This is the AlertType generated for failures during "
            "enumerateInstanceNames or enumerateInstances. \n"
        "5 - Device Alert. An Indication of this type is principally "
            "associated with an equipment or hardware fault. \n"
```

```
    "6 - Environmental Alert. An Indication of this type is "
        "principally associated with a condition relating to an "
        "enclosure in which the hardware resides, or other "
        "environmental considerations. \n"
    "7 - Model Change. The Indication addresses changes in the "
        "Information Model. For example, it may embed a Lifecycle "
        "Indication to convey the specific model change being "
        "alerted. \n"
    "8 - Security Alert. An Indication of this type is associated "
        "with security violations, detection of viruses, user account changes "
        "and similar issues. \n"
        "The 'Description' field will describe the type of indication."),
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8"},
Values {"Other", "Communications Alert", "Quality of Service Alert",
    "Processing Error", "Device Alert", "Environmental Alert",
    "Model Change", "Security Alert"}]
uint16 AlertType;

[Override("OtherAlertType"), Description (
    "A string describing the Alert type - used when "
    "the AlertType property is set to 1, \"Other\". Other Alert types "
    "include notifications that are intended as state change "
    "notifications for fabric/switch/port/device/connection, etc."),
ModelCorrespondence {"CISCO_AlertIndication.AlertType"},
ValueMap {"Switch Alert", "Port Alert", "Fabric Alert", "Device Alert",
    "Zone Alert", "Platform Alert", "Firmware Alert", "Connection Alert",
    "HA Alert", "Agent Alert","Virtual Fabric Alert"}]
string OtherAlertType;

[Description (
    "A string describing the Alert subtype. "),
MappingStrings {"API.CISCO | Event | EventType"},
ModelCorrespondence {"CISCO_AlertIndication.OtherAlertType"},
ValueMap {"New Login", "Failed Login", "Logout", "Config Changed",
    "Track Change On", "Track Change Off", "Up", "Down",
    "Fabric Changed", "Connected Area Online", "Connected Area Offline",
    "Connected Area State Unknown", "Connected Port Online",
    "Connected Port Offline", "Connected Port State Unknown",
    "Database Merge Completed", "Database Change Completed",
    "Config Enabled", "Config Disabled", "Config Saved",
    "Config Committed", "Session Aborted", "Session Status Changed",
    "Registered", "Deregistered", "State Changed", "Firmware Download Started",
    "Firmware Download Completed", "Connection Merged", "Connection Deactivated",
    "Connection Reactivated", "Failover Completed",
    "Persistence Failure", "Initialization Failure",
    "RPC Handles Initialization Failure","Event Registration Failure",
    "SMIAgent Config Update Failure","Account Locked Out","Account Added",
    "Account Deleted","Account Role Changed","Account Membership Changed",
    "Password Expiring","Password Expired"}]
string AlertSubType;

[Description (
    "The identifying information for the admin "
    "domain (Fabric) for which this indication is generated. "
    "The property is the path of the Fabric instance encoded "
    "as a string. The entity within this domain for which the "
    "indication is generated is the called the AlertingManagedElement.")]
string AlertingAdminDomain;

[Description (
    "The identifying information of the entity ")]
string AlertingManagedElements[];

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

```
            "The severity of the event. "
            "The values are: "
            "1 - Other, by CIM convention, is used to indicate that the "
                "Severity's value can be found in the OtherSeverity property. \n"
            "3 - Degraded/Warning should be used when its appropriate to let "
                "the user decide if action is needed.  \n"
            "4 - Minor should be used to indicate action is needed, but the "
                "situation is not serious at this time.  \n"
            "5 - Major should be used to indicate action is needed NOW. \n"
            "6 - Critical should be used to indicate action is needed NOW "
                "and the scope is broad (perhaps an imminent outage to a "
                "critical resource will result). \n"
            "7 - Fatal/NonRecoverable should be used to indicate an error "
                "occurred, but it's too late to take remedial action. \n"
            "2 and 0 - Information and Unknown (respectively) follow common "
                "usage. Literally, the AlertIndication is purely informational "
                "or its severity is simply unknown."),
        Values {"Unknown", "Other", "Information", "Degraded/Warning",
            "Minor", "Major", "Critical", "Fatal/NonRecoverable"},
        ValueMap{"0", "1", "2", "3", "4", "5", "6", "7"}]
        uint16 PerceivedSeverity = 0;

        [Override("ProbableCause"), Description (
            "An enumerated value that describes the probable cause "
            "of the situation which resulted in the AlertIndication."),
        Values {"Unknown", "Other"},
        ValueMap{"0", "1"}]
        uint16 ProbableCause = 0;

        [Description (
            "The debug level of this event."),
        Values {"Level_0", "Level_1", "Level_2", "Level_3", "Level_4",
            "Level_5", "Level_6", "Level_7", "Level_8", "Level_9"},
        ValueMap{"0", "1", "2", "3", "4", "5", "6", "7", "8", "9"}]
        uint16 DebugLevel = 0;

        [Override("Description"), Description (
            "Textual (ASCII) description of indication.")]
        string Description;

};
```

CISCO_Component.mof

```
CISCO_Component
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_Component : CIM_Component
{};
```

CISCO_ComputerSystem.mof

```
CISCO_ComputerSystem
[Abstract,
       Description ("This is an abstract class." )]
class CISCO_ComputerSystem : CIM_ComputerSystem
{};
```

CISCO_ComputerSystemPackage.mof

```
CISCO_ComputerSystemPackage
[Association,
       Description("ComputerSystem may be realized realized in "
           "one or more PhysicalPackages. The ComputerSystemPackage "
           "association explicitly defines this relationship."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ComputerSystemPackageProviderImpl"
)]
class CISCO_ComputerSystemPackage : CIM_ComputerSystemPackage
{
      [Override ( "Antecedent" ),
       Description (
          "The PhysicalPackage(s) that realize a Unitary ComputerSystem.")]
   CISCO_PhysicalPackage REF Antecedent;

      [Override ( "Dependent" ),
       Description ( "The UnitaryComputerSystem." )]
   CISCO_PhysicalComputerSystem REF Dependent;

};
```

CISCO_ComputerSystemRemoteService.mof

```
CISCO_ComputerSystemRemoteService
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ComputerSystemRemoteServiceProvide
rImpl")]
class CISCO_ComputerSystemRemoteService : CISCO_HostedAccessPoint
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The hosting System." )]
   CISCO_PhysicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Weak, Description (
          "The SAPs that are hosted on this System." )]
   CISCO_RemoteServiceAccessPoint REF Dependent;
};
```

CISCO_ConnectivityCollection.mof

```
CISCO_ConnectivityCollection
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ConnectivityCollectionProviderImp
l")]
class CISCO_ConnectivityCollection : CIM_ConnectivityCollection
{};
```

CISCO_ConnectivityCollectionInVsan.mof

```
CISCO_ConnectivityCollectionInVsan
[Association,
```

```
Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ConnectivityCollectionInVsanProvid
erImpl")]
class CISCO_ConnectivityCollectionInVsan : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_Vsan REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_ConnectivityCollection REF Dependent;
};
```

CISCO_ConnectivityMemberOfCollection.mof

```
CISCO_ConnectivityMemberOfCollection
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ConnectivityMemberOfCollectionProv
iderImpl")]
class CISCO_ConnectivityMemberOfCollection : CIM_MemberOfCollection
{
      [Override ( "Collection" ), Min ( 1 ), Description (
          "Collection representing Connectivity.")]
    CISCO_ConnectivityCollection REF Collection;

      [Override ( "Member" ), Min ( 1 ), Description (
          "The protocol endpoints that are members of the connectivity collection.")]
    CISCO_ProtocolEndPoint REF Member;

};
```

CISCO_ContainedDomain.mof

```
CISCO_ContainedDomain
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ContainedDomainProviderImpl")]
class CISCO_ContainedDomain : CIM_ContainedDomain
{
      [Aggregate, Override ( "GroupComponent" ),
       Description (
          "An AdminDomain that aggregates other AdminDomains." )]
   CISCO_AdminDomain REF GroupComponent;

      [Override ( "PartComponent" ),
       Description (
          "An AdminDomain aggregated by another AdminDomain." )]
   CISCO_Vsan REF PartComponent;
};
```

CISCO_CopyRunning.mof

```
CISCO_CopyRunning
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_CopyRunningProviderImpl")]
class CISCO_CopyRunning:CIM_Service
```

```
{
    uint32 Execute();
};
```

CISCO_DeviceAlias.mof

```
CISCO_DeviceAlias
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_DeviceAliasProviderImpl")]
class CISCO_DeviceAlias : CIM_NamedAddressCollection
{
    string name;
    string pwwn;
};
```

CISCO_DeviceSAPImplementation.mof

```
CISCO_DeviceSAPImplementation
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_DeviceSAPImplementation : CIM_DeviceSAPImplementation
{};
```

CISCO_ElementCapabilities.mof

```
CISCO_ElementCapabilities
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_ElementCapabilities : CIM_ElementCapabilities
{};
```

CISCO_ElementSettingData.mof

```
CISCO_ElementSettingData
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_ElementSettingData : CIM_ElementSettingData
{};
```

CISCO_ElementSoftwareIdentity.mof

```
[Association, Version("3.1.0"),
   Provider("jsr48:com.wbemsolutions.wbem.cimom."
               "GenericReadOnlyProvider")
 ]
class CISCO_ElementSoftwareIdentity : WBEMSolutions_ElementSoftwareIdentity {

      [Override ( "Antecedent" ), Description (
          "A LogicalElement's Software Asset.")]
   CISCO_ServerSoftware REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The ManagedElement that requires or uses the software.")]
   CISCO_RegisteredProfile REF Dependent;
```

```
};
```

CISCO_ElementStatisticalData.mof

```
CISCO_ElementStatisticalData
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_ElementStatisticalData : CIM_ElementStatisticalData
{};
```

CISCO_EndPort.mof

```
CISCO_EndPort
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EndPortProviderImpl")]
class CISCO_EndPort : 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;

    [Description (
        "IP Address of the actual node.")]
    string NodeIpAddress;

    [Experimental, Description (
          "The availability of the port for client to "
          "determine whether the port can be made operational.  The "
          "values: \n"
          "\"Available\" indicates that the port can be made operational, \n"
          "\"Not Installed\" indicates some aspect of the port has not been "
          "installed preventing it from being operational but is discoverable through "
          "instrumentation, \n"
          "\"No Transceiver\" indicates that the transceiver is "
          "not installed to allow the port to become operational, "
         "\"Incompatible Transceiver\" indicates the installed transceiver is not correct
and is preventing "
          "the port from being operational, \n"
          "\"Not Licensed\" indicates that the port "
          "cannot be made operational due to a license not existing for the port."),
       ValueMap { "0", "1", "2", "3", "4", "5", "6" },
       Values { "Unknown", "Available", "Not Installed", "No Transceiver",
          "Incompatible Transceiver", "Not Licensed", "DMTF Reserved" }]
    uint16 PortAvailability = 2;
};
```

CISCO_EndPortControlledByPortController.mof

```
CISCO_EndPortControlledByPortController
[Association,
       Description ("This association represents the relationship between a "
           "device and ports."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EndPortControlledByPortControllerP
roviderImpl")]
class CISCO_EndPortControlledByPortController: CIM_ControlledBy {
    [Override ("Antecedent"), Description (
        "The device that controls the port.")]
    CISCO_PortController REF Antecedent;

    [Override ("Dependent"), Description (
        "The port being controlled.")]
    CISCO_EndPort 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."),
        MappingStrings {"FC-GS-4 | FDMI | OS Device Name"}]
    string DeviceNumber;
};
```

CISCO_EndPortSAPImplementation.mof

```
CISCO_EndPortSAPImplementation
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EndPortSAPImplementationProviderIm
pl")]
class CISCO_EndPortSAPImplementation : CISCO_DeviceSAPImplementation
{
      [Override ( "Antecedent" ),
       Description ( "The LogicalDevice." )]
   CISCO_EndPort REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The ServiceAccessPoint implemented using the LogicalDevice."
           )]
   CISCO_ProtocolEndPoint REF Dependent;

};
```

CISCO_EndPortsInHostComputerSystem.mof

```
CISCO_EndPortsInHostComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EndPortsInHostComputerSystemProvid
erImpl")]
class CISCO_EndPortsInHostComputerSystem : CISCO_SystemDevice
{
      [Aggregate, Override ( "GroupComponent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The parent system in the Association." )]
```

```
   CISCO_HostComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ),
       Weak, Description (
          "The LogicalDevice that is a component of a System." )]
   CISCO_EndPort REF PartComponent;

};
```

CISCO_EndPortsInStorageComputerSystem.mof

```
CISCO_EndPortsInStorageComputerSystem
    [Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EndPortsInStorageComputerSystemPro
viderImpl")]
class CISCO_EndPortsInStorageComputerSystem : CISCO_SystemDevice
{
      [Aggregate, Override ( "GroupComponent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The parent system in the Association." )]
   CISCO_StorageComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ),
       Weak, Description (
          "The LogicalDevice that is a component of a System." )]
   CISCO_EndPort REF PartComponent;

};
```

CISCO_EnvironmentalAlert.mof

```
CISCO_EnvironmentalAlert
[Abstract,
       Indication]
class CISCO_EnvironmentalAlert: CISCO_AlertIndication
{
    string EnvAlertDescription;
        uint32  PhysicalIndex;
    uint32  OperationalStatus;
};
```

CISCO_EthernetPort.mof

```
CISCO_EthernetPort
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EthernetPortProviderImpl")]
class CISCO_EthernetPort : CIM_EthernetPort
{};
```

CISCO_EthernetPortProtocolEndpoint.mof

```
CISCO_EthernetPortProtocolEndpoint
[Association,
       Description ("CISCO_EthernetPortProtocolEndpoint associates a "
           "CISCO_EthernetPort with CISCO_LANEndpoint . "),
```

```
Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EthernetPortProtocolEndpointProvid
erImpl")]
class CISCO_EthernetPortProtocolEndpoint : CIM_DeviceSAPImplementation {

    [Override ( "Antecedent" ), Description (
        "The EthernetPort that represents the Device behind the "
        "ProtocolEndpoint.")]
    CISCO_EthernetPort REF Antecedent;

    [Override ( "Dependent" ), Description (
        "The LANEndpoint implemented on the LogicalPort.")]
    CISCO_LANEndpoint REF Dependent;
};
```

CISCO_EthernetPortsInPhysicalComputerSystem.mof

```
CISCO_EthernetPortsInPhysicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EthernetPortsInPhysicalComputerSys
temProviderImpl")]
class CISCO_EthernetPortsInPhysicalComputerSystem : CISCO_SystemDevice
{
      [Aggregate, Override ( "GroupComponent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The parent system in the Association." )]
   CISCO_PhysicalComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ),
       Weak, Description (
          "The LogicalDevice that is a component of a System." )]
   CISCO_EthernetPort REF PartComponent;

};
```

CISCO_EthernetPortStatisticalData.mof

```
CISCO_EthernetPortStatisticalData
[Association,
      Description (
                "CISCO_IPEndPointStatistics is an association that associates "
                "CISCO_IPProtocolEndPoint to its StatisticalData "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EthernetPortStatisticalDataProvide
rImpl")]
class CISCO_EthernetPortStatisticalData :  CISCO_ElementStatisticalData {

        [Override ("ManagedElement"), Description (
                "Reference to CISCO_EthernetPort instance.")]
         CISCO_EthernetPort REF ManagedElement;

         [Override("Stats"), Key, Description (
                "The statistic information.")]
         CISCO_EthernetPortStatistics REF Stats;
};
```

CISCO_EthernetPortStatistics.mof

```
CISCO_EthernetPortStatistics
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_EthernetPortStatisticsProviderImp
l")]
class CISCO_EthernetPortStatistics : CIM_EthernetPortStatistics {

     [Override("InstanceID"), Key, Description (
         "Within the scope of the instantiating Namespace, InstanceID "
          "opaquely and uniquely identifies an instance of this class. ")]
   string InstanceID;

      [Required, Override ( "ElementName" ), Description (
          "The user friendly name for this instance of "
         "StatisticalData. In addition, the user friendly name can be "
         "used as a index property for a search of query. (Note: Name "
          "does not have to be unique within a namespace.)")]
   string ElementName ="Undefined";

    [Override ( "PacketsTransmitted" ), Description (
          "The total number of packets transmitted. This is "
       "calculated as the sum of the packets transmitted "
       "through each EPort associated to the GigEPort. "), Counter,
      MappingStrings { "MIF.DMTF|Network Adapter 802 Port|001.6" }]
   uint64 PacketsTransmitted;

     [Override ( "PacketsReceived" ), Description (
          "The total number of packets received. This is "
       "calculated as the sum of the packets received "
       "by each EPort associated to the GigEPort. "), Counter,
      MappingStrings { "MIF.DMTF|Network Adapter 802 Port|001.8" }]
   uint64 PacketsReceived;

    [Override ( "SymbolErrors" ), Description (
          "The number of times there was an invalid data symbol when a "
         "valid carrier was present. The count is incremented at most "
         "once per carrier event, even if multiple symbol errors "
          "occur during the carrier event."), Counter,
      MappingStrings { "MIB.IETF|EtherLike-MIB.dot3StatsSymbolErrors"  }]
   uint32 SymbolErrors;

    [Override ( "AlignmentErrors" ), Description (
         "A count of frames received on a particular interface that "
          "are not an integral number of octets in length and do not "
          "pass the FCS check. The count represented by an instance of "
         "this object is incremented when the alignment error status "
       "is returned by the MAC layer to the LLC (or other MAC "
       "user). Received frames for which multiple error conditions "
        "obtain are, according to the conventions of IEEE 802.3 "
        "Layer Management, counted exclusively according to the "
        "error status presented to the LLC."), Counter,
        MappingStrings {"MIB.IETF|EtherLike-MIB.dot3StatsAlignmentErrors" }]
   uint32 AlignmentErrors;

    [Override ( "CarrierSenseErrors" ),
    Description (
         "The number of times that the carrier sense condition was "
          "lost or never asserted when attempting to transmit a frame "
          "on a particular interface. The count represented by an "
          "instance of this object is incremented at most once per "
          "transmission attempt, even if the carrier sense condition "
         "fluctuates during a transmission attempt."), Counter,
      MappingStrings {"MIB.IETF|EtherLike-MIB.dot3StatsCarrierSenseErrors" }]
```

```
uint32 CarrierSenseErrors;

 [Override ( "FrameTooLongs" ),
 Description (
      "A count of frames received on a particular interface that "
       "exceed the maximum permitted frame size. The count "
       "represented by an instance of this object is incremented "
       "when the FrameTooLong status is returned by the MAC layer "
      "to the LLC (or other MAC user). Received frames for which "
       "multiple error conditions obtain are, according to the "
       "conventions of IEEE 802.3 Layer Management, counted "
      "exclusively according to the error status presented to the "
       "LLC."), Counter,
    MappingStrings { "MIB.IETF|EtherLike-MIB.dot3StatsFrameTooLongs"    }]
uint32 FrameTooLongs;

 [Override ( "BytesTransmitted" ), Description (
       "The total number of bytes transmitted, including framing "
       "characters."), Units ( "Bytes" ), Counter,
    MappingStrings { "MIB.IETF|MIB-II.ifOutOctets",
       "MIF.DMTF|Network Adapter 802 Port|001.7" }]
uint64 BytesTransmitted;

   [Override ( "BytesReceived" ), Description (
       "The total number of bytes received, including framing "
       "characters."), Units ( "Bytes" ), Counter,
    MappingStrings { "MIB.IETF|MIB-II.ifInOctets",
       "MIF.DMTF|Network Adapter 802 Port|001.9" }]
uint64 BytesReceived;

 [Description (
       "The total number of unicast frames transmitted. " )]
uint64 UnicastFramesTransmitted;

 [Description (
       "The total number of unicast frames received. " )]
uint64 UnicastFramesReceived;

 [Description (
       "The total number of multicast frames transmitted. " )]
uint64 MulticastFramesTransmitted;

 [Description (
       "The total number of multicast frames received. " )]
uint64 MulticastFramesReceived;

 [Description (
       "The total number of broadcast frames transmitted. " )]
uint64 BroadcastFramesTransmitted;

  [Description (
       "The total number of broadcast frames received. " )]
uint64 BroadcastFramesReceived;

 [Description (
       "The total number of pause frames transmitted. " )]
uint64 PauseFramesTransmitted;

 [Description (
      "The total number of pause frames received. " )]
uint64 PauseFramesReceived;

 [Description (
      "A count of frames received with less than allowed minimum "
```

```
        "frame length (64 bytes) and have CRC errors(Runt)." )]
    uint32 FrameTooShorts;

     [Description (
          "A count of frames aborted because of excessive or "
        "late collisions. " )]
    uint32 Collisions;

     [Description (
          "A count of frames discarded because they are abruptly cut short "
        "and miss valid CRC. " )]
    uint32 FrameAborts;

     [Description (
        "A count of frames which are dropped because of lack of "
        "receive buffer. " )]
    uint32 Overruns;

     [Description (
           "A count of frames which are dropped because of "
        "FIFO overflow. " )]
    uint32 FIFOOverflow;

     [Description (
           "Number of compressed bytes through the GigEPort. This is "
        "calculated as the sum of the compressed bytes transmitted "
        "through each EPort associated to this GigEPort. ")]
    uint64 CompressedBytes;

     [Description (
           "Number of uncompressed bytes through the GigEPort. This is "
        "calculated as the sum of the uncompressed byes transmitted "
        "through each EPort associated to this GigEPort. ")]
    uint64 UncompressedBytes;

     [Description (
           "Number of IOAccelerated bytes.")]
    uint64 IOAccelerated;
};
```

CISCO_FabricProfile.mof

```
[Version ( "3.1.0" ), Description (
       "A RegisteredProfile describes a set of CIM Schema classes with "
       "required properties and/or methods, necessary to manage a "
       "real-world entity or to support a usage scenario, in an "
       "interoperable fashion. RegisteredProfiles can be defined by "
       "the DMTF or other standards organizations. Note that this "
       "class should not be confused with CIM_Profile, which collects "
       "SettingData instances, to be applied as a 'configuration "
       "profile' for an element. \n"
       "A RegisteredProfile is a named 'standard' for CIM-based "
       "management of a particular System, subsystem, Service or other "
       "entity, for a specified set of uses. It is a complete, "
       "standalone definition, as opposed to the subclass "
       "RegisteredSubProfile, which requires a scoping profile for "
       "context. \n"
       "The uses for a RegisteredProfile or SubProfile MUST be "
       "specified in the document that defines the profile. Examples "
       "of Profiles are to manage various aspects of an Operating "
       "System, Storage Array, or Database. The name of the profile is "
       "defined and scoped by its authoring organization."),
```

```
    Provider("jsr48:com.wbemsolutions.wbem.cimom."
               "GenericReadOnlyProvider")
   ]
class CISCO_FabricProfile : CISCO_RegisteredProfile {
};
```

CISCO_FabricService.mof

```
CISCO_FabricService
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FabricServiceProviderImpl")]
class CISCO_FabricService : CIM_Service {

    uint32 ZoneSetDistribute (
            [Required, IN, Description (
                "A reference to the ZoneSet to be activated.")]
            CISCO_VSAN ref VSAN);
};
```

CISCO_FabricServiceInAdminDomain.mof

```
CISCO_FabricServiceInAdminDomain
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FabricServiceInAdminDomainProvider
Impl")]
class CISCO_FabricServiceInAdminDomain : CISCO_HostedService
{       [Override ( "Antecedent" ),
        Min ( 1 ),
        Max ( 1 ),
        Description ( "The hosting System." )]
   CISCO_AdminDomain REF Antecedent;

       [Override ( "Dependent" ),
        Weak, Description ( "The Service hosted on the System." )]
   CISCO_FabricService REF Dependent;

};
```

CISCO_FabricServiceInVsan.mof

```
CISCO_FabricServiceInVsan
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FabricServiceInVsanProviderImpl")]
class CISCO_FabricServiceInVsan : CISCO_HostedService
{       [Override ( "Antecedent" ),
        Min ( 1 ),
        Max ( 1 ),
        Description ( "The hosting System." )]
   CISCO_Vsan REF Antecedent;

       [Override ( "Dependent" ),
        Weak, Description ( "The Service hosted on the System." )]
   CISCO_FabricService REF Dependent;

};
```

CISCO_FanAlert.mof

```
CISCO_FanAlert
[Indication,
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FanAlertProviderImpl")]
class CISCO_FanAlert: CISCO_EnvironmentalAlert
{};
```

CISCO_FCIPElementSettingData.mof

```
CISCO_FCIPElementSettingData
[Association,
        Description ("This corresponds to the association between
CISCO_FCIPProtocolEndpoint "
                "and CISCO_FCIPSettings. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCIPElementSettingDataProviderImpl
")]
class CISCO_FCIPElementSettingData : CISCO_ElementSettingData
{
        [Override ("ManagedElement"), Description (
                "Reference to CISCO_FCIPProtocolEndpoint instance.")]
        CISCO_FCIPProtocolEndpoint REF ManagedElement;

        [Override ("SettingData"), Description (
                "Reference to CISCO_FCIPSettings instance.")]
        CISCO_FCIPSettings REF SettingData;
};
```

CISCO_FCIPPEBasedOn.mof

```
CISCO_FCIPPEBasedOn
[Association, Aggregation,
        Description ("CISCO_FCIPPEBasedOn is an association used to establish "
                "membership relationships between a fcipprofile and the fcip protocol
endpoints"
                "within that switch. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCIPPEBasedOnProviderImpl")]
class CISCO_FCIPPEBasedOn: CISCO_Component {

        [Override("GroupComponent"), Description (
                "The switch that has contained Ethernet ports.")]
        CISCO_FCIPProfile REF GroupComponent;

      [Override("PartComponent"), Description (
               "The FCIP ProtocolEndPoint in this switch.")]
       CISCO_FCIPProtocolEndPoint REF PartComponent;
};
```

CISCO_FCIPProfile.mof

```
CISCO_FCIPProfile
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCIPProfileProviderImpl")]
class CISCO_FCIPProfile : CIM_Profile
{

    [Description ("The type of Internet address by which the entity is reachable. " )]
    uint32  FcipEntityAddressType;
```

```
    [Description (
            "The Internet address for this entity. "
             )]
    string FcipEntityAddress;

    [Description (
    "A TCP port other than the FCIP Well-Known port on which the"
    " FCIP entity listens for new TCP connection requests. It "
    " contains the value zero (0) if the FCIP Entity only listens on "
    " the Well-Known port. "
    )]

    uint32 FcipEntityTcpConnPort;

    [Description (
    "An indication of whether the TCP Selective Acknowledgement"
    " Option is enabled to allow the receiver end to acknowledge "
    " multiple lost packets in a singel ACK, enabling faster"
    " recovery."
    "enabled(1) - acknowledge option is enabled."
    " disabled(2) - acknowledge option is disabled. "
    )]

    boolean  FcipEntitySACKOption;

    [Description (
            "An indication of whether the FCIP Entity supports the "
            "protection against sequence number wrap. "
            "If true(1), the FCIP Entity supports protection against "
            "sequence number wrap. If false(2), the FCIP Entity does not "
            "support protection against sequence number wrap. "
    )]

    boolean FcipEntitySeqNumWrap;

    [Description (
       "An indication of whether the FCIP Entity supports PHB IP QoS. "
    )]

    boolean FcipEntityPHBSupport;
};
```

CISCO_FCIPProtocolEndpoint.mof

```
CISCO_FCIPProtocolEndpoint
[Description ("A communication point from which data may be "
            "sent or received. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCIPProtocolEndpointProviderImpl")
]
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;

         [Override("ProtocolIFType"), Description (
                "ProtocolIFType's enumeration is limited to IP-related and "
                "reserved values for this subclass of ProtocolEndpoint."),
                ValueMap { "1", "56", "222..4095", "4096", "4097", "4098",
                        "4116..32767", "32768.." },
                Values { "Other","Fibre Channel", "IANA Reserved", "IPv4", "IPv6",
"IPv4/v6",
                        "DMTF Reserved", "Vendor Reserved" }]
        uint16 ProtocolIFType = 56;

};
```

CISCO_FCIPSettings.mof

```
CISCO_FCIPSettings
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCIPSettingsProviderImpl")]
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'. "
//  )]

    uint32 KeepAliveTimeout;
    uint32 SpecialFrameTimeout;
    uint16 ConnectionUsageFlags;
};
```

CISCO_FCIPTCPEndpoint.mof

```
CISCO_FCIPTCPEndpoint
[Association,
        Description ("This corresponds to the association between
CISCO_FCIPProtocolEndpoint "
            "and CISCO_TCPProtocolEndPoint."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCIPTCPEndpointProviderImpl")]
class CISCO_FCIPTCPEndpoint : CIM_BindsTo {
    [Override ( "Antecedent" ), Description (
        "The underlying TCPEndpoint, which is depended upon.")]
    CISCO_TCPProtocolEndPoint REF Antecedent;

    [Override ( "Dependent" ), Description (
        "The FCIP ProtocolEndpoint dependent on the TCP Endpoint.")]
    CISCO_FCIPProtocolEndpoint REF Dependent;
};
```

CISCO_FCLogicalSwitchCapabilities.mof

```
CISCO_FCLogicalSwitchCapabilities
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCLogicalSwitchCapabilitiesProvid
erImpl")]
class CISCO_FCLogicalSwitchCapabilities : CIM_FCSwitchCapabilities
{};
```

CISCO_FCLogicalSwitchSettings.mof

```
CISCO_FCLogicalSwitchSettings
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCLogicalSwitchSettingsProviderIm
pl")]
class CISCO_FCLogicalSwitchSettings : CIM_FCSwitchSettings
{};
```

CISCO_FCNodeMemberOfCollection.mof

```
CISCO_FCNodeMemberOfCollection
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCNodeMemberOfCollectionProviderIm
pl")]
class CISCO_FCNodeMemberOfCollection : CIM_MemberOfCollection
{
      [Override ( "Collection" ), Min ( 1 ), Description (
          "Collection representing Connectivity.")]
    CISCO_LogicalPortGroup REF Collection;

      [Override ( "Member" ), Min ( 1 ), Description (
          "The protocol endpoints that are members of the connectivity collection.")]
    CISCO_EndPort REF Member;
};
```

CISCO_FCPort.mof

```
CISCO_FCPort
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortProviderImpl")]
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", "19", "16004", "16010", "16011", "16012", "16000..65535"},
       Values { "Unknown", "Other", "N", "NL", "F/NL", "Nx", "E", "F",
          "FL", "B", "G", "NP", "PortChannel", "FCIP", "ISCSI-F", "ISCSI-N", "Vendor
Reserved"} ]
    uint16 PortType;

    [Description (
        "IP Address of the actual node.")]
    string NodeIpAddress;

    [Experimental, Description (
          "The availability of the port for client to "
          "determine whether the port can be made operational.  The "
          "values: \n"
          "\"Available\" indicates that the port can be made operational, \n"
          "\"Not Installed\" indicates some aspect of the port has not been "
          "installed preventing it from being operational but is discoverable through "
```

```
          "instrumentation, \n"
          "\"No Transceiver\" indicates that the transceiver is "
          "not installed to allow the port to become operational, "
         "\"Incompatible Transceiver\" indicates the installed transceiver is not correct
and is preventing "
          "the port from being operational, \n"
          "\"Not Licensed\" indicates that the port "
          "cannot be made operational due to a license not existing for the port."),
       ValueMap { "0", "1", "2", "3", "4", "5", "6" },
       Values { "Unknown", "Available", "Not Installed", "No Transceiver",
          "Incompatible Transceiver", "Not Licensed", "DMTF Reserved" }]
    uint16 PortAvailability = 2;
};
```

