Intel PROSet For Windows* Device Manager WMI Provider User's Guide

White Paper

Revision 1.6

Table of Contents

Introduction

Intel® PROSet for Windows* Device Manager deploys Network Configuration Services version 2.0, an easy to use solution for deploying and managing all Intel end-station networking technologies using industry standard methods. The NCS2 architecture works closely with the Windows Management Instrumentation (WMI) service to provide remote management of Intel network devices. This document describes the WMI classes and providers supplied by Intel® PROSet for Windows Device Manager.

This document is divided into several sections:

- *Technology overview* – an overview of WMI technology.
- *Class summaries* – the class and namespace details for the NCS2 architecture.
- *Working examples* – how to use the NCS2 architecture to manage Intel® network devices.

Intel® PROSet for Windows Device Manager WMI providers offer the following features:

Category	Features
Adapter	▪ Enumerate all supported physical network adapters ▪ Update settings for an adapter ▪ Obtain an adapter's physical device information ▪ Monitor adapter link ▪ Uninstall an adapter driver ▪ Query IPv4 and IPv6 adapter addresses
Boot	▪ Change an adapter's boot agent settings ▪ View and modify adapter iSCSI settings
Diagnostics	▪ Enumerate diagnostic tests, settings, and results ▪ Run or stop a diagnostic test on an installed adapter
Team	▪ Enumerate supported team types ▪ Create or remove a team of adapters ▪ Update team settings ▪ Add or remove team member adapters ▪ Change team member priorities ▪ Obtain the IPv4 protocol settings for a team
VLAN	▪ Create, discover, or remove Virtual LANs on an adapter or team ▪ Update VLAN settings ▪ Obtain the IPv4 protocol settings for a VLAN

Technology Overview

This section offers an overview of Windows Management Instrumentation in Microsoft operating systems and is recommended for anyone not familiar with the architecture. Further reading on this topic is encouraged and additional are links are provided at the end of this section.

Web-based Enterprise Management

Web-based Enterprise Management (WBEM) is a Distributed Management Task Force (DMTF) initiative providing enterprise system managers with a standardized, cost-effective method for end station management. The WBEM initiative encompasses a multitude of tasks, ranging from simple workstation configuration to full-scale enterprise management across multiple platforms. Central to the initiative is the Common Information Model (CIM), an extensible data model representing objects in typical management environments, and the Managed Object Format (MOF) language for defining and storing modeled data.

Windows Management Instrumentation

Windows Management Instrumentation (WMI) is the Microsoft implementation of WBEM for Windows operating systems. It exposes a programmable interface to view and interact with management objects. Running as a system service, this operating system component offers many powerful capabilities.

WMI consists of the following components:

- Management applications
- Managed objects
- Providers
- Management infrastructure
- A COM API to allow access to management information.

Management applications process or display data from managed objects, which are logical or physical enterprise components. These components are modeled using CIM and accessed by applications through Windows Management. Providers supply Windows Management with data from managed objects, handle requests from applications and notification of events. The providers for Intel® PROSet for Windows Device Manager play a central role in network card configuration management.

Windows management consists of the CIM Object Manager (for handling the communication between management applications and providers) and a central storage area (CIMOM object repository). Data is placed in the repository using either the MOF language compiler or the Windows Management API.

The following diagram shows the interrelationship of these components:

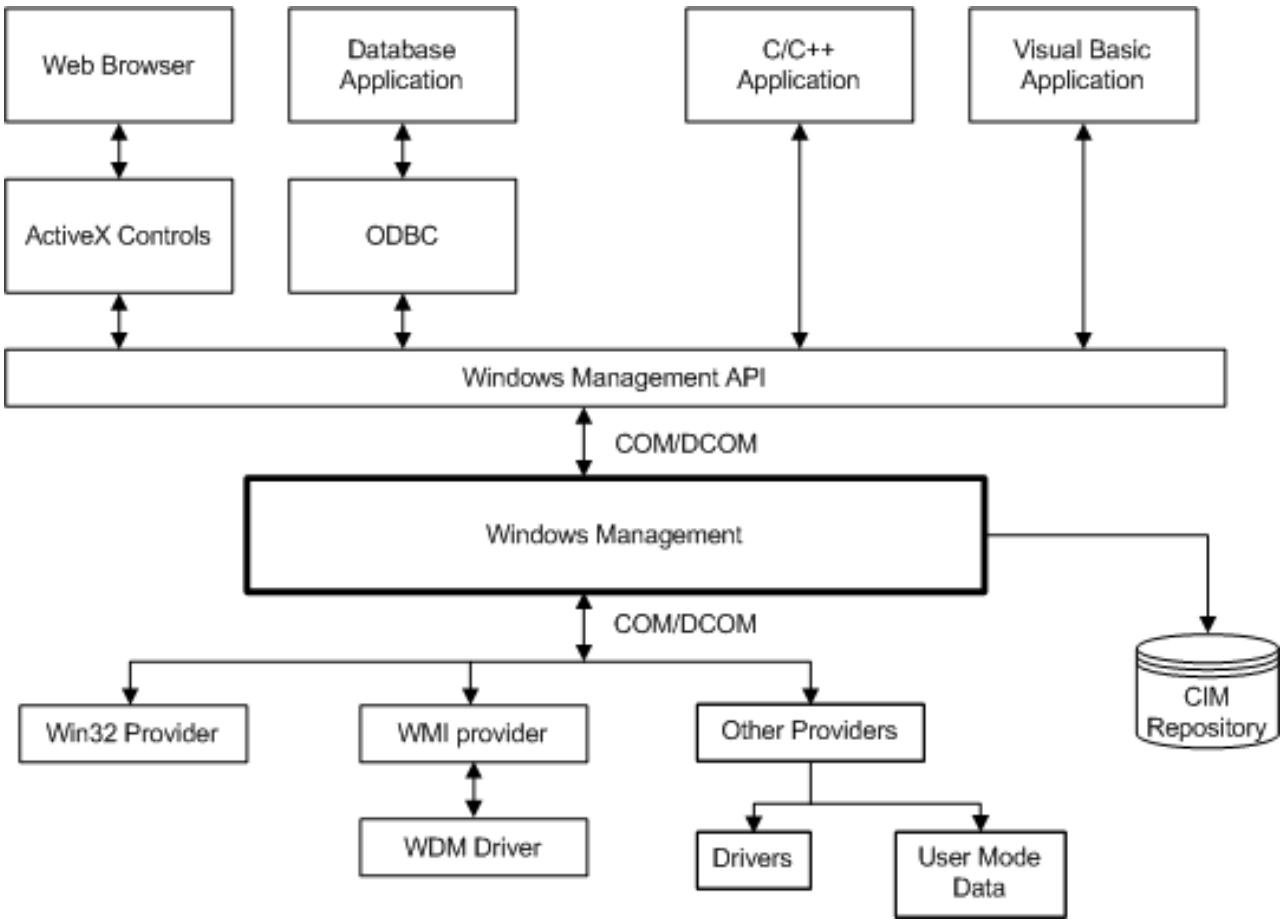


Figure 1 – Windows Management Architecture

Common Information Model

The Common Information Model (CIM) presents a consistent and unified view of all types of logical and physical objects in a managed environment. Managed objects are represented using object-oriented constructs as classes. The classes include properties to describe data and methods. CIM was designed by the DMTF to be operating system and platform independent, but the Microsoft implementation predominates the specification. WBEM technology includes an extension of CIM for Microsoft Windows operating system platforms. Please refer to the DMTF CIM schema on the DMTF web site for more information. Intel® PROSet for Windows Device Manager is based on CIM Schema version 2.6.

CIM defines three levels of classes:

- Classes representing managed objects that apply to all areas of management. These classes provide a basic vocabulary for analyzing and describing managed systems and are part of what is referred to as the "core model."
- Classes representing managed objects that apply to a specific management area but are independent of a particular implementation or technology. These classes are part of what is referred to as the common model - an extension of the core model.
- Classes representing managed objects that are technology-specific additions to the common model. These classes typically apply to specific platforms such as UNIX or the Microsoft Win32 environment.

Inheritance Relationships

Classes can be related by *inheritance*, where a child class includes data and methods from its parent class. Inheritance relationships are not typically visible to the management application using them, nor are the applications required to know the inheritance hierarchy. Class hierarchies can be viewed with CIM repository viewers.

Association Classes

Windows Management also supports *association* classes. Association classes link two different classes to model a user-defined relationship, and are visible to management applications. Third-party developers can also define association classes for their management environment. Associations represent a relationship between two WMI objects (classes). The properties of the association class include two pointers or references, each linking to a different instance. The relationships are maintained by path only; the association class does not have the capabability to modify the instances it links. For additional information on CIM, visit http://www.dmtf.org

CIM Tools

- *Wbemtest.exe* has native support any Windows operating system where WMI has been installed. Examples are included at the end of this document.
- *CIM Studio* is a browser based implementation of WBEMTest.exe and much easier to use. However, it requires download and install an additional program. To locate this tool, search for "WMI Administrative Tools" on Microsoft's web site (www.microsoft.com).