CISCO_FCPortCapabilities.mof

```
CISCO_FCPortCapabilities
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortCapabilitiesProviderImpl")]
class CISCO_FCPortCapabilities : CIM_FCPortCapabilities
{};
```

CISCO_FCPortElementCapabilities.mof

```
CISCO_FCPortElementCapabilities
[Association,
       Description("ElementCapabilities represents the association between "
        "ManagedElements and their Capabilities."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortElementCapabilitiesProviderI
mpl")]
class CISCO_FCPortElementCapabilities : CISCO_ElementCapabilities
{
      [Override ( "ManagedElement" ), Key, Min ( 1 ),
       Max ( 1 ),
       Description ( "The managed element." )]
   CISCO_FCPort REF ManagedElement;

      [Override ( "Capabilities" ), Key, Description (
          "The Capabilities object associated with the element." )]
   CISCO_FCPortCapabilities REF Capabilities;
};
```

CISCO_FCPortProtocolEndPoint.mof

```
CISCO_FCPortProtocolEndPoint
[Association,
        Description ("CISCO_FcPortProtocolEndpoint associates a "
                "CISCO_FcPort with CISCO_LANEndpoint . "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortProtocolEndPointProviderImpl
")]
class CISCO_FCPortProtocolEndPoint : CIM_DeviceSAPImplementation {

        [Override ( "Antecedent" ), Description (
                "The FcPort that represents the Device behind the "
                "ProtocolEndpoint.")]
        CISCO_FCPort REF Antecedent;

        [Override ( "Dependent" ), Description (
                "The FCIPProtocolEndpoint implemented on the LogicalPort.")]
```

```
        CISCO_FCIPProtocolEndpoint REF Dependent;
};
```

CISCO_FCPortSAPImplementation.mof

```
CISCO_FCPortSAPImplementation
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortSAPImplementationProviderImp
l")]
class CISCO_FCPortSAPImplementation : CISCO_DeviceSAPImplementation
{
      [Override ( "Antecedent" ),
       Description ( "The LogicalDevice." )]
   Cisco_LogicalFcPort  REF Antecedent;

      [Override ( "Dependent" ),
       Description (
           "The ServiceAccessPoint implemented using the LogicalDevice."
            )]
   CISCO_ProtocolEndPoint REF Dependent;

};
```

CISCO_FCPortSettingData.mof

```
CISCO_FCPortSettingData
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortSettingDataProviderImpl")]
class CISCO_FCPortSettingData : CISCO_ElementSettingData
{

      [Override ( "ManagedElement" ), Key,
      Description ( "The managed element." )]
   CISCO_FCPort REF ManagedElement;

      [Override ( "SettingData" ), Key, Description (
          "The SettingData object associated with the element." )]
   CISCO_FCPortSettings REF SettingData;
};
```

CISCO_FCPortSettings.mof

```
CISCO_FCPortSettings
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortSettingsProviderImpl")]
class CISCO_FCPortSettings : CIM_FCPortSettings
{};
```

CISCO_FCPortsInLogicalComputerSystem.mof

```
CISCO_FCPortsInLogicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortsInLogicalComputerSystemProv
iderImpl")]
class CISCO_FCPortsInLogicalComputerSystem : CISCO_SystemDevice
{
      [Aggregate, Override ( "GroupComponent" ),
```

```
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The parent system in the Association." )]
   CISCO_LogicalComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ),
       Weak, Description (
          "The LogicalDevice that is a component of a System." )]
   CISCO_LogicalFCPort REF PartComponent;

};
```

CISCO_FCPortsInPhysicalComputerSystem.mof

```
CISCO_FCPortsInPhysicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortsInPhysicalComputerSystemPro
viderImpl")]
class CISCO_FCPortsInPhysicalComputerSystem : CISCO_SystemDevice
{
      [Aggregate, Override ( "GroupComponent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The parent system in the Association." )]
   CISCO_PhysicalComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ),
       Weak, Description (
          "The LogicalDevice that is a component of a System." )]
   CISCO_FCPort REF PartComponent;

};
```

CISCO_FCPortsInPortChannel.mof

```
CISCO_FCPortsInPortChannel
[Association,
        Description ("This corresponds to the association between CISCO_PortChannel"
                "and CISCO_FCPort. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortsInPortChannelProviderImpl")
]
class CISCO_FCPortsInPortChannel : CIM_MemberOfCollection
{
      [Override ( "Collection" ), Min ( 1 ), Description (
          "PortChannel.")]
    CISCO_PortChannel REF Collection;

      [Override ( "Member" ), Min ( 1 ), Max ( 8 ), Description (
          "The ports that are members of the port channel.")]
    CISCO_FCPort REF Member;
};
```

CISCO_FCPortStatisticalData.mof

```
CISCO_FCPortStatisticalData
[Association,
```

```
Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortStatisticalDataProviderImpl"
)]
class CISCO_FCPortStatisticalData : CISCO_ElementStatisticalData
{
      [Override ( "ManagedElement" ), Key, Min ( 1 ),
       Max ( 1 ),
       Description (
          "The ManagedElement for which statistical or metric data "
          "is defined." )]
   CISCO_FCPort REF ManagedElement;

      [Override ( "Stats" ), Key, Description (
      "The statistic information/object." )]
   CISCO_FCPortStatistics REF Stats;

};
```

CISCO_FCPortStatistics.mof

```
CISCO_FCPortStatistics
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCPortStatisticsProviderImpl")]
class CISCO_FCPortStatistics : CIM_FCPortStatistics
{};
```

CISCO_FCSwitchCapabilities.mof

```
CISCO_FCSwitchCapabilities
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCSwitchCapabilitiesProviderImpl"
)]
class CISCO_FCSwitchCapabilities : CIM_FCSwitchCapabilities
{};
```

CISCO_FCSwitchSettings.mof

```
CISCO_FCSwitchSettings
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_FCSwitchSettingsProviderImpl")]
class CISCO_FCSwitchSettings : CIM_FCSwitchSettings
{};
```

CISCO_HBAProduct.mof

```
CISCO_HBAProduct
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_HBAProductProviderImpl"),
        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 ),
        MappingStrings { "FC-GS-4 | FDMI | IdentifyingNumber " }]
    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 ),
        MappingStrings { "FC-GS-4 | FDMI | Manufacturer" }]
    string Vendor;

    [Override("Version"),Key, Description (
        "A string indicating the version of the HBA card."),
        MaxLen ( 64 ),
        MappingStrings { "FC-GS-4 | FDMI | Version" }]
    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 "),
        MappingStrings {"FC-GS-4 | FDMI | Model Description"}]
    string ElementName;
};
```

CISCO_HBASoftwareIdentity.mof

```
CISCO_HBASoftwareIdentity
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_HBASoftwareIdentityProviderImpl")
]
class CISCO_HBASoftwareIdentity : CIM_SoftwareIdentity
{};
```

CISCO_HBASoftwareInstalledOnPlatform.mof

```
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."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_HBASoftwareInstalledOnPlatformProv
iderImpl")]
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_HostComputerSystem.mof

```
CISCO_HostComputerSystem
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_HostComputerSystemProviderImpl")]
class CISCO_HostComputerSystem : CISCO_ComputerSystem
{
    String IpAddress;
};
```

CISCO_HostComputerSystemsInAdminDomain.mof

```
CISCO_HostComputerSystemsInAdminDomain
[Association,
       Description (
       "CISCO_HostComputerSystemsInAdminDomain is a association between CISCO_AdminDomain
and "
       "CISCO_HostComputerSystem."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_HostComputerSystemsInAdminDomainPr
oviderImpl")]
class CISCO_HostComputerSystemsInAdminDomain : CISCO_Component
{
      [Override ("GroupComponent"), Key, Aggregate, Description (
          "The parent element in the association." )]
   CISCO_AdminDomain REF GroupComponent;

      [Override("PartComponent"), Key, Description (
       "The child element in the association." )]
   CISCO_HostComputerSystem REF PartComponent;

};
```

CISCO_HostedAccessPoint.mof

```
CISCO_HostedAccessPoint
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_HostedAccessPoint : CIM_HostedAccessPoint
{};
```

CISCO_HostedCollection.mof

```
CISCO_HostedCollection
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_HostedCollection : CIM_HostedCollection
{};
```

CISCO_HostedDependency.mof

```
CISCO_HostedDependency
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_HostedDependency : CIM_HostedDependency
```

```
{};
```

CISCO_HostedService.mof

```
CISCO_HostedService
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_HostedService : CIM_HostedService
{};
```

CISCO_InstalledSoftwareIdentity.mof

```
CISCO_InstalledSoftwareIdentity
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_InstalledSoftwareIdentity : CIM_InstalledSoftwareIdentity
{};
```

CISCO_IPElementSettingData.mof

```
CISCO_IPElementSettingData
[Association,
       Description ("This corresponds to the association between CISCO_IPProtocolEndpoint
"
                "and CISCO_IPSettings. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_IPElementSettingDataProviderImpl")
]
class CISCO_IPElementSettingData : CIM_ElementSettingData
{
        [Override ("ManagedElement"), Description (
                "Reference to CISCO_IPProtocolEndpoint instance.")]
        CISCO_IPProtocolEndpoint REF ManagedElement;

        [Override ("SettingData"), Description (
                "Reference to CISCO_FCIPSettings instance.")]
        CISCO_IPSettings REF SettingData;
};
```

CISCO_IPEndPointStatisticalData.mof

```
CISCO_IPEndPointStatisticalData
[Association,
       Description ("CISCO_IPEndPointStatistics is an association that associates "
                    "CISCO_IPProtocolEndPoint to its StatisticalData "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_IPEndPointStatisticalDataProviderI
mpl")]
class CISCO_IPEndPointStatisticalData :  CIM_ElementStatisticalData {

        [Override ("ManagedElement"), Description (
                "Reference to CISCO_IPProtocolEndpoint instance.")]
     CISCO_IPProtocolEndPoint REF ManagedElement;

         [Override("Stats"), Key, Description (
                "The statistic information.")]
```

```
     CISCO_IPEndPointStatistics REF Stats;
};
```

CISCO_IPEndPointStatistics.mof

```
CISCO_IPEndPointStatistics
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_IPEndPointStatisticsProviderImpl"
)]
class CISCO_IPEndPointStatistics : CIM_IPEndpointStatistics {

    [Description (
         "The number of times FC synchronization lost on this FCIP Link. "
    )]
         uint64  FcipLossofFcSynchs;

    [Description (
         "The number of times FCIP Special Frame not received on this FCIP Link. "
    )]
                uint64 FcipSfNotRcv;

    [Description (
    "The number of times FCIP Special Frame Response not received on this FCIP Link. "
    )]
         uint64 FcipSfRespNotRcv;

    [Description (
         "The number of times FCIP Special Frame Bytes mismatch"
         " happened on this FCIP Link. "
    )]
                uint64 FcipSfRespMismatch;
    [Description (
         "The number of times FCIP Special Frame Invalid connections"
         " once happened on this FCIP Link. "
    )]

    uint64 FcipSfInvalidNonce;

    [Description (
    "The number of times FCIP Special Frame Response not received on this FCIP Link. "
    )]
    uint64 FcipDuplicateSfRcv;

    [Description (
         "The number of times FCIP Special Frames with invalid "
         " destination FC Fabric Entity WWN received on this FCIP Link. "
    )]
    uint64 FcipSfInvalidWWN;

    [Description (
         "The number of times B_Access Link Keep Alive Time out"
         " happened on this FCIP Link. "
    )]

    uint64 FcipBB2LkaTimeOut;

    [Description (
         "The number of times SNTP Time Stamp expired on this FCIP Link. "
    )]
    uint64 FcipSntpTimeStampExp;
};
```

CISCO_IPEthernetEndpoint.mof

```
CISCO_IPEthernetEndpoint
[Association,
       Description ("This corresponds to the association between
CISCO_IPProtocollEndpoint "
           "and CISCO_LANEndPoint." ),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_IPEthernetEndpointProviderImpl")]
class CISCO_IPEthernetEndpoint : CIM_BindsTo {

    [Override ( "Antecedent" ), Description (
        "The underlying IP Endpoint, which is depended upon.")]
    CISCO_IPProtocolEndpoint REF Antecedent;

    [Override ( "Dependent" ), Description (
        "The LAN Endpoint dependent on the IP Endpoint.")]
    CISCO_LANEndPoint REF Dependent;
};
```

CISCO_IPProtocolEndpoint.mof

```
CISCO_IPProtocolEndpoint
[Description ("A protocol endpoint that is dedicated to running IP."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_IPProtocolEndpointProviderImpl")]
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;

//  [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_IPSettings.mof

```
CISCO_IPSettings
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_IPSettingsProviderImpl")]
class CISCO_IPSettings : CIM_IPSettings
{};
```

CISCO_LANEndpoint.mof

```
CISCO_LANEndpoint
[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."),
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LANEndpointProviderImpl")]
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;
};
```

CISCO_LinkDown.mof

```
CISCO_LinkDown
[Indication,
       Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LinkDownProviderImpl")]
class CISCO_LinkDown : CISCO_LinkStateChange
{};
```

CISCO_LinkStateChange.mof

```
CISCO_LinkStateChange
[Abstract,
       Indication,
       Description ("This is an abstract indication class." )]
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 {"0", "1", "2"},
      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;
};
```

CISCO_LinkUp.mof

```
CISCO_LinkUp
[Indication,
       Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LinkUpProviderImpl")]
class CISCO_LinkUp : CISCO_LinkStateChange
{};
```

CISCO_LogicalComputerSystem.mof

```
CISCO_LogicalComputerSystem
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalComputerSystemProviderImpl
")]
class CISCO_LogicalComputerSystem : CISCO_ComputerSystem
{
};
```

CISCO_LogicalComputerSystemsInAdminDomain.mof

```
CISCO_LogicalComputerSystemsInAdminDomain
[Association,
       Description (
       "CISCO_LogicalComputerSystemsInAdminDomain is a association between
CISCO_AdminDomain and "
       "CISCO_LogicalComputerSystem."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalComputerSystemsInAdminDomai
nProviderImpl")]
class CISCO_LogicalComputerSystemsInAdminDomain : CISCO_Component
{
      [Override ("GroupComponent"), Key, Aggregate, Description (
          "The parent element in the association." )]
   CISCO_AdminDomain REF GroupComponent;

      [Override("PartComponent"), Key, Description (
       "The child element in the association." )]
   CISCO_LogicalComputerSystem REF PartComponent;

};
```

CISCO_LogicalFCPort.mof

```
CISCO_LogicalFCPort
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalFCPortProviderImpl")]
class CISCO_LogicalFCPort : 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;

    [Description (
        "IP Address of the actual node.")]
    string NodeIpAddress;

    [Experimental, Description (
          "The availability of the port for client to "
          "determine whether the port can be made operational.  The "
          "values: \n"
          "\"Available\" indicates that the port can be made operational, \n"
          "\"Not Installed\" indicates some aspect of the port has not been "
          "installed preventing it from being operational but is discoverable through "
          "instrumentation, \n"
          "\"No Transceiver\" indicates that the transceiver is "
          "not installed to allow the port to become operational, "
         "\"Incompatible Transceiver\" indicates the installed transceiver is not correct
and is preventing "
          "the port from being operational, \n"
          "\"Not Licensed\" indicates that the port "
          "cannot be made operational due to a license not existing for the port."),
       ValueMap { "0", "1", "2", "3", "4", "5", "6" },
       Values { "Unknown", "Available", "Not Installed", "No Transceiver",
          "Incompatible Transceiver", "Not Licensed", "DMTF Reserved" }]
    uint16 PortAvailability = 2;
};
```

CISCO_LogicalFCPortForFCPort.mof

```
CISCO_LogicalFCPortForFCPort
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalFCPortForFCPortProviderImpl
")]
class CISCO_LogicalFCPortForFCPort : CISCO_HostedDependency
{
      [Override ( "Antecedent" ),
       Max ( 1 ),
       Description ( "The scoping ManagedElement." )]
   CISCO_FCPort REF Antecedent;

      [Override ( "Dependent" ),
       Description ( "The hosted ManagedElement." )]
   CISCO_LogicalFCPort REF Dependent;
};
```

CISCO_LogicalForPhysicalComputerSystem.mof

```
CISCO_LogicalForPhysicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalForPhysicalComputerSystemPr
oviderImpl")]
class CISCO_LogicalForPhysicalComputerSystem : CISCO_HostedDependency
{
      [Override ( "Antecedent" ),
       Max ( 1 ),
       Description ( "The scoping ManagedElement." )]
   CISCO_PhysicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description ( "The hosted ManagedElement." )]
   CISCO_LogicalComputerSystem REF Dependent;
};
```

CISCO_LogicalModule.mof

```
CISCO_LogicalModule
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalModuleProviderImpl")]
class CISCO_LogicalModule : CIM_LogicalModule
{};
```

CISCO_LogicalModulesInPhysicalComputerSystem.mof

```
CISCO_LogicalModulesInComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalModulesInPhysicalComputerSy
stemProviderImpl")]
class CISCO_LogicalModulesInPhysicalComputerSystem : CISCO_SystemDevice
{
      [Aggregate, Override ( "GroupComponent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The parent system in the Association." )]
   CISCO_PhysicalComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ),
       Weak, Description (
          "The LogicalDevice that is a component of a System." )]
   CISCO_LogicalModule REF PartComponent;

};
```

CISCO_LogicalPortGroup.mof

```
CISCO_LogicalPortGroup
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalPortGroupProviderImpl")]
class CISCO_LogicalPortGroup : CIM_LogicalPortGroup
{};
```

CISCO_LogicalPortGroupInHostComputerSystem.mof

```
CISCO_LogicalPortGroupInHostComputerSystem
```

```
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalPortGroupInHostComputerSyst
emProviderImpl")]
class CISCO_LogicalPortGroupInHostComputerSystem : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_HostComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_LogicalPortGroup REF Dependent;
};
```

CISCO_LogicalSwitchConformsToSwitchProfile.mof

```
[Association, Version ( "3.1.0" ), Description (
       "The SMISConformsToProfile association defines the "
       "RegisteredProfiles that are conformant with a specific "
       "verision of SIM-S. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalSwitchConformsToSwitchProfi
leProviderImpl")
    ]
class CISCO_LogicalSwitchConformsToSwitchProfile : CIM_ElementConformsToProfile {

      [Key, Override ( "ConformantStandard" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The RegisteredProfile to which the ManagedElement conforms.")]
   CISCO_SwitchProfile REF ConformantStandard;

      [Key, Override ( "ManagedElement" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The ManagedElement that conforms to the RegisteredProfile.")]
   CISCO_LogicalComputerSystem REF ManagedElement;
};
```

CISCO_LogicalSwitchElementCapabilities.mof

```
CISCO_LogicalSwitchElementCapabilities
[Association,
       Description("ElementCapabilities represents the association between "
       "ManagedElements and their Capabilities."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalSwitchElementCapabilitiesPr
oviderImpl")]
class CISCO_LogicalSwitchElementCapabilities : CISCO_ElementCapabilities
{
      [Override ( "ManagedElement" ), Key, Min ( 1 ),
       Max ( 1 ),
       Description ( "The managed element." )]
   CISCO_LogicalComputerSystem REF ManagedElement;

      [Override ( "Capabilities" ), Key, Description (
          "The Capabilities object associated with the element." )]
   CISCO_FCLogicalSwitchCapabilities REF Capabilities;
};
```

CISCO_LogicalSwitchInstalledSoftwareIdentity.mof

```
CISCO_LogicalSwitchInstalledSoftwareIdentity
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalSwitchInstalledSoftwareIden
tityProviderImpl")]
class CISCO_LogicalSwitchInstalledSoftwareIdentity : CISCO_InstalledSoftwareIdentity
{
      [Override ( "System" ), Key, Description (
          "The system on which the software is installed." )]
   CISCO_LogicalComputerSystem REF System;

      [Override ( "InstalledSoftware" ), Key, Description (
      "The SoftwareIdentity that is installed." )]
   CISCO_LogicalSwitchSoftwareIdentity REF InstalledSoftware;
};
```

CISCO_LogicalSwitchSettingData.mof

```
CISCO_LogicalSwitchSettingData
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalSwitchSettingDataProviderIm
pl")]
class CISCO_LogicalSwitchSettingData : CISCO_ElementSettingData
{

      [Override ( "ManagedElement" ), Key,
      Description ( "The managed element." )]
   CISCO_LogicalComputerSystem REF ManagedElement;

      [Override ( "SettingData" ), Key, Description (
          "The SettingData object associated with the element." )]
   CISCO_FCLogicalSwitchSettings REF SettingData;
};
```

CISCO_LogicalSwitchSoftwareIdentity.mof

```
CISCO_LogicalSwitchSoftwareIdentity
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalSwitchSoftwareIdentityProv
iderImpl")]
class CISCO_LogicalSwitchSoftwareIdentity : CISCO_SoftwareIdentity
{};
```

CISCO_LogicalIdentity.mof

```
[Association,
       Description ("This association represents relation between the ServiceAccessPoints
i.e. "
       "two CISCO_ProtocolEndPoints of the NPV link."
       "Basically it the connection between the "
       "F port of the core NPIV  to the F port of NPV which connects to the end
devices"),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalIdentityProviderImpl")]
class CISCO_LogicalIdentity : CIM_LogicalIdentity
{
```

```
      [Override ( "SystemElement" ) ,
      Description ("SystemElement represents one aspect of the Managed "
          "Element. The use of \'System\' in the role name does not "
          "limit the scope of the association. The role name was "
          "defined in the original association, where the "
          "referenced elements were limited to LogicalElements. "
          "Since that time, it has been found valuable to "
          "instantiate these types of relationships for "
          "ManagedElements, such as Collections. So, the referenced "
          "elements of the association were redefined to be "
          "ManagedElements. Unfortunately, the role name could not "
          "be changed without deprecating the entire association. "
          "This was not deemed necessary just to correct the role "
          "name." )]
    CISCO_ProtocolEndPoint REF SystemElement;
    [Override ( "SameElement" ),
    Description ("SameElement represents an alternate aspect of the "
        "ManagedElement.") ]
    CISCO_ProtocolEndPoint REF SameElement;
};
```

CISCO_LogicalPortGroupInStorageComputerSystem.mof

```
// ==================================================================
// CISCO_LogicalPortGroupInStorageComputerSystem
// ==================================================================
    [Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_LogicalPortGroupInStorageComputerS
ystemProviderImpl")]
class CISCO_LogicalPortGroupInStorageComputerSystem : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_StorageComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_LogicalPortGroup REF Dependent;
};
```

CISCO_MediaFRU.mof

```
CISCO_MediaFRU
[Abstract,
       Indication,
       Description ("This is an abstract indication class." )]
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;

};
```

CISCO_MediaFRUChanged.mof

```
CISCO_MediaFRUChanged
[Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_MediaFRUChangedProviderImpl")]
class CISCO_MediaFRUChanged: CISCO_AlertIndication
{
  uint32  PhysicalIndex;
    [Description (
        "Module Operational Status"),
     ValueMap {"1", "2", "3", "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;
};
```

CISCO_MediaFRUInserted.mof

```
CISCO_MediaFRUInserted
[Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_MediaFRUInsertedProviderImpl")]
class CISCO_MediaFRUInserted : CISCO_MediaFRU
{};
```

CISCO_MediaFRURemoved.mof

```
CISCO_MediaFRURemoved
[Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_MediaFRURemovedProviderImpl")]
class CISCO_MediaFRURemoved : CISCO_MediaFRU
{};
```

CISCO_ModuleEthernetPort.mof

```
CISCO_ModuleEthernetPort
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ModuleEthernetPortProviderImpl")]
class CISCO_ModuleEthernetPort : CISCO_ModulePort
{
      [Aggregate, Override ( "GroupComponent" ),
       Max ( 1 ),
       Description ( "A module that has ports." )]
   CISCO_LogicalModule REF GroupComponent;

      [Override ( "PartComponent" ),
```

```
       Description ( "A Port that is associated with a module." )]
   CISCO_EthernetPort REF PartComponent;
};
```

CISCO_ModuleFcPort.mof

```
CISCO_ModuleFcPort
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ModuleFcPortProviderImpl")]
class CISCO_ModuleFcPort : CISCO_ModulePort
{
      [Aggregate, Override ( "GroupComponent" ),
       Max ( 1 ),
       Description ( "A module that has ports." )]
   CISCO_LogicalModule REF GroupComponent;

      [Override ( "PartComponent" ),
       Description ( "A Port that is associated with a module." )]
   CISCO_FCPort REF PartComponent;
};
```

CISCO_ModulePort.mof

```
CISCO_ModulePort
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_ModulePort : CIM_ModulePort
{};
```

CISCO_NameServerDatabaseChanged.mof

```
CISCO_NameServerDatabaseChanged
[Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_NameServerDatabaseChangedProviderI
mpl")]
class CISCO_NameServerDatabaseChanged: CISCO_AlertIndication
{
};
```

CISCO_PhysicalComputerSystem.mof

```
CISCO_PhysicalComputerSystem
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalComputerSystemProviderImp
l")]
class CISCO_PhysicalComputerSystem : CISCO_ComputerSystem
{
    string IpAddress;
};
```

CISCO_PhysicalComputerSystemsInAdminDomain.mof

```
CISCO_PhysicalComputerSystemsInAdminDomain
```

```
[Association,
        Description (
        "CISCO_PhysicalComputerSystemsInAdminDomain is a association between
CISCO_AdminDomain and "
        "CISCO_PhysicalComputerSystem."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalComputerSystemsInAdminDoma
inProviderImpl")]
class CISCO_PhysicalComputerSystemsInAdminDomain : CISCO_Component
{
      [Override ("GroupComponent"), Key, Aggregate, Description (
          "The parent element in the association." )]
   CISCO_AdminDomain REF GroupComponent;

      [Override("PartComponent"), Key, Description (
       "The child element in the association." )]
   CISCO_PhysicalComputerSystem REF PartComponent;

};
```

CISCO_PhysicalElement.mof

```
CISCO_PhysicalElement
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalElementProviderImpl")]
class CISCO_PhysicalElement : CIM_PhysicalElement
{};
```

CISCO_PhysicalElementEthernetPortRealizes.mof

```
CISCO_PhysicalElementEthernetPortRealizes
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalElementEthernetPortRealize
sProviderImpl")]
class CISCO_PhysicalElementEthernetPortRealizes : CISCO_Realizes
{
      [Override ( "Antecedent" ),
       Description (
          "The physical component that implements the Device." )]
   CISCO_PhysicalElement REF Antecedent;

      [Override ( "Dependent" ),
       Description ( "The LogicalDevice." )]
   CISCO_EthernetPort REF Dependent;
};
```

CISCO_PhysicalElementFcPortRealizes.mof

```
CISCO_PhysicalElementFcPortRealizes
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalElementFcPortRealizesProvi
derImpl")]
class CISCO_PhysicalElementFcPortRealizes : CISCO_Realizes
{
      [Override ( "Antecedent" ),
       Description (
          "The physical component that implements the Device." )]
   CISCO_PhysicalElement REF Antecedent;
```

```
      [Override ( "Dependent" ),
       Description ( "The LogicalDevice." )]
   CISCO_FCPort REF Dependent;
};
```

CISCO_PhysicalHBA.mof

```
CISCO_PhysicalHBA
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalHBAProviderImpl"),
       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 ( "ElementName" ),
       MappingStrings { "MIB.IETF|Entity-MIB.entPhysicalName" }]
    string ElementName;

    [Override("CreationClassName"), Key, MaxLen (256), 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.")]
    string CreationClassName= "CISCO_PhysicalHBA";

    [Override("Manufacturer"), MaxLen (256), Description (
        "The name of the organization responsible for "
        "manufacturing the HBA."),
        MappingStrings {"FC-GS-4 | FDMI | Manufacturer"}]
    string Manufacturer;

    [Override("Model"), MaxLen (64), Description (
        "The name by which the HBA is generally known."),
        MappingStrings {"FC-GS-4 | FDMI | Model"}]
    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),
        MappingStrings {"FC-GS-4 | FDMI | Model Description"}]
    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."),
        MappingStrings {"FC-GS-4 | FDMI | Serial Number"}]
    string SerialNumber;

    [Override("Version"), MaxLen (64), Description (
        "A string indicating the version of the HBA card."),
        MappingStrings {"FC-GS-4 | FDMI | Version"}]
    string Version;
} ;
```

CISCO_PhysicalPackage.mof

```
CISCO_PhysicalPackage
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalPackageProviderImpl")]
class CISCO_PhysicalPackage : CIM_PhysicalPackage
{};
```

CISCO_PhysicalPackageLogicalModuleRealizes.mof

```
CISCO_PhysicalPackageLogicalModuleRealizes
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PhysicalPackageLogicalModuleRealiz
esProviderImpl")]
class CISCO_PhysicalPackageLogicalModuleRealizes : CISCO_Realizes
{
      [Override ( "Antecedent" ),
       Description (
          "The physical component that implements the Device." )]
   CISCO_PhysicalPackage REF Antecedent;

      [Override ( "Dependent" ),
       Description ( "The LogicalDevice." )]
   CISCO_LogicalModule REF Dependent;
};
```

CISCO_Platform.mof

```
CISCO_Platform
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PlatformProviderImpl"),
       Description ("CISCO_Platform represents a fabric-connected entity, "
           "containing one or more Node objects, that has registered "
           "with a fabric's Management Server service."
           "Instances of this class can be created and deleted by a "
           "client using createInstance and deleteInstance respectively "
           "This class also represents the HBA host and if the HBA host "
           "information is got through FDMI call then the setInstance will "
           "not be supported for setting the properties like Dedicated, "
           "Management ServerList.")]
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.  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.")]
    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"
        "\"Proxy 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. \n"
        "Note that ManagedSystemElement's Name property is also "
        "defined as a user-friendly name. But, it is often "
        "subclassed to be a Key. It is not reasonable that the same "
        "property can convey both identity and a user friendly name, "
        "without inconsistencies. Where Name exists and is not a Key "
        "(such as for instances of LogicalDevice), the same "
        "information MAY be present in both the Name and ElementName "
        "properties.")]
    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"},
        MappingStrings {"API.CISCO | Platform | PlatformType"}]
    uint16 Dedicated[];

    [Override ("OtherIdentifyingInfo"), Description(
        "Platform name: for example, host name."),
        MappingStrings {"API.CISCO | Platform | PlatformName"}]
    string OtherIdentifyingInfo[];

    [Write, Description(
        "The set of management IP Addresses used to access this "
        "platform."),
        MappingStrings {"API.CISCO | Platform | ManagementAddrSet"}]
    string MgmtAddressList[];
};
```

CISCO_PlatformHostedSANAccessPoint.mof

```
CISCO_PlatformHostedSANAccessPoint
[Association,
       Description ("CISCO_PlatformHostedSANAccessPoint is an association "
           "between a ProtocolEndPoint and the platform on which it is "
           "provided.  The cardinality of this association is "
           "one-to-many and is weak with respect to the platform.  Each "
```

```
            "platform can host many ProtocolEndPoints."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PlatformHostedSANAccessPointProvid
erImpl")]
class CISCO_PlatformHostedSANAccessPoint: CIM_HostedAccessPoint {

    [Override ("Antecedent"), Max (1), Min (1), Description (
        "The hosting system.")]
    CISCO_Platform REF Antecedent;

    [Override ("Dependent"), Weak, Description (
        "The SAPs that are hosted on this system.")]
    CISCO_ProtocolEndPoint REF Dependent;
};
```

CISCO_PlatformPackage.mof

```
CISCO_PlatformPackage
[Association,
        Description ("This association denotes one or more physical HBAs that "
            "realize a Platform."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PlatformPackageProviderImpl")]
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_PortAdded.mof

```
CISCO_PortAdded
[Indication,
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortAddedProviderImpl")]
class CISCO_PortAdded: CIM_InstCreation
{};
```

CISCO_PortChannel.mof

```
CISCO_PortChannel
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortChannelProviderImpl")]
class CISCO_PortChannel : CIM_RedundancySet
{
    [Override("Caption"), Description (
            "An user-friendly name of the port channel. ")]
        string Caption;

    [Override("Description"), Description (
            "An user-friendly name of the port channel. ")]
        string Description;

    [Override("ElementName"), Description (
            "An user-friendly name of the port channel. ")]
        string ElementName;
```

```
    [Override("InstanceID"), Description (
            "The ifIndex of port channel interface concatenated with number of members in
that portchannel. (Eg: 67108866:1) ")]
        string InstanceID;

    [Description ("CreationTime of the port channel interface.")]
        datetime CreationTime;
    [Description ("The timestamp indicating the time of last action performed on the port
channel interface.")]
        datetime LastActionTime;
    boolean LastActionStatus;
//  [Description ("The minimum number of members required for PortChannel ")]
//      uint32 MinNumberNeeded = 1;
//  [Description ("The maximum number of members that can be part for PortChannel ")]
//      uint32 MaxNumberSupported = 8;
    [Description ("Vsan to which the port channel belongs to. It is zero, if the
portchannel is trunked.")]
        uint16 VsanId;
    boolean isTrunked;
    [Description ("If the addition was forced, it will be set to true.")]
        boolean ChannelAdditionForced;

    [Description ("The number of interface that are part of the portchannel.")]
    uint32 NumberOfMembers;

    [Description("Indicates the Admin Status of the port channel mode."),
    ValueMap {  "1", "2", "3", "4", "5"},
    Values { "Auto", "On", "Off", "Desirable", "Active"}]
        uint16 AdminStatus;

    [Description("Indicates the Operational Status of the port channel mode." ),
    ValueMap {  "1", "2", "3", "4", "5"},
    Values { "Auto", "On", "Off", "Desirable", "Active"}]
    uint16 OperStatus;


    [Description (
        "Indicates the current status of the port channel."),
        ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
            "10", "11", "12", "13", "14", "15", "16", "17", "18"},
        Values {"Unknown", "Other", "OK", "Degraded", "Stressed",
            "Predictive Failure", "Error", "Non-Recoverable Error",
            "Starting", "Stopping", "Stopped", "In Service",
            "No Contact", "Lost Communication", "Aborted",
            "Dormant", "Supporting Entity in Error", "Completed",
            "Power Mode"},
        ArrayType ("Indexed")]
    uint16 OperationalStatus[];
};
```

CISCO_PortChannelsInSwitch.mof

```
CISCO_PortChannelsInSwitch
[Association,
        Description ("This corresponds to the association between
CISCO_PhysicalComputerSystem"
                "and CISCO_PortChannel. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortChannelsInSwitchProviderImpl")
]
```

```
class CISCO_PortChannelsInSwitch : CIM_HostedCollection
{
      [Override ( "Antecedent" ), Description (
          "The scoping system.")]
   CISCO_PhysicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The collection defined in the context of a system.")]
    CISCO_PortChannel REF Dependent;
};
```

CISCO_PortController.mof

```
CISCO_PortController
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortControllerProviderImpl"),
       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"),
        MappingStrings {"API.CISCO | HBA | WWN",
            "API.CISCO | Node | WWN"}]
    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" },
        ModelCorrespondence { "CIM_PortController.OtherControllerType" }]
    uint16 ControllerType = 4;
};
```

CISCO_PortControllerInFabric.mof

```
CISCO_PortControllerInFabric
[Association,
       Description ("CISCO_PortControllerInFabric defines a SystemSpecificCollection "
           "in the context of a scoping system.  This association is "
           "created ONLY if CISCO_PortController cannot be scoped within "
           "CISCO_Platform.  This can happen if the scoping platform "
           "cannot be determined for some reason: for example, if it "
           "was not registered in the platform database."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortControllerInFabricProviderImpl
")]
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_PortControllerInPlatform.mof

```
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."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortControllerInPlatformProviderIm
pl")]
class CISCO_PortControllerInPlatform: CIM_SystemDevice {

    [Override ("GroupComponent"), Description (
        "A platform hosts a collection of devices."),
        MappingStrings {"API.CISCO | Platform | PlatformNodeSet "}]
    CISCO_Platform REF GroupComponent;

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

CISCO_PortControllerRealizes.mof

```
CISCO_PortControllerRealizes
[Association,
       Description ("CISCO_PortControllerRealizes is the association that defines "
           "the mapping between devices and the physical elements "
           "that implement them."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortControllerRealizesProviderImpl
")]
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_PortControllerSoftwareIdentity.mof

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

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortControllerSoftwareIdentityProv
iderImpl")]
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_PortRemoved.mof

```
CISCO_PortRemoved
[Indication,
       Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PortRemovedProviderImpl")]
class CISCO_PortRemoved: CIM_InstDeletion
{};
```

CISCO_PowerAlert.mof

```
CISCO_PowerAlert
[Indication,
       Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_PowerAlertProviderImpl")]
class CISCO_PowerAlert: CISCO_EnvironmentalAlert
{
      uint32  FRUPowerAdminStatus;
      uint32  FRUCurrent;
};
```

CISCO_Product.mof

```
CISCO_Product
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ProductProviderImpl")]
class CISCO_Product : CIM_Product
{};
```

CISCO_ProductPhysicalComponent.mof

```
CISCO_ProductPhysicalComponent
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ProductPhysicalComponentProviderIm
pl")]
class CISCO_ProductPhysicalComponent : CIM_ProductPhysicalComponent
{
      [Override ( "GroupComponent" ),
       Max ( 1 ),
       Description ( "The Product." )]
   CISCO_Product REF GroupComponent;

      [Override ( "PartComponent" ),
       Description (
          "The PhysicalElement which is a part of the Product." )]
   CISCO_PhysicalElement REF PartComponent;

};
```

CISCO_ProductPhysicalHBA.mof

```
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."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ProductPhysicalHBAProviderImpl")]
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_ProductSoftwareComponent.mof

```
[Association, Version("3.1.0"),
   Provider("jsr48:com.wbemsolutions.wbem.cimom."
              "GenericReadOnlyProvider")
   ]
class CISCO_ProductSoftwareComponent : WBEMSolutions_ProductSoftwareComponent {
};
```

CISCO_ProtocolEndPoint.mof

```
CISCO_ProtocolEndPoint
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ProtocolEndPointProviderImpl")]
class CISCO_ProtocolEndPoint : CIM_ProtocolEndPoint
{};
```

CISCO_ProtocolEndPointHostComputerSystem.mof

```
CISCO_ProtocolEndPointHostComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ProtocolEndPointHostComputerSystem
ProviderImpl")]
class CISCO_ProtocolEndPointHostComputerSystem : CISCO_HostedAccessPoint
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The hosting System." )]
    CISCO_HostComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Weak, Description (
          "The SAPs that are hosted on this System." )]
    CISCO_ProtocolEndPoint REF Dependent;
};
```

CISCO_ProtocolEndPointLogicalComputerSystem.mof

```
CISCO_ProtocolEndPointLogicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ProtocolEndPointLogicalComputerSys
temProviderImpl")]
class CISCO_ProtocolEndPointLogicalComputerSystem : CISCO_HostedAccessPoint
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The hosting System." )]
   CISCO_LogicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Weak, Description (
          "The SAPs that are hosted on this System." )]
   CISCO_ProtocolEndPoint REF Dependent;
};
```

CISCO_ProtocolEndPointStorageComputerSystem.mof

```
CISCO_ProtocolEndPointStorageComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ProtocolEndPointStorageComputerSys
temProviderImpl")]
class CISCO_ProtocolEndPointStorageComputerSystem : CISCO_HostedAccessPoint
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The hosting System." )]
    CISCO_StorageComputerSystem REF Antecedent;

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

```
            "The SAPs that are hosted on this System." )]
    CISCO_ProtocolEndPoint REF Dependent;
};
```

CISCO_Realizes.mof

```
CISCO_Realizes
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_Realizes : CIM_Realizes
{};
```

CISCO_ReferencedProfile.mof

```
[Association, Version ( "2.6.0" ), Description (
       "A subprofile requires another RegisteredProfile for context. "
       "This association mandates the scoping relationship between a "
       "subprofile and its scoping profile."),
    Provider("jsr48:com.wbemsolutions.wbem.cimom."
               "GenericReadOnlyProvider")
    ]
class CISCO_ReferencedProfile : WBEMSolutions_ReferencedProfile {

      [Override ( "Antecedent" ), Min ( 1 ), Description (
          "The RegisteredProfile that is referenced/required by the "
          "subprofile.")]
   CISCO_RegisteredProfile REF Antecedent;

      [Override ( "Dependent" ), Description (
          "A RegisteredSubProfile that requires a scoping profile, for "
          "context.")]
   CISCO_RegisteredSubProfile REF Dependent;
};
```

CISCO_RegisteredProfile.mof

```
[Version ( "3.1.0" ), Description (
       "A RegisteredProfile describes a set of CIM Schema classes with "
       "required properties and/or methods, necessary to manage a "
       "real-world entity or to support a usage scenario, in an "
       "interoperable fashion. RegisteredProfiles can be defined by "
       "the DMTF or other standards organizations. Note that this "
       "class should not be confused with CIM_Profile, which collects "
       "SettingData instances, to be applied as a 'configuration "
       "profile' for an element. \n"
       "A RegisteredProfile is a named 'standard' for CIM-based "
       "management of a particular System, subsystem, Service or other "
       "entity, for a specified set of uses. It is a complete, "
       "standalone definition, as opposed to the subclass "
       "RegisteredSubProfile, which requires a scoping profile for "
       "context. \n"
       "The uses for a RegisteredProfile or SubProfile MUST be "
       "specified in the document that defines the profile. Examples "
       "of Profiles are to manage various aspects of an Operating "
       "System, Storage Array, or Database. The name of the profile is "
       "defined and scoped by its authoring organization."),
    //Provider("jsr48:com.wbemsolutions.wbem.cimom."
    //           "GenericReadOnlyProvider")
```

```
Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_RegisteredProfileProviderImpl")

   ]
class CISCO_RegisteredProfile : WBEMSolutions_RegisteredProfile {
};
```

CISCO_RegisteredProfileInstances.mof

```
instance of CISCO_RegisteredProfile as $DMTF100_PRP {

   InstanceID = "CISCO:CISCO_RegisteredProfile.2.Profile Registration.1.0.0";
   RegisteredOrganization = 2;
   RegisteredName = "Profile Registration";
   RegisteredVersion = "1.0.0";
   AdvertiseTypes = {3};
   Caption = "2:Profile Registration:1.0.0";
   Description = "2:Profile Registration:1.0.0";
   ElementName = "2:Profile Registration:1.0.0";
};

instance of CISCO_ServerSoftware as $software {

   InstanceID = "CISCO:CISCO_ServerSoftware-CISCO SMI-S Agent";
   MajorVersion = "1";
   MinorVersion = "1";
   RevisionNumber = "0";
   VersionString = "1.1.0";
   Manufacturer = "Cisco Systems";
   SerialNumber = "CISCO SMI-S Agent" " " "1.1.0";
   Caption = "CISCO SMI-S Agent";
   Description = "CISCO SMI-S Agent";
   ElementName = "CISCO SMI-S Agent";
   Name = "CISCO SMI-S Agent";
   Classifications = {5};
};

instance of CISCO_ElementSoftwareIdentity {
    Antecedent = $software;
    Dependent = $DMTF100_PRP;
};

instance of CISCO_RegisteredSubProfile as $SMIS100_PRP {

   InstanceID = "CISCO:CISCO_RegisteredSubProfile.11.Profile Registration.1.0.0";
   RegisteredOrganization = 11;
   RegisteredName = "Profile Registration";
   RegisteredVersion = "1.0.0";
   AdvertiseTypes = {2};
   Caption = "11:Profile Registration:1.0.0";
   Description = "11:Profile Registration:1.0.0";
   ElementName = "11:Profile Registration:1.0.0";
};

//The following instance reference the SNIA PRP to the DMTF PRP
instance of CISCO_ReferencedProfile {
   Antecedent = $DMTF100_PRP;
   Dependent = $SMIS100_PRP;
};

instance of CISCO_SubProfileSoftwareIdentity {
   Antecedent = $software;
```

```
   Dependent = $SMIS100_PRP;
};

instance of WBEMSolutions_RegisteredProfile as $SMIS140 {

   InstanceID = "WBEMSolutions:WBEMSolutions_RegisteredProfile.11.SMI-S.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "SMI-S";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {3};
   Caption = "SMI-S";
   Description = "SMI-S";
   ElementName = "SMI-S";
};

instance of CISCO_ServerProduct as $product {

   Name = "CISCO SMI-S Agent";
   IdentifyingNumber = "CISCO SMI-S Agent 1.1.0";
   Vendor = "Cisco Systems";
   Version = "1.1.0";
   Caption = "CISCO SMI-S Agent";
   Description = "CISCO SMI-S Agent";
   ElementName = "CISCO SMI-S Agent";
   SKUNumber = "CISCO SMI-S Agent";
};

instance of WBEMSolutions_ProductSoftwareComponent {

  GroupComponent = $product;
  PartComponent = $software;
};

instance of CISCO_FabricProfile as $SMI140_Fabric {

   InstanceID = "CISCO:CISCO_RegisteredProfile.11.Fabric.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "Fabric";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {3};
   Caption = "11:Fabric:1.4.0";
   Description = "11:Fabric:1.4.0";
   ElementName = "11:Fabric:1.4.0";
};

instance of CISCO_RegisteredSubProfile as $SMI140_Fabric_ZoneControl {

   InstanceID = "CISCO:CISCO_RegisteredSubProfile.11.Zone Control.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "Zone Control";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {2};
   Caption = "11:ZoneControl:1.4.0";
   Description = "11:ZoneControl:1.4.0";
   ElementName = "11:ZoneControl:1.4.0";
};

instance of CISCO_RegisteredSubProfile as $SMI140_Fabric_EnhancedZoneControl {

   InstanceID = "CISCO:CISCO_RegisteredSubProfile.11.Enhanced Zoning and Enhanced Zoning
Control.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "Enhanced Zoning and Enhanced Zoning Control";
   RegisteredVersion = "1.4.0";
```

```
   AdvertiseTypes = {2};
   Caption = "11:Enhanced Zoning and Enhanced Zoning Control:1.4.0";
   Description = "11:Enhanced Zoning and Enhanced Zoning Control:1.4.0";
   ElementName = "11:Enhanced Zoning and Enhanced Zoning Control:1.4.0";
};

instance of CISCO_RegisteredSubProfile as $SMI140_Fabric_FDMI {

   InstanceID = "CISCO:CISCO_RegisteredSubProfile.11.FDMI.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "FDMI";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {2};
   Caption = "11:FDMI:1.4.0";
   Description = "11:FDMI:1.4.0";
   ElementName = "11:FDMI:1.4.0";
};

instance of CISCO_RegisteredSubProfile as $SMI140_Fabric_VirtualFabrics {

   InstanceID = "CISCO:CISCO_RegisteredSubProfile.11.Virtual Fabrics.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "Virtual Fabrics";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {2};
   Caption = "11:Virtual Fabrics:1.4.0";
   Description = "11:Virtual Fabrics:1.4.0";
   ElementName = "11:Virtual Fabrics:1.4.0";
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $DMTF100_PRP;
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMIS100_PRP;
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Fabric;
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Fabric_ZoneControl;
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Fabric_EnhancedZoneControl;
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Fabric_FDMI;
```

```
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Fabric_VirtualFabrics;
};

instance of CISCO_SubProfileRequiresProfile {
   Antecedent = $SMI140_Fabric;
   Dependent = $SMIS100_PRP;
};

instance of CISCO_SubProfileRequiresProfile {

   Antecedent = $SMI140_Fabric;
   Dependent = $SMI140_Fabric_ZoneControl;
};

instance of CISCO_SubProfileRequiresProfile {

   Antecedent = $SMI140_Fabric;
   Dependent = $SMI140_Fabric_FDMI;
};

instance of CISCO_SubProfileRequiresProfile {

   Antecedent = $SMI140_Fabric;
   Dependent = $SMI140_Fabric_EnhancedZoneControl;
};

instance of CISCO_SubProfileRequiresProfile {

   Antecedent = $SMI140_Fabric;
   Dependent = $SMI140_Fabric_VirtualFabrics;
};

instance of CISCO_ElementSoftwareIdentity {

   Antecedent = $software;
   Dependent = $SMI140_Fabric;
};

instance of CISCO_SubProfileSoftwareIdentity {

   Antecedent = $software;
   Dependent = $SMI140_Fabric_FDMI;
};

instance of CISCO_SubProfileSoftwareIdentity {

   Antecedent = $software;
   Dependent = $SMI140_Fabric_ZoneControl;
};

instance of CISCO_SubProfileSoftwareIdentity {

   Antecedent = $software;
   Dependent = $SMI140_Fabric_EnhancedZoneControl;
};

instance of CISCO_SubProfileSoftwareIdentity {

   Antecedent = $software;
```

```
   Dependent = $SMI140_Fabric_VirtualFabrics;
};

instance of CISCO_SwitchProfile as $SMI140_Switch {

   InstanceID = "CISCO:CISCO_RegisteredProfile.11.Switch.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "Switch";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {3};
   Caption = "11:Switch:1.4.0";
   Description = "11:Switch:1.4.0";
   ElementName = "11:Switch:1.4.0";
};

instance of CISCO_RegisteredSubProfile as $SMI140_Switch_SwitchPartitioning {

   InstanceID = "CISCO:CISCO_RegisteredSubProfile.11.Switch Partitioning.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "Switch Partitioning";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {2};
   Caption = "11:SwitchPartitioning:1.4.0";
   Description = "11:SwitchPartitioning:1.4.0";
   ElementName = "11:SwitchPartitioning:1.4.0";
};

instance of CISCO_RegisteredSubProfile as $SMI140_Switch_Blades {

   InstanceID = "CISCO:CISCO_RegisteredSubProfile.11.Blades.1.4.0";
   RegisteredOrganization = 11;
   RegisteredName = "Blades";
   RegisteredVersion = "1.4.0";
   AdvertiseTypes = {2};
   Caption = "11:Blades:1.4.0";
   Description = "11:Blades:1.4.0";
   ElementName = "11:Blades:1.4.0";
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Switch;
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Switch_Blades;
};

instance of WBEMSolutions_SMISConformsToProfile {

   ConformantStandard = $SMIS140;
   ManagedElement = $SMI140_Switch_SwitchPartitioning;
};

instance of CISCO_SubProfileRequiresProfile {
   Antecedent = $SMI140_Switch;
   Dependent = $SMIS100_PRP;
};

instance of CISCO_SubProfileRequiresProfile {
```

```
   Antecedent = $SMI140_Switch;
   Dependent = $SMI140_Switch_Blades;
};

instance of CISCO_SubProfileRequiresProfile {

   Antecedent = $SMI140_Switch;
   Dependent = $SMI140_Switch_SwitchPartitioning;
};

instance of CISCO_ElementSoftwareIdentity {

   Antecedent = $software;
   Dependent = $SMI140_Switch;
};

instance of CISCO_SubProfileSoftwareIdentity {

   Antecedent = $software;
   Dependent = $SMI140_Switch_Blades;
};

instance of CISCO_SubProfileSoftwareIdentity {

   Antecedent = $software;
   Dependent = $SMI140_Switch_SwitchPartitioning;
};
```

CISCO_RegisteredSubProfile.mof

```
[Version ( "3.1.0" ), Description (
       "A RegisteredSubProfile subclasses RegisteredProfile to "
       "indicate that a scoping profile is required to provide "
       "context. The latter is specified by the mandatory association, "
       "SubProfileRequiresProfile."),
    Provider("jsr48:com.wbemsolutions.wbem.cimom."
               "GenericReadOnlyProvider")
   ]
class CISCO_RegisteredSubProfile : WBEMSolutions_RegisteredSubProfile {
};
```

CISCO_RemoteFCIPPort.mof

```
CISCO_RemoteFCIPPort
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_RemoteFCIPPortProviderImpl")]
class CISCO_RemoteFCIPPort : CIM_RemotePort
{};
```

CISCO_RemoteIPServiceAccessPoint.mof

```
CISCO_RemoteIPServiceAccessPoint
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_RemoteIPServiceAccessPointProvide
rImpl")]
class CISCO_RemoteIPServiceAccessPoint: CIM_RemoteServiceAccessPoint
{};
```

CISCO_RemoteServiceAccessPoint.mof

```
CISCO_RemoteServiceAccessPoint
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_RemoteServiceAccessPointProviderI
mpl")]
class CISCO_RemoteServiceAccessPoint : CIM_RemoteServiceAccessPoint
{};
```

CISCO_RemoteTCPPort.mof

```
CISCO_RemoteTCPPort
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_RemoteTCPPortProviderImpl")]

class CISCO_RemoteTCPPort : CIM_RemotePort
{};
```

CISCO_SANFCIPEndpoint.mof

```
CISCO_SANFCIPEndpoint
[Association,
       Description ("This corresponds to the association between
CISCO_FCIPProtocolEndpoint "
           "and CISCO_ProtocolEndPoint. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SANFCIPEndpointProviderImpl")]
class CISCO_SANFCIPEndpoint : CIM_BindsTo {

    [Override ("Antecedent"), Description (
        "The underlying SANEndpoint, which is depended upon.")]
    CISCO_FCIPProtocolEndPoint REF Antecedent;

    [Override ( "Dependent" ), Description (
        "The FCIP ProtocolEndpoint dependent on the SANEndpoint.")]
    CISCO_RemoteFCIPPort REF Dependent;
};
```

CISCO_SANIPEndpoint.mof

```
CISCO_SANIPEndpoint
[Association,
       Description ("This corresponds to the association between CISCO_IPProtocolEndpoint
"
                "and CISCO_RemoteFCIPServiceAccessPoint. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SANIPEndpointProviderImpl")]
class CISCO_SANIPEndpoint : CIM_BindsTo {

        [Override ("Antecedent"), Description (
                "The underlying SANEndpoint, which is depended upon.")]
        CISCO_IPProtocolEndpoint REF Antecedent ;

        [Override ( "Dependent" ), Description (
                "The FCIP ProtocolEndpoint dependent on the SANEndpoint.")]
        CISCO_RemoteIPServiceAccessPoint REF Dependent;
};
```

CISCO_SANTCPEndpoint.mof

```
CISCO_SANTCPEndpoint
[Association,
       Description ("This corresponds to the association between CISCO_IPProtocolEndpoint
"
                "and CISCO_TCPProtocolEndPoint. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SANTCPEndpointProviderImpl")]
class CISCO_SANTCPEndpoint : CIM_BindsTo {

        [Override ( "Antecedent" ), Description (
                "The underlying IP Endpoint, which is depended upon.")]
        CISCO_TCPProtocolEndpoint REF Antecedent;

        [Override ( "Dependent" ), Description (
                "The TCP ProtocolEndpoint dependent on the IP Endpoint.")]
        CISCO_RemoteTCPPort REF Dependent;
};
```

CISCO_SAPAvailableForElement.mof

```
CISCO_SAPAvailableForElement
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SAPAvailableForElementProviderImpl
")]
class CISCO_SAPAvailableForElement : CIM_SAPAvailableForElement
{
      [Override ("AvailableSAP"), Key, Description (
          "The Service Access Point that is available." )]
   CISCO_RemoteServiceAccessPoint REF AvailableSAP;

      [Override ("ManagedElement"), Key, Description (
          "The ManagedElement for which the SAP is available." )]
   CISCO_PhysicalComputerSystem REF ManagedElement;

};
```

CISCO_SecurityAlert.mof

```
CISCO_SecurityAlert
[Abstract,
       Indication,
       Description ("This is an abstract indication class." )]
class CISCO_SecurityAlert : CISCO_AlertIndication
{};
```

CISCO_ServerProduct.mof

```
[Version("3.1.0"),
    Provider("jsr48:com.wbemsolutions.wbem.cimom."
               "GenericReadOnlyProvider")
 ]
class CISCO_ServerProduct : WBEMSolutions_WBEMServerProduct {
};
```

CISCO_ServerSoftware.mof

```
[Version("3.1.0"), Description("WBEM Server Software"),
    Provider("jsr48:com.wbemsolutions.wbem.cimom."
                "GenericReadOnlyProvider")
 ]
class CISCO_ServerSoftware : WBEMSolutions_WBEMServerSoftware {
};
```

CISCO_SoftwareIdentity.mof

```
CISCO_SoftwareIdentity
    [Abstract,
        Description ("This is an abstract class." )]
class CISCO_SoftwareIdentity : CIM_SoftwareIdentity
{};
```

CISCO_StatisticsCollection.mof

```
CISCO_StatisticsCollection
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_StatisticsCollectionProviderImpl"
)]
class CISCO_StatisticsCollection : CIM_StatisticsCollection
{};
```

CISCO_StatisticsHostedCollection.mof

```
CISCO_StatisticsHostedCollection
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_StatisticsHostedCollectionProvider
Impl")]
class CISCO_StatisticsHostedCollection : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_PhysicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_StatisticsCollection REF Dependent;
};
```

CISCO_StatisticsHostedCollectionInComputerSystem.mof

```
CISCO_StatisticsHostedCollectionInComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_StatisticsHostedCollectionInComput
erSystemProviderImpl")]
class CISCO_StatisticsHostedCollectionInComputerSystem : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
```

```
        Max ( 1 ),
        Description ( "The scoping system." )]
    CISCO_PhysicalComputerSystem REF Antecedent;

       [Override ( "Dependent" ),
        Description (
           "The collection defined in the context of a system." )]
    CISCO_StatisticsHostedCollection REF Dependent;
};
```

CISCO_StatisticsMemberOfCollection.mof

```
CISCO_StatisticsMemberOfCollection
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_StatisticsMemberOfCollectionProvid
erImpl")]
class CISCO_StatisticsMemberOfCollection : CIM_MemberOfCollection
{
      [Override ("Collection"), Key, Aggregate, Description (
          "The Collection that aggregates members." )]
   CISCO_StatisticsCollection REF Collection;

      [Override ("Member"), Key, Description (
      "The aggregated member of the Collection." )]
   CISCO_FCPortStatistics REF Member;

};
```

CISCO_StorageComputerSystem.mof

```
CISCO_StorageComputerSystem

[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_StorageComputerSystemProviderImpl
")]
class CISCO_StorageComputerSystem : CISCO_ComputerSystem
{
    String IpAddress;
};
```

CISCO_StorageComputerSystemsInAdminDomain.mof

```
CISCO_StorageComputerSystemsInAdminDomain
    [Association,
        Description (
        "CISCO_StorageComputerSystemsInAdminDomain is a association between
CISCO_AdminDomain and "
        "CISCO_StorageComputerSystem."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_StorageComputerSystemsInAdminDomai
nProviderImpl")]
class CISCO_StorageComputerSystemsInAdminDomain : CISCO_Component
{
      [Override ("GroupComponent"), Key, Aggregate, Description (
          "The parent element in the association." )]
   CISCO_AdminDomain REF GroupComponent;
```

```
      [Override("PartComponent"), Key, Description (
       "The child element in the association." )]
   CISCO_StorageComputerSystem REF PartComponent;

};
```

CISCO_SubProfileRequiresProfile.mof

```
[Association, Version ( "3.1.0" ), Description (
       "A subprofile requires another RegisteredProfile for context. "
       "This association mandates the scoping relationship between a "
       "subprofile and its scoping profile.\n"
    "Note that this is only used for SMI profiles. Newer profiles"
    "do not have the concept of sub profiles."),
    Provider("jsr48:com.wbemsolutions.wbem.cimom."
               "GenericReadOnlyProvider")
    ]
class CISCO_SubProfileRequiresProfile : WBEMSolutions_SubProfileRequiresProfile {

      [Override ( "Antecedent" ), Min ( 1 ), Description (
          "The RegisteredProfile that is referenced/required by the "
          "subprofile.")]
   CISCO_RegisteredProfile REF Antecedent;

      [Override ( "Dependent" ), Description (
          "A RegisteredSubProfile that requires a scoping profile, for "
          "context.")]
   CISCO_RegisteredSubProfile REF Dependent;
};
```

CISCO_SubProfileSoftwareIdentity.mof

```
[Association, Version("3.1.0"),
   Provider("jsr48:com.wbemsolutions.wbem.cimom."
              "GenericReadOnlyProvider")
 ]
class CISCO_SubProfileSoftwareIdentity : WBEMSolutions_SubProfileSoftwareIdentity {

      [Override ( "Antecedent" ), Description (
          "A LogicalElement's Software Asset.")]
   CISCO_ServerSoftware REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The ManagedElement that requires or uses the software.")]
   CISCO_RegisteredSubProfile REF Dependent;

};
```

CISCO_SwitchAdded.mof

```
CISCO_SwitchAdded
    [Indication,
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchAddedProviderImpl")]
class CISCO_SwitchAdded: CIM_InstCreation
{};
```

CISCO_SwitchConformsToFabricProfile.mof

```
[Association, Version ( "3.1.0" ), Description (
       "The SMISConformsToProfile association defines the "
       "RegisteredProfiles that are conformant with a specific "
       "verision of SIM-S. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchConformsToFabricProfileProvi
derImpl")
    ]
class CISCO_SwitchConformsToFabricProfile : CIM_ElementConformsToProfile {

      [Key, Override ( "ConformantStandard" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The RegisteredProfile to which the ManagedElement conforms.")]
   CISCO_FabricProfile REF ConformantStandard;

      [Key, Override ( "ManagedElement" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The ManagedElement that conforms to the RegisteredProfile.")]
   CISCO_LogicalComputerSystem REF ManagedElement;
};
```

CISCO_SwitchConformsToSwitchProfile.mof

```
[Association, Version ( "3.1.0" ), Description (
       "The SMISConformsToProfile association defines the "
       "RegisteredProfiles that are conformant with a specific "
       "verision of SIM-S. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchConformsToSwitchProfileProvi
derImpl")
    ]
class CISCO_SwitchConformsToSwitchProfile : CIM_ElementConformsToProfile {

      [Key, Override ( "ConformantStandard" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The RegisteredProfile to which the ManagedElement conforms.")]
   CISCO_SwitchProfile REF ConformantStandard;

      [Key, Override ( "ManagedElement" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The ManagedElement that conforms to the RegisteredProfile.")]
   CISCO_PhysicalComputerSystem REF ManagedElement;
};
```