Installed Files

Executables

When information is requested about Intel® PROSet for Windows Device Manager through a WMI service call, one or both of the following executable files will launched. Both of these providers will be referred to as "NCS2 WMI Providers" in this document. There is no need to directly manipulate these files; it is enough to know they exist. Execution and shutdown of these programs is completely transparent to user.

Filename	Description
Ncs2Prov.exe	The instance and method provider for the NCS2 architecture in the "root\IntelNCS2" namespace. This provider will be called "NCS2 WMI Provider" for the remainder of this document.
NCS2Diag.exe	The instance and method provider for the NCS2 architecture in the "root\CIMv2" namespace. This provider will be called "NCS2 CDM Provider" for the remainder of this document.

Dynamically Linked Libraries

The following dynamically linked libraries are used by Intel® PROSet for Windows Device Manager.

Filename	Description
Ncs2Core.dll	Implements the Ethernet Adapter Schema.
Ncs2Diag.dll	Implements the Diagnostics Schema.
Ncs2Boot.dll	Implements the Boot Agent Schema.
Ncs2Team.dll	Implements the Team Schema.
Ncs2VLAN.dll	Implements the VLAN Schema.
Ncs2InstUtility.dll	Implements the common utility functions.

MOF Files

A "MOF" file is a Managed Object Format file which contains information about WMI classes. A set of basic MOF files are included on distribution media for reference only. There are separate MOF files for language neutral and language specific data, which become available upon installation.

The following are .mof files for the 'root\IntelNCS2' namespace:

Filename	Description
ICmLn.mof	CIM base classes on which the NCS2 classes depend.
ICmEnu.mfl	US English version of the CIM base classes.
ICoreLn.mof	Classes for the IEEE 802.3 adapters.
ICoreEnu.mfl	US English textual amendments to the adapter classes.
IBootLn.mof	Classes for the IEEE 802.3 boot service
IBootEnu.mfl	US English textual amendments to the 802.3 boot service classes.
IDiagLn.mof	Classes for the CDM (Common Diagnostic Model).
IDiagEnu.mfl	US English textual amendments to the CDM classes.
ITeamLn.mof	Classes for the IEEE 802.3 teams.
ITeamEnu.mfl	US English textual amendments to the team classes.
IVLANLn.mof	Classes for the IEEE 802.3 VLANs.
IVLANEnu.mfl	US English textual amendments to the VLAN classes.

Security

The NCS2 WMI Providers uses client impersonation to manage the security. Every call will be made in the client's own security context. This context is passed down to the lower layers. An operation may fail if the user does not have suitable administrative rights on the target machine.

Namespaces

CIM classes are organized into namespaces, a logical partitioning of the CIM object management repository. Installation of Intel® PROSet for Windows Device Manager will create a new namespace "root\IntelNCS2" and add information to the existing "root\CIMv2" namespace. The NCS2 architecture uses both namespaces to organize management information and make it available to clients.

IntelNCS2

The root\IntelNCS2 namespace contains the majority of information about Intel® PROSet for Windows Device Manager configuration and is based on CIM version 2.6. The root\CIMv2 namespace was not used as a primary because it is based on CIM version 2.2 and has object key differences. Classes in this namespace have been extended through class inheritance to contain information specific to the NCS2 architecture. All operations regarding adapters*, teams, VLANs, boot agent settings, and diagnostics* must interact with this namespace.

CIMv2

New classes are installed into the root\CIMv2 namespace to support network card diagnostics for legacy applications. The decision to add support for diagnostics in this namespace was made for backward compatibility. Although some classes in the root\CIMv2 namespace have the same nomenclature and properties as their counterparts in the root\IntelNCS2 namespace, any properties not relating to diagnostics have been disabled in the root\CIMv2 versions. These differences are outlined later.

Providers

Each supported namespace has its own provider, which handles requests specific to Intel® PROSet for Windows Device Manager WMI class methods and properties. Although these providers are separate executable files, usage rules and limitations for one apply to the other.

Name	Executable	Description
NCS2 WMI Provider	NCS2Prov.exe	Primary provider for queries to the root\IntelNCS2 namespace.
NCS2 CDM Provider	NCSDiag.exe	Diagnostics provider which interfaces with some classes installed in the root\CIMv2 namespace.

WBEM Context

IWbemContext

IWbemContext is a WMI programming interface which allows users to optionally communicate additional parameters to providers when submitting function calls. If you plan on making any changes to the NCS2 configuration through a WMI call, then you must pass a IWbemContext parameter. These optional parameters are constructed by the user and passed as part of a WbemServices call. Interaction with NCS2 is dependent upon IWbemContext objects when modify operations are requested. Thus, any request to NCS2 for a configuration change requires a IWbemContext object to be constructed by the user and passed in the WbemServices function call. Use of these context qualifiers facilitates exclusive client locks to prevent more

* Adapter and diagnostic information is available in either the root\CIMv2 or root\IntelNCS2 namespaces.

than one change request. A lock is obtained, used, and released whenever a change in NCS2 configuration is required. Data polling operations do not require use of this object. The following table contains the context qualifiers (named values) used by the NCS2 Providers. ClientSetId is only used in conjunction with specific functional areas of the NCS2 WMI Provider, whereas MachineName can be set for all IWbemServices calls. A NULL context can be used for read operations

Context Qualifier	Variant Type	Description
ClientSetId	VT_BSTR	A client handle allows the NCS2 software to manage single access to the configuration. The application cannot make any changes to classes without first establishing this; see the section on the IANet_NetService class to see how to establish and use a client handle.
MachineName	VT_BSTR	The name of the machine that is connecting to the IntelNCS2 provider. This is required for logging.

Use Cases

A session handle is required to change a configuration and is managed through the root\IntelNCS2 namespace classes. This identification number allows the NCS2 software to manage single access to the configuration, thereby preventing changes from more than one source at a time. Understanding the role of these client handles is crucial for successful remote management changes.

Getting a Client Handle

The client must get the object path of the single instance of IANet_NetService before accessing the client handle. Call IWbemServices::CreateInstanceEnum and pass the name of the class: IANet_NetService. (this is equivalent to calling IWbemServices::ExecQuery with the query "SELECT * FROM IANet_NetService). Before making any changes to the configuration, the client must get a client handle. Use the BeginApply method to obtain a numeric handle and lock the software from additional access requests. This lock will remain in place until an apply operation is performed or it times out (usually 2 minutes).

Using a Client Handle in the IWbemContext Object

After the client handle is obtained, a IWbemContext object has to be created. Store the client handle in the ClientSetId qualifier of this object. A pointer to this COM object should be passed to every call into IWbemServices. The client handle is not required when making calls to access the IANet_NetService object as this takes the handle as an explicit argument. By passing the client handle as an argument with the method, the software stack can identify the source of the request.

Finishing with a Client Handle

After changing the configuration, call the IANet_NetService::Apply() method to commit the changes. The client handle integer is passed as an argument to the Apply( ) method. This may return a follow-up action code (e.g., reboot the system before the changes can take effect). If any devices became disabled during change operations, committing an Apply( ) method will enable them.

Locales and Localization

Localized MOF files

All the MOF files used by the NCS2 WMI Provider are localized according to the Microsoft Windows Management Instrumentation globalization model. To accomplish this, each class definition is separated into the following:

- a language-neutral version that contains only the basic class definition in the .mof file.
- a language-specific version that contains localized information, such as property descriptions that are specific to a locale in the corresponding .mfl file.

Class Storage

The language-specific class definitions are stored in a child sub-namespace beneath the namespace that contains a language-neutral basic class definition. For example, for the NCS2 WMI Provider, a child namespace ms_409 will exist beneath the root/IntelNCS2 namespace for the English locale. Similarly, there exists a child sub-namespace for each supported language beneath the root/IntelNCS2 namespace.

Runtime Support

To retrieve localized data, a WMI application can specify the locale using strLocale parameter in SWbemLocator::ConnectServer and IWbemLocator::ConnectServer calls. If the locale is not specified, the default

locale for that system will be used. (e.g. MS_409 for US English). This locale is used to select the correct namespace when adding in the English strings. In addition, IWbemServices::GetObject, SWbemServices.GetObject, IWbemServices:: ExecQuery, and SWbemServices.ExecQuery must specify the WBEM_FLAG_USER_AMENDED_QUALIFIERS flag to request localized data stored in the localized namespace, along with the basic definition. This is required in all functions that produce displayable values using value maps or display descriptions or other amended qualifiers from the MOF files.