CISCO_SwitchElementCapabilities.mof

```
CISCO_SwitchElementCapabilities
[Association,
       Description("ElementCapabilities represents the association between "
       "ManagedElements and their Capabilities."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchElementCapabilitiesProviderI
mpl")]
class CISCO_SwitchElementCapabilities : CISCO_ElementCapabilities
{
      [Override ( "ManagedElement" ), Key, Min ( 1 ),
       Max ( 1 ),
       Description ( "The managed element." )]
```

```
   CISCO_PhysicalComputerSystem REF ManagedElement;

      [Override ( "Capabilities" ), Key, Description (
          "The Capabilities object associated with the element." )]
   CISCO_FCSwitchCapabilities REF Capabilities;
};
```

CISCO_SwitchHostedFCIPAccessPoint.mof

```
CISCO_SwitchHostedFCIPAccessPoint
[Association,
       Description (
           "CISCO_SwitchHostedFCIPAccessPoint is an association between an "
           "FCIP protocol endpoint and the switch on which it is provided. "
           "The cardinality of this association is one-to-many and is "
           "weak with respect to the switch.  Each switch can host many "
           "FCIP protocol endpoints."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchHostedFCIPAccessPointProvide
rImpl")]
class CISCO_SwitchHostedFCIPAccessPoint: CIM_HostedAccessPoint {

    [Override ("Antecedent"), Max (1), Min (1), Description (
        "The hosting system.")]
    CISCO_LogicalComputerSystem REF Antecedent;

    [Override ("Dependent"), Weak, Description (
        "The SAPs that are hosted on this system.")]
    CISCO_FCIPProtocolEndPoint REF Dependent;
};
```

CISCO_SwitchHostedIPAccessPoint.mof

```
CISCO_SwitchHostedIPAccessPoint
[Association,
       Description (
           "CISCO_SwitchHostedIPAccessPoint is an association between an"
           "IP protocol endpoint and the switch on which it is provided. "
           "The cardinality of this association is one-to-many and is "
           "weak with respect to the switch.  Each switch can host many "
           "IP protocol endpoints."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchHostedIPAccessPointProviderI
mpl")]
class CISCO_SwitchHostedIPAccessPoint: CIM_HostedAccessPoint {

    [Override ("Antecedent"), Max (1), Min (1), Description (
        "The hosting system.")]
    CISCO_LogicalComputerSystem REF Antecedent;

    [Override ("Dependent"), Weak, Description (
        "The SAPs that are hosted on this system.")]
    CISCO_IPProtocolEndPoint REF Dependent;
};
```

CISCO_SwitchHostedTCPAccessPoint.mof

```
CISCO_SwitchHostedTCPAccessPoint
```

```
[Association,
       Description (
           "CISCO_SwitchHostedTCPAccessPoint is an association between a"
           "TCP protocol endpoint and the switch on which it is provided. "
           "The cardinality of this association is one-to-many and is "
           "weak with respect to the switch.  Each switch can host many "
           "TCP protocol endpoints."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchHostedTCPAccessPointProvider
Impl")]
class CISCO_SwitchHostedTCPAccessPoint: CIM_HostedAccessPoint {

    [Override ("Antecedent"), Max (1), Min (1), Description (
        "The hosting system.")]
    CISCO_LogicalComputerSystem REF Antecedent;

    [Override ("Dependent"), Weak, Description (
        "The SAPs that are hosted on this system.")]
    CISCO_TCPProtocolEndPoint REF Dependent;
};
```

CISCO_SwitchInstalledSoftwareIdentity.mof

```
CISCO_SwitchInstalledSoftwareIdentity
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchInstalledSoftwareIdentityPro
viderImpl")]
class CISCO_SwitchInstalledSoftwareIdentity : CISCO_InstalledSoftwareIdentity
{
      [Override ( "System" ), Key, Description (
          "The system on which the software is installed." )]
   CISCO_PhysicalComputerSystem REF System;

      [Override ( "InstalledSoftware" ), Key, Description (
      "The SoftwareIdentity that is installed." )]
   CISCO_SwitchSoftwareIdentity REF InstalledSoftware;
};
```

CISCO_SwitchProfile.mof

```
[Version ( "3.1.0" ), Description (
       "A RegisteredProfile describes a set of CIM Schema classes with "
       "required properties and/or methods, necessary to manage a "
       "real-world entity or to support a usage scenario, in an "
       "interoperable fashion. RegisteredProfiles can be defined by "
       "the DMTF or other standards organizations. Note that this "
       "class should not be confused with CIM_Profile, which collects "
       "SettingData instances, to be applied as a 'configuration "
       "profile' for an element. \n"
       "A RegisteredProfile is a named 'standard' for CIM-based "
       "management of a particular System, subsystem, Service or other "
       "entity, for a specified set of uses. It is a complete, "
       "standalone definition, as opposed to the subclass "
       "RegisteredSubProfile, which requires a scoping profile for "
       "context. \n"
       "The uses for a RegisteredProfile or SubProfile MUST be "
       "specified in the document that defines the profile. Examples "
       "of Profiles are to manage various aspects of an Operating "
       "System, Storage Array, or Database. The name of the profile is "
```

```
       "defined and scoped by its authoring organization."),
    Provider("jsr48:com.wbemsolutions.wbem.cimom."
               "GenericReadOnlyProvider")
   ]
class CISCO_SwitchProfile : CISCO_RegisteredProfile {
};
```

CISCO_SwitchRemoved.mof

```
CISCO_SwitchRemoved
    [Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchRemovedProviderImpl")]
class CISCO_SwitchRemoved: CIM_InstDeletion
{};
```

CISCO_SwitchSettingData.mof

```
CISCO_SwitchSettingData
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchSettingDataProviderImpl")]
class CISCO_SwitchSettingData : CISCO_ElementSettingData
{

      [Override ( "ManagedElement" ), Key,
      Description ( "The managed element." )]
   CISCO_PhysicalComputerSystem REF ManagedElement;

      [Override ( "SettingData" ), Key, Description (
          "The SettingData object associated with the element." )]
   CISCO_FCSwitchSettings REF SettingData;
};
```

CISCO_SwitchSoftwareIdentity.mof

```
CISCO_SwitchSoftwareIdentity
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_SwitchSoftwareIdentityProviderImp
l")]
class CISCO_SwitchSoftwareIdentity : CISCO_SoftwareIdentity
{};
```

CISCO_SystemDevice.mof

```
CISCO_SystemDevice
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_SystemDevice : CIM_SystemDevice
{};
```

CISCO_TCPElementSettingData.mof

```
CISCO_TCPElementSettingData
[Association,
```

```
        Description ("This corresponds to the association between CISCO_IPProtocolEndpoint
"
                "and CISCO_IPSettings. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_TCPElementSettingDataProviderImpl"
)]
class CISCO_TCPElementSettingData : CIM_ElementSettingData
{
        [Override ("ManagedElement"), Description (
                "Reference to CISCO_IPProtocolEndpoint instance.")]
        CISCO_TCPProtocolEndpoint REF ManagedElement;

        [Override ("SettingData"), Description (
                "Reference to CISCO_FCIPSettings instance.")]
        CISCO_TCPSettings REF SettingData;
};
```

CISCO_TCPEndPointStatisticalData.mof

```
CISCO_TCPEndPointStatisticalData
[Association,
      Description ("CISCO_IPEndPointStatistics is an association that associates "
                "CISCO_IPProtocolEndPoint to its StatisticalData "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_TCPEndPointStatisticalDataProvider
Impl")]
class CISCO_TCPEndPointStatisticalData :  CIM_ElementStatisticalData {

        [Override ("ManagedElement"), Description (
                "Reference to CISCO_IPProtocolEndpoint instance.")]
         CISCO_TCPProtocolEndPoint REF ManagedElement;

         [Override("Stats"), Key, Description (
                "The statistic information.")]
         CISCO_TCPEndPointStatistics REF Stats;
};
```

CISCO_TCPEndPointStatistics.mof

```
CISCO_TCPEndPointStatistics
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_TCPEndPointStatisticsProviderImpl
")]
class CISCO_TCPEndPointStatistics : CIM_TCPEndpointStatistics
{

    [Description ("The TCP Keep alive timeout for all links within this entity.")]
    uint32 KeepAliveTimeOut;

    [Description ("The Maximum number of times that the same item of data"
          "will be retransmitted over a TCP connection. If delivery"
          "is not acknowledged after this number of retransmissions"
         " then the TCP connection is terminated. ")]
    uint32 MaxReTransmission;

    [Description (
         "The path MTU discovery is enabled if the value of this object is true(1), else
it is disabled, and"
         "has the value false(2). "
    )]
    boolean PathMTU;
```

```
    [Description (
         "The time interval for which the discovered pathMTU is valid, before MSS reverts
back to the negotiated"
         "TCP value. This object is writeble only if"
         "cfmFcipEntityExtPMTUEnable is 'true'. "
    )]
    uint32 PathMTUResetTimeOut;

    [Description(
         "The TCP minimum retransmit timeout for all the links on"
         "this entity. "
    )]
    uint32 MinimumRTO;

    [Description (
         "The aggregate TCP send window for all TCP connections on all"
         " Links within this entity. This value is used for Egress"
          "Flow Control. When the aggregate of the data queued"
          "on all connections within this entity reaches this value,"
         " the sender is flow controlled. "
    )]
    uint32 SendBufSize ;

    [Description (
         "This is an estimate of the Bandwidth of the network pipe used for the B-D
product computation,which lets decide"
         "the TCP receive window to advertise."
         "The cfmFcipEntityExtTcpMaxBW,cfmFcipEntityExtTcpMinAvailBW,"
         "cfmFcipEntityExtTcpRndTrpTimeEst must be set in the same"
         "SNMP set request. SET operation would fail if this object"
         "is set individually."
    )]
    uint32 TcpMaxBW;

    [Description (
         "The minimum available bandwidth for the TCP connections"
        " on the Links within this entity."
        " The cfmFcipEntityExtTcpMaxBW,cfmFcipEntityExtTcpMinAvailBW,"
        " cfmFcipEntityExtTcpRndTrpTimeEst must be set in the same"
        " SNMP set request. SET operation would fail if this object"
        " is set individually. "
    )]
    uint32 TcpMinAvailBW;


    [Description (
         "This is an estimate of the round trip delay of the network"
         " pipe used for the B-D product computation, which lets us"
         " derive the TCP receive window to advertise."
         " The cfmFcipEntityExtTcpMaxBW,cfmFcipEntityExtTcpMinAvailBW,"
         " cfmFcipEntityExtTcpRndTrpTimeEst must be set in the same"
         " SNMP set request. SET operation would fail if this object"
         " is set individually. microseconds "
    )]
    uint32 TcpRndTrpTimeEst;

    [Description (
         "This object is used for enabling/disabling the congestion"
          "window monitoring. If the value of this object is true(1),"
         " it is enabled. It is disabled if the value is false(2). "
    )]
    boolean ExtCWMEnable=true;
```

```
    [Description (
         "The maximum burst sent after a tcp sender idle period. This object is writeble
only if"
         "  cfmFcipEntityExtCWMEnable is 'true'. kilobytes"
    )]
    uint32 CWMBurstSize;

    [Description (
         "The maximum delay variation that is not due to"
         " congestion that can be experienced by TCP"
         " connections for all links on this enti ty in milliseconds."
    )]
    uint32 MaxJitter;
};
```

CISCO_TCPIPEndpoint.mof

```
CISCO_TCPIPEndpoint
[Association,
       Description ("This corresponds to the association between CISCO_IPProtocolEndpoint
"
           "and CISCO_TCPProtocolEndPoint. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_TCPIPEndpointProviderImpl")]
class CISCO_TCPIPEndpoint : CIM_BindsTo {

    [Override ( "Antecedent" ), Description (
        "The underlying IP Endpoint, which is depended upon.")]
    CISCO_IPProtocolEndpoint REF Antecedent;

    [Override ( "Dependent" ), Description (
        "The TCP ProtocolEndpoint dependent on the IP Endpoint.")]
    CISCO_TCPProtocolEndPoint REF Dependent;
};
```

CISCO_TCPProtocolEndpoint.mof

```
CISCO_TCPProtocolEndpoint
[Description ("A protocol endpoint that is dedicated to running TCP. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_TCPProtocolEndPointProviderImpl")]
class CISCO_TCPProtocolEndPoint: CIM_TCPProtocolEndPoint {

    [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("PortNumber"), Description (
        "The TCP port number for data connection traffic.")]
    uint32 PortNumber = 3226;

    [Override("ProtocolIFType"), Description (
        "ProtocolIFType's enumeration is limited to TCP and reserved "
        "values for this subclass of ProtocolEndpoint."),
        ValueMap { "1", "222..4095", "4111", "4116..32767", "32768.." },
        Values { "Other", "IANA Reserved", "TCP", "DMTF Reserved", "Vendor Reserved" }]
    uint16 ProtocolIFType = 4111;


    [Description ("The default maximum TCP Receiver Window size for this TCP connection")]
    uint32 ReceiverWindowSize;

    [Description ("The TCP Maximum Segment Size (MSS) for this TCP connection. ")]
    uint32 MaximumSegmentSize;

    [Description ("The timeout value for this TCP connection.")]
    uint32 ConnectionTimeOut;

    [Description ("Remote Port number for this TCP Connection.")]
    uint32 RemotePortNumber;

    [Description (
         "The nature of messages that get transmitted on this TCP connection."),
        ValueMap { "1", "2", "3"},
        Values { "Control", "Data", "Both" }]
    string ConnectionPurpose;
};
```

CISCO_TCPSettings.mof

```
CISCO_TCPSettings
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_TCPSettingsProviderImpl")]
class CISCO_TCPSettings : CIM_TCPSettings {

      [Write, Description (
          "Sets the TCP/IP retransmit time for an initial SYN segment "
          "when establishing a connection. The suggested value is 240 "
```

```
       "seconds based on RFC 793. "),
     Units ("Seconds"), MinValue (30), MaxValue (240)  ]
uint8 WaitTimeTotalTime = 240;
   [Write, Description (
       "Selects TCP/IP delayed acknowledgements (ACKs) if set to 1 "
       "(default), and selects immediate ACKs if set to 0. If "
       "delayed ACKs are set, TCP/IP does not send an ACK "
       "immediately on receiving data. TCP/IP normally delays "
       "sending an ACK to improve the chance that it can bundle "
       "it with transmitted data ") ]
boolean DelayedACKsEnabled;
   [Write, Description (
       "Control system-wide implementation of TCP/IP performance "
       "extensions including timestamps and large window scaling "
       "(as defined in RFC 1323). These features provide more "
       "efficient and reliable usage of high-bandwidth, "
       "high-latency links. If set to 1 (the default), negotiation "
       "is turned on and will permit a TCP/IP receive window size "
       "as large as 1,073,725,440 bytes (just under 1GB). "
       "If set to 0, negotiation is disabled and the largest "
       "possible window size is 65,535 bytes (64KB-1)."
       "Window size negotiation may be disabled on a per-interface "
       "basis by specifying the -rfc1323 option to ifconfig(1Mtcp). "
       "This is necessary for PPP and SLIP interfaces that allow "
       "header compression. ") ]
boolean WindowSizeNegotiationEnabled;
   [Write, Description (
       "Enable RFC1323 TCP timestamps. ") ]
boolean TimeStampsEnabled;
   [Write, Description (
       "Sets the idle time before TCP/IP keepalives are sent "
       "(if enabled). The default value is 7200 seconds. "
       "The suggested value is 7200 seconds. "),
     Units ("Seconds"), MinValue (300), MaxValue (86400)  ]
uint32 KeepAliveIdleTime = 7200;
   [Write, Description (
       "Sets the number of TCP/IP keepalives that will be sent before "
       "giving up. The suggested value is 8. "),
     MinValue (1), MaxValue (256)  ]
uint16 KeepAlivePackets = 8;
   [Write, Description (
       "Sets the TCP/IP keepalive interval between keepalive packets "
       "once they start being sent. The suggested value is 75."),
     Units ("Seconds"), MinValue (1), MaxValue (43200)  ]
uint16 KeepAliveInterval = 75;
[Write, Description (
       "Sets the number of keep-alive probes to be sent per slow timer "
       "run. The suggested value is 5. "),
     MinValue (1), MaxValue (64)  ]
uint8 MaxKeepAliveProbes = 5;
   [Write, Description (
       "Default maximum segment size used in communicating with remote "
       "networks. The suggested value is 512."),
     Units ("Bytes"), MinValue (512), MaxValue (32768)  ]
uint32 DefaultSegmentSize = 512;
   [Write, Description (
       "Enable RFC2018 TCP Selective Acknowledgements. ") ]
boolean SelectiveACKsEnabled;
   [Write, Description (
       "Enable TCP syncookies. The kernel must be compiled with "
       "CONFIG_SYN_COOKIES. Syncookies protects a socket from overload "
       "when too many connection attempts arrive. Client machines may "
       "not be able to detect an overloaded machine with a short timeout "
       "anymore when syncookies are enabled. ") ]
```

```
   boolean SYNCookiesEnabled;
};
```

CISCO_TempAlert.mof

```
CISCO_TempAlert
[Indication,
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_TempAlertProviderImpl")]
class CISCO_TempAlert: CISCO_EnvironmentalAlert
{
       uint32 SensorValue;
       uint32 SensorThresholdValue;
       uint32 SensorThresholdIndex;
};
```

CISCO_UserAddedOnSwitch.mof

```
CISCO_UserAddedOnSwitch
    [Indication,
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_UserAddedProviderImpl")]
class CISCO_UserAddedOnSwitch: CISCO_SecurityAlert
{};
```

CISCO_UserLoginFailed.mof

```
CISCO_UserLoginFailed
[Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_UserLoginFailedProviderImpl")]
class CISCO_UserLoginFailed: CISCO_SecurityAlert
{};
```

CISCO_UserModifiedOnSwitch.mof

```
CISCO_UserModifiedOnSwitch
    [Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_UserModifiedProviderImpl")]
class CISCO_UserModifiedOnSwitch: CISCO_SecurityAlert
{};
```

CISCO_UserRemovedOnSwitch.mof

```
CISCO_UserRemovedOnSwitch
    [Indication,
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_UserRemovedProviderImpl")]
class CISCO_UserRemovedOnSwitch: CISCO_SecurityAlert
{};
```

CISCO_Vsan.mof

```
CISCO_Vsan
    [Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_VsanProviderImpl")]
class CISCO_Vsan : CIM_AdminDomain {
```

```
    string LoadBalancingType;
    string  InteropMode;
};
```

CISCO_VSANChanged.mof

```
CISCO_VSANChanged
[Indication,
       Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_VsanChangedProviderImpl")]
class CISCO_VSANChanged: CISCO_AlertIndication
{};
```

CISCO_VsanComputerSystemComponent.mof

```
CISCO_VsanComputerSystemComponent
[Association,
       Description (
       "CISCO_VsanComputerSystemComponent is a association between CISCO_Vsan and "
       "CISCO_LogicalComputerSystem."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_VsanComputerSystemComponentProvide
rImpl")]
class CISCO_VsanComputerSystemComponent : CISCO_Component
{
      [Override ("GroupComponent"), Key, Aggregate, Description (
          "The parent element in the association." )]
   CISCO_Vsan REF GroupComponent;

      [Override("PartComponent"), Key, Description (
       "The child element in the association." )]
   CISCO_LogicalComputerSystem REF PartComponent;

};
```

CISCO_VsanConformsToFabricProfile.mof

```
[Association, Version ( "3.1.0" ), Description (
       "The SMISConformsToProfile association defines the "
       "RegisteredProfiles that are conformant with a specific "
       "verision of SIM-S. "),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_VsanConformsToFabricProfileProvide
rImpl")
    ]
class CISCO_VsanConformsToFabricProfile : CIM_ElementConformsToProfile {

      [Key, Override ( "ConformantStandard" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The RegisteredProfile to which the ManagedElement conforms.")]
   CISCO_FabricProfile REF ConformantStandard;

      [Key, Override ( "ManagedElement" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The ManagedElement that conforms to the RegisteredProfile.")]
   CISCO_Vsan REF ManagedElement;
};
```

CISCO_VsanZoneCapabilities.mof

```
CISCO_VsanZoneCapabilities
    [Association,
        Description("ElementCapabilities represents the association between "
        "ManagedElements and their Capabilities."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_VsanZoneCapabilitiesProviderImpl")
]
class CISCO_VsanZoneCapabilities : CISCO_ElementCapabilities
{
      [Override ( "ManagedElement" ), Key, Min ( 1 ),
       Max ( 1 ),
       Description ( "The managed element." )]
   CISCO_Vsan REF ManagedElement;

      [Override ( "Capabilities" ), Key, Description (
          "The Capabilities object associated with the element." )]
   CISCO_ZoneCapabilities REF Capabilities;
};
```

CISCO_Zone.mof

```
CISCO_Zone
    [Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneProviderImpl")]
class CISCO_Zone : CIM_Zone
{};
```

CISCO_ZoneAlert.mof

```
CISCO_ZoneAlert
    [Indication,
        Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneAlertProviderImpl")]
class CISCO_ZoneAlert : CISCO_AlertIndication
{
    uint32 VsanId;
};
```

CISCO_ZoneAlias.mof

```
CISCO_ZoneAlias
    [Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneAliasProviderImpl")]
class CISCO_ZoneAlias : CIM_NamedAddressCollection
{};
```

CISCO_ZoneAliasForZone.mof

```
CISCO_ZoneAliasForZone
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneAliasForZoneProviderImpl")]
class CISCO_ZoneAliasForZone : CISCO_ZoneMemberOfCollection
{
      [Override("Collection"), Key, Aggregate, Description (
          "The Collection that aggregates members." )]
   CISCO_Zone REF Collection;
```

```
      [Override("Member"), Key, Description ( "The aggregated member of the Collection."
)]
   CISCO_ZoneAlias REF Member;

};
```

CISCO_ZoneAliasInVsan.mof

```
CISCO_ZoneAliasInVsan
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneAliasInVsanProviderImpl")]
class CISCO_ZoneAliasInVsan : CISCO_ZoneHostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_Vsan REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_ZoneAlias REF Dependent;
};
```

CISCO_ZoneAliasSettingData.mof

```
CISCO_ZoneAliasSettingData
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneAliasSettingDataProviderImpl")
]
class CISCO_ZoneAliasSettingData : CISCO_ElementSettingData
{

      [Override ( "ManagedElement" ), Key,
      Description ( "The managed element." )]
   CISCO_ZoneAlias REF ManagedElement;

      [Override ( "SettingData" ), Key, Description (
          "The SettingData object associated with the element." )]
   CISCO_ZoneMemberSettingData REF SettingData;
};
```

CISCO_ZoneCapabilities.mof

```
CISCO_ZoneCapabilities
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneCapabilitiesProviderImpl")]
class CISCO_ZoneCapabilities : CIM_ZoneCapabilities
{};
```

CISCO_ZoneCapInAdminDomain.mof

```
CISCO_ZoneCapInAdminDomain
[Association,
       Description("ElementCapabilities represents the association between "
```

```
        "ManagedElements and their Capabilities."),

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneCapInAdminDomainProviderImpl")
]
class CISCO_ZoneCapInAdminDomain : CISCO_ElementCapabilities
{
      [Override ( "ManagedElement" ), Key, Min ( 1 ),
       Max ( 1 ),
       Description ( "The managed element." )]
   CISCO_AdminDomain REF ManagedElement;

      [Override ( "Capabilities" ), Key, Description (
          "The Capabilities object associated with the element." )]
   CISCO_ZoneCapabilities REF Capabilities;
};
```

CISCO_ZoneHostedCollection.mof

```
CISCO_ZoneHostedCollection
    [Abstract,
        Association,
       Description ("This is an abstract association." )]
class CISCO_ZoneHostedCollection : CISCO_HostedCollection
{};
```

CISCO_ZoneInLogicalComputerSystem.mof

```
CISCO_ZoneInLogicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneInLogicalComputerSystemProvide
rImpl")]
class CISCO_ZoneInLogicalComputerSystem : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_LogicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_Zone REF Dependent;
};
```

CISCO_ZoneInPhysicalComputerSystem.mof

```
CISCO_ZoneInPhysicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneInPhysicalComputerSystemProvid
erImpl")]
class CISCO_ZoneInPhysicalComputerSystem : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
```

```
        Description ( "The scoping system." )]
   CISCO_PhysicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_Zone REF Dependent;
};
```

CISCO_ZoneInVsan.mof

```
CISCO_ZoneInVsan
[Association,
       Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneInVsanProviderImpl")]
class CISCO_ZoneInVsan : CISCO_ZoneHostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_Vsan REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_Zone REF Dependent;
};
```

CISCO_ZoneMemberOfCollection.mof

```
CISCO_ZoneMemberOfCollection
[Abstract,
       Association,
       Description ("This is an abstract association." )]
class CISCO_ZoneMemberOfCollection : CIM_MemberOfCollection
{};
```

CISCO_ZoneMemberSettingData.mof

```
CISCO_ZoneMemberSettingData
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneMemberSettingDataProviderImpl
")]
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. \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."
     "* A ConnectivityMemberType of 13 (Device alias) "
            "indicates that a device alias"
            "should be specified in the "
            "related ConnectivityMemberID property.")]
    uint16 ConnectivityMemberType;
};
```

CISCO_ZoneService.mof

```
CISCO_ZoneService
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneServiceProviderImpl")]
class CISCO_ZoneService : CIM_ZoneService
{};
```

CISCO_ZoneServiceInAdminDomain.mof

```
CISCO_ZoneServiceInAdminDomain
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneServiceInAdminDomainProviderIm
pl")]
class CISCO_ZoneServiceInAdminDomain : CISCO_HostedService
{       [Override ( "Antecedent" ),
        Min ( 1 ),
        Max ( 1 ),
        Description ( "The hosting System." )]
    CISCO_AdminDomain REF Antecedent;

       [Override ( "Dependent" ),
        Weak, Description ( "The Service hosted on the System." )]
    CISCO_ZoneService REF Dependent;
```

```
};
```

CISCO_ZoneServiceInVsan.mof

```
CISCO_ZoneServiceInVsan
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneServiceInVsanProviderImpl")]
class CISCO_ZoneServiceInVsan : CISCO_HostedService
{       [Override ( "Antecedent" ),
        Min ( 1 ),
        Max ( 1 ),
        Description ( "The hosting System." )]
   CISCO_Vsan REF Antecedent;

       [Override ( "Dependent" ),
        Weak, Description ( "The Service hosted on the System." )]
   CISCO_ZoneService REF Dependent;

};
```

CISCO_ZoneSet.mof

```
CISCO_ZoneSet
[Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneSetProviderImpl")]
class CISCO_ZoneSet : CIM_ZoneSet
{};
```

CISCO_ZoneSetAlert.mof

```
CISCO_ZoneSetAlert
[Indication,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneSetAlertProviderImpl")]
class CISCO_ZoneSetAlert: CISCO_AlertIndication
{
    string ZoneSetName;
    uint32 VsanId;
};
```

CISCO_ZoneSetInAdminDomain.mof

```
CISCO_ZoneSetInAdminDomain
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneSetInAdminDomainProviderImpl")
]
class CISCO_ZoneSetInAdminDomain : CISCO_ZoneHostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_AdminDomain REF Antecedent;

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

```
          "The collection defined in the context of a system." )]
   CISCO_ZoneSet REF Dependent;
};
```

CISCO_ZoneSetInLogicalComputerSystem.mof

```
CISCO_ZoneSetInLogicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneSetInLogicalComputerSystemProv
iderImpl")]
class CISCO_ZoneSetInLogicalComputerSystem : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_LogicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_ZoneSet REF Dependent;
};
```

CISCO_ZoneSetInPhysicalComputerSystem.mof

```
CISCO_ZoneSetInPhysicalComputerSystem
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneSetInPhysicalComputerSystemPro
viderImpl")]
class CISCO_ZoneSetInPhysicalComputerSystem : CISCO_HostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_PhysicalComputerSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_ZoneSet REF Dependent;
};
```

CISCO_ZoneSetInVsan.mof

```
CISCO_ZoneSetInVsan
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneSetInVsanProviderImpl")]
class CISCO_ZoneSetInVsan : CISCO_ZoneHostedCollection
{
      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   CISCO_Vsan REF Antecedent;
```

```
      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   CISCO_ZoneSet REF Dependent;
};
```

CISCO_ZoneSettingData.mof

```
CISCO_ZoneSettingData
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZoneSettingDataProviderImpl")]
class CISCO_ZoneSettingData : CISCO_ElementSettingData
{

      [Override ( "ManagedElement" ), Key,
      Description ( "The managed element." )]
   CISCO_Zone REF ManagedElement;

      [Override ( "SettingData" ), Key, Description (
          "The SettingData object associated with the element." )]
   CISCO_ZoneMemberSettingData REF SettingData;
};
```

CISCO_ZonesInZoneSet.mof

```
CISCO_ZonesInZoneSet
[Association,

Provider("jsr48:com.cisco.dcbu.smis.provider.impl.CISCO_ZonesInZoneSetProviderImpl")]
class CISCO_ZonesInZoneSet : CISCO_ZoneMemberOfCollection
{
      [Override("Collection"), Key, Aggregate, Description (
          "The Collection that aggregates members." )]
   CISCO_Zoneset REF Collection;

      [Override("Member"), Key, Description ( "The aggregated member of the Collection."
)]
   CISCO_Zone REF Member;

};
```

DCNM for SAN Web Services API

This chapter describes the DCNM for SAN (DCNM-SAN) Web Services application program interface (API). This chapter defines the APIs exposed by the Cisco DCNM for SAN Web Services feature.

This chapter includes the following sections:

Introduction to Cisco DCNM for SAN Web Services

Cisco DCNM for SAN (DCNM-SAN) Web Services provide application programming interfaces (APIs) that expose Cisco DCNM-SAN core software functionalities as remote procedure calls to third-party vendors. Software developers can use the APIs to design computer applications that interact with the Cisco DCNM-SAN Server over the network.

With Cisco DCNM-SAN, you can monitor MDS switch events, performance, and inventory, and you can perform administrative tasks. Applications can access Cisco DCNM-SAN Web Services through many protocols and data formats such as HTTP, HTTPS, XML, SOAP, and WSDL.

Cisco DCNM-SAN Web Services provide cross-platform operations. Web Services can interact with .NET applications, C++ applications, and applications written in other programming languages and Web Services must adhere to accepted conventions to make services interoperable with other applications. For this reason, Cisco DCNM-SAN Web Services must follow the Java API for XML Web Services (JAX-WS) specification. Cisco DCNM-SAN Web Services relies on JBoss Web Service (JBoss WS) as

a service endpoint engine and as an entry point into the JAX-WS programming model. The framework also allows Cisco DCNM-SAN Web Services to become an integral part of the Cisco DCNM-SAN Server run-time environment.

Web Services Specifications

Web Services specifications combine together to provide interoperable protocols for security, communication, and syntax for representing data.

XML

XML is the data format that defines the structure of the message. XML Web Services architecture allows programs written in different languages on different platforms to communicate with each other in a standards-based way. XML Web Services expose useful functionality to Web users through a standard Web protocol (SOAP).

SOAP

Simple Object Access Protocol (SOAP) is the communications protocol for Web Services. SOAP is a specification that defines the XML format for messages. The advantage of SOAP is that it has been implemented on many different hardware and software platforms.

HTTP/HTTPS

HTTP/HTTPS is the transport layer of the service. HTTP/HTTPS allows data to traverse the network easily and is widely accepted. It is also considered as platform neutral. Every Cisco DCNM-SAN Web Services operation is through HTTP/HTTPS.

WDSL

A WSDL definition is an XML document with a root definition element from the http://schemas.xmlsoap.org/wsdl/ namespace. Cisco DCNM-SAN Web Services uses the WSDL document to publish which operations of DCNM-SAN are available. The definitions element can contain several other elements including types, message, portType, binding, and service, all of which come from the namespace. WSDL is published on FMServer at:

http://localhost/LogonWSService/LogonWS?wsdl

Logon Service

LogonWS makes IdentityManager's operations available as Web Service calls. LogonWS allows the following operations:

requestToken

This method returns a token string that must be passed in as the header of the SOAP message. Once the username and password is authenticated using DCNM-SAN's SecurityManager, the token is generated and is kept valid for the number of milliseconds specified in the expiration argument.

Parameters

username—Name of the user.

password—Password of the user.

expiration—Time (in milliseconds).

Return Value

Session token.

Error

Error code: 201—Invalid argument in Web Service exception.

requestLogonRole

Parameters

username—Name of the user.

password—Password of the user.

expiration—Time (in milliseconds).

Return Value

Session token.

Error

Error code: 201—Invalid argument in Web Service exception.

requestLogonToken

Parameters

username—Name of the user.

password—Password of the user.

expiration—Time (in milliseconds).

Return Value

Session token.

Error

Error code: 201—Invalid argument in Web Service exception.

getCredentialByToken

Parameters

username—Name of the user.

password—Password of the user.

expiration—Time (in milliseconds).

Return Value

Session token.

Error

Error code: 201—Invalid argument in Web Service exception.

validateToken

This method returns true or false depending on the validity of the token. If the token has expired, it returns false, or else it returns true.

Parameters

token—Session Token.

Return Value

Boolean value is True if DCNM-SAN accepts the token.

Error

Error code: 201—Invalid argument in Web Service exception.

Authentication or Token

To interact with DCNM-SAN Web Services, you must obtain a token through LogonWS and attach this token to the header message of every SOAP requests. DCNM-SAN Web Services verifies user credentials using a unique token string that is administered by LogonWS. At any given time, HTTPS should be deployed to secure the communication channel. The following example displays the format of the header message:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
```

```
    <m:Token xmlns:m="http://www.w3schools.com/transaction/">
      token string is put here
    </m:Token></SOAP-ENV:Header>
  <SOAP-ENV:Body>
   <getFabrics xmlns="http://ep.jaxws.dcbu.cisco.com/"/>
  </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

IdentityManager

IdentityManager provides identity services and manages the user credentials that are required by Web Services. It is the token provider that administers and maintains tokens. It authenticates the user, generates tokens, and validates or expires tokens by periodically checking and clearing the cache.

Sas WS

SAN Service is an Enterprise Java Beans (EJB) component that manages SAN-related service requests and executes queries on DCNM-SAN for information. San WS checks with IdentityManager for authentication before performing the request. A valid token string indicates to San Service that the user is a DCNM-SAN user and that it must honor and execute the request. After retrieving the required information, it sends the result back to the user. SanWS logs errors in fms_ws.log. The service end-point interface (SEI) of SanWS defines the operations of the service, and sends them to the end users.

getFabrics

Returns the list of all open fabrics.

Return Value

An array of open fabrics.

Error

Error Code: 300— General SAN Service exception.

getFabricByIP

Returns the list of fabrics associated with the IP address of a given switch.

Parameters

ipAddress—IP address of the switch.

Return Value

List of all fabrics associated with the specific IP address.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getPmEntity

Returns the PM entity from the database.

Return Value

PM entity from the database.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getPmChartData

Returns the PM statistics of a specific RRD file.

Return Value

PM statistics of a specific RRD file.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFabricByKey

Returns the list of fabrics associated with the specified key.

Parameters

key—Key of the fabric.

Return Value

List of all fabrics associated with the specified key.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFabricBySwitchKey

Returns the list of fabrics associated with the specified seed switch key (WWN).

Parameters

swkey—Seed switch key of the fabric.

Return Value

List of all fabrics associated with the specified seed switch key.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getSwitchesByFabric

Returns the list of switches associated with the specified fabric key.

Parameters

key—Key of the fabric.

Return Value

List of all fabrics associated with the specified fabric key.

Error

Error Code: 300— General SAN Service exception.

getNeighborSwitches

Returns the list of neighboring switches associated with the specified WwnKey.

Parameters

key—WwnKey object.

Return Value

List of neighboring switches associated with the specified WwnKey.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

Error Code: 302—SAN does not find objects by query key exception.

getActiveServerNodes

Returns the list of all active DCNM for SAN servers.

Parameters

key—Key of the fabric.

Return Value

List of all fabric servers that are active.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFabricWithSnmpCredentail

Returns the list of fabrics (except the fabric with opening status) with the SNMP credentials.

Parameters

key—Key of the fabric.

Return Value

List of all fabrics with their SNMP credentials.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getSwitchesByFabric

Returns the list of switches associated with the specified fabric key.

Parameters

key—Key of the fabric.

Return Value

List of all fabrics associated with the specified fabric key.

Error

Error Code: 300— General SAN Service exception.

getSwitch

Returns the list of switches on all the fabrics.

Parameters

key—Key of the fabric.

Return Value

List of all fabrics associated with the specified fabric key.

Error

Error Code: 300— General SAN Service exception.

getSwitchByKey

Returns the switch associated with the specified switch key object.

Parameters

key—Key of the fabric.

Return Value

Switch associated with the specified switch key.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getSwitchIPByName

Returns the IP address associated with the specified system name or switch name.

Parameters

sysname—Name of the system or switch.

Return Value

IP address associated with the specified system name.

Error

Error Code: 300— General SAN Service exception.

Error code: 201—Invalid argument in Web Service exception.

getSwitchIPByKey

Returns the IP address of the switch associated with the specified WwnKey object.

Parameters

key—WwnKey object.

Return Value

IP address associated with the specified WwnKey object.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getNeighborSwitches

Returns the list of neighboring switches associated with the specified WwnKey.

Parameters

key—WwnKey object.

Return Value

List of neighboring switches associated with the specified WwnKey.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

Error Code: 302—SAN does not find objects by query key exception.

getVsans

Returns the list of VSANs in the fabric associated with the specified fabric key.

Parameters

key—Fabric key object.

Return Value

List of VSANs in the fabric associated with the specified fabric key.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getVsan

Returns the VSAN in the fabric associated with the specified VSAN key object.

Parameters

key—VSAN key object.

Return Value

VSANs in the fabric associated with the specified VSAN key object.

Error

Error Code: 300— General SAN Service exception.

Error code: 201—Invalid argument in Web Service exception.

getIsls

Returns the list of ISLs in the VSAN associated with the specified VSAN key.

Parameters

key—VSAN key.

Return Value

Array of ISL objects in the VSAN associated with the specified VSAN key.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

discoverFabric

This API opens the fabric. This function requires the IP address of the seed switch and SNMP credentials.

Parameters

seed—IP address of the seed switch.

user—SNMP credential.

Return Value

Boolean value is True if the discovery was successful.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

Error Code: 100— Authentication failure exception.

Error Code: 101—Invalid credentials exception.

manageFabric

Returns true or false depending on manageability of the fabric.

Parameters

key—Fabric key.

Return Value

Returns true if the fabric can be identified or managed. Returns false if the fabric cannot be identified or managed.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

unManageFabric

This function unmanages a fabric.

Parameters

key—Fabric key.

Return Value

None.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

closeFabric

This function unmanages and closes a fabric.

Parameters

key—Fabric key.

Return Value

None.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

purgeFabric

This function purges the specified fabric data both from the DCNM-SAN cache and database.

Parameters

key—Fabric key.

Return Value

None.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

Error Code: 302—SAN does not find objects by query key exception.

getEndports

Returns the list of all the end ports known to DCNM-SAN.

Return Value

An array of all the end ports.

Error

Error Code: 300— General SAN Service exception.

getEnclosures

Returns the list of all the enclosures known to DCNM-SAN.

Return Value

An array of enclosure objects.

Error

Error Code: 300— General SAN Service exception.

getEndPortByKey

Returns the end port based on the switch WWN.

Parameters

key—WWN of the node.

Return Value

Returns the end port based on the switch WWN. Returns null if there are no end ports associated with the switch.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEndPortAttachedToSw

Returns the end ports that are associated with a switch.

Parameters

key—IP address of the switch.

Return Value

Returns the end ports based on switch.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnclosureByName

Returns the enclosure based on the name.

Parameters

name—Name of the enclosure object.

Return Value

Returns the enclosure object.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnclosureByKey

Returns the enclosure based on the name.

Parameters

name—Name of the enclosure object.

Return Value

Returns the enclosure object.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnclosureByPWwn

Returns the enclosures that are associated with a physical WWN.

Parameters

wwn—Physical WWN of the switch.

Return Value

Returns the enclosure based on physical WWN.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

updateEnclosure

Update the enclosure with the value that is passed as parameter.

Parameters

value—Value to update the enclosure.

Return Value

None.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

updateEndportEnclosure

Update the end port enclosure with the value that is passed as parameter.

Parameters

endportKey—Value for the end port key.

enclosureKey—Value for the enclosure key.

Return Value

None.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getHosts

Returns the list of all the host enclosures known to DCNM-SAN.

Return Value

Returns the list of all the host enclosures known to DCNM-SAN.

Error

Error Code: 300— General SAN Service exception.

getHost

Returns the name of hosts in a VSAN.

Parameters

key—Name of the VSAN.

Return Value

Returns the name of the hosts in tthe specified VSAN.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getHostByFabric

Returns the name of hosts in a fabric.

ValidationException is displayed if any of the following situation occurs:

- If the argument passed is null.
- If the argument does not contain a valid key.

Parameters

key—Name of the fabric.

Return Value

Returns the name of the hosts in the specified VSAN.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getStorages

Returns the list of all the storage device enclosures known to DCNM-SAN.

Return Value

An array of all the storage device enclosures known to DCNM-SAN.

Error

Error Code: 300— General SAN Service exception.

getStorageByFabric

Returns the name of storage device enclosures in a fabric.

Parameters

key—Name of the fabric.

Return Value

Returns the name of the storages in the specified fabric.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getHostPorts

Returns the list of all the host end ports in a fabric.

Parameters

key—Name of the fabric.

Return Value

An array of all the host ports in a fabric.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getDomainId

Returns the domain address.

Parameters

key—Wwn

vsanid—Unique identifier of the VSAN.

Return Value

Domain IP address.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getVsanIp

Returns the IP address of a VSAN.

Parameters

key—Wwn

vsanid—Unique identifier of the VSAN.

Return Value

IP address of a VSAN.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getVsanDomains

Returns all VSAN domains in a switch.

Parameters

Key—Wwnkey

Return Value

VSAN domains in a switch.

Error

Error Code: 300— General SAN Service exception.

Error code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneSetName

Returns the fabric IVR-enforced zone set name.

Parameters

Key—Fabric key

Return Value

Zone set name.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneSetNumber

Returns the fabric IVR-enforced zone number.

Parameters

Key—Fabric key

Return Value

Zone number.

Error

Error Code: 300— General SAN Service exception.

Error code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneSetActivateTime

Returns the fabric IVR-enforced zone set activate time.

Parameters

Key—Fabric key

Return Value

Time stamp in the long integer format.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneSet

Returns the fabric IVR-enforced zone set.

Parameters

Key—Fabric key

Return Value

List of zone objects.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrActiveZonesetChecksum

Returns the IVR active zone set checksum.

Parameters

Key—Fabric key

Return Value

Checksum value.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getAliases

Returns all the aliases used by the fabric.

Parameters

Key—Fabric key

Return Value

Aliases used by the fabric.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

useFcAlias

Returns all the FC aliases used by the fabric.

Parameters

Key—Fabric key

Return Value

FC aliases used by the fabric.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnfZoneSet

Returns all theVSAN enforced zone sets.

Parameters

Key—Fabric key

Return Value

List of zones.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnfZoneSetName

Returns all the VSAN enforced zone set names.

Parameters

Key—Fabric key

Return Value

List of zone set names.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnfZoneSetName

Returns all the VSAN enforced zone set name.

Parameters

Key—Fabric key

Return Value

List of zone set names.

Error

Error Code: 300— General SAN Service exception.

Error code: 201—Invalid argument in Web Service exception.

getFCAliases

Returns all the FC aliases for the fabric.

Parameters

Key—Fabric key

Return Value

List of aliases.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFCAliasesByVsan

Returns all the FC aliases for the VSAN.

Parameters

Key—Fabric key

Return Value

List of aliases for the VSAN.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getCFS

Returns CFS.

Parameters

Key—Fabric key

Return Value

List of CFS features.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getCFSBySwitch

Returns CFS.

Parameters

Key—Switch key

Return Value

List of CFS features.

Error

Error Code: 300— General SAN Service exception.

Error code: 201—Invalid argument in Web Service exception.

getZoneMode

Returns zone operation modes.

Parameters

Key—Fabric key

Return Value

List of zone operation modes.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneModeByVsan

Returns zone operation modes for VSAN.

Parameters

Key—VSAN key

Return Value

List of zone operation modes.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneAttributes

Returns zone attributes.

Parameters

Key—Fabric key

Return Value

Zone attributes.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneAttributesByVsan

Returns zone attributes for VSAN.

Parameters

Key—VSAN key

Return Value

Zone attributes for VSAN.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getSwitchPorts

Returns ports for the switch.

Parameters

Key—Fabric key

Return Value

List of ports for a given switch.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

isIVREnabled

Returns a boolean value depending on whether the IVR is enabled on the switch or not.

Parameters

Key—Switch key

Return Value

Boolean value.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getSwitchDateAndTime

Returns the switch time and date.

Parameters

Key—Switch key

Return Value

Boolean value.

Error

Error Code: 400—SnmpException.

Zone Manager WS - SEI

activateZoneset

Activates the zone set.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Operational status of the zone set.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

addZone

Add a new zone to the list of zones.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Zone object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

addZoneAlias

Adds a zone alias.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Zone object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

addZoneMemberToZone

Adds a new zone member to the specific zone.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

ID of the zone member.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

addZoneMemberToZoneAlias

Adds a zone member to the zone alias.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Zone member.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

createZone

Creates a new zone.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Zone object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

createZoneAlias

Creates a zone alias.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Zone alias object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

createZoneMemberInZone

Creates a zone member in the specified zone.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Member ID.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

createZoneMemberInZoneAlias

Creates a zone member in the specified zone alias.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Member ID.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

createZoneSet

Creates a zone set.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

ID of the zone set.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

deActivateZoneset

Returns the operational status of the zone set.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

Operational status of the zone set.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnfZoneSet

Returns the enforced zone set.

Parameters

key—WwnKey.

key—Name of the VSAN.

Return Value

ID of the zone set.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getEnfZoneSetName

Returns the name of the enforced zone set.

Parameters

key—WwnKey

key—Name of the VSAN

Return Value

ID of the zone set.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrActiveZonesetChecksum

Returns the IVR active zoneset checksum.

Parameters

key—FabricKey

Return Value

Checksum value.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneNumber

Returns the fabric IVR enforced zone number.

Parameters

key—FabricKey

Return Value

Zone number.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneSet

Returns the fabric IVR enforced zone set ID.

Parameters

key—FabricKey

Return Value

Zone set ID.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneSetActivateTime

Returns the fabric IVR enforced zone set activate time.

Parameters

key—FabricKey

Return Value

System time as the number of seconds elapsed since the start of the Unix epoch at 1 January 1970 00:00:00 UT.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIvrEnfZoneSetName

Returns the fabric IVR enforced zone set name.

Parameters

key—WWNKey

Return Value

Zone ID.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZone

Returns an array of zone IDs.

Parameters

key—WwnKey

Return Value

Zone object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneAlias

Returns zone alias.

Parameters

key—WwnKey

key—VSAN Key

Return Value

Zone alias object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneAliases

Returns an array of zone aliases.

Parameters

key—WwnKey

key—VSAN Key

Return Value

List of zone aliases.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneAttributes

Returns all the attributes for the zone.

Parameters

key—WwnKey

key—VSAN Key

Return Value

List of zone attributes.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneAttributesByVsan

Returns all the zone attributes for the VSAN.

Parameters

key—WwnKey

Return Value

List of zone attributes for the VSAN.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneCapabilitiesByFabric

Returns the zone information associated with the specified fabric key.

Parameters

key—Fabric key

Return Value

ZoneCapabilities object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneCapabilitiesByVsan

Returns all zone information for the VSAN.

Parameters

key—WwnKey

Return Value

ZoneCapabilities object.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneMode

Returns the list of zone modes.

Parameters

key—WwnKey

Return Value

Zone information.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneModeByVsan

Returns the list of zone modes for the VSAN.

Parameters

key—VSAN key

Return Value

Zone information for the VSAN.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneSet

Returns the zone set.

Parameters

key—WwnKey

Return Value

Zone sets.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZoneSets

Returns the list of zone sets.

Parameters

key—WwnKey

Return Value

List of zone sets.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getZones

Returns the list of zones.

Parameters

key—WwnKey

Return Value

List of zones.

Error

Error Code: 300—General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

Statistics WS

getEndDeviceStatistics

Returns the statistics of the end device.

Parameters

key—WwnKey

Return Value

Statistics of the end device.

Error

Error Code: 201—Invalid argument in Web Service exception.

getEndDeviceStatisticsByAlias

Returns the statistics of the end devices by device alias.

Parameters

key—WwnKey

Return Value

Statistics of the end devices.

Error

Error Code: 201—Invalid argument in Web Service exception.

getEthPortStatisticsByKey

Returns the statistics of the Ethernet port.

Parameters

key—WwnKey

Return Value

Statistics of the Ethernet port.

Error

Error Code: 201—Invalid argument in Web Service exception.

getEthPortStatisticsBySwitch

Returns the statistics of the Ethernet port based on a switch.

Parameters

key—WwnKey

Return Value

Statistics of the Ethernet port based on a switch.

Error

Error Code: 201—Invalid argument in Web Service exception.

getFcPortStatistics

Returns the statistics of the Fibre Channel port.

Parameters

key—WwnKey

Return Value

Statistics of the Fibre Channel port.

Error

Error Code: 201—Invalid argument in Web Service exception.

getFcPortStatisticsByKey

Returns the statistics of the Fibre Channel port based on the switch WWN.

Parameters

key—WwnKey

Return Value

Statistics of the Fibre Channel port.

Error

Error Code: 201—Invalid argument in Web Service exception.

getFcPortStatisticsBySwitch

Returns the statistics of the Fibre Channel port based on a switch.

Parameters

key—WwnKey

Return Value

Port statistics of the Fibre Channel port.

Error

Error Code: 201—Invalid argument in Web Service exception.

getIPEndPointStatisticsByKey

Returns the statistics of the IPEndPoint based on a switch WWN.

Parameters

key—WwnKey

Return Value

Statistics of the IPEndPoint.

Error

Error Code: 201—Invalid argument in Web Service exception.

getIPEndPointStatisticsBySwitch

Returns the statistics of the IPEndPoint based on a switch.

Parameters

key—WwnKey

Return Value

Statistics of the IPEndPoint.

Error

Error Code: 201—Invalid argument in Web Service exception.

getTCPEndPointStatisticsByKey

Returns the statistics of the TCPEndPoint based on a switch WWN.

Parameters

key—WwnKey

Return Value

Statistics of the TCPEndPoint.

Error

Error Code: 201—Invalid argument in Web Service exception.

getTCPEndPointStatisticsBySwitch

Returns the statistics of the TCPEndPoint based on a switch.

Parameters

key—WwnKey

Return Value

Statistics of the TCPEndPoint on a switch.

Error

Error Code: 201—Invalid argument in Web Service exception.

Security WS

getAaaMaxServer

Returns a value for the maximum number of server entries in a server group of the AAA configuration.

Parameters

key—WwnKey

Return Value

Maximum number of server entries in a server group of the AAA configuration.

Error

Error Code: 400—SnmpException

getAaaMaxAppServer

Returns a value for the maximum number of server entries in the AAA configuration for an application type.

Parameters

key—WwnKey

Return Value

Maximum number of server entries in a server group in the AAA configuration.

Error

Error Code: 400—SnmpException

isClearAcctLogSet

Checks if the clear accounting log is set.

Parameters

key—WwnKey

Return Value

Checks if the clear accounting log is set.

Error

Error Code: 400—SnmpException

isMSCHAPRequired

Returns a boolean value to indicate if MSCHAP authentication mechanism is required for authenticating a user.

Parameters

key—WwnKey

Return Value

Boolean value to indicate if MSCHAP authentication mechanism is required for authenticating a user.

Error

Error Code: 400—SnmpException

getAaaSetup

Returns the AAA configuration.

Parameters

key—WwnKey

Return Value

AAA configuration.

Error

Error Code: 400—SnmpException

getAaaAppServerGroups

Returns the AAA server groups for a specific application type.

Parameters

key—WwnKey

Return Value

AAA server groups for a specific application type.

Error

Error Code: 400—SnmpException

getAaaServerGroups

Returns all the AAA server group entries (a server group consists of a number of AAA servers implementing the same AAA protocol).

Parameters

key—WwnKey

Return Value

AAA server group entries.

Error

Error Code: 400—SnmpException

getSnmpUsers

Returns information about SNMP users.

Parameters

key—WwnKey

Return Value

Information about SNMP users.

Error

Error Code: 400—SnmpException

getIPACLProfiles

Returns all the IP ACL profiles.

Parameters

key—WwnKey

Return Value

All the IP ACL profiles.

Error

Error Code: 400—SnmpException

getSSHConfig

Returns the SSH configuration information.

Parameters

key—WwnKey

Return Value

SSH configuration information.

Error

Error Code: 400—SnmpException

getSSHEnabled

Returns a boolean value to indicate if the SSH is enabled.

Parameters

key—WwnKey

Return Value

Boolean value to indicate if the SSH is enabled.

Error

Error Code: 400—SnmpException

isTelnetEnabled

Returns a boolean value to indicate if Telnet is enabled.

Parameters

key—WwnKey

Return Value

Boolean value to indicate if Telnet is enabled.

Error

Error Code: 400—SnmpException.

getPkiRsaKeys

Returns the PKI RSA key pair entries.

Parameters

key—WwnKey

Return Value

PKI RSA key pair entries.

Error

Error Code: 400—SnmpException.

getPkiTrustPointNames

Returns a list of PKI trustpoint names.

Parameters

key—WwnKey

Return Value

A list of PKI trustpoint names.

Error

Error Code: 400—SnmpException.

getPkiTrustPointNames

Returns a list of PKI trustpoint names.

Parameters

key—WwnKey

Return Value

A list of PKI trustpoint names.

Error

Error Code: 400—SnmpException.

getPkiCert

Returns certificate information of a PKI trustpoint.

Parameters

key—WwnKey

Return Value

Certificate information of a PKI trustpoint.

Error

Error Code: 400—SnmpException.

getPkiAction

Returns the PKI support action of a trustpoint.

Parameters

key—WwnKey

Return Value

PKI support action of a trustpoint.

Error

Error Code: 400—SnmpException.

getPkiTrustPoint

Returns the PKI trust point information, which consists of the key pair name, a list of revocation methods, and the contact HTTP URL of the external OCSP server for certificate revocation.

Parameters

key—WwnKey

Return Value

PKI trust point information.

Error

Error Code: 400—SnmpException.

getFeatureControls

Returns all of the feature control names and their respective statuses.

Parameters

key—WwnKey

Return Value

Feature control names and statuses.

Error

Error Code: 400—SnmpException.

getIkeFailRecoveryCfg

Returns the IKE configuration.

Parameters

key—WwnKey

Return Value

IKE configuration.

Error

Error Code: 400—SnmpException

getIkeCfgPolicies

Returns the policy that is used to set up the IKE tunnels.

Parameters

key—WwnKey

Return Value

Policy that is used to set up the IKE tunnels.

Error

Error Code: 400—SnmpException.

getIkeCfgInitiators

Returns the IKE initiator configuration information.

Parameters

key—WwnKey

Return Value

IKE initiator configuration information.

Error

Error Code: 400—SnmpException.

getIkeTunnels

Returns the IKE tunnels information.

Parameters

key—WwnKey

Return Value

IKE tunnels information.

Error

Error Code: 400—SnmpException.

getIPsecGlobalCfg

Returns the IPsec tunnel configuration information.

Parameters

key—WwnKey

Return Value

IPsec tunnel configuration information.

Error

Error Code: 400—SnmpException.

getIPsecXformSets

Returns the IPsec transform set information.

Parameters

key—WwnKey

Return Value

IPsec transform set information.

Error

Error Code: 400—SnmpException.

getIPsecCryptoMaps

Returns the IPsec cryptomap set.

Parameters

key—WwnKey

Return Value

IPsec cryptomap set.

Error

Error Code: 400—SnmpException.

getIfsFromCryptoMap

Returns the interface name from the IPsec cryptomap.

Parameters

key—WwnKey

Return Value

Interface name from the IPsec cryptomap.

Error

Error Code: 400—SnmpException.

getIPsecTunnels

Returns the information about IPsec tunnels.

Parameters

key—WwnKey

Return Value

Information about IPsec tunnels.

Error

Error Code: 400—SnmpException.

isFipsModeEnabled

Returns information about the FIPS mode.

Parameters

key—WwnKey

Return Value

Information about FIPS mode.

Error

Error Code: 400—SnmpException.

Protocol WS

getNtpPeers

Returns NTP peer information.

Parameters

key—WwnKey

Return Value

NTP peer information.

Error

Error Code: 400—SnmpException

getNtpInfo

Returns NTP system information.

Parameters

key—WwnKey

Return Value

NTP system information.

Error

Error Code: 400—SnmpException

getFspfConfig

Returns the FSPF protocol configuration.

Parameters

key—WwnKey

Return Value

Configuration settings of the FSPF protocol.

Error

Error Code: 400—SnmpException

queryInterfaceFspfConfig

Returns the FSPF configuration on the interface pertaining to the specified VSAN.

Parameters

key—WwnKey

vsanid—Unique identifier of the VSAN

Return Value

FSPF configuration settings on the interface of a specified VSAN.

Error

Error Code: 400—SnmpException

getFcipProfiles

Returns FCIP profiles.

Parameters

Key—Fabric key

Return Value

List of FCIP profiles.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFcipProfilesBySwitch

Returns FCIP profiles based on a switch.

Parameters

Key—Switch key

Return Value

List of FCIP profiles.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFcipTunnels

Returns FCIP tunnels.

Parameters

Key—Fabric key

Return Value

List of FCIP tunnels.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFcipTunnelsBySwitch

Returns FCIP tunnels based on a switch.

Parameters

Key—Switch key

Return Value

List of FCIP tunnels.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFcipTunnelByLinkIndex

Returns FCIP tunnels based on a switch.

Parameters

Key—Switch key

Return Value

List of FCIP tunnels.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFcipTunnelByLinkIfIndex

Returns FCIP tunnels based on a switch.

Parameters

Key—Switch key

Return Value

List of FCIP tunnels.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFcipTunnelErrors

Returns FCIP tunnels errors.

Parameters

Key—Fabric key

Return Value

List of FCIP tunnel errors.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getFcipTunnelErrorsBySwitch

Returns FCIP tunnels errors based on a switch.

Parameters

Key—Switch key

Return Value

List of FCIP tunnels errors.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIpSettingsBySwitch

Returns IP settings from the switch.

Parameters

Key—Switch key

Return Value

IP settings from the switch.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getIpSettings

Returns IP settings from the switch and port.

Parameters

Key—Port key

Return Value

IP settings from the switch and port.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getTcpSettingsBySwitch

Returns TCP settings from the switch.

Parameters

Key—Switch key

Return Value

TCP settings from the switch.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

getTcpSettings

Returns TCP settings from the switch.

Parameters

Key—Switch key

Return Value

TCP settings from the switch.

Error

Error Code: 300— General SAN Service exception.

Error Code: 201—Invalid argument in Web Service exception.

Cluster WS - SEI

Service end point interface (SEI) of Cluster WS defines the operations of the service. These methods are published to the end users.

getSwitchesByFabricKey

Returns all the switches related to a fabric key.

Parameters

key—Name of the fabric.

Return Value

All the switches associated with the fabric.

Error

Error Code: 400—SnmpException

getServerIpByFabricKey

Returns the managing server IP address from a fabric key.

Parameters

key—Name of the fabric.

Return Value

IP address of the server.

Error

Error Code: 400—SnmpException

getServerIpBySwitchKey

Returns the managing server IP address from a switch key.

Parameters

key—Name of the switch.

Return Value

IP address of the server.

Error

Error Code: 400—SnmpException

getFabricsByServerIp

Returns name of the fabric by server IP address.

Parameters

key—IP address

Return Value

Name of the fabric.

Error

Error Code: 400—SnmpException

getAllServers

Returns all the servers in this federation.

Parameters

key—IP address

Return Value

All the servers in the federation.

Error

Error Code: 400—SnmpException

getFabricByEnclosureKey

Returns the fabric key from an enclosure key.

Parameters

key—Name of the enclosure.

Return Value

Name of the fabric.

Error

Error Code: 400—SnmpException

getServerIpByEnclosureKey

Returns the server IP address from an enclosure key.

Parameters

key—Name of the fabric.

Return Value

IP address of the server.

Error

Error Code: 400—SnmpException

getServerIpByVsanKey

Returns the server IP address from a VSAN key.

Parameters

key—Name of the VSAN

Return Value

IP address of the server.

Error

Error Code: 400—SnmpException

Event WS - SEI

Service end point interface (SEI) of Event WS defines the operations of the service. These methods are published to the end users.

isCallHomeEnabled

Returns a boolean value depending upon the activation of the CallHome feature.

Parameters

key—WwnKey

Return Value

Boolean value

Error

Error Code: 400—SnmpException

getCallHomeDestProfile

Returns the CallHome destination profile.

Parameters

key—WwnKey

Return Value

Error

Error Code: 400—SnmpException

getCallHomeSysInfo

Returns system information about the CallHome feature.

Parameters

key—WwnKey

Return Value

System information about the CallHome feature.

Error

Error Code: 400—SnmpException

getEmailMaxEntries

Returns the maximum number of e-mail address entries for the CallHome feature.

Parameters

key—WwnKey

Return Value

Number of e-mail address entries for the CallHome feature

Error

Error Code: 400—SnmpException

getEmailSetup

Returns the e-mail setup details of the CallHome feature.

Parameters

key—WwnKey

Return Value

E-mail setup details of CallHome feature.

Error

Error Code: 400—SnmpException

getSyslogServers

Returns the list of syslog servers.

Parameters

key—WwnKey

Return Value

List of syslog servers.

Error

Error Code: 400—SnmpException

getSyslogMessageControl

Returns a list of syslog message configuration.

Parameters

key—WwnKey

Return Value

List of syslog message configuration.

Error

Error Code: 400—SnmpException

getSyslogLoggingCfg

Returns syslog logging configuration.

Parameters

key—WwnKey

Return Value

Configuration information on syslog credentials.

Error

Error Code: 400—SnmpException

Inventory WS - SEI

Service end point interface (SEI) of Inventory WS defines the service end point interface for Inventory Web Service. These methods are published to the end users.

getPowerSuppliesBySwitchWwnKey

Returns name of the power suppies by switch.

Parameters

key—WwnKey

Return Value

List of power supply names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getPowerSuppliesBySwitchSnKey

Returns name of the power suppies by switch.

Parameters

key—swKey

Return Value

List of power supply names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getPowerSuppliesBySwitchIP

Returns name of the power supplies by switch IP address.

Parameters

key—swKey

Return Value

List of power supply names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getCardsBySwitchWwnKey

Returns name of the cards from a switch key.

Parameters

key—swKey

Return Value

List of card names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getCardsBySwitchSnkey

Returns name of the cards from a switch key.

Parameters

key—swKey

Return Value

List of card names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getCardsBySwitchIP

Returns name of the cards by switch IP address.

Parameters

key—swKey

Return Value

List of card names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getFansBySwitchWwnKey

Returns name of the fans by switch key.

Parameters

key—swKey

Return Value

List of fan names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getFansBySwitchSnKey

Returns name of the fans by switch key.

Parameters

key—swKey

Return Value

List of fan names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getFansBySwitchIP

Returns name of the fans by switch IP address.

Parameters

key—swKey

Return Value

List of fan names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getChassisBySwitchWwnKey

Returns name of the chassis by switch key.

Parameters

key—swKey

Return Value

List of chassis names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getChassisBySwitchSnKey

Returns name of the chassis by switch key.

Parameters

key—swKey

Return Value

List of chassis names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getChassisBySwitchIP

Returns name of the chassis by switch IP address.

Parameters

key—swKey

Return Value

List of chassis names.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getAllHbas

Returns name of the hosts by host IP address.

Parameters

key—hba WWN

Return Value

List of HBAs.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getHbaByWwn

Returns name of the hosts by host IP address.

Parameters

key—hba WWN

Return Value

List of HBAs.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getLicensesBySwitchWwnKey

Returns name of the licenses used by the switch.

Parameters

key—swKey

Return Value

List of licenses.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getLicensesBySwitchIP

Returns name of the licenses used by the switch.

Parameters

key—swKey

Return Value

List of licenses.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getLicenseFlags

Returns name of the licenses used by the switch.

Parameters

key—swKey

Return Value

List of licenses.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

getCardByPhysicalIndex

Returns the physical and switch index.

Parameters

key—swKey

Return Value

List of indexes.

Error

Error Code: 300 or 201—SAN service error or Invalid Argument

Error Codes

Error Code	Description
100	Authentication failure
101	Invalid credential
102	Invalid privilege
103	Invalid token
200	Web Service error
201	Invalid argument in Web Service function
202	Unreachable Web Service server
300	SAN service error
301	Invalid query key
400	SNMP error
201	Invalid argument

Note DCNM-SAN Web Services supports server federation. Service requests to SanWS, SecurityWS, ProtocolWS, EventWS, and InventoryWS automatically dispatches the calls to the correct server in the federation. If you are using server federation, the following methods do not automatically mediate to the corresponding server in the federation:

SanWS:
getEnclosures()
getEndports()
getFabricByIP()
getHosts()
getStorages()
getSwitchIPByName()
getSwitches()

InventoryWS:
getAllHbas()
getLicenseFlags()

In those specific instances, you might need to rely on ClusterWS to determine the server that you need to send the request.

CHAPTER 8

Discovery Automation Tool

You can use the Discovery Automation tool to discover DCNM-SAN, DCNM-LAN, and vCenter. After Cisco DCNM is installed, this tool requires the IP address and user credentials of the seed switch in DCNM-SAN, DCNM-LAN, and vCenter.

Installing Discovery Automation

After you install Cisco DCNM, you can find the Discovery Automation tool installation informaiton here:

- Batch files exeute the Discover Fabric tool are located:
  - On Windows at<*DCNM install di*r>/dcm/fm/bin/DiscoverFabric.bat
  - On Linux at DiscoverFabric.sh
- The jars are available at <*DCNM install dir*>dcm/fm/lib/.

The webservices used are:

- SAN discovery— SANWS to discoverFabricWithServer
- LAN discovery— SANWS to discoverLan
- vCenter discovery— SANWS to addVirtualCenter

Using the Discovery Automation Tool

To use the discovery automation tool, enter the following command:

DiscoverFabric <*FMServerIP*> <*FMUserName*> <*FMPassword*> <*Fabric-info file name- absolute path*> [FM Server Port] (Optional-Default value is 80).

The fabric-info filename contains the DCNM-SAN, DCNM-LAN, and vCenter information and should be in the following format:

- For SAN SNMP v2— SeedSwitch IP, Community String, FM Server IP/Hostname
- For SAN SNMP v3— SeedSwitch IP, UserName,Password, Protocol, FM Server IP/Hostname
- For LAN SNMP v2— SeedSwitch IP, Community String, FM Server IP/Hostname

- For LAN SNMP v3— SeedSwitch IP, Snmp Version, UserName, Password, maxHop, enablePwd, groupDbId, Protocol, FM Server IP/Hostname
- For VCENTER—VirtualCenter IP, UserName, Password
- For deepDiscover— between protcol and serverIP.

Sample Client Programs to Use Web Services

This appendix describes the sample client programs that use Cisco DCNM-SAN Web Services. Cisco DCNM-SAN supports the following web services.

SOAP Web Services

The following client programs use SOAP web services in Cisco DCNM.

Java Client

This section provides an example of the Java client that uses the SOAP web services.

Example A-1 Zone Manager Web Services API