Error Reporting

IANet_ExtendedStatus

This section details how to handle errors generated by the NCS2 Providers. How and when an error object is returned depends on whether a call is synchronous, semi-synchronous or asynchronous. In most cases, the HRESULT is set to WBEM_E_FAILED when an error occurs. At this point, however, it is unknown whether WMI or a NCS2 Provider generated the error.

Getting the Error Object

Synchronous Calls

Use GetErrorInfo() to get the IErrorInfo object. Use QueryInterface() to get the IWbemClassObject that contains the error information.

Asynchronous Calls

The IWbemClassObject is passed back as the last item in the last SetStatus() call. After you get the error object instance, you can check the __Class property to determine the origin of the error. WMI creates an instance of __ExtendedStatus, and the NCS2 WMI Provider creates an instance of IANet_ExtendedStatus for errors relating to IANet_ classes and NCS2 WMI Provider. IANet_ExtendedStatus is derived from __ExtendedStatus and contains the following attributes:

Error Object Qualifiers

Context Qualifier	Description
Description	Description of the error tailored to the current locale.
File	Code file where the error was generated.
Line	Line number in the code file with the error.
ParameterInfo	Class or attribute that was being utilized when the error occurred.
Operation	Operation being attempted when the error occurred.
ProviderName	Name of the provider that caused the error.
StatusCode	Code returned from the internal call that failed.

ClientSetHandle	Client Set handle used for the operation.
RuleFailureReasons	Reason for operation failure. An operation can fail because a technical rule has failed. (e.g., you must have a management adapter in certain teams).

Error Codes

For all error codes, the NCS2 WMI Providers gives a description customized to the locale. Below is a list of possible error codes. Error codes are in the form of HRESULT with severity set to one (1) and facility set to ITF. An application may use these codes as a basis for a recovery action.

0x80040901	"WMI: Put property failed"
0x80040902	"WMI: No class object"
0x80040903	"WMI: Failed to create class"
0x80040904	"WMI: Failed to spawn instance of class"
0x80040905	"WMI: Failed to create safe array"
0x80040906	"WMI: Failed to put safe array"
0x80040907	"WMI: Failed to return object to WMI"
0x80040908	"WMI: Get property failed"
0x80040909	"WMI: Unexpected type while getting property"
0x8004090A	"WMI: Class not implemented by this provider"
0x8004090B	"WMI: Unable to parse WQL statement"
0x8004090C	"WMI: Provider only supports WQL"
0x8004090D	"WMI: Parameter in context has the wrong type"
0x8004090E	"WMI: Error formatting debug log"
0x8004090F	"WMI: bad object path"
0x80040910	"WMI: Failed to update setting"
0x80040911	"WMI:[Null parameter passed to method"
0x80040912	"Setting value too small"
0x80040913	"Setting value too big"
0x80040914	"Setting not in step"
0x80040915	"String setting is too long"
0x80040916	"Setting is not one of the allowed values"
0x80040917	"WMI: Qualifier not found"
0x80040918	"WMI: Qualifier set not found"
0x80040919	"WMI: Safe array access failed"
0x8004091A	"WMI: Unhandled exception"
0x8004091B	"WMI: Operation is not supported for this class"
0x8004091C	"WMI: Unexpected event class"
0x8004091D	"WMI: Bad event data"
0x8004091E	"WMI: Operation succeeded with warnings"
0x8004081F	"WMI: The NCS2 Service has been stopped"

Class Summary

The following classes are used by Intel® PROSet for Windows Device Manager.

Namespace : IntelNCS2

IANet_802dot1QVLANService

IANet_AdapterSetting

IANet_AdapterSettingEnum

IANet_AdapterSettingInt

IANet_AdapterSettingMultiSelection

IANet_AdapterSettingMultiString

IANet_AdapterSettingSlider

IANet_AdapterSettingString

IANet_AdapterToSettingAssoc [A]

IANet_BootAgent

IANet_BootAgent_iSCSI_Adapters

IANet_BootAgentSetting

IANet_BootAgentSettingEnum

IANet_BootAgentSettingInt

IANet_BootAgentSettingString

IANet_BootAgentToBootAgentSettingAssoc [A]

IANet_Device802dot1QVLANServiceImplementation [A]

IANet_DeviceBootServiceImplementation [A]

IANet_DiagResult

IANet_DiagResultForMSE [A]

IANet_DiagResultForTest [A]

IANet_DiagSetting

IANet_DiagSettingForTest [A]

IANet_DiagTest

IANet_DiagTestForMSE [A]

IANet_EthernetAdapter

IANet_ExtendedStatus

IANet_LogicalEthernetAdapter

IANet_NetService

IANet_NetworkVirtualAdapter

IANet_PhysicalEthernetAdapter

IANet_Setting

IANet_TeamedMemberAdapter

IANet_TeamOfAdapters

IANet_TeamSetting

IANet_TeamSettingEnum

IANet_TeamSettingInt

IANet_TeamSettingMultiSelection

IANet_TeamSettingSlider

IANet_TeamSettingString

IANet_TeamToTeamSettingAssoc [A]

IANet_VLAN

IANet_VLANFor [A]

IANet_VLANSetting

IANet_VLANSettingEnum

IANet_VLANSettingInt

IANet_VLANSettingMultiSelection

IANet_VLANSettingSlider

IANet_VLANSettingString

IANet_VLANToVLANSettingAssoc [A]

Namespace : CIMv2

IANet_DiagResult

IANet_DiagResultForMSE [A]

IANet_DiagResultForTest [A]

IANet_DiagSetting

IANet_DiagSettingForTest [A]

IANet_DiagTest

IANet_DiagTestForMSE [A]

IANet_EthernetAdapter

IANet_PhysicalEthernetAdapter

[A] – Association class.

IntelNCS2 Class Definitions

The following section contains information for all classes in the root\IntelNCS2 namespace supported by Intel® PROSet for Windows Device Manager.

Graphic Conventions

These conventions will describe relationships between classes defined in this namespace and those inherited from CIM specification classes.

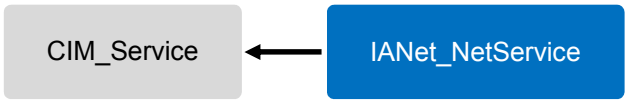
Denotes a class defined in the CIM specification. The properties of these parent classes are inherited by child class definitions.

Represents a class defined in the NCS2 architecture, specific to Intel PROSet for Windows Device Manager.

Represents an *association* class defined in the NCS2 architecture, specific to Intel PROSet for Windows Device Manager.

This arrow indicates inheritance between one class and another.

IANet_NetService



Purpose

This class enables the client to establish active sessions where changes can be made to the configuration. When requesting or applying a client lock handle, this class must be used : it exposes two methods for performing these operations.

Instances

There is one instance of this object. The client should not rely on the key used for this class. Instead, the client should get the instance of the class by enumerating all instances of IANet_NetService. The user cannot create or delete instances of this class.

Supported Properties

Version - Contains the current version of the core provider.

Unsupported Properties

The following properties not supported: Caption, Description, Install Date, Started, Start Mode, Status.

Modifiable Properties

There are no user modifiable properties of this class.

Supported Methods

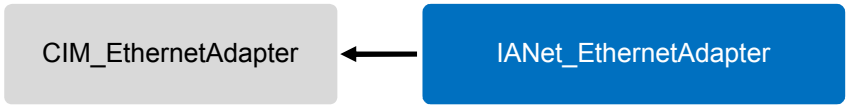
Method	Returns	Parameters	Detail
BeginApply	void	[OUT] uint32 ClientSetHandle	Used to get a Client session handle , which should be placed in the context object in the ClientSetId qualifier.
Apply	void	[IN] uint32 ClientSetHandle [OUT] uint32 FollowupAction	Applies changes made with a particular session handle and releases the session handle after it has been used. The uint32 argument returned is used by the provider to tell the application the server must be rebooted before the changes will take effect. FollowupAction 1 (system reboot required) 0 (no reboot required)

Unsupported Methods

The following methods are not supported: StartService and StopService.

Adapter Classes

IANet_EthernetAdapter



Purpose

This is an abstract base class which objectifies network characteristics of an Intel network card. The IANet_EthernetAdapter class is inherited by IANet_LogicalEthernetAdapter and contains properties common to both virtual and physical network devices. If you need information on teaming classes, reference IANet_LogicalEthernetAdapter.

Associations

Association Class	Association Partner
IANet_Device802dot1QVLANServiceImplementation	IANet_802dot1QVLANService

IANet_PhysicalEthernetAdapter

Purpose

IANet_PhysicalEthernetAdapter defines the capabilities and status of all the installed Intel adapters.

Instances

Instances of this class will exist for all installed network adapters. Non-Intel network cards will be represented by an instance of this class, although only a subset of the supported properties will have values. Such adapters do not support some properties specific to Intel network drivers. The user cannot create instances of IANet_PhysicalEthernetAdapter. Deleting an instance of IANet_PhysicalEthernetAdapter will uninstall a physical adapter; a client handle is required for this operation.

Supported Properties

Name	Type	Description	Values	
AdapterStatus	uint32	Adapter status specifies the current status of the adapter. This value is the sum of any of the values which apply. Example: 51 = 1 + 2 + 16 + 32	1 2 4 16 32 1024 2048	Installed DriverLoaded HardwareMissing HasDiag HasLink HasTCOEnabled DeviceError
AdditionalAvailability	uint16[ ]	This is an inherited property; refer to parent class.		

Availability	uint16	This is an inherited property; refer to parent class.		
BusType	uint16	Bus Type indicates the bus type.	0	Unknown
			1	ISA
			2	EISA
			3	PCMCIA
			4	Cardbus
			5	PCI
			6	PCI-X
			7	PCI Express
Capabilities	uint16[ ]	Capabilities of the PhysicalEthernetAdapter. For example, the Device may support AlertOnLan, WakeOnLan, Load Balancing and/or FailOver. If failover or load balancing capabilities are listed, a SpareGroup (failover) or ExtraCapacityGroup (load balancing) should also be defined to completely describe the capability. Some capabilities are dependent upon feature discovery in the operating system. Therefore, a capability may not be present because operating system requirements have not been met.	0	Unknown
			1	Other
			2	AlertOnLan
			3	WakeOnLan
			4	Adapter Fault Tolerance
			5	Adaptive Load Balancing
			6	IPSec Offload
			7	ASF
			8	GEC/802.3ad Static Link Aggregation
			9	Static Link Aggregation
			10	IEEE 802.3ad Dynamic Link Aggregation
			11	Checksum Offload
			12	Switch Fault Tolerance
			13	Basic AlertOnLan
			14	AlertOnLan 2
			15	Security Offload AH
			16	Security Offload ESP
			17	Security Payload Tunnel
			18	Security Payload Transport
			19	Security IPV4 Packets
			20	Authentication Algorithm MD5
			21	Authentication Algorithm SHA1
			22	Encryption Algorithm EAS
			23	Encryption Algorithm DES
			24	Encryption Algorithm 3DES
			25	ESP Xmit Checksum Encryption
			26	ESP Xmit Checksum Authentication
			27	ESP Receive Checksum Encryption
			28	ESP Receive Checksum Authentication
			29	TCO Capability
			30	Wake Up Capabilities
			31	IP Checksum Offload

			32	10 Mbps
			33	100 Mbps
			34	1000 Mbps
			35	10000 Mbps
			36	Teaming
			37	VLAN
			38	IEEE VLAN
			39	ISL VLAN
			40	Uninstallable
			41	Identify Adapter Support
			42	Cable Test Support
			43	Diagnostic Support
			44	Flash support
			45	ICH Support
			46	Usage Server
			47	Vendor Intel
			48	Phoneline PHY
			49	Mobile
			50	PowerManagement Support
			51	Feature Not Supported
			52	MFO
			53	Pass Through
			54	Quad-Port Support
			55	Dedicated MAC Address
			56	Jumbo Frame Support
			57	Feature Not Supported
			58	Signal Quality Test
			59	Cable Offline Test
			60	Adapter is LOM
			61	Scalable Networking Pack Capability
			62	CB Platform Capability
			63	iSCSI Capability
CapabilityDescriptions	string[ ]	This property is deprecated and is not in use.		
Caption	string	This is an inherited property; refer to parent class		
ControllerID	uint32	The Controller ID identifies the Ethernet controller that the adapter uses. Adapters with different DeviceIDs can have the same Controller ID.	0	Unknown
			1	Intel 82542
			3	Intel 82543
			6	Intel 82544 Controller
			7	Intel 82540 Controller
			8	Intel 82545 Controller
			11	Intel 82541 Controller
			13	Intel 82547 Controller
			20	Intel 82571 Controller
			30	Intel 82573 Controller
			31	Intel 82574 Controller
			40	Intel ESB2LAN Controller

			50 Intel ICH8 Controller 51 Intel ICH9 Controller 52 Intel ICH10 Controller 60 Intel 82575 Controller 62 Intel 82576 Controller 63 Intel ADORAM_VIRTUAL Controller 65537 Intel D100_A_STEP Controller 65538 Intel D100_B_STEP Controller 65539 Intel D100_C_STEP Controller 65540 Intel D101_A_STEP Controller 65541 Intel D101_B0_STEP Controller 65542 Intel D101M_A_STEP Controller 65543 Intel D101S_A_STEP Controller 65544 Intel D102_A_STEP Controller 65545 Intel D102_B_STEP Controller Controller 65546 Intel D102_C_STEP Controller Controller 65547 Intel D102_D_STEP Controller Controller 65548 Intel D102_E_STEP Controller Controller 65549 Intel D102_F_STEP Controller 65550 Intel 82562_G Controller 65551 Intel 82562_GZ Controller 65552 Intel 82562_GX_GT Controller 65553 Intel 82562 Controller 131073 Intel 82597 EX Controller 196609 Intel 82598 Controller 196610 Intel 82599 Controller
Description	string	This is an inherited property; refer to parent class.	
DeviceID	string	This is an inherited property; refer to parent class.	
EEPROMVersion	string	EEPROMVersion contains the EEPROM version of the device.	
EnabledCapabilities	uint16[ ]	Specifies which capabilities are enabled from the list of all supported ones, defined in the	Please refer to the .Capabilities property definition (above) to resolve

		Capabilities array	uint16 values to strings.
HardwareStatus	uint32	Hardware status specifies the current status of the hardware.	0 Unknown 1 Ready 2 Initializing 3 Reset 4 Closing 5 Not Ready
MaxSpeed	uint16	This is an inherited property; refer to parent class.	
MediaType	uint16	MediaType indicates the media which interfaces to this PHY.	0 Unknown 1 Copper 2 Fiber 3 Phone Line 4 CX4 Copper 5 Other
MiniPortInstance	string	This is an inherited property; refer to parent class.	
MiniPortName	string	This is an inherited property; refer to parent class.	
Name	string	This is an inherited property; refer to parent class.	
NegotiatedLinkWidth	uint16	Negotiated Link Width specifies the negotiated link width of the bus. Only PCI-Express adapters will have a non zero value.	0 Unknown 1 x1 2 x2 4 x4
NetworkAddresses	string[ ]	This is an inherited property; refer to parent class.	
OriginalDisplayName	string	If teaming is enabled on this adapter OriginalDisplayName will contain the original display name of the adapter.	
OtherCapabilityDescriptions	string[ ]	This property is deprecated and is not in use.	
OtherEnabledCapabilities	string[ ]	This property is deprecated and is not in use.	
OtherEnabledCapabilityIDs	uint16[ ]	This property is deprecated and is not in use.	
OtherMediaType	string	This property is deprecated and is not in use.	
OtherPhyDevice	string	This property is deprecated and is not in use.	
PartNumber	string	PartNumber is the NIC's PBA manufacturing part number.	
PCIDeviceID	string	PCI device Id of the device.	
PermanentAddress	string	This is an inherited property; refer to parent class.	
PHYDevice	uint16	PHYDevice indicates the particular PHY used on this NIC.	0 No PHY detected 1 Intel 82553 (PHY 100) A or B step 2 Intel 82553 (PHY 100) C step 3 Intel 82503 10Mps 4 National DP83840A (10BaseT and 100Base-TX) 5 Seeq 80C240 - 100BASE-T4 6 Seeq 80C24 - 10Mps 7 Intel 82555 100Base-TX PHY 8 Microlinear 10Mps 9 Level One 10Mbps 10 National DP83840 100Base-TX, C step

			11	ICS 100Base-TX PHY
			12	Gilad
			13	Kinnereth
			14	Kinnereth Plus
			15	Other
			16	Unknown
			50	Intel 82562 EH Phoneline PLC
			60	Intel 82562 ET 100 Base-TX PHY
			70	Intel 82562 EM 100 Base-TX PHY
PortNumber	uint16	PortNumber indicates the port number on PCIe Quad port adapters	0	A
			1	B
			2	C
			3	D
SlotID	string	SlotID field of the System Slot structure provides a mechanism to correlate the physical attributes of the slot to its logical access method.		
Speed	uint64	This is an inherited property; refer to parent class.		
StaticIPAddress	string	StaticIPAddress shows the static IP address if Static IP Address is configured, else this is set to 0.0.0.0.		
Status	string	This is an inherited property; refer to parent class.		
StatusInfo	uint16	This is an inherited property; refer to parent class.		
SubnetMask	string	SubnetMask shows the current adapter's subnet mask. This field is populated only if the adapter has a static IP Address configured or else this is set to 0.0.0.0.		