```
package com.cisco.dcbu.smis.jaxws.test;


import static com.cisco.dcbu.smis.jaxws.test.ZoneWsUtil.*;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import com.cisco.dcbu.smis.jaxws.ep.Fabric;
import com.cisco.dcbu.smis.jaxws.ep.OperationStatus;
import com.cisco.dcbu.smis.jaxws.ep.Vsan;
import com.cisco.dcbu.smis.jaxws.ep.WwnKey;
import com.cisco.dcbu.smis.jaxws.ep.Zone;
import com.cisco.dcbu.smis.jaxws.ep.ZoneMember;
import com.cisco.dcbu.smis.jaxws.ep.ZoneSet;
import com.cisco.dcbu.smis.jaxws.ep._switch;
public class ZoneWsClient {

    /**
     * Default constructor
     */
    public ZoneWsClient() {
```

```
        input_ = new BufferedReader(new InputStreamReader(System.in));
    }

    /**
     * Main method is the starting point of execution.
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception {
        menu();
    }

    /**
     * This method show all available zone webservice as user menu.
     * @throws Exception
     */
    public static void menu() throws Exception {
        ZoneWsClient zwsClient = new ZoneWsClient();
        int choice = 0;

        for (;;) {
            header("================================Zone WS
Client=============================================");
            System.out.println("Available Zone services:\n\n " + "\t1:Discover Zones in
fabric\n " + "\t2:Discover Zonesets in fabric\n " + "\t3:Discover Zonemenbers in fabric\n
"
                    + "\t4:Create new Zone in fabric\n " + "\t5:Create new ZoneSet in
fabric\n " + "\t6:Add Zonemember to existing Zone in fabric\n " + "\t7:Add existing Zone
to existing Zoneset in fabric\n "
                    + "\t8:Activate/Deactivate a ZoneSet in fabric\n " + "\t9:Manual Zone
Cache Invalidation\n " + "\t10:Exit ");
            System.out.println("\n\tPlease enter your Choice: ");
            try {
                choice = Integer.parseInt(input_.readLine());
            } catch (NumberFormatException e) {
                System.out.println("Enter Integer choices only...");
                continue;
            }

            switch (choice) {
            case 1:
                zwsClient.displayZone();
                break;
            case 2:
                zwsClient.displayZonesets();
                break;
            case 3:
                zwsClient.displayZonemembers();
                break;
            case 4:
                zwsClient.createZone();
                break;
            case 5:
                zwsClient.createZoneSet();
                break;
            case 6:
                zwsClient.addZoneMember();
                break;
            case 7:
                zwsClient.addZone();
                break;
            case 8:
                zwsClient.activateAndDeactivateZoneSet();
                trailer("ACTIVATE/DEACTIVATE ZONESET");
```

```
                break;
            case 9:
                zwsClient.zoneCacheInvalidation();
                break;
            case 10:
                System.out.println("Exiting ZoneWSClient..");
                trailer("Zone WS CLIENT");
                System.exit(1);
            default:
                System.out.println("Wrong Choice entered.");
            }
        }

    }

    /**
     * This method invalidates zonedata cache by making use of
     * ZoneManagerWS: invalidateZoneDataCache()
     */
    public void zoneCacheInvalidation() {
        try {
            System.out.println("Invalidation of Zone data cache started");
            zoneManager.invalidateZoneDataCache();
            System.out.println("Succesfully Invalidated Zone data cache");
        } catch (Exception e) {
            exception(e);
        }finally{
            trailer("Zone Cache Invalidation");
        }

    }

    /**
     * This method takes promts user to enter already existing zoneset name to
     * activate/deactivate.
     * ZoneManagerWS: ActivateZoneset()
     * ZoneManagerWS: deActivateZoneset()
     */
    public void activateAndDeactivateZoneSet() throws Exception {
        int flag = 0,choice=0;
        do{
        try {
            System.out.println("Enter your choice\n 1:Activate a ZoneSet\n 2:Deactivate a
ZoneSet\n 3:Main menu");
            choice = Integer.parseInt(input_.readLine());
            OperationStatus oper = null;
            if(choice==3)return;
            if (choice != 1 && choice != 2 && choice !=3) {
                System.out.println("Enter right Choice: 1 or 2 or 3");
                flag = 0;
                continue;
            }
            oper = activation(choice);
            if (oper != null){
                System.out.println("Zoneset activation/deactivation is done successfully.\n
Please wait for 1-5 minutes to get reflected.");
                flag =1;
            }
            else if (oper == null){
                System.out.println("Zoneset activation/deactivation is not successfull");
                flag =1;
            }
            break;
        } catch(NumberFormatException ne){
```

```
            System.out.println("NumberFormatException: Integer type expected.");
            continue;
        }
        catch (Exception e) {
            exception(e);
            flag=0;
            continue;
        }
        }while(choice !=3||flag==0);
    }

    private OperationStatus activation(int choice) throws Exception {
        OperationStatus oper = null;
        System.out.println("Enter the ZoneSet name to be activated/deactivated ");
        String zName = input_.readLine();
        Fabric[] fabrics = getFabrics();
        if(fabrics!=null){
        for (Fabric fab : fabrics) {
            _switch[] sws = getSwitchesByFabric(fab);
            Vsan[] vsans = getVsans(fab);
            if(sws!=null){
            for (_switch sw : sws) {
                if(vsans!=null){
                for (Vsan vsan : vsans) {
                    ZoneSet enfZoneset = zoneManager.getEnfZoneSetDO(vsan.getKey());
                    ZoneSet[] zoneSets = zoneManager.getZoneSets(vsan.getKey(),new
WwnKey(sw.getWwn()));
                    if(zoneSets==null){
                        System.out.println("Zoneset is null for VSan
"+vsan.getKey().getVsanID());
                        continue;
                    }
                    for (ZoneSet zs : zoneSets) {
                        if (enfZoneset != null) {
                            if (enfZoneset.getName().equals(zName) && choice == 1 &&
enfZoneset.isActive()) {
                                System.out.println("Selected ZoneSet is already activated");
                                return null;
                            }
                            else if (!enfZoneset.getName().equals(zName) && choice == 2 &&
zs.getName().equals(zName)) {
                                System.out.println("Selected ZoneSet is already
deactivated");
                                return null;
                            }
                        } else if (enfZoneset == null && choice == 2) {
                            System.out.println("No ZoneSet is active to deactivate");
                            return null;
                        } else if (zs.getName().equals(zName) && choice == 2 &&
!zs.isActive()) {
                            System.out.println("Selected ZoneSet is already deactivated");
                            return null;
                        }
                        if (zs.getName().equals(zName)) {
                            if (choice == 1)
                                oper = zoneManager.activateZoneset(vsan.getKey(), new
WwnKey(sw.getWwn()), zs.getName());
                            else if (choice == 2)
                                oper = zoneManager.deActivateZoneset(vsan.getKey(), new
WwnKey(sw.getWwn()), zs.getName());
                            return oper;
                        }
                    }
                }
```

```
            }
            }

            System.out.println("Invalid Zoneset is selected");
        }
        }}
        return oper;


    }

    /**
     * This method promts user to enter a unique zone name and creates the
     * zone in default vsan.
     * ZoneManagerWS:createZone()
     */
    public void createZone() {
        try {
            String zName=null;
            do{
            System.out.println("Enter the new Zone name");
            zName= input_.readLine();
            }while(zName.length()<1);
            Fabric[] fabrics = getFabrics();
            if(fabrics!=null){
            for (Fabric fab : fabrics) {
                _switch[] sws = getSwitchesByFabric(fab);
                Vsan[] vsans = getVsans(fab);
                if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                    for (Vsan vsan : vsans) {
                        int qosPriority = -1;
                        boolean readOnly = false;
                        boolean broadcast = false;
                        boolean qos = false;
                        Zone zone1 = zoneManager.createZone(vsan.getKey(),new
WwnKey(sw.getWwn()), zName, readOnly, broadcast, qos, qosPriority);
                        if (zone1 == null)
                            System.out.println("Failed to create Zone: " + zName);
                        else
                            System.out.println("Zone " + zName + " created successfuly in
Vsan:" + vsan.getKey().getVsanID() + ",Switch: " + sw.getName());
                        return;
                    }
                }
                }}
            }
        }
        } catch (Exception e) {
            exception(e);
        }finally{
            trailer("CREATE ZONE");
        }

    }

    /**
     * This method promts user to enter an existing zone,zonememberid and zonemembertype
and creates a zonemember in zone provided.
     * ZoneManagerWS:addZoneMemberToZone
     */
    public void addZoneMember() {
        try {
```

```
            for (;;) {
                System.out.println("\nEnter existing zonename to which member has to be
added\n");
                String zoneName = input_.readLine();
                Fabric[] fabrics = getFabrics();
                if(fabrics!=null){
                for (Fabric fab : fabrics) {
                    _switch[] sws = getSwitchesByFabric(fab);
                    Vsan[] vsans = getVsans(fab);
                    if(sws!=null){
                    for (_switch sw : sws) {
                        if(vsans!=null){
                        for (Vsan vsan : vsans) {
                            Zone existingzones[] = zoneManager.getZones(vsan.getKey(), new
WwnKey(sw.getWwn()));
                            if(existingzones==null){
                                continue;
                            }
                            boolean flag=false;
                            for (Zone zs : existingzones) {
                                if (zs.getName().equals(zoneName)) {
                                    flag=true;
                                    System.out.println("Enter zonemember type which is in
number format. \n 1. For WWN ");
                                    int zonememtype = Integer.parseInt(input_.readLine());
                                    String hexstr = null;
                                    byte[] zonememberid = null;
                                    if(zonememtype==1){
                                        System.out.println("Enter zonemember id which is in
hexa format Eg: 1122334455667788 or 11:22:33:44:55:66:77:88");
                                        hexstr=input_.readLine();
                                        if (hexstr.contains(":"))
                                            zonememberid = ZoneWsUtil.fromHexString(hexstr,
true);
                                        else
                                            zonememberid = ZoneWsUtil.fromHexString(hexstr,
false);
                                    }
                                    else{
                                        System.out.println("Demo ZoneWsClient do not support
other member types\n");
                                        continue;
                                    }

                                    ZoneMember zoneMemberAdded =
zoneManager.addZoneMemberToZone(vsan.getKey(),new WwnKey(sw.getWwn()),zoneName,
zonememtype, -1,-1, -1, zonememberid, null);
                                    if (zoneMemberAdded == null)
                                        System.out.println("Failed to add Zonemember to
specified zone");
                                    else
                                        System.out.println("ZoneMember added succesfully to
zone "+ zoneName);
                                    return;

                                }
                            }
                            if(flag==false)
                            System.out.println("Zone " + zoneName + " not found. Please
enter an existing zone");
                        }}
                    }}
                }}
            }
```

```
        } catch(NumberFormatException ne){
            System.out.println("NumberFormatException: Please check the input
type/format");
        }
        catch (Exception e) {
            exception(e);
        }finally{
            trailer("ADD ZONE");
        }
    }

    /**
     * This method promts user to enter existing zoneset, existing zone
     * and adds zone to zoneset.
     * ZoneManagerWS:addZoneToZoneset()
     *
     */
    public void addZone() {
        try {
            for (;;) {
                System.out.println("Enter already existing zoneset to which the zone has to
be added: ");
                String zonesetName = input_.readLine();
                Fabric[] fabrics = getFabrics();
                if(fabrics!=null){
                for (Fabric fab : fabrics) {
                    _switch[] sws = getSwitchesByFabric(fab);
                    Vsan[] vsans = getVsans(fab);
                    if(sws!=null){
                    for (_switch sw : sws) {
                        if(vsans!=null){
                        for (Vsan vsan : vsans) {
                            ZoneSet existingzonesets[] =
zoneManager.getZoneSets(vsan.getKey(), new WwnKey(sw.getWwn()));
                            for (ZoneSet zset : existingzonesets) {
                                if (zset.getName().equals(zonesetName)) {
                                    System.out.println("Enter existing zone to be added to
zoneset: ");
                                    String zonename = input_.readLine();
                                    Zone existingzones[] =
zoneManager.getZones(vsan.getKey(),new WwnKey(sw.getWwn()));
                                    for (Zone zone : existingzones) {
                                        if (zone.getName().equals(zonename)) {
                                            ZoneSet znst =
zoneManager.addZoneToZoneset(vsan.getKey(),new WwnKey(sw.getWwn()),zset, zone);
                                            if (znst == null)
                                                System.out.println("Failed to add Zone to
specified Zoneset");
                                            else
                                                System.out.println("Zone "+ zonename+ " added
succesfully to zoneset "+ zonesetName);
                                            return;
                                        }
                                    }
                                    System.out.println("Zone "+ zonename+ " not found.
Please enter an existing zone");

                                }
                            }
                        }}
                    }}
                }}
```

```
                System.out.println("ZoneSet " + zonesetName+ " not found. Please enter an
existing zoneset");
            }
        } catch (Exception e) {
            exception(e);
        }finally{
            trailer("ADD ZONE");
        }

    }

    /**
     * This method promts user to enter zoneset name which will be created
     * in default vsan.
     * ZoneManagerWS:createZoneSet()
     */
    public void createZoneSet() {
        try {
            System.out.println("Enter the new unique Zoneset name");
            String zonsetName =input_.readLine();
            Fabric[] fabrics = getFabrics();
            if(fabrics!=null){
            for (Fabric fab : fabrics) {
                _switch[] sws = getSwitchesByFabric(fab);
                Vsan[] vsans = getVsans(fab);
                if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                    for (Vsan vsan : vsans) {

                        ZoneSet newzoneset = zoneManager.createZoneSet(vsan.getKey(), new
WwnKey(sw.getWwn()), zonsetName);
                        if (newzoneset == null)
                            System.out.println("Failed to create Zoneset: "+ zonsetName);
                        else
                            System.out.println("ZoneSet " + zonsetName+ " created in
Vsan:"+ vsan.getKey().getVsanID() + ",Switch: "+ sw.getName());
                        return;
                    }}
                }}
            }}
        } catch (Exception e) {
            exception(e);
        }finally{
            trailer("CREATE ZONESET");
        }
    }

    /**
     * This method displays the Zone member setting data in the fabric.
     * ZoneManagerWS:getZoneMembers()
     */
    public void displayZonemembers() {
        try {
            Fabric[] fabrics = getFabrics();
            if(fabrics!=null){
            for (Fabric fab : fabrics) {
                _switch[] sws = getSwitchesByFabric(fab);
                Vsan[] vsans = getVsans(fab);
                if(vsans!=null){
                for (Vsan vsan : vsans) {
                    Zone[] enfZones = getEnfZones(vsan);
                    if (enfZones != null) {
                        for (Zone ezone : enfZones) {
```

```
                            ZoneMember[] zms = getZoneMembers(ezone);
                            for (ZoneMember zm : zms)
                                System.out.println("vsan id:"+ vsan.getKey().getVsanID()+
",vsan wwn:"+ vsan.getKey().getPWwn().getValue()+ ",Zone Name:" + ezone.getName()
                                        + ", ZoneMemberType: " + zm.getType()+ ",
ZoneMemberId: "+ ZoneWsUtil.toHexString(zm.getId())+ ",Status:" + ezone.isActive());
                        }
                    }
                }}
                if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                    for (Vsan vsan : vsans) {
                        Zone[] zones = getZones(vsan, sw);
                        if (zones == null)
                            return;
                        for (Zone zone : zones) {
                            ZoneMember[] zms = getZoneMembers(zone);
                            for (ZoneMember zm : zms)
                                System.out.println("vsan id:"+ vsan.getKey().getVsanID()+
",vsan wwn:"+ vsan.getKey().getPWwn().getValue()+ ",Zone Name:" + zone.getName()
                                        + ", ZoneMemberType: " + zm.getType()+ ",
ZoneMemberId: "+ ZoneWsUtil.toHexString(zm.getId())+ ",Status:" + zone.isActive());
                        }
                    }}
                }}
            }}
        } catch (Exception e) {
            exception(e);
        }finally{
                trailer("Zone Members");
            }


    }

    private ZoneMember[] getZoneMembers(Zone zone) {
        header("Now getting ZoneMembers for Zone " + zone.getName());
        ZoneMember[] zoneMemz = null;
        try {
            zoneMemz = zone.getMembers();
            if (zoneMemz != null)
                System.out.println(zoneMemz.length+ " ZoneMembers found for the Zone " +
zone.getName());
            else
                System.out.println("No ZoneMembers found for the Zone "+ zone.getName());
            return zoneMemz;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }

    /**
     * This method displays the active and local zonesets in the fabric.
     * ZoneManagerWS:getZoneSets()
     */
    public void displayZonesets() {
        try {
            Fabric[] fabrics = getFabrics();
            if(fabrics!=null){
            for (Fabric fab : fabrics) {
                _switch[] sws = getSwitchesByFabric(fab);
                Vsan[] vsans = getVsans(fab);
```

```
                if(vsans!=null){
                for (Vsan vsan : vsans) {
                    getEnfZonesets(vsan);
                }}
                if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                    for (Vsan vsan : vsans) {
                        ZoneSet[] zoneSets = zoneManager.getZoneSets(vsan.getKey(), new
WwnKey(sw.getWwn()));
                        header("Now getting local ZoneSets for Vsan "+
vsan.getKey().getVsanID());
                        if (zoneSets == null) {
                            System.out.println("No Zoneset found for switch"+ sw.getName()
+ " Vsan "+ vsan.getKey().getVsanID());

                        }
                        else{
                        System.out.println(zoneSets.length+ " Zonesets found for the switch
"+ sw.getName() + " Vsan "+ vsan.getKey().getVsanID());
                        for (ZoneSet zoneset : zoneSets) {
                            System.out.println("VsanId:"+ vsan.getKey().getVsanID() +
",Vsan WWN:"+ vsan.getKey().getPWwn().getValue()+ ",ZonesetName:" + zoneset.getName()+
",Status:" + zoneset.isActive());
                        }
                        }
                    }}
                }}
            }}
        } catch (Exception e) {
            exception(e);
        }finally{
            trailer("ZONESETS");
        }


    }

    private static Fabric[] getFabrics() {
        try {
            Fabric[] fabrics = san.getFabrics();
            if (fabrics == null) {
                System.out.println("fabrics is null" + fabrics.length);
                trailer("No Fabric - Return to Main Menu");
                menu();
            }
            return fabrics;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }


    private static _switch[] getSwitchesByFabric(Fabric fabric) {
        try {
            _switch[] sws = san.getSwitchesByFabric(fabric.getFabricKey());
            if (sws == null)
                System.out.println("No switches found for fabric");
            else {
                header(sws.length + " switch(es) found for fabric");
                for (_switch sw : sws) {
                    System.out.println("Name:" + sw.getName() + ",IP:"+ sw.getIpAddress() +
",WWN:"+ sw.getWwn().getValue());
```

```
                }
            }
            return sws;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }

    private static Vsan[] getVsans(Fabric fabric) {
        try {
            Vsan[] vsans = san.getVsans(fabric.getFabricKey());
            if (vsans == null)
                System.out.println("No vsans found for fabric");
            else {
                header(vsans.length + " vsan(s) found for fabric");
                for (Vsan vsan : vsans) {
                    System.out.println("VanID:" + vsan.getKey().getVsanID()+ ",VsanWWN:" +
vsan.getKey().getPWwn().getValue());
                }
            }

            return vsans;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }



    /**
     * This method displays all the active and local zones in the fabric.
     * ZoneManagerWS:getEnfZones()
     * ZoneManagerWS:getZones()
     */
    public void displayZone() throws Exception {

        Fabric[] fabrics=getFabrics();
        if(fabrics!=null){
        for (Fabric fabric : fabrics) {
            System.out.println("Discovered the Fabric with WWN:" +
fabric.getSeedSwWwn().getValue());
            _switch[] sws = getSwitchesByFabric(fabric);
            Vsan[] vsans = getVsans(fabric);
            if(vsans!=null){
            for (Vsan vsan : vsans) {
                Zone[] enfZones = getEnfZones(vsan);
                if (enfZones != null) {
                    for (Zone ezone : enfZones) {
                        System.out.println("vsan id:"+ vsan.getKey().getVsanID() + ",vsan
wwn:"+ vsan.getKey().getPWwn().getValue()+ ",Zone Name:" + ezone.getName() + ",Status:"+
ezone.isActive());
                    }
                }
            }}
            if(sws!=null){
            for (_switch sw : sws) {
                if(vsans!=null){
                for (Vsan vsan : vsans) {
                    Zone[] zones = getZones(vsan, sw);
                    if (zones == null) continue;
                    for (Zone zone : zones) {
```

```
                        System.out.println("vsan id:"+ vsan.getKey().getVsanID() + ",vsan
wwn:"+ vsan.getKey().getPWwn().getValue()+ ",Zone Name:" + zone.getName() + ",Status:"+
zone.isActive());
                    }
                }}
            }}
        }}
        trailer("ZONES");
    }

    private Zone[] getEnfZones(Vsan vsan) throws Exception {
        header(" Getting active zones for Vsan " + vsan.getKey().getVsanID());
        Zone[] enfZones = null;
        try {
            enfZones = zoneManager.getEnfZoneSet(vsan.getKey());
            if (enfZones != null)
                System.out.println(enfZones.length+ " Active zones found for the Vsan "+
vsan.getKey().getVsanID());
            else
                System.out.println("No Active zones found");
            return enfZones;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }

    private ZoneSet getEnfZonesets(Vsan vsan) throws Exception {
        header("Now getting active ZoneSets for Vsan " + vsan.getKey().getVsanID());
        ZoneSet enfZoneset = null;
        try {
            enfZoneset = zoneManager.getEnfZoneSetDO(vsan.getKey());
            if (enfZoneset != null)
                System.out.println("Active zoneset for the VsanId: "+
vsan.getKey().getVsanID() + ", Pwwn: "+ vsan.getKey().getPWwn().getValue()
                        + ", ZonesetName: " + enfZoneset.getName()+ ", Status " +
enfZoneset.isActive());
            else
                System.out.println("Active zoneset is null");
            return enfZoneset;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }

    private static Zone[] getZones(Vsan vsan, _switch sw) {
        header("Now getting Local Zones for switch " + sw.getName() + " vsan "+
vsan.getKey().getVsanID());
        try {
            Zone[] zones = zoneManager.getZones(vsan.getKey(), new WwnKey(sw.getWwn()));
            if (zones != null)
                System.out.println("Found " + zones.length + " zones");
            else
                System.out.println("No zones found ");
            return zones;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }

    static {
        getCredentials();
```

```
            getLoginService();
            getLogin();
            getZoneManagerService();
            getSanService();
        }
    }
    public class ZoneWsUtil {

        public static String url = null;

        public static String username = null;

        public static String password = null;

        public static String token = null;

        public static Logon logon = null;

        public static San san = null;

        public static ZoneManager zoneManager = null;

        static BufferedReader input_ = null;

        static byte[] fromHexString(String hexStr, boolean hasColon)
                throws NumberFormatException {
            hexStr = hexStr.toLowerCase();
            int len = hexStr.length();
            byte bytes[] = new byte[hasColon ? ((len / 3) + 1) : (len / 2)];
            int sPos = 0; // position in hexStr
            int bPos = 0; // position in bytes
            try {
                while (sPos < len) {
                    char a = hexStr.charAt(sPos);
                    char b = hexStr.charAt(sPos + 1);
                    if (hasColon && (a == ':' || b == ':'))
                        throw new NumberFormatException("bad Hex format");
                    int v1 = Character.digit(a, 16);
                    int v2 = Character.digit(b, 16);
                    if (v1 < 0 || v2 < 0)
                        throw new NumberFormatException("bad Hex format");
                    int v3 = (int) (v1 * 16 + v2);
                    bytes[bPos] = (byte) v3;
                    sPos += hasColon ? 3 : 2;
                    bPos++;
                }
            } catch (Exception ex) {
                throw new NumberFormatException("bad Hex format");
            }

            if (bPos < bytes.length)
                throw new NumberFormatException("bad Hex format");
            return bytes;
        }

        static String toHexString(byte[] inputByte) {
            char[] HEX_DIGIT = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
                    'a', 'b', 'c', 'd', 'e', 'f' };
            if (inputByte == null || inputByte.length == 0)
                return null;
            StringBuffer outputstr = new StringBuffer(inputByte.length * 3);
            for (int i = 0; i < inputByte.length; i++) {
                int n = (int) (inputByte[i] & 0xFF);
                outputstr.append(HEX_DIGIT[(n >> 4) & 0x0F]);
```

```
            outputstr.append(HEX_DIGIT[n & 0x0F]);
            if (i + 1 < inputByte.length)
                outputstr.append(':');
        }
        return outputstr.toString();
    }

    static void header(String text) {
        System.out.println(" ");
        System.out.println(text);
    }

    static void trailer(String string) {
        System.out.println("\n******************************** END OF "
                + string + " ******************************** ");
    }

    /**
     * This static method is used read the login credentials from the properties
     * file named "configuration.properties"
     */
    public static void getCredentials() {
        Properties props = new Properties();
        try {
            if(props!=null){
            System.out.println("Reading credentials from the 'configuration.properties'
file");
            props.load(new FileInputStream("configuration.properties"));
            url = props.getProperty("FMServer-ipadress");
            username = props.getProperty("UserName");
            password = props.getProperty("Password");
            }
        }
        catch (FileNotFoundException fe) {
            System.out.println("ZoneWsClient: 'configuration.properties' File Not
found!!");
            url="localhost";
            username="admin";
            password="cisco123";
        }
        catch (IOException e) {
            exception(e);
        }
    }

    /**
     * This method is to check LogonWSService is available or not. If available
     * bind to the stub.
     */
    public static void getLoginService() {

        try {
            String loginServiceAddr = "http://" + url
                    + "/LogonWSService/LogonWS";

            LogonServiceLocator logonService = new LogonServiceLocator();
            logonService.setEndpointAddress("LogonPort", loginServiceAddr);

            logon = logonService.getLogonPort();

            if (logon == null)
                System.out.println("login service not available" + logon);
            else
                System.out.println("Logon success.");
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * This method is to get login token.
 */
public static void getLogin() {

    try {
        if(logon!=null){
        token = logon.requestToken(username, password, 1000000000L);
        if (token == null)
            System.out.println("Token not found" + token);
        else {
            System.out.println("Token found for configured user. ");
        }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * This method is to check SanWSService is available or not. If available
 * bind to the stub.
 */
public static void getZoneManagerService() {
    try {