Unsupported Properties

The following properties are not supported: AlignmentErrors, AutoSense, CarrierSenseErrors, DeferredTransmissions, ErrorCleared, ErrorDescription, ExcessiveCollisions, FCSErrors, FlowControlPacketsReceived, FlowControlPacketsTransmitted, FrameTooLongs, FullDuplex, GeneralReceiveErrors, GeneralTransmitErrors, IdentifyingDescriptions, InstallDate, InternalMACReceiveErrors, InternalMACTransmitErrors, LastErrorCode, LateCollisions, MaxDataSize, MaxQuiesceTime, MultipleCollisionFrames, OctetsReceived, OctetsTransmitted, OtherIdentifyingInfo, PowerManagementCapabilities (this is exposed as a method), PowerManagementSupported (this is exposed as a method), PowerOnHours, SingleCollisionFrames, SymbolErrors, TotalPacketsReceived, TotalPacketsTransmitted, TotalPowerOnHours.

Modifiable Properties

There are no user modifiable properties of this class.

Supported Methods

Method	Returns	Parameters	Detail
GetNDISVersion	uint32	[OUT] uint32 dwMajorVersion [OUT] uint32 dwMinorVersion	This method can be used to get the NDIS version
GetPowerUsageOptions	uint32	[OUT] uint32 AutoPowerSaveEnabled [OUT] uint32 ReduceSpeedOnPowerDown [OUT] uint32 SmartPowerDown [OUT] uint32 SavePowerNowEnabled	Detects any optional power usage settings (e.g., power usage for standby, battery operation, etc.). 0 = Off 1 = On

		[OUT] uint32 EnhancedASPMPowerSaver [OUT] uint32 ACBSMode [OUT] uint32 LinkSpeedBatterySaver	
GetWakeOnLanPowerOptions	uint32	[IN] uint32 WakeFromPoweroff [IN] uint32 WakeOnLink [IN] uint32 WakeOnMagicPacket [IN] uint32 WakeOnDirectedPacket	GetWakeOnLanPowerOptions returns WakeOnLan power settings. For example, information about wakeonlink, wakeonmagicpacket etc.. 0 = Off 1 = On If an adapter does not support this feature, the returned structure will be empty.
IdentifyAdapter	uint32	[IN] uint16 nSeconds	Identifies adapter by flashing the light on the adapter for a few seconds. This method will only work for physical adapters.
IsISCSIEnabled	uint32	[OUT] uint32 iSCSIStatus	This method can be used to check if iSCSI is enabled on that adapter. iSCSIStatus 0 – Unavailable 1 – Disabled 2 – Primary 3 - Secondary
IsiSCSISupported	uint32	[OUT] boolean bIsiSCSIOS [OUT] boolean bIsISCSIPatch [OUT] boolean bIsISCSIHotFix	This method can be used to check if iSCSI is supported by the OS and iSCSI patch and hot fix are installed. The "hot fix" is also known as the Microsoft iSCSI intitiator.
IsSetPowerMgmtCapabilitiesReq	uint32	[OUT] boolean bIsSetRequired	This method can be used to check if SetPowerMgmt-Capabilities() needs to be called.
SetPowerMgmtCapabilities	uint32	This method is used to makes changes to the Power management capabilities during NCS2 install so that any upgrade scenarios from earlier releases will have the right options for all the WakeOnLAN options and NCS2 will not have reinterpret them dynamically.	
SetPowerUsageOptions	uint32	[IN] uint32 AutoPowerSaveModeEnabled [IN] uint32 ReduceSpeedOnPowerDown [IN] uint32 SmartPowerDown [IN] uint32 SavePowerNowEnabled [IN] uint32 EnchancedASPM-PowerSaver [IN] uint32 ACBSMode [IN] uint32 LinkBatterySaver	Changes power usage options (e.g., method can be used to reduce power usage for standby, battery operation, etc.) Note: Power usage settings are stored and used for subsequent reboots. 0 = Off 1 = On
SetWakeOnLanPowerOptions	uint32	[IN] uint32 WakeFromPoweroff [IN] uint32 WakeOnLink [IN] uint32 WakeOnMagicPacket [IN] uint32 WakeOnDirected-	This method can be used to makes changes to the WakeOnLan options. For example, this method could be used to set options like wakefromPoweroff, wakeOnlink,

	Packet	WakeOn-MagicPacket, WakeOn-DirectedPacket etc. Note WakeOnLan settings are stored and used for every boot. 0 = Off 1 = On
ValidateSettingOnNewTeam	Internal use only.	

Unsupported Methods

The following methods are not required for Intel PROSet and are, therefore, not supported:

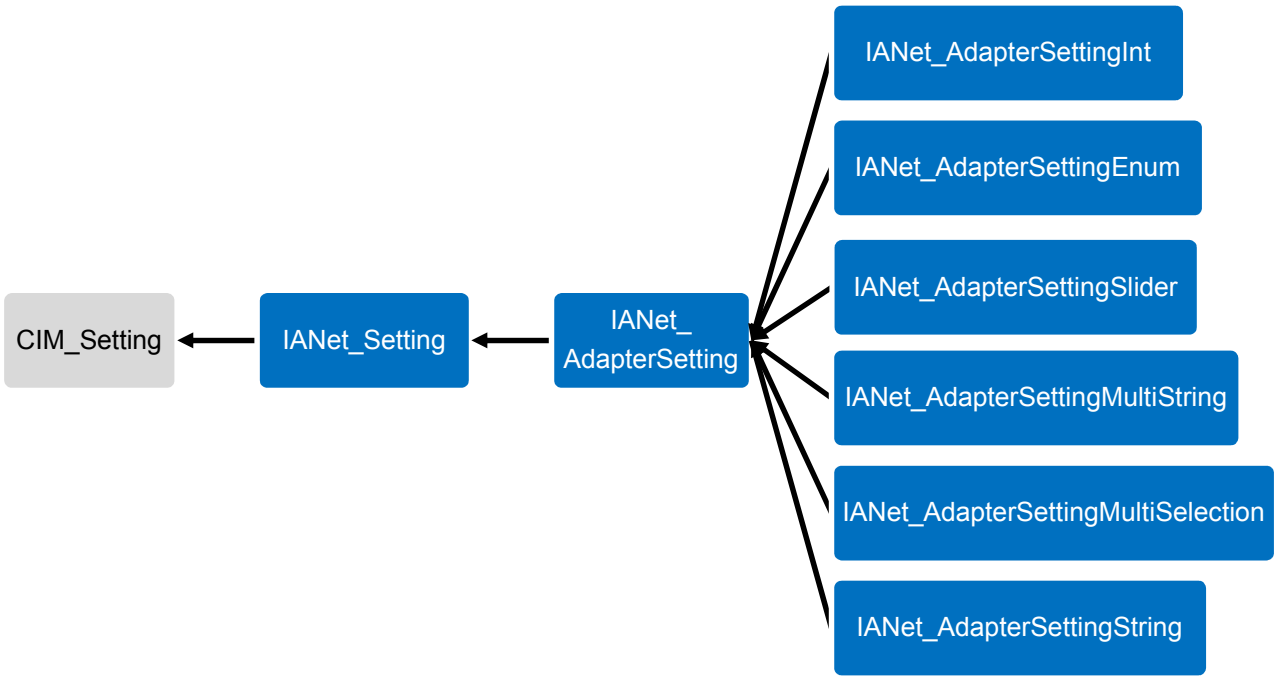
EnableDevice, OnlineDevice, QuiesceDevice, Reset, RestoreProperties, SaveProperties, SetPowerState.

Associations

Association Class	Association Partner
IANet_Device802dot1QVLANServiceImplementation	IANet_802dot1QVLANService
IANet_DiagTestForMSE	IANet_DiagTest
IANet_DiagResultForMSE	IANet_DiagResult
IANet_DeviceBootServiceImplementation	IANet_BootAgent
IANet_AdapterToSettingAssoc	IANet_AdapterSetting
IANet_TeamedMemberAdapter	IANet_TeamOfAdapters

Adapter Settings

All adapter settings are derived from a common base class.



IANet_Setting

Purpose

This is an abstract super class for a set of concrete classes of different types. This set of classes allows open ended usage of a variable number of settings. These will be different between adapters, teams, or VLANs and it may not always be possible to predict what parameters are required. Between the setting categories, this class groups the most common parameters for inheritance.

IANet_AdapterSetting

Purpose

This abstract class is used to describe a settable property in a configuration. The class is derived from IANet_Setting. Instances of this class will exist for each setting on each adapter. There are several sub-classes for IANet_AdapterSetting. The sub-classes correspond to the different types and ranges of values that settings can take. Each sub-class corresponds to a different style of GUI that may be used to display or change the settings.

Supported Common Properties

Each of the five IANet_AdapterSetting-[String, Enum, Slider, MultiSelection, Int, MultiString] classes support a similar set of properties. To reduce reproduction of the same information, the common properties are listed here:

Name	Type	Description
Caption	string	This is an inherited property; refer to parent class.
CurrentValue	sint64	Actual value of the parameter – tihs is the only attribute at the user can change.

DefaultValue	sint64	The initial value of the parameter.
Description	string	This is an inherited property; refer to parent class.
ExposeLevel	uint32	This is an inherited property; refer to parent class.
Grouped	boolean	This is an inherited property; refer to parent class.
GroupId	uint16	This is an inherited property; refer to parent class.
MiniHelp	string	This is an inherited property; refer to parent class.
ParentId	string	This is an inherited property; refer to parent class.
ParentType	string	This is an inherited property; refer to parent class.
Writable	boolean	This is an inherited property; refer to parent class.

Common Methods

No methods are supported by the IANet_AdapterSetting-[String, Enum, Slider, MultiSelection, Int, MultiString] classes.

Associations

Association Class	Association Partner
IANet_AdapterToSettingAssoc	IANet_PhysicalEthernetAdapter

IANet_AdapterSettingInt

Purpose

The class models a setting that takes an integer value. There are several IANet setting classes used to model integers. The differences between these classes concerns how the integer is displayed and modified by the GUI, and how validation is done by the NCS2 WMI Provider. For IANet_AdapterSettingInt, it is expected that the GUI will display an edit box with a spin control.

Instances

An instance of this class exists for each setting that should be displayed as an integer edit box. Users can neither create or remove instances.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
base	uint64	Base is the root from which an integer value may take values.
max	sint64	The maximum value the integer can take.
min	sint64	The minimum value the integer can take.
Scale	sint64	The unit of measurement to set or estimate series of marks or points at known intervals to measure value of the parameter.
step	sint64	Granularity of the integer value.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. The user can modify this property by using IWbemClassObject::Put() to change the value, then call "IWbemServices::PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue <= max, CurrentValue >= min, (CurrentValue – min) is a multiple of step.

Associations

Association Class	Association Partner
IANet_AdapterToSettingAssoc	IANet_PhysicalEthernetAdapter

IANet_AdapterSettingEnum

Purpose

The class models a enumeration setting value. For IANet_AdapterSettingEnum, it is expected that the GUI will display a list of strings which map onto a small number of enumerated values. (e.g., a drop list combo box).

Instances

An instance of this class exists for each setting that will be displayed as an enumeration.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
DescriptionMap	[ ] string	Contains what each value means
PossibleValues	[ ] sint64	An array of possible values allowed for the enum.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue ∈ PossibleValues[]

Associations

Association Class	Association Partner
IANet_AdapterToSettingAssoc	IANet_PhysicalEthernetAdapter

IANet_AdapterSettingSlider

Purpose

The class models a setting that specifically handles Slider settings. For IANet_AdapterSettingSlider, it is expected that the user interface will display a slider which will allow the user to choose the value in a graphical manner – the actual value chosen need not be displayed.

Instances

An instance of this class exists for each setting that will be displayed as a slider. Users can neither create or remove instances.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
FirstLabel	string	The label that should be displayed on the left side of the slider.
LastLabel	string	The label that should be displayed on the right side of the slider.
PossibleValues	[ ] sint64	The initial value of the parameter.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue $\in$ PossibleValues[]

Associations

Association Class	Association Partner
IANet_AdapterToSettingAssoc	IANet_PhysicalEthernetAdapter

IANet_AdapterSettingMultiString

Purpose

The class objectifies adapter related driver and network device settings; specifically, it handles multi-string settings.

Instances

An instance of this class exists for each setting that will be as a list of string values. Users can neither create or remove instances.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
Maxlength	uint32	The maximum length of the string.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue $\in$ PossibleValues[]

Associations

Association Class	Association Partner
IANet_AdapterToSettingAssoc	IANet_PhysicalEthernetAdapter

IANet_AdapterSettingMultiSelection

Purpose

This class models a setting whereby the user can select several options from a list of options. For IANet_AdapterSettingMultiSelection, it is expected that the GUI will display multi-selection list box which will allow the user to choose any (or no) option(s).

Instances

An instance of this class exists for each setting that should be displayed as a list of options.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
FirstLabel	string	The label that should be displayed on the left side of the slider.
LastLabel	string	The label that should be displayed on the right side of the slider.
PossibleValues	[ ] sint64	The initial value of the parameter.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then use "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue $\in$ PossibleValues[]

Associations

Association Class	Association Partner
IANet_AdapterToSettingAssoc	IANet_PhysicalEthernetAdapter

IANet_AdapterSettingString

Purpose

This class models a setting whereby the user can enter a free-form string value. For IANet_AdapterSettingString, it is expected that the user interface will display an edit box.

Instances

An instance of this class exists for each setting that should be displayed as a string.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
MaxLength	uint32	The maximum length of the string.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

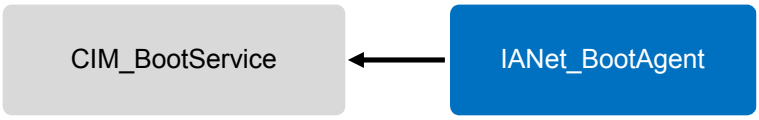
The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting.

Associations

Association Class	Association Partner
IANet_AdapterToSettingAssoc	IANet_PhysicalEthernetAdapter

Boot Agent Classes

IANet_BootAgent



Purpose

This class is used to capture information about the network boot capabilities of an adapter (e.g., settings for the PXE Boot Agent supported by some Intel adapters).

Instances

An IANet_BootAgent instance exists for each adapter that supports boot agent capabilities, even if the boot agent is not currently installed. Users can neither create or remove instances.

Supported Properties

Name	Type	Description	Values	
FlashImageType	uint32	Boot Agent Flash Image type.	0	PXE
			1	PXE_EFI
			3	EFI
			4	DISABLED
			5	BLANK
			6	MISSING
			7	iSCSI
			255	Unknown
InstalledFlashImageTypes	uint32	Boot Agent flash image types that are currently installed in the ROM.	1	PXE
			2	EFI
			4	ISCSI
			255	Unknown
InvalidImageSignature	boolean	Will be set to true if the boot agent has a corrupted flash image.		
iSCSI_Status	uint32	Boot Agent iSCSI status.	0	iSCSI_PRIMARY
			1	iSCSI_SECONDARY
			2	iSCSI_DISABLED
			255	Unknown
UpdateAvailable	boolean	Indicates if install or upgrade to boot agent software is available.		
Version	string	String describing boot agent version.		
VersionNumber	uint32	Boot agent version in the format x.x.x		

Unsupported Properties

The following properties are not required by Intel PROSet and are, therefore, not supported:

Caption, Description, InstallDate, Started, StartMode, Status.

Modifiable Properties

There are no user modifiable properties of this class.

Methods

There are two methods on this class that can be used to update the Flash ROM on the NIC:

Method	Returns	Parameters	Detail
ProgramFlash	uint32	[IN] uint32 Action [IN] array of uint8 NewFlashData [OUT] uint32 FlashRetCode	This method is used to update the Flash ROM on the NIC. This will cause the NIC to stop communicating with the network while the flash is updated.
ReadFlash	uint32	[OUT] array of uint8 FlashData	This method reads the Flash ROM on the NIC.

Unsupported Methods

StartService, StopService are not supported.

Associations

Association Class	Association Partner
IANet_BootAgentToBootAgentSettingAssoc	IANet_BootAgentSetting
IANet_DeviceBootServiceImplementation	IANet_PhysicalEthernetAdapter

IANet_BootAgent_iSCSI_Adapters

Purpose

This class is used to capture information about iSCSI supported adapters installed in the system.

Instances

There will be one instance of each adapter which supports iSCSI boot. Users can neither create or remove instances.

Supported Properties

Name	Type	Description	Values	
AdapterName	string	Friendly name of the adapter.		
Caption	This is an inherited property; refer to parent class.			
iSCSI_Status	uint32	The boot agent iSCSI status.	0	iSCSI_PRIMARY
			1	iSCSI_SECONDARY
			2	iSCSI_DISABLED
			255	Unknown
Name	This is an inherited property; refer to parent class.			

Unsupported Properties

The following properties are not required by Intel PROSet and are, therefore, not supported:

Description, InstallDate, Started, StartMode, Status

Modifiable Properties

There are no user modifiable properties of this class.

Methods

There is one method of this class which can be used to set the iSCSI priority of adapters:

Method	Returns	Parameters	Detail
SetiSCSI_Status	uint32	[IN] uint32 iSCSI_State [OUT] uint32 RetCode	This method will update the status of adapters that support iSCSI Boot. The function only takes the primary and secondary adapter IDs and sets them accordingly. The remaining adapters are set to disabled. iSCSI_State 0 – Set adapter to Primary 1 – Set adapter to Secondary 2 – Set adapter to Disabled RetCode 0 – The state change was successful 1 – The state change failed

Unsupported Methods

StartService, StopService are not supported.