        String zmServiceAddr = "http://" + url
                + "/ZoneManagerWSService/ZoneManagerWS";
        ZoneManagerServiceLocator zmService = new ZoneManagerServiceLocator();
        zmService.setEndpointAddress("ZoneManagerPort", zmServiceAddr);
        zoneManager = zmService.getZoneManagerPort();

        if (zoneManager == null) {
            System.out.println("zonemanager service not available");
            return;
        }
        SOAPHeaderElement hdrElement = setSoapHeader(token);
        ZoneManagerBindingStub zmStub = (ZoneManagerBindingStub) zoneManager;
        zmStub.setHeader(hdrElement);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * This method is to check ZoneManagerWSService is available or not. If
 * available bind to the stub.
 */
public static void getSanService() {
    try {

        String sanServiceAddr = "http://" + url + "/SanWSService/SanWS";
        SanServiceLocator sanService = new SanServiceLocator();
        sanService.setEndpointAddress("SanPort", sanServiceAddr);
        san = sanService.getSanPort();
        if (san == null) {
            System.out.println("service not available");
```

```
                return;
            }
            SOAPHeaderElement hdrElement = setSoapHeader(token);
            SanBindingStub sStub = (SanBindingStub) san;
            sStub.setHeader(hdrElement);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static SOAPHeaderElement setSoapHeader(String token) {

        SOAPHeaderElement hdrElement = new SOAPHeaderElement(
                "http://ep.cisco.dcbu.cisco.com", "token");

        hdrElement.setPrefix("m");

        hdrElement.setMustUnderstand(false);

        hdrElement.setValue(token);
        return hdrElement;
    }

    static void exception(Exception e){
        header("ZonwWsWclient: Some exception occurred!! \n"+e.toString()+"\nEnter [y] to
view the stack trace or any key to continue...");
        try {
            String view=input_.readLine();
            if(view.equalsIgnoreCase("y")){
                e.printStackTrace();

            }
        } catch (IOException e1) {
            exception(e);
        }
    }

}
```

Perl Client

This section describes the Perl client that use the SOAP web services, installing ActiveState Perl, and the SOAP:Lite package. Also, this section provides information about installing ActiveState Perl, the SOAP:Lite package and the Perl client to consume the exposed SOAP web services along with the sample code.

This section contains the following sections:

- Installing Perl
- Installing SOAP:Lite
- Sample Code for Perl Client
- Running the Zone Client
- Running the Statistics Client

Before executing the Perl script, install ActiveState Perl and the SOAP:Lite package.

Installing Perl

Step 1 Download Perl from http://www.perl.org/get.html

Note Choose and download the appropriate Perl version for your operating system.

Step 2 Run the downloaded executable file to install Perl.

Step 3 To verify whether Perl has been installed properly, use the **perl -v** CLI command in the command prompt.
If the installation is proper, the details of the installed version of Perl are displayed.

Installing SOAP:Lite

Download SOAP:Lite from http://search.cpan.org/dist/SOAP-Lite/.

Note Choose and download the appropriate SOAP:Lite version for your operating system.

To install SOAP:lite, follow these steps:

Step 1 Using the command prompt, browse to the directory where the Makefile.PL is located in the downloaded SOAP:Lite package.

Step 2 Use the **perl Makefile.PL -noprompt** CLI command to install the SOAP:Lite package.

Step 3 Download nmake utility for Windows from ftp://ftp.microsoft.com/Softlib/MSLFILES/Nmake15.exe.

Step 4 Extract the downloaded Nmake15.exe
You will find two files, NMAKE.exe and NMAKE.ERR.
Copy both the NMAKE.exe and NMAKE.ERR files to the installed Perl bin directory.

Step 5 Complete the SOAP:Lite installation by entering the following commands for Windows and Linux machines:

```
nmake
nmake test
nmake install
```

For more information on installing SOAP:Lite, see http://soaplite.com/install.html.

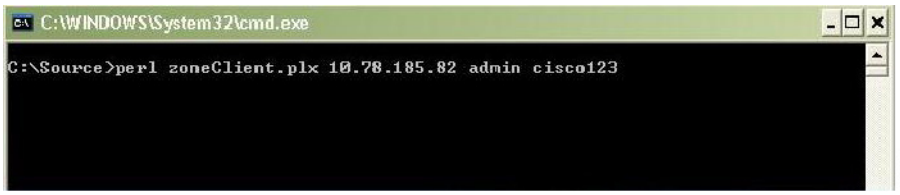
Running the Perl Clients

Note To run a Perl client, the client file and its module file need to be in the same Perl folder.

From the Perl folder where the source files exist, enter the following command:

```
perl Client-File-Name FM-Server-IP Username password
```

Figure A-1 Running a Perl Client



Running the Zone Client

To run the zone client, the files zoneClient.plx and zoneModule.plx need to be in same directory.

From the Perl folder where the source files exist, enter the following command:

perl *zoneClient.plx FM-Server-IP Username password*

Running the Statistics Client

To run the statistics client, the files statistics.plx and statisticsModule.plx need to be in same directory.

From the Perl folder where the source files exist, enter the following command:

perl *statistics.plx FM-Server-IP Username password*

Sample Code for Perl Client

Note

- The sample code provided here works only with DCNM release 6.1(x) and later. For prior DCNM versions, you need to make changes to the namespace base on the WSDL.
- To collect statistics related data, you need to turn on the PM.

Example A-2 ZoneClient.plx

```
# File: zoneClient.plx
# Name: Shailin Saraiya
# Description: zoneClient

# Variables

my $username;
my $password;
my $vsanId = 01;
my $vsan_wwn = '20:01:00:0d:ec:19:6a:81';
my $wwn = '20:00:00:0d:ec:19:6a:80';
my $zoneName = 'test5';
my $readOnly = 0;
my $broadcast = 0;
my $qos = 0;
my $qosPriority = -1;
my $zoneMemberType = 1;
my $zoneMemberFormat = 1;
my $zoneMemberIvrFabricIndex = -1;
my $zoneMemberIvrVsanIndex = -1;
my $zoneMemberId = 'b1a2334455667788';
my $zoneLastModtime = 0;
```

```
my $zoneMemberLunId = '';
my $zoneMemberDbId = 0;
my $zoneisIvr = 0;
my $zoneSetName = 'ZoneSet5';
my $active = 0;

require 'zoneModule.plx';
 ($url,$username,$password) = @ARGV;
# Getting Token
my $token = getToken($url,$username,$password);
if(!$token) {
print "\nError in getToken call \n";
goto exit;
}
# Creating Zone
my $zoneIndex =
createZone($token,$vsanId,$vsan_wwn,$wwn,$zoneName,$readOnly,$broadcast,$qos,$qosPriority)
;

if(!$zoneIndex) {
print "\nError in createZone call \n";
goto exit;
}
# Adding Member to Zone
my $status =
addZoneMemberToZone($token,$vsanId,$vsan_wwn,$wwn,$zoneName,$zoneMemberType,$zoneMemberFor
mat,$zoneMemberIvrFabricIndex,$zoneMemberIvrVsanIndex,$zoneMemberId,$zoneMemberLunId);
if(!$status) {
print "\nError in addZoneMemberToZone call \n";
goto exit;
}
# Creating ZoneSet
my $zoneSetIndex = createZoneSet($token,$vsanId,$vsan_wwn,$wwn,$zoneSetName);
  if(!$zoneSetIndex) {
 print "\nError in createZoneSet call \n";
 goto exit;
 }
# Add Zone to ZoneSet
my $status =
addZoneToZoneset($token,$vsanId,$vsan_wwn,$wwn,$zoneIndex,$zoneName,$readOnly,$broadcast,$
qos,$qosPriority,$zoneMemberDbId,$zoneMemberType,$zoneLastModtime,$zoneMemberLunId,$zoneis
Ivr,$zoneSetIndex,$zoneSetName,$active);
if(!$status) {
print "\nError in addZoneToZoneset call \n";
goto exit;
}
# Activate ZoneSet
$status = activateZoneset($token,$vsanId,$vsan_wwn,$wwn,$zoneSetName);
if(!$status) {
print "\nError in activateZoneset call \n";
goto exit;
}
else {
print "successful \n";
}
exit:
```

Example A-3 ZoneModule.plx

```
use SOAP::Lite;
# Get Token
sub getToken {
```

```
      my $url;
      my $username;
      my $password;
   ($url,$username,$password) = @_;
      # Setting uri and proxy


      my $client = SOAP::Lite
      #->uri('http://ep.jaxws.dcbu.cisco.com/')
    ->proxy('https://'.$url.'/LogonWSService/LogonWS?wsdl')
    ->ns('http://ep.jaxws.dcbu.cisco.com/','namesp1');
      my $username = SOAP::Data->type('string')->name('username')->value($username);
      my $password = SOAP::Data->type('string')->name('password')->value($password);
      my $expiration = SOAP::Data->type('long')->name('expiration')->value(100000000);
      # Calling requestToken method
         my $result = $client->requestToken($username,$password,$expiration);
      # check for error
      unless ($result->fault) {
      my $token =  $result->result();
      return $token;
      } else {
      # error handling
      print join ', ',
        $result->faultcode,
        $result->faultstring,
        $result->faultdetail;
      return 0
      }
}


# Create Zone

sub createZone {

    my $token;
    my $arg_vsanId;
    my $arg_vsanwwn;
    my $arg_wwn;
    my $arg_zoneName;
    my $arg_readOnly;
    my $arg_broadcast;
    my $arg_qos;
    my $arg_qosPriority;

($token,$arg_vsanId,$arg_vsanwwn,$arg_wwn,$arg_zoneName,$arg_readOnly,$arg_broadcast,$arg_
qos,$arg_qosPriority) = @_;
my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
                              SOAP::Data->name("vsanID" => $arg_vsanId),
    SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value"    =>
$arg_vsanwwn)))->type("Wwn")))->type("VsanKey");
my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
     SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn")))->type("WwnKey");
    my $zName = SOAP::Data->name("zoneName"=>$arg_zoneName);
    my $readOnly = SOAP::Data->name("readOnly")->value($arg_readOnly);
    my $broadcast = SOAP::Data->name("broadcast")->value($arg_broadcast);
    my $qos = SOAP::Data->type('boolean')->name("qos")->value($arg_qos);
    my $qosPriority =
SOAP::Data->type('int')->name("qosPriority")->value($arg_qosPriority);

    # Setting header
```

```
     my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');
    my $client = SOAP::Lite
            #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
            ->proxy('https://'.$url.'/ZoneManagerWSService/ZoneManagerWS?wsdl')
                   ->ns('http://ep.san.jaxws.dcbu.cisco.com/','namesp2');
    # Calling createZone method
my $result =
$client->createZone($header,$VsanKey,$Wwn,$zName,$readOnly,$broadcast,$qos,$qosPriority);
    unless ($result->fault) {
          return $result->valueof('//index');
    } else {
        print join ', ',
              $result->faultcode,
              $result->faultstring,
              $result->faultdetail;
        return 0;
           }
    }

# Adding Zone Member To Zone

sub addZoneMemberToZone {
    my $token;
    my $arg_vsanId;
    my $arg_wwn;
    my $arg_vsanwwn;
    my $arg_zoneName;
    my $arg_zoneMemberType;
    my $arg_zoneMemberFormat;
    my $arg_zoneMemberIvrFabricIndex;
    my $arg_zoneMemberIvrVsanIndex;
    my $arg_zoneMemberId;
    my $arg_zoneMemberLunId;


($token,$arg_vsanId,$arg_vsanwwn,$arg_wwn,$arg_zoneName,$arg_zoneMemberType,$arg_zoneMembe
rFormat,$arg_zoneMemberIvrFabricIndex,$arg_zoneMemberIvrVsanIndex,$arg_zoneMemberId,$arg_z
oneMemberLunId) = @_;
    my @hex    = ($arg_zoneMemberId =~ /(..)/g);
    my @dec    = map { hex($_) } @hex;
    my @zoneMemberId  = map { pack('C',$_) } @dec;
    my $arg_zoneMemberId = join("",@zoneMemberId);

    my $client = SOAP::Lite
              #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
              ->proxy('https://'.$url.'/ZoneManagerWSService/ZoneManagerWS?wsdl')
              ->ns('http://ep.san.jaxws.dcbu.cisco.com/','namesp3');
    my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
              SOAP::Data->name("vsanID" => $arg_vsanId),
              SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value" =>
$arg_vsanwwn)))->type("Wwn")))->type("VsanKey");
    my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
          SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn")))->type("WwnKey");
    my $zName = SOAP::Data->name("zoneName"=>$arg_zoneName);
    my $zMemberType = SOAP::Data->name("zoneMemberType")->value($arg_zoneMemberType);
    my $zMemberFormat =
SOAP::Data->name("zoneMemberFormat")->value($arg_zoneMemberFormat);
    my $zMemberIvrFabricIndex =
SOAP::Data->name('zoneMemberIvrFabricIndex')->value($arg_zoneMemberIvrFabricIndex);
```

```
    my $zMemberIvrVsanIndex =
SOAP::Data->type('int')->name("zoneMemberIvrVsanIndex")->value($arg_zoneMemberIvrVsanIndex
);
    my $zMemberId = SOAP::Data->name("zoneMemberId")->value($arg_zoneMemberId);
    my $zMemberLunId = SOAP::Data->name("zoneMemberLunId")->value(arg_zoneMemberLunId);


    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');
         my $result =
$client->addZoneMemberToZone($header,$VsanKey,$Wwn,$zName,$zMemberType,$zMemberFormat,$zMe
mberIvrFabricIndex,$zMemberIvrVsanIndex,$zMemberId,$zMemberLunId);
    unless ($result->fault) {
            return 1;
    }
    else {
    print join ', ',
          $result->faultcode,
          $result->faultstring,
          $result->faultdetail;
    return 0;
    }
}

# Creating ZoneSet

sub createZoneSet {

    my $token;
    my $arg_vsanId;
    my $arg_vsanwwn;
    my $arg_wwn;
    my $arg_zoneSetName;

    ($token,$arg_vsanId,$arg_vsanwwn,$arg_wwn,$arg_zoneSetName) = @_;
    my $client = SOAP::Lite
           #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
           ->proxy('https://'.$url.'/ZoneManagerWSService/ZoneManagerWS?wsdl')
               ->ns('http://ep.san.jaxws.dcbu.cisco.com/','namesp4');
    # Creating complex data type

    my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
              SOAP::Data->name("vsanID" => $arg_vsanId),
              SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value" =>
$arg_vsanwwn)))->type("Wwn")))->type("VsanKey");
    my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
         SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn")))->type("WwnKey");
    my $zSetName = SOAP::Data->name("zoneSetName"=>$arg_zoneSetName);

    # Setting header

    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');

    # Calling createZone method

    my $result = $client->createZoneSet($header,$VsanKey,$Wwn,$zSetName);

    unless ($result->fault) {
          return $result->valueof('//index')."\n";
     } else {
```

```
        print join ', ',
              $result->faultcode,
              $result->faultstring,
              $result->faultdetail;
    return 0;
      }
    }


# Adding Zone to ZoneSet
sub addZoneToZoneset {
    my $token;
    my $arg_vsanId;
    my $arg_vsanwwn;
    my $arg_wwn;
    my $arg_index;
    my $arg_zoneName;
    my $arg_readOnly;
    my $arg_broadcast;
    my $arg_qos;
    my $arg_qosPriority;
    my $arg_zoneMemberDbId;
    my $arg_zoneLastModtime;
    my $arg_memberType;
    my $arg_zoneMemberLunId;
    my $arg_isIvr;
    my $arg_zoneSetindex;
    my $arg_zoneSetName;
    my $active;


($token,$arg_vsanId,$arg_vsanwwn,$arg_wwn,$arg_index,$arg_zoneName,$arg_readOnly,$arg_broa
dcast,$arg_qos,$arg_qosPriority,$arg_zoneMemberDbId,$arg_memberType,$arg_zoneLastModtime,$
arg_zoneMemberLunId,$arg_isIvr,$arg_zoneSetindex,$arg_zoneSetName,$active) = @_;

    my $client = SOAP::Lite
            #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
            ->proxy('https://'.$url.'/ZoneManagerWSService/ZoneManagerWS?wsdl')
               ->ns('http://ep.san.jaxws.dcbu.cisco.com/','namesp5');
    # Creating complex data type

    my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
              SOAP::Data->name("vsanID" => $arg_vsanId),
              SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value" =>
$arg_vsanwwn)))->type("Wwn")))->type("VsanKey");
    my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
          SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn")))->type("WwnKey");
    my $zoneDO = SOAP::Data->name("zoneDo"=> \SOAP::Data->value(
              SOAP::Data->name("index"=>$arg_index),
                  SOAP::Data->name("name"=>$arg_zoneName),
              SOAP::Data->name("readonly"=>$arg_readonly),
              SOAP::Data->name("qos"=>$arg_qos),
              SOAP::Data->name("qosPriority"=>arg_qosPriority),
              SOAP::Data->name("broadcast"=>$arg_broadcast),
                  SOAP::Data->name("active"=>$active),
              SOAP::Data->name("zoneLastModtime"=>arg_zoneLastModtime),
                  SOAP::Data->name("zoneMemberDbId"=>$arg_zoneMemberDbId),
                  SOAP::Data->name("vsanId"=>$arg_vsanId),
              SOAP::Data->name("isIvr"=>$arg_isIvr),
              SOAP::Data->name("memberType"=>$arg_memberType),
              SOAP::Data->name("lunId"=>$arg_zoneMemberLunId)))->type("zone");
    my $zoneSetDO = SOAP::Data->name("zoneSetDo"=> \SOAP::Data->value(
            SOAP::Data->name("index"=>$arg_zoneSetindex),
```

```
            SOAP::Data->name("name"=>$arg_zoneSetName),
            SOAP::Data->name("active"=>$active),
            SOAP::Data->name("zoneLastModified"=>$arg_zoneLastModtime)))->type("ZoneSet");

    # Setting header

    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');

    # Calling addZonetoZoneSet method

    my $result = $client->addZoneToZoneset($header,$VsanKey,$Wwn,$zoneDO,$zoneSetDO);

    unless ($result->fault) {
        return 1;
    } else {
        print join ', ',
              $result->faultcode,
              $result->faultstring,
              $result->faultdetail;
      return 0;
      }
}

#Actvating ZoneSet

sub activateZoneset {

    my $token;
    my $arg_vsanId;
    my $arg_vsanwwn;
    my $arg_wwn;
    my $arg_zoneSetName;


    ($token,$arg_vsanId,$arg_vsanwwn,$arg_wwn,$arg_zoneSetName) = @_;

    my $client = SOAP::Lite
        #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
        ->proxy('https://'.$url.'/ZoneManagerWSService/ZoneManagerWS?wsdl')
            ->ns('http://ep.san.jaxws.dcbu.cisco.com/','namesp6');
      # Creating complex data type

      my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
                SOAP::Data->name("vsanID" => $arg_vsanId),
                  SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value"
=> $arg_vsanwwn)))->type("Wwn")))->type("VsanKey");
    my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
          SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn")))->type("WwnKey");
    my $zSetName = SOAP::Data->name("zoneSetName"=>$arg_zoneSetName);

      # Setting header

    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');

    # Calling activateZoneset method

      my $result = $client->activateZoneset($header,$VsanKey,$Wwn,$zSetName);
      unless ($result->fault) {
      return 1;
```

```
      } else {
          print join ', ',
                $result->faultcode,
                 $result->faultstring,
                $result->faultdetail;
        return 0;
        }
    }
    1;
```

Example A-4 StatisticsClient.plx

```
# File: StatisticsClient.plx
# Name: Shailin Saraiya
# Description: Statistics Client

# Variables

my $username;
my $password;
my $url;
my $fabricDbId = -1;
my $switchDbId = -1;
my $vsanDbId = -1;
my $sortField = 'rxTxStr';
my $sortType = 'DESC';
my $limit = NaN;
my $groupId = -1;
my $isGroup = 0;
my $networkType ='SAN';
my $filterStr = '';
my $temp = '24 Hours';
my $startIdx = 0;
my $recordSize = 45;
my $interval = 1;
require 'statisticsModule.plx';

($url,$username,$password) = @ARGV;


# Getting Token
my $token = getToken($url,$username,$password);
 ($rrdFile,$fid,$pmType) =
getIslStatList($token,$fabricDbId,$switchDbId,$vsanDbId,$sortField,$sortType,$limit,$group
Id,$isGroup,$networkType,$filterStr,$temp,$startIdx,$recordSize);
for (my $i=0;$i<1;$i++) {
    getPmChartData($token,@$rrdFile[$i],@$pmType[$i],@$fid[$i],$interval) . "\n";

}
```

Example A-5 StatisticsModule.plx

```
use SOAP::Lite;
use Time::Local;

# Get Token
$url = '';

sub getToken {
      my $username;
```

```
      my $password;
       ($url,$username,$password) = @_;

      # Setting uri and proxy

      my $client = SOAP::Lite
      #->uri('http://ep.jaxws.dcbu.cisco.com/')
    ->proxy('https://'.$url.'/LogonWSService/LogonWS?wsdl')
        -> ns('http://ep.jaxws.dcbu.cisco.com/','namesp1');
        #-> autotype(0)
        #-> readable(1);
      my $username = SOAP::Data->type('string')->name('username')->value($username);
      my $password = SOAP::Data->type('string')->name('password')->value($password);
      my $expiration = SOAP::Data->type('long')->name('expiration')->value(100000000);

      # Calling requestToken method

      my $result = $client->requestToken($username,$password,$expiration);
           # check for error
      unless ($result->fault) {
          my $token =  $result->result();
      $token;
      } else {

      # error handling
      print join ', ',
        $result->faultcode,
        $result->faultstring,
        $result->faultdetail;
      }
}
sub getIslStatList {
    my $token;
    my $arg_fabricDbId;
    my $arg_switchDbId;
    my $arg_vsanDbId;
    my $arg_sortField;
    my $arg_sortType;
    my $arg_limit;
    my $arg_groupId;
    my $arg_isGroup;
    my $arg_networkType;
    my $arg_filterStr;
    my $arg_interval;
    my $arg_startIdx;
    my $arg_recordSize;



($token,$arg_fabricDbId,$arg_switchDbId,$arg_vsanDbId,$arg_sortField,$arg_sortType,$arg_li
mit,$arg_groupId,$arg_isGroup,$arg_networkType,$arg_filterStr,$arg_interval,$arg_startIdx,
$arg_recordSize) = @_;
        my $DbFilter= SOAP::Data->name("arg0" => \SOAP::Data->value(
              SOAP::Data->name("fabricDbId" => $arg_fabricDbId),
              SOAP::Data->name("switchDbId" => $arg_switchDbId),
              SOAP::Data->name("vsanDbId" => $arg_vsanDbId),
              SOAP::Data->name("sortField" => $arg_sortField),
              SOAP::Data->name("sortType" => $arg_sortType),
              SOAP::Data->name("limit" => $arg_limit),
              SOAP::Data->name("groupId" => $arg_groupId),
              SOAP::Data->name("isGroup" => $arg_isGroup)->type("boolean"),
              SOAP::Data->name("networkType" => $arg_networkType),
              SOAP::Data->name("filterStr" => $arg_filterStr)))->type("DbFilter");
```

```
    my $interval = SOAP::Data->type("string")->name("arg1")->value($arg_interval);
    my $data_startIdx = SOAP::Data->name("arg2"=>$arg_startIdx);
    my $data_recordSize = SOAP::Data->name("arg3"=>$arg_recordSize);

    # Setting header

    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');
    #my $header = SOAP::Header->name('token')->prefix('m')->value($token)->type('string');
    my $client = SOAP::Lite
            #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
            ->proxy('https://'.$url.'/StatisticsWSService/StatisticsWS?wsdl')
                        -> ns('http://ep.san.jaxws.dcbu.cisco.com/','namesp2');
                        #-> autotype(0)
                        #-> readable(1);

    # Calling getIslStatDataLength method
      my $result =
$client->getIslStatList($header,$DbFilter,$interval,$data_startIdx,$data_recordSize);
    unless ($result->fault) {
    print "\ngetIslStatList\n\n";
         my @fabric = $result->valueof('//item/fabric');
         my @title = $result->valueof('//item/title');
         my @maxTxStr = $result->valueof('//item/maxTxStr');
         my @maxRxStr = $result->valueof('//item/rxTxStr');
         my @rxTx = $result->valueof('//item/rxTx');

         my $count = @fabric;

         for(my $i=0;$i<10;$i++)
         {

            print "------------------------------------------------------------\n";
         print "Fabric -> " . $fabric[$i] . "\n";
         print "Title -> " . $title[$i] . "\n";
         print "maxTxStr -> " . $maxTxStr[$i] . "\n";
         print "maxRxStr -> " . $maxRxStr[$i] . "\n";
         print "rxTx -> " . $rxTx[$i] . "\n";
         print "------------------------------------------------------------\n";

         }

         my @rrdFile = $result->valueof('//item/rrdFile');
         my @fid = $result->valueof('//item/fid');
         my @pmType = $result->valueof('//item/pmtype');
         return (\@rrdFile,\@fid,\@pmType);

    } else {
        print join ', ',
              $result->faultcode,
              $result->faultstring,
              $result->faultdetail;
          }


}


sub getPmChartData {
    my $token;
    my $arg_rrd;
    my $arg_pmType;
    my $arg_fid;
```

```
    my $arg_interval;
    ($token,$arg_rrd,$arg_pmType,$arg_fid,$arg_interval) = @_;
    my $data_rrd = SOAP::Data->type("string")->name("arg0")->value($arg_rrd);
    my $data_pmType = SOAP::Data->name("arg1")->value($arg_pmType);

    my $data_fid = SOAP::Data->name("arg2"=>$arg_fid);
    my $data_interval = SOAP::Data->name("arg3"=>$arg_interval);

    # Setting header
    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');
    my $client = SOAP::Lite
            #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
            ->proxy('https://'.$url.'/StatisticsWSService/StatisticsWS?wsdl')
                        -> ns('http://ep.san.jaxws.dcbu.cisco.com/','namesp3');
                        #-> autotype(0)
                        #-> readable(1);

    # Calling getPmChartData method
       my $result =
$client->getPmChartData($header,$data_rrd,$data_pmType,$data_fid,$data_interval);
    unless ($result->fault) {
    print "\n\nGetPmChartData\n\n";

         my @item = $result->valueof('//item/item');
         for(my $i=2;$i<30;$i=$i+3) {
         print "------------------------------------------------------------\n";
         print "Rx -> ".$item[$i-2]."\n";
         print "Tx -> ".$item[$i-1]."\n";
         print "Time -> ".scalar localtime($item[$i]) . "\n";
         print "------------------------------------------------------------\n";
         }
        } else {
        print join ', ',
              $result->faultcode,
              $result->faultstring,
              $result->faultdetail;
          }
}

1;
```

REST Web Services

The following client programs use REST web services in Cisco DCNM.

This section contains the following sections:

- Python Client
- JavaScript Client

Python Client

This section provides information about installing Python, and using the python client to consume the exposed REST web services along with the sample code.

This section contains the following sections:

Installing Python

Step 1 You need to use the Python version 3.6.1.

Step 2 Download Python from https://www.python.org/downloads/release/python-361/.

Installing Pycharm (IDE)

Step 1 You need to use the Pycharm version 3.6.1.

Step 2 Download Pycharm from https://www.jetbrains.com/pycharm/download/#section=windows.

Running Python Client Using IDE

Step 1 Using Pycharm (Python IDE), set the interpreter to the location where python is installed. For example, C:\Users\ Local\Programs\Python\Python36-32\python.exe.

Note In the *MyClient.py* file, set the queryKey, zone name, zoneset name, fabricDBID accordingly.

Step 2 Go to **Run > Edit Configurations**. In the Script parameters tab provide the arguments such as FMServerIP, UserName, Password that need to be separated by blank spaces. For example, FMServerIP username password)

Step 3 Right-click on the file, and select “Run FileName”.

Running Python Client Using Command Prompt

Step 1 Using the command prompt, go to the location where python is installed.

Note In the *MyClient.py* file, set the queryKey, zone name, zoneset name, fabricDBID accordingly.