IANet_BootAgentSetting

Purpose

This abstract class is used to describe a settable property in a configuration. The class is derived from IANet_Setting. Instances will exist for each Boot Agent setting. There are several sub-classes for IANet_BootAgentSetting which correspond to the different types and ranges of values that settings can take.

Supported Common Properties

Each of the IANet_BootAgentSetting-[String, Enum, Int] classes support a similar set of properties. To reduce reproduction of the same information, the common properties are listed here:

Name	Type	Description
Caption	string	This is an inherited property; refer to parent class.
CurrentValue	sint64	Actual value of the parameter – tihs is the only attribute at the user can change.
DefaultValue	sint64	The initial value of the parameter.
Description	string	This is an inherited property; refer to parent class.
ExposeLevel	uint32	This is an inherited property; refer to parent class.
Grouped	boolean	This is an inherited property; refer to parent class.
GroupId	uint16	This is an inherited property; refer to parent class.
MiniHelp	string	This is an inherited property; refer to parent class.
ParentId	string	This is an inherited property; refer to parent class.
ParentType	string	This is an inherited property; refer to parent class.
Writable	boolean	This is an inherited property; refer to parent class.

Common Methods

No methods are supported by the IANet_BootAgentSetting-[String, Enum, Int] classes.

Associations

Association Class	Association Partner
IANet_BootAgentToBootAgentSettingAssoc	IANet_BootAgent

IANet_BootAgentSettingEnum

Purpose

The class models a enumeration setting value.

Instances

An instance of this class exists for each setting that will be displayed as an enumeration.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
DescriptionMap	[ ] string	Contains what each value means.
PossibleValues	[ ] sint64	An array of possible values allowed for the Enum.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue $\in$ PossibleValues[]

Associations

Association Class	Association Partner
IANet_BootAgentToBootAgentSettingAssoc	IANet_BootAgent

IANet_BootAgentSettingInt

Purpose

This class objectifies Boot Agent related driver and network device settings. IANet_BootAgentSettingInt specifically handles Integer settings.

Instances

An instance of this class exists for each setting that should be displayed as an integer edit box. Users can neither create or remove instances.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
base	uint64	Base is the root from which an integer value may take values.
max	sint64	The maximum value the integer can take.
min	sint64	The minimum value the integer can take.
scale	sint64	The unit of measurement to set or estimate series of marks or points at known intervals to measure value of the parameter.
step	sint64	Granularity of the integer value.

Unsupported Properties

SettingID and RequiresSession are not used.

Associations

Association Class	Association Partner
IANet_BootAgentToBootAgentSettingAssoc	IANet_BootAgent

IANet_BootAgentSettingString

Purpose

This class objectifies Boot Agent related driver and network device settings. IANet_BootAgentSettingString specifically handles Integer settings

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
MaxLength	uint32	The maximum length of the string.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting.

Associations

Association Class	Association Partner
IANet_BootAgentToBootAgentSettingAssoc	IANet_BootAgent

Team Classes

IANet_LogicalEthernetAdapter

Purpose

This class objectifies the general network characteristics of an Intel ANS team portrayed as a logical device. For every team instance there will be one instance of this class. This class implements CIM_EthernetAdapter for a virtual team interface.

Supported Properties

Name	Type	Description
Caption	string	This is an inherited property; refer to parent class.
Description	string	This is an inherited property; refer to parent class.
DeviceID	string	This is an inherited property; refer to parent class.
MiniPortInstance	string	This is an inherited property; refer to parent class.
MiniPortName	string	This is an inherited property; refer to parent class.
Name	string	This is an inherited property; refer to parent class.
StatusInfo	string	This is an inherited property; refer to parent class.

Unsupported Properties

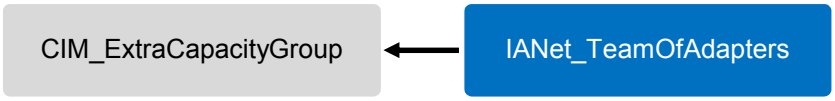
The following properties are not required by Intel PROSet and are, therefore, not supported:

AdditionalAvailability, AlignmentErrors, AutoSense, Availability, Capabilities, CapabilityDescriptions, CarrierSenseErrors, DeferredTransmissions, EnabledCapabilities, ErrorCleared, ErrorDescription, ExcessiveCollisions, FCSErrors, FrameTooLongs, FullDuplex, IdentifyingDescriptions, InstallDate, InternalMACReceiveErrors, InternalMACTransmitErrors, LastErrorCode, LateCollisions, MaxDataSize, MaxQuiesceTime, MaxSpeed, MultipleCollisionFrames, NetworkAddresses, OctetsReceived, OctetsTransmitted, OtherIdentifyingInfo, PowerManagementCapabilities, PowerManagementSupported, PowerOnHours, SingleCollisionFrames, SymbolErrors, TotalPacketsReceived, TotalPacketsTransmitted, TotalPowerOnHours.

Associations

Association Class	Association Partner
IANet_NetworkVirtualAdapter	IANet_TeamOfAdapters
IANet_TeamToTeamSettingAssoc	IANet_TeamSetting

IANet_TeamOfAdapters



Purpose

This class has members that describe the type of the team, the number of adapters in the team, and the maximum number of adapters that can be in the team.

Instances

There is an instance of this class for each Intel adapter team. To create an empty team, the user will create an instance of IANet_TeamOfAdapters. The user must set the correct "TeamingMode" before calling IWbemServices::PutInstance() to create the object. The NCS2 WMI Provider will return a string containing the object path of the new object. Correspondingly, to remove a team the user should delete the instance of IANet_TeamOfAdapters. The NCS2 WMI Provider will delete the associations to the team members, and will also delete the virtual adapter and settings for the team.

Supported Properties

Name	Type	Description	Values	
AdapterCount	uint32	The number of adapters currently in the team.		
Caption	string	This is an inherited property; refer to parent class.		
Description	string	This is an inherited property; refer to parent class.		
LoadBalancedGroup	boolean	This is an inherited property; refer to parent class.		
MaxAdapterCount	uint32	The maximum number of adapters that can be placed in this team.		
MFOEnabled	boolean	The MFO status in the current team.		
Name	string	This is an inherited property; refer to parent class.		
RedundancyStatus	uint16	This is an inherited property; refer to parent class.		
StaticIPAddress	string	The static IP address assigned to the team, otherwise this is 0.0.0.0		
Status	string	This is an inherited property; refer to parent class.		
SubnetMask	string	The subnet mask assigned to the team, otherwise this is 0.0.0.0		
TeamingMode *	uint32	The type of the current team.	0 1 2 4 5 15 255	AFT ALB SLA IEEE 802.3ad SFT Detect Mode Unknown
TeamMACAddress	string	The configured MAC address of this team.		
TeamPrefix	[ ] uint16	This property is deprecated and is not in use.		

** Use Put() to change the value of the "TeamingMode" property, then call PutInstance() to update the team.*

Unsupported Properties

The following properties are not supported : InstallDate and Status.

Supported Methods

Method	Returns	Parameters	Detail
TestSwitchConfiguration	uint32	[out] uint16 [ ] CauseMessageId [out] string [ ] strCause [out] uint16 [ ] SolutionMessageId [out]string [ ] strSolution	Tests the switch configuration to ensure that the team is functioning correctly with the switch. This test can be used to check that link partners i.e., a device that an adapter links to, such as another adapter, hub, switch, etc., support the chosen adapter teaming mode. For example, if the adapter is a member of a Link Aggregration team, then this test can verify that link partners connected to the adapter support Link Aggregation
RenameTeam	uint32	[IN] string TeamName	Changes the name of an existing Intel team in the system.
ValidateAddAdapters	uint32	[in] [ ] ref: IANet_PhysicalEthernetAdapter Adapters [out] uint16 ValResult	Validates the adapters which will be added to this team.
ValidateSetting	uint32	[in] ref: IANet_PhysicalEthernetAdapter Adapter [in] string SettingName [in] sint64 Value [out] uint16 ValResult	Validates the member adapter setting.