```
C:\users\exampleuser\AppData\Local\Programs\Python\Python36>python.exe
C:\Users\exampleuser\PycharmProjects\PythonRestClient\MyClient.py FMServerIP username
password
```

Step 2 Execute "**python.exe** *<complete path for MyClient.py file> FMServerIP username password*". For example:

```
C:\Python\Python36-32>python.exe C:\Users\PycharmProjects\PythonRestClient\MyClient.py
FMServerIP username password
```

Sample Code for Python Client

Example A-6 MyClient.py

```
import sys
import time
from restfactory import RestFactory

try:
    fmserver = sys.argv[1]
    user = sys.argv[2]
    password = sys.argv[3]
    rf = RestFactory(fmserver, user, password)
    i_rest = rf.get_webzonemanager_rest();

    zonesetName = "PythonZoneSetTest"
    zoneName = "PythonTest"
    vsanID = 200
    switchName = "sw-9148-248"
    fabricName = "Fabric_sw-9250i-244"
    cacheBust = i_rest.getCacheBust()
    navId = "-1";
    fabricDBID = i_rest.get_fabrics(cacheBust, navId,fabricName)
    print("fabricDBID "+str(fabricDBID))
    vsanDBID = i_rest.get_vsanDBID(fabricDBID,cacheBust,navId,vsanID,switchName)
    print("VSANDBID "+str(vsanDBID))
    swwn =
i_rest.get_switchesForVsan(fabricDBID,vsanDBID,cacheBust,navId,vsanID,switchName)
    print("SwitchWWN "+swwn)

    #queryKey is the combination of <FDBID>,<VSANIndex>,<SWWWN>,<client ID>,<otherclient
data>
    queryKey = i_rest.generate_queryKey(fabricDBID,vsanID,swwn,"-1","-1")
    print(queryKey)

    time.sleep(2)
    createZoneSetResponse = i_rest.createZoneSet(queryKey, zonesetName, fabricDBID)

    # creatingZONE
    createZoneResponse = i_rest.create_zone(queryKey, zoneName, fabricDBID)
    #
    # # addingZoneToZoneSet
    addZoneResponse = i_rest.addZoneToZoneSet(queryKey, zoneName, zonesetName, fabricDBID)
    #
    # # deletingZoneFromZoneSet
    removeZoneFromZoneSet = i_rest.removeZoneToZoneSet(queryKey, zoneName, zonesetName,
fabricDBID)
    #
    # # deletingZone
    deleteZoneResponse = i_rest.deleteZone(queryKey, zoneName, fabricDBID)
    #
    # # deleting ZoneSet
```

```
    deleteZoneSet = i_rest.deleteZoneset(queryKey, zonesetName, fabricDBID)

except Exception as e:
    print(e)
```

Example A-7 restfactory.py

```
from san_rest_api import RestApi
from webzonemanagerrest import WebZoneManager

class RestFactory(object):
    _cookie = {}
    _dcnm_url = None
    _username = None
    _password = None
    _rest_api = None
    _is_remote = False
    _expiration_time = "9999999"

    def __init__(self, dcnm_url , username = None, password = None):
        if username is not None:
            self._username = username
            self._password = password
            print(dcnm_url)
            dcnm_url = "https://" + dcnm_url + "/"
            print(dcnm_url)
            self._rest_api = RestApi(dcnm_url,
                             self._username,
                             self._password,
                             self._expiration_time)
            print(dcnm_url)
            print("here")
            print(self._rest_api)
            dcnm_token = self._rest_api.dcnm_token['Dcnm-Token']
            if dcnm_token is None:
                raise Exception("Failed to instantiate RestFactory as Dcnm-Token is :" +
dcnm_token)
            else:
                self._rest_api.headers['Dcnm-Token'] = dcnm_token
                print("headers with token")
                print(self._rest_api.headers)
        self._dcnm_url = dcnm_url

    def get_webzonemanager_rest(self):
        return WebZoneManager(self._dcnm_url, self._rest_api)
```

Example A-8 san_rest_api.py

```
"""This class handles all the REQUEST modules required for DCNM"""
import ast
import base64
import json
import requests


LOGON_URL = 'rest/logon'

class RestApi(object):
```

```
    """This handles all the request modules"""

    def __init__(self, *args):
        self.headers = {'Content-Type': 'application/json; charset=utf8'}
        self.verify = False
        if len(args) != 0:
            self.dcnm_url = args[0]
            self.username = args[1]
            self.password = args[2]
            self.logon_url = self.dcnm_url + LOGON_URL
            self.expiration_time = args[3]
            self.dcnm_token = None
            self.fdbid = None
            self.get_dcnmtoken()

    def get_dcnmtoken(self):
        self.credentials = self.generate_encoded_credentials(
            self.username, self.password)
        self.dcnm_token = self.generate_dcnmtoken(self.credentials)
        print("dcnm-token")
        print(self.dcnm_token)
        return self.dcnm_token

    def generate_encoded_credentials(self, username, password):
        """This generates the credentials"""
        data = {'Content-Type': 'application/json; charset=utf8'}
        credentials = username + ':' + password
        encoded = base64.b64encode(credentials.encode())
        print("///////////////")
        print(encoded)
        print(encoded.decode('ascii'))
        print("/////////////")
        data.update({"Authorization": 'Basic ' + encoded.decode('ascii')})
        print(data)
        return data

    def generate_dcnmtoken(
            self,
            headers,
            js_token=None):
        """This generates valid DCNM-Token for the encoded credentials"""
        dcnm_token = None
        payload = {'expirationTime': self.expiration_time}
        print("//////////Header///////")
        print(headers)
        try:
            print(self.logon_url)
            response = requests.post(
                self.logon_url,
                data=json.dumps(payload),
                headers=headers,
                verify=self.verify)
            if response.status_code == 200:  # Successful token generation
                dcnm_token = ast.literal_eval(response.text)
        except:
            return None
        return dcnm_token
```

Example A-9 webzonemanagerrest.py

```
import requests
```

```
import time
import json

class WebZoneManager(object):

    ZONE_REST_PATH = "fm/fmrest/WebZoneManager/"
    CREATE_ZONE = "createZoneMod"
    CREATE_ZONESET = "createZoneSetMod"
    ADD_ZONE_TO_ZONESET = "addZonesToZonesetMod"
    REGISTER_AND_POPULATE = "registerAndPopulateLocalZoneDBGet"
    DELETE_ZONE_FROM_ZONESET = "deleteZonesFromZonesetMod"
    DELETE_ZONESET = "deleteZonesetsMod"
    DELETE_ZONE = "deleteZonesMod"
    GET_FABRICS_URL = 'fm/fmrest/dcnm/fabrics'
    GET_VSANS_MOD_URL = "getVsansMod"
    GET_SWITCHES_IN_VSAN = "getSwitchesInVSANMod"

    def __init__(self, dcnm_url, restapi):
        self._dcnm_url = dcnm_url + self.ZONE_REST_PATH
        self._rest_api = restapi
        self.get_fabrics_url = dcnm_url + self.GET_FABRICS_URL
        self.verify = False
        self.headers = {'Content-Type': 'application/json; charset=utf8'}

    def createZoneSet(self, queryKey, zonesetName, fabricDBID):
        print("creating zoneset")
        zoneset_url = self._dcnm_url+self.CREATE_ZONESET;
        print(zoneset_url)
        self._rest_api.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        payload = {}
        payload['queryKey'] = queryKey
        payload['zoneSetName'] = zonesetName
        payload['fabricDBID'] = fabricDBID
        response = requests.post(url=zoneset_url, data=payload,
headers=self._rest_api.headers, verify=self.verify)
        print(response.text)

    def create_zone(self, queryKey, zoneName, fabricDBID):
        print("creating zone")
        zone_url = self._dcnm_url + self.CREATE_ZONE
        print(zone_url)
        self._rest_api.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        payload = {}
        payload['queryKey'] = queryKey
        payload['zoneName'] = zoneName
        payload['fabricDBID'] = fabricDBID
        response = requests.post(zone_url, data=payload, headers=self._rest_api.headers,
verify=self.verify)
        print("printing response")
        print(response.text)

    def addZoneToZoneSet(self, queryKey, zoneNames, zonesetName, fabricDBID):
        print("adding zone to zoneset")
        addzoneurl = self._dcnm_url + self.ADD_ZONE_TO_ZONESET
        print(addzoneurl)
        self._rest_api.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        payload = {}
        payload['queryKey'] = queryKey
        payload['zoneNames'] = zoneNames
        payload['zonesetName'] = zonesetName
        payload['fabricDBID'] = fabricDBID
       response = requests.post(addzoneurl, data=payload, headers=self._rest_api.headers,
verify=self.verify)
        print("printing response")
```

```
        print(response.text)

    def removeZoneToZoneSet(self, queryKey, zoneNames, zonesetName, fabricDBID):
        print("deleting zone from zoneset")
        removezoneurl = self._dcnm_url + self.DELETE_ZONE_FROM_ZONESET
        print(removezoneurl)
        self._rest_api.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        payload = {}
        payload['queryKey'] = queryKey
        payload['zoneName'] = zoneNames
        payload['zonesetName'] = zonesetName
        payload['fabricDBID'] = fabricDBID
        response = requests.post(url=removezoneurl, data=payload,
headers=self._rest_api.headers, verify=self.verify)
        print("printing response")
        print(response.text)

    def deleteZone(self, queryKey, zoneName, fabricDBID):
        print("deleting zone")
        deletezoneurl = self._dcnm_url + self.DELETE_ZONE
        print(deletezoneurl)
        self._rest_api.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        payload = {}
        payload['queryKey'] = queryKey
        payload['zoneName'] = zoneName
        payload['fabricDBID'] = fabricDBID
        response = requests.post(url=deletezoneurl, data=payload,
headers=self._rest_api.headers, verify=self.verify)
        print("printing response")
        print(response.text)

    def deleteZoneset(self, queryKey, zonesetName, fabricDBID):
        print("deleting zoneset")
        delete_ZoneSet_url = self._dcnm_url+self.DELETE_ZONESET
        print(delete_ZoneSet_url)
        self._rest_api.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        payload = {}
        payload['queryKey'] = queryKey
        payload['zoneSetName'] = zonesetName
        payload['fabricDBID'] = fabricDBID
        response = requests.post(url=delete_ZoneSet_url, data=payload,
headers=self._rest_api.headers, verify=self.verify)
        print(response.text)

    def registerAndPopulate(self,queryKey,fabricDBID,clientRequestID,navId):
        registerAndPopulate_url = self._dcnm_url+self.REGISTER_AND_POPULATE
        registerAndPopulate_url += '/?queryKey=' + queryKey + '&fabricDBID=' + fabricDBID
+ '&clientRequestID=' + clientRequestID + \
                            '&cacheBust=' + self.getCacheBust() + '&navId=' + navId
        response = requests.get(registerAndPopulate_url, params=None,
headers=self._rest_api.headers, verify=self.verify)
        resp_dict = json.loads(response.text)
        req_key = resp_dict['requestKey']
        return req_key

    def getCacheBust(self):
        return str(int(time.time() * 1000))

    def get_fabrics(self, cachebust, navId, fabricName):
        print("fetching fabrics")
        get_fabric_url = self.get_fabrics_url+"/?"+'&cacheBust=' + cachebust + '&navId=' +
navId
        print(get_fabric_url)
        self.headers['Content-Type'] = 'application/x-www-form-urlencoded'
```

```
        response = requests.get(url=get_fabric_url, params=None,
headers=self._rest_api.headers, verify=self.verify)
        print(response.text)
        resp = json.loads(response.text)
        for res in resp:
            if res['name'] == fabricName:
                self.fdbid = res['id']
        return self.fdbid

    def get_vsanDBID(self, fabricid, cachebust, navId, vsanID, switchName):
        print("fetching vsans")
        get_vsans_url = self._dcnm_url+self.GET_VSANS_MOD_URL+"/?"+'&fabricDBID=' +
str(fabricid) +'&cacheBust=' + cachebust + '&navId=' + navId
        print(get_vsans_url)
        self.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        self.vsan_dbid = None
        response = requests.get(url=get_vsans_url, params=None,
headers=self._rest_api.headers, verify=self.verify)
        print(response.text)
        resp = json.loads(response.text)
        print(resp)
        for res in resp:
            if res['vsanIndex'] == vsanID:
                if res['prinswName'] == switchName:
                    self.vsan_dbid = (res['DBID'])
        return self.vsan_dbid

    def get_switchesForVsan(self, fabricid, vsanid, cachebust, navId, vsanID, switchName):
        print("fetching switches")
        get_switches_for_vsans_url = self._dcnm_url+self.GET_SWITCHES_IN_VSAN + "/?" +
'&fabricDBID=' + str(
            fabricid) + '&vsanDBID=' + str(vsanid) +'&cacheBust=' + cachebust + '&navId='
+ navId
        print(get_switches_for_vsans_url)
        self.headers['Content-Type'] = 'application/x-www-form-urlencoded'
        self.switchWWN = None
        response = requests.get(url=get_switches_for_vsans_url, params=None,
headers=self._rest_api.headers, verify=self.verify)
        resp = json.loads(response.text)
        print(resp)
        for res in resp[1]:
            if res['sys_name'] == switchName:
                self.switchWWN = res['swWWN']
        return self.switchWWN

    def generate_queryKey(self,fDBID,vsanIndex,SWWN,clientID,otherClientData):
        key = str(fDBID)+","+str(vsanIndex)+","+SWWN+","+clientID+","+otherClientData;
        self.queryKey = self.registerAndPopulate(key,str(fDBID),clientID,"-1")
        print("query key "+self.queryKey)
        return self.queryKey
```

JavaScript Client

This section provides information about using the JavaScript client to access REST Web Services along with sample code and configuring the CORS plugin.

This section contains the following sections:

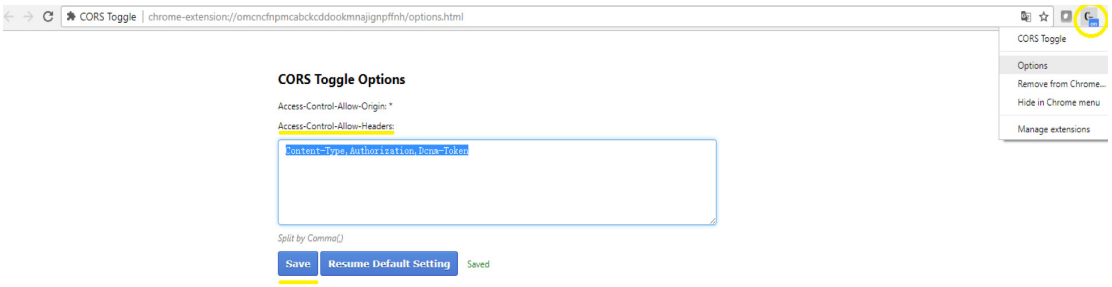
- Downloading and Configuring the CORS Toggle Plug-in

Downloading and Configuring the CORS Toggle Plug-in

Step 1 Download the CORS Toggle plugin from Chrome web store using the following URL:

https://chrome.google.com/webstore/search/CORSToggle?hl=en-US

Step 2 Add the Access control Allow-Headers in the space provided and save. (For example, Content-Type, Authorization, Dcnm-Token).

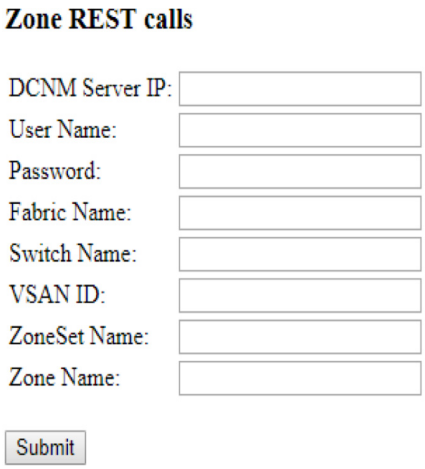


Step 3 Ensure CORS status is ‘ON’.

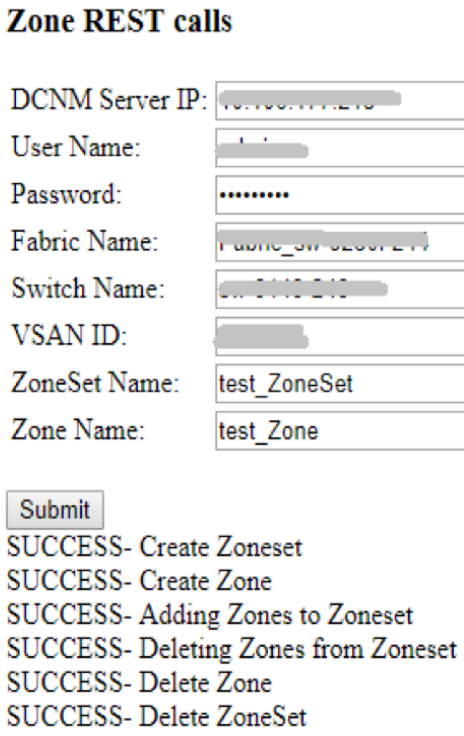
Step 4 Click on the plug-in to change the status.

Running JavaScript REST Client

Step 1 Double-click on *JavaScript_Client.html, which* will show the below UI in web browser.



Step 2 Fill the above fields, and click the **Submit** button. Cisco DCNM performs the supported zone-related operations and displays the status message.



Step 3 After you get a success message, you can verify this in Cisco DCNM GUI (/SAN/Zoning).

Sample Code for JavaScript Client

Example A-10 JavaScript_Client.html

```
<!DOCTYPE html>
<html>
<body>
    <script
        src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.1.min.js"></script>
    <script type="text/javascript">
        function loadDoc() {
            debugger;
            var user = document.getElementById("un").value;
            var password = document.getElementById("pw").value;
            var serverIP = document.getElementById("ip").value;
            var fabricName = document.getElementById("fabricName").value;
            var switchName = document.getElementById("switchName").value;
            var zoneSetName = document.getElementById("zoneSetName").value;
            var zoneName = document.getElementById("zoneName").value;
            var vsanID = document.getElementById("vsanID").value;
            var cacheBust = new Date().getTime() * 1000;
            var navID = -1;
            var encodeUP = btoa(user + ':' + password);
            var authorization='Basic ' + encodeUP;
            var fabricDBID;
            var vsanDbId;
            var swWWN;
            var queryKey;
            var dcnm_Token;
            // Below ajax is for Getting Dcnm-token from serverIP
            $
                    .ajax({
                        type : "POST",
                        headers : {
                            "Authorization" : authorization,
                        },
                        url : "https://" + serverIP + "/rest/logon",
                        data : JSON.stringify({
                            expirationTime : '9999999'
                        }),
                        processData : false,
                        async : true,
                        httpNodeCors : {
                            origin : "*",
                            methods : "GET,PUT,POST,DELETE"
                        },
                        error : function(xhr, status, error) {
                            alert('error !!');
                            document.getElementById("result").innerHTML =
data.statusMessage
                                    + " " + status;

                        },
                        success : function(msg, status, xhr) {
                            console.log(msg);
                            var dcnmToken = xhr.responseText;
                            dcnm_Token = dcnmToken.substring(15,
```

```
                                    dcnmToken.length - 2);
                            getFabrics();
                        }
                    });

        // getting the faricDBID
                function getFabrics() {

                    $.ajax({
                        url : 'https://' + serverIP
                                + "/fm/fmrest/dcnm/fabrics/?&cacheBust="
                                + cacheBust + "&navId=" + navID,
                        type : 'GET',
                        headers : {
                            "Content-Type" : "application/x-www-form-urlencoded",
                            "Authorization" : authorization,
                            "Dcnm-Token" : dcnm_Token
                        },
                        success : function(data, textStatus, xhr) {
                            console.log("data= " + fabrics);
                            var fab = JSON.stringify(data);
                            var fabrics = JSON.parse(fab);
                            for (var i = 0; i < fabrics.length; i++) {
                                if (fabricName == fabrics[i].name) {
                                    fabricDBID = fabrics[i].id;
                                }

                            }
                            getVsanDbId(fabricDBID);
                        },
                        error : function(xhr, textStatus, errorThrown) {
                            console.log(textStatus);
                            document.getElementById("result").innerHTML = " "
                                    + textStatus;
                        }
                    });
                }
    //getting the VsanDBID from the vsanID (vsanID is provided as the input)
                function getVsanDbId(fabricDBID) {

                    $
                            .ajax({
                                url : 'https://'
                                        + serverIP
                                        + '/fm/fmrest/WebZoneManager/getVsansMod/?&fabricDBID='
                                        + fabricDBID + '&cacheBust=' + cacheBust
                                        + '&navId=' + navID,
                                type : 'GET',
                                headers : {
                                    "Content-Type" : "application/x-www-form-urlencoded",
                                    "Authorization" : authorization,
                                    "Dcnm-Token" : dcnm_Token
                                },
                                success : function(data, textStatus, xhr) {
                                    console.log("data= " + data);
                                    console.log(textStatus);
                                    var vsanSwitches = JSON.stringify(data);
                                    var vsanSwitch = JSON.parse(vsanSwitches);
                                    for (var i = 0; i < vsanSwitch.length; i++) {
                                        if (vsanID == vsanSwitch[i].vsanIndex) {
                                            if (switchName == vsanSwitch[i].prinswName) {
                                                vsanDbId = vsanSwitch[i].DBID;
                                            }
                                        }
```

```
                                }
                                getSwitchesInVSANMod(vsanDbId);
                            },
                            error : function(xhr, textStatus, errorThrown) {
                                console.log(textStatus);
                                document.getElementById("result").innerHTML =
data.statusMessage
                                        + " " + textStatus;
                            }

                        });

            }
//getting switch wwn
            function getSwitchesInVSANMod(vsanDbId) {
                        $
                        .ajax({
                            url : 'https://'
                                    + serverIP
                                    +
'/fm/fmrest/WebZoneManager/getSwitchesInVSANMod/?&fabricDBID='
                                    + fabricDBID + '&vsanDBID=' + vsanDbId
                                    + '&cacheBust=' + cacheBust + '&navId='
                                    + navID,
                            type : 'GET',
                            headers : {
                                "Content-Type" : "application/x-www-form-urlencoded",
                                "Authorization" : authorization,
                                "Dcnm-Token" : dcnm_Token
                            },
                            success : function(data, textStatus, xhr) {
                                console.log("data " + data);
                                console.log(textStatus);
                                var sw_wwn = JSON.stringify(data);
                                var sw_WWN = JSON.parse(sw_wwn);
                                for (var i = 0; i < sw_WWN[1].length; i++) {
                                    if (switchName == sw_WWN[1][i].sys_name) {
                                        swWWN = sw_WWN[1][i].swWWN;
                                        console.log(swWWN);
                                    }
                                }
                                generateQueryKey(fabricDBID, vsanID, swWWN, -1,
                                        -1);
                            },
                            error : function(xhr, textStatus, errorThrown) {
                                console.log(textStatus);
                                document.getElementById("result").innerHTML = " "
                                        + textStatus;
                            }

                        });

            }
//generating QueryKey
            function generateQueryKey(fabricId, vsanID, SWWN, clientRequestID,
                    otherClientData) {
                var key = fabricId + ',' + vsanID + ',' + SWWN + ','
                        + clientRequestID + ',' + otherClientData;

                $
                        .ajax({
                            url : 'https://'
                                    + serverIP
```

```
                                    +
'/fm/fmrest/WebZoneManager/registerAndPopulateLocalZoneDBGet/?queryKey='
                                    + key + '&fabricDBID=' + fabricDBID
                                    + '&clientRequestID=' + clientRequestID
                                    + '&cacheBust=' + cacheBust + '&navId='
                                    + navID,
                                type : 'GET',
                                headers : {
                                    "Content-Type" : "application/x-www-form-urlencoded",
                                    "Authorization" : authorization,
                                    "Dcnm-Token" : dcnm_Token
                                },
                                success : function(data, textStatus, xhr) {

                                    console.log("data  " + data);
                                    console.log(textStatus);
                                    var query_Key = JSON.stringify(data);
                                    var query = JSON.parse(query_Key);
                                    queryKey = query.requestKey;
                                    createZoneset(queryKey, zoneSetName, fabricDBID)
                                },
                                error : function(xhr, textStatus, errorThrown) {
                                    console.log(textStatus);
                                    document.getElementById("result").innerHTML = " "
                                            + textStatus;
                                }

                            });
                    }
            // 1. Creating Zoneset
                    function createZoneset(queryKey, zoneSetName, fabricDBID) {
                        debugger;
                        var payload = {};
                        payload['queryKey'] = queryKey;
                        payload['zoneSetName'] = zoneSetName;
                        payload['fabricDBID'] = fabricDBID;
                        $
                                .ajax({
                                    url : 'https://'
                                            + serverIP
                                            + '/fm/fmrest/WebZoneManager/createZoneSetMod',
                                    type : 'POST',
                                    headers : {
                                        "Content-Type" : "application/x-www-form-urlencoded",
                                        "Authorization" : authorization,
                                        "Dcnm-Token" : dcnm_Token
                                    },
                                    data : payload,
                                    success : function(data, textStatus, xhr) {
                                        debugger;
                                        console.log(data);
                                        document.getElementById("result").innerHTML = data.status
                                                + "- Create Zoneset ";
                                        createZone(queryKey, zoneName, fabricDBID);
                                    },
                                    error : function(xhr, textStatus, errorThrown) {
                                        console.log(textStatus);
                                        document.getElementById("result").innerHTML = " "
                                                + textStatus;
                                    }
                                });
                    }
            // 2. Creating Zone
                    function createZone(queryKey, zoneName, fabricDBID) {
```

```
                var payload = {};
                payload['queryKey'] = queryKey;
                payload['zoneName'] = zoneName;
                payload['fabricDBID'] = fabricDBID;
                $
                        .ajax({
                            url : 'https://' + serverIP
                                    + '/fm/fmrest/WebZoneManager/createZoneMod',
                            type : 'POST',
                            headers : {
                                "Content-Type" : "application/x-www-form-urlencoded",
                                "Authorization" : authorization,
                                "Dcnm-Token" : dcnm_Token
                            },
                            data : payload,
                            success : function(data, textStatus, xhr) {
                                debugger;
                                console.log(data);
                                document.getElementById("result1").innerHTML = data.status
                                        + "- Create Zone";
                                addZonesToZoneset(queryKey, zoneName,
                                        zoneSetName, fabricDBID);
                            },
                            error : function(xhr, textStatus, errorThrown) {
                                console.log(textStatus);
                                document.getElementById("result1").innerHTML = " "
                                        + textStatus;
                            }
                        });
            }
// 3. Adding Zones to Zoneset
            function addZonesToZoneset(queryKey, zoneNames, zonesetName,
                    fabricDBID) {
                var payload = {};
                payload['queryKey'] = queryKey;
                payload['zoneNames'] = zoneNames;
                payload['zonesetName'] = zoneSetName;
                payload['fabricDBID'] = fabricDBID;
                $
                        .ajax({
                            url : 'https://'
                                    + serverIP
                                    + '/fm/fmrest/WebZoneManager/addZonesToZonesetMod',
                            type : 'POST',
                            headers : {
                                "Content-Type" : "application/x-www-form-urlencoded",
                                "Authorization" : authorization,
                                "Dcnm-Token" : dcnm_Token
                            },
                            data : payload,
                            success : function(data, textStatus, xhr) {
                                debugger;
                                console.log(data);
                                document.getElementById("result2").innerHTML = data.status
                                        + "- Adding Zones to Zoneset ";
                                deleteZonesFromZoneset(queryKey, zoneName,
                                        zoneSetName, fabricDBID);
                            },
                            error : function(xhr, textStatus, errorThrown) {
                                console.log(textStatus);
                                document.getElementById("result2").innerHTML = " "
                                        + textStatus;
                            }
                        });
```

```
            }
// 4. Deleting Zones from zoneset
            function deleteZonesFromZoneset(queryKey, zoneNames, zonesetName,
                    fabricDBID) {
                debugger;
                var payload = {};
                payload['queryKey'] = queryKey;
                payload['zoneName'] = zoneNames;
                payload['zonesetName'] = zonesetName;
                payload['fabricDBID'] = fabricDBID;
                $
                        .ajax({
                            url : 'https://'
                                    + serverIP
                                    + '/fm/fmrest/WebZoneManager/deleteZonesFromZonesetMod',
                            type : 'POST',
                            headers : {
                                "Content-Type" : "application/x-www-form-urlencoded",
                                "Authorization" : authorization,
                                "Dcnm-Token" : dcnm_Token
                            },
                            data : payload,
                            success : function(data, textStatus, xhr) {
                                debugger;
                                console.log(data);
                                document.getElementById("result3").innerHTML = data.status
                                        + "- Deleting Zones from Zoneset ";
                                deleteZones(queryKey, zoneName, fabricDBID);
                            },
                            error : function(xhr, textStatus, errorThrown) {
                                console.log(textStatus);
                                document.getElementById("result3").innerHTML = " "
                                        + textStatus;
                            }
                        });
            }
// 5. Deleting Zones
            function deleteZones(queryKey, zoneName, fabricDBID) {
                debugger;
                var payload = {};
                payload['queryKey'] = queryKey;
                payload['zoneName'] = zoneName;
                payload['fabricDBID'] = fabricDBID;
                $
                        .ajax({
                            url : 'https://'
                                    + serverIP
                                    + '/fm/fmrest/WebZoneManager/deleteZonesMod',
                            type : 'POST',
                            headers : {
                                "Content-Type" : "application/x-www-form-urlencoded",
                                "Authorization" : authorization,
                                "Dcnm-Token" : dcnm_Token
                            },
                            data : payload,
                            success : function(data, textStatus, xhr) {
                                debugger;
                                console.log(data);
                                document.getElementById("result4").innerHTML = data.status
                                        + "- Delete Zone ";
                                deleteZoneset(queryKey, zoneSetName, fabricDBID);
                            },
                            error : function(xhr, textStatus, errorThrown) {
                                console.log(textStatus);
```

```
                                document.getElementById("result4").innerHTML = " "
                                        + textStatus;
                                }
                            });
                    }
//6. Delete Zoneset
                    function deleteZoneset(queryKey, zonesetName, fabricDBID) {
                        debugger;
                        var payload = {};
                        payload['queryKey'] = queryKey;
                        payload['zonesetName'] = zonesetName;
                        payload['fabricDBID'] = fabricDBID;
                        $
                                .ajax({
                                    url : 'https://'
                                            + serverIP
                                            + '/fm/fmrest/WebZoneManager/deleteZonesetsMod',
                                    type : 'POST',
                                    headers : {
                                        "Content-Type" : "application/x-www-form-urlencoded",
                                        "Authorization" : authorization,
                                        "Dcnm-Token" : dcnm_Token
                                    },
                                    data : payload,
                                    success : function(data, textStatus, xhr) {
                                        debugger;
                                        console.log(data);
                                        document.getElementById("result5").innerHTML = data.status
                                                + "- Delete ZoneSet ";
                                    },
                                    error : function(xhr, textStatus, errorThrown) {
                                        console.log(textStatus);
                                        document.getElementById("result5").innerHTML = " "
                                                + textStatus;
                                    }
                                });
                    }

            }
        </script>
    </body>
    <body>
        <br>
        <br>

        <h3>Zone REST calls</h3>
        <table>
            <tr>
                <td>DCNM Server IP:</td>
                <td><input id="ip" type="text" value=""></td>
            </tr>
            <tr>
                <td>User Name:</td>
                <td><input id="un" type="text" value=""></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input id="pw" type="password" value=""></td>
            </tr>
            <tr>
                <td>Fabric Name:</td>
                <td><input id="fabricName" type="text"
                    value=""></td>
            </tr>
```

```
            <tr>
                <td>Switch Name:</td>
                <td><input id="switchName" type="text" value=""></td>
            </tr>
            <tr>
                <td>VSAN ID:</td>
                <td><input id="vsanID" type="text" value=""></td>
            </tr>
            <tr>
                <td>ZoneSet Name:</td>
                <td><input id="zoneSetName" type="text" name="zoneSetName"
                    value="" /></td>
            </tr>
            <tr>
                <td>Zone Name:</td>
                <td><input id="zoneName" type="text" name="zoneName" value="" /></td>
            </tr>
        </table>
        <br>
        <button type="button" onclick="loadDoc()">Submit</button>

    <!-- below divs for displaying status messages -->
        <div id="result"></div>
        <div id="result1"></div>
        <div id="result2"></div>
        <div id="result3"></div>
        <div id="result4"></div>
        <div id="result5"></div>

    </body>
    </html>
```

APPENDIX B

Command Line Storage Dumps Tool

Command Line storage dumps tool is used get Dumps of Storage arrays by passing details through command line arguments. These dumps are further used to analyze for required purpose.

This appendix contains the following sections:

Installing Command Line Storage Dumps Tool

The Command Line Storage Dumps Tool is integrated into DCNM. Also it can be installed as a standalone application. Follow below steps to run the tool in both the ways.

Installing with DCNM

Step 1 After installation of Cisco DCNM, the Command Line Storage Dumps Tool can be found in *<DCNM install dir>/dcnm/smis/client/bin*.

Step 2 Batch files to execute the Command Line Storage Dumps Tool are located:

- On Windows at *<DCNM install dir>/dcm/smis/client/bin/StorageDiscovery.bat*.
- On Linux at *StorageDiscovery.sh*.

Installing as Standalone Tool

Step 1 After installation of Cisco DCNM, the standalone Command Line Storage Dumps Tool can be found in *<DCNM install dir>*/dcm/smis/client/standaloneTools.

Step 2 Zip files to execute the Command Line Storage Dumps Tool are located:

- On Windows at*<DCNM install dir>/dcm/smis/client/standaloneTools/smisclient_standalone.zip.*

Contents of smisclient_standalone.zip can be extracted to any directory and *StorageDiscovery.bat/StorageDiscovery.sh* can be run from the extracted bin directory.

Note Standalone tool requires JAVA_HOME to be set to 1.7 version to execute successfully.

Using Command Line Storage Dumps Tool

To use the Command Line Storage Dumps Tool, enter the following commands:

Windows

<*DCNM install dir*>/dcm/smis/client/bin>**StorageDiscovery.bat http://0.0.0.0:5988/namespace -u** *username* **-p** *password* **-v** *vendorName*

-u: username

-p: password

-v: vendor name(emc or netapp or hds or others)

Linux

<*DCNM install dir*>/dcm/smis/client/bin>**StorageDiscovery.sh http://0.0.0.0:5988/namespace -u** *username* **-p** *password* **-v** *vendorName*

-u: username

-p: password

-v: vendor name(emc or netapp or hds or others)

Once dumps are collected, the dumps are stored under <*DCNM install dir*>*/dcm/smis/client/bin /results* or <*DCNM install dir*>*/dcm/smis/client/bin /results* directory. The related logs can also be found under the results directory.