Modifiable Properties

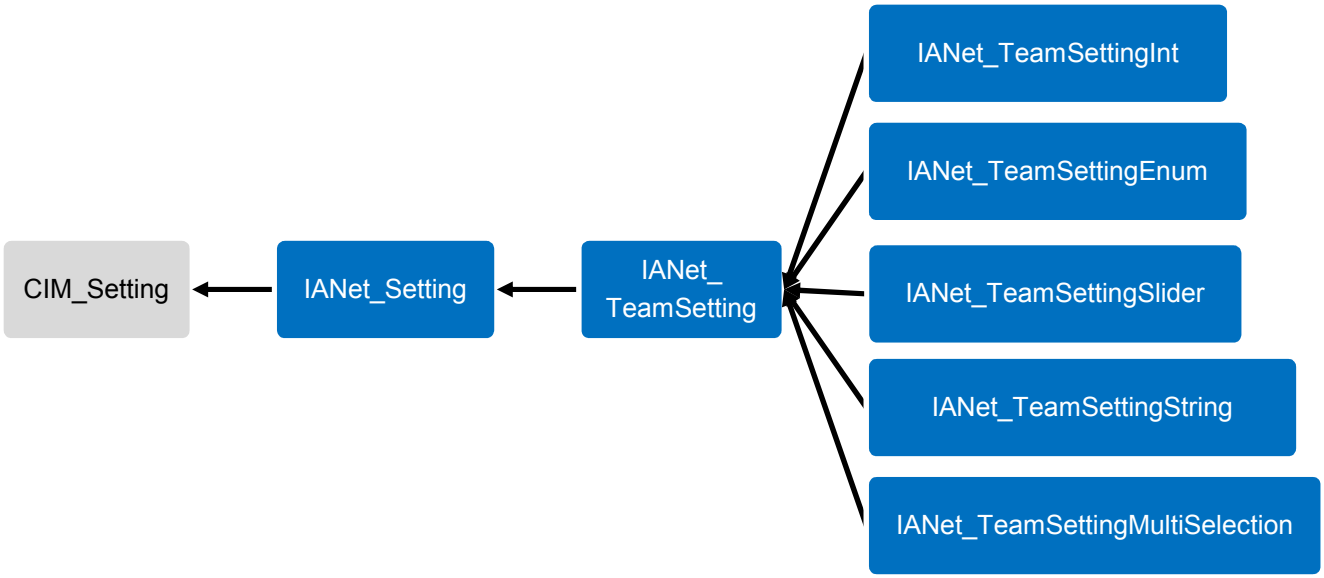
There are no user modifiable properties of this class.

Associations

Association Class	Association Partner
IANet_VirtualNetworkAdapter	IANet_LogicalEthernetAdapter
IANet_TeamedMemberAdapter	IANet_PhysicalEthernetAdapter

Team Settings

All team settings have a common base class and inheritance hierarchy.



IANet_TeamSetting

Purpose

This abstract class is used to describe a settable property in a configuration. The class is derived from IANet_Setting.

Instances

Instances of this class will exist for each setting on each Team. There are several sub-classes for IANet_TeamSetting. The sub-classes correspond to the different types and ranges of values that settings can take. Each sub-class corresponds to a different style of GUI that may be used to display or change the settings.

Supported Common Properties

Each of the five IANet_TeamSetting-[String, Enum, Slider, MultiSelection, Int] classes support a similiar set of properties. To reduce reproduction of the same information, the common properties are listed here:

Name	Type	Description
Caption	string	This is an inherited property; refer to parent class.
CurrentValue	sint64	Actual value of the parameter – this is the only attribute at the user can change.
DefaultValue	sint64	The initial value of the parameter.
Description	string	This is an inherited property; refer to parent class.
ExposeLevel	uint32	This is an inherited property; refer to parent class.

Grouped	boolean	This is an inherited property; refer to parent class.
GroupId	uint16	This is an inherited property; refer to parent class.
MiniHelp	string	This is an inherited property; refer to parent class.
ParentId	string	This is an inherited property; refer to parent class.
ParentType	string	This is an inherited property; refer to parent class.
Writable	boolean	This is an inherited property; refer to parent class.

Methods

No methods are supported by the IANet_TeamSetting-[String, Enum, Slider, MultiSelection, Int] classes.

Associations

Association Class	Association Partner
IANet_TeamToTeamSettingAssoc	IANet_LogicalEthernetAdapter

IANet_TeamSettingInt

Purpose

The class models a setting that takes an integer value. There are several IANet setting classes used to model integers. The differences between these classes concerns how the integer is displayed and modified by the GUI, and how validation is done by the NCS2 WMI Provider. For IANet_TeamSettingInt, it is expected that the GUI will display an edit box with a spin control.

Instances

An instance of this class exists for each setting that should be displayed as an integer edit box.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
base	uint64	Base is the root from which an integer value may take values.
max	sint64	The maximum value the integer can take.
min	sint64	The minimum value the integer can take.
Scale	sint64	The unit of measurement to set or estimate series of marks or points at known intervals to measure value of the parameter.
step	sint64	Granularity of the integer value.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. The user can modify this property by using IWbemClassObject::Put() to change the value, then call "IWbemServices::PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue <= max, CurrentValue >= min, (CurrentValue – min) is a multiple of step where max, min, CurrentValue and step are all properties of IANet_TeamSettingInt.

IANet_TeamSettingEnum

Purpose

The class models an enumeration setting value. For IANet_TeamSettingEnum, it is expected that the GUI will display a list of strings which map onto a small number of enumerated values. (e.g., a drop list combo box).

Instances

An instance of this class exists for each setting that will be displayed as an enumeration.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
DescriptionMap	[ ] string	Contains what each value means
PossibleValues	[ ] sint64	An array of possible values allowed for the Enum.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue $\in$ PossibleValues[]

IANet_TeamSettingSlider

Purpose

The class models a setting that specifically handles Slider settings. For IANet_AdapterSettingSlider, it is expected that the GUI will display a slider which will allow the user to choose the value in a graphical manner – the actual value chosen need not be displayed.

Instances

An instance of this class exists for each setting that will be displayed as a slider. Users can neither create or remove instances.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
FirstLabel	string	The label that should be displayed on the left side of the slider.
LastLabel	string	The label that should be displayed on the right side of the slider.
PossibleValues	[ ] sint64	The initial value of the parameter.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: urrentValue ∈ PossibleValues[]

IANet_TeamSettingMultiSelection

Purpose

This class models a setting whereby the user can select several options from a list of options. For IANet_AdapterSettingMultiSelection, it is expected that the GUI will display multi-selection list box which will allow the user to choose any (or no) option(s).

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
FirstLabel	string	The label that should be displayed on the left side of the slider.
LastLabel	string	The label that should be displayed on the right side of the slider.
PossibleValues	[ ] sint64	The initial value of the parameter.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then use "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue ∈ PossibleValues[]

IANet_TeamSettingString

Purpose

This class models a setting whereby the user can enter a free-form string value. For IANet_AdapterSettingString, it is expected that the GUI will display an edit box.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
MaxLength	uint32	The maximum length of the string.

Unsupported Properties

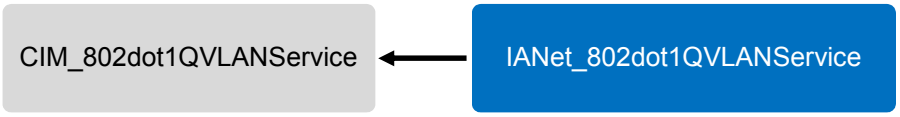
SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting.

VLAN Classes

IANet_802dot1QVLANService



Purpose

This class is used to hold the IEEE 802.1Q properties of a network adapter. This class implements the CIM class CIM_802dot1QVLANService.

Instances

An instance of this class exists for each adapter or team that supports IEEE 802.1Q. Each adapter or team can have just one IANet_802dot1QVLANService. Some teams, such as multi-vendor fault tolerant teams do not support this service. The user cannot create instances of this class If the adapter does not have an instance associated with it, then the adapter does not support this service. The user cannot delete instances of this class.

Modifiable Properties

There are no modifiable properties of this class.

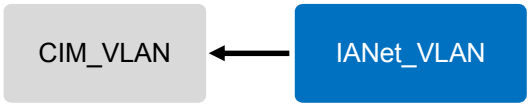
Methods

Method	Returns	Parameters	Detail
CreateVLAN	uint16	[in] uint32 VLANNumber [in] string Name [out] ref:IANet_VLAN	Used to create a VLAN on the adapter or team. The client must supply the VLAN number and the VLAN name, and will get the object path of the newly created VLAN.

Associations

Association Class	Association Partner
IANet_802dot1QVLANService	IANet_Device802dot1QVLANServiceImplementation

IANet_VLAN



Purpose

This class holds the information for each Intel VLAN. This class implements CIM_VLAN.

Instances

An instance of this class will exist of each Intel VLAN. To create a VLAN, call CreateVLAN from the appropriate instance of IANet_802dot1QVLANService.The user can remove an instance of this class to remove the corresponding VLAN.

Modifiable Properties

The user is able to modify the VLANNumber and Caption attribute.

Supported Properties

Name	Type	Description	Values	
Caption	string	This is an inherited property; refer to parent class.		
Description	string	This is an inherited property; refer to parent class.		
Name	string	This is an inherited property; refer to parent class.		
ParentID	uint16	Contains the VLAN's parent device ID.		
ParentType	uint16	Contains the VLAN's parent device type.	0 1 2	Adapter Team Unknown
StaticIPAddress	string	This field has a value if the VLAN is configured to have a static IP address. Otherwise, it will be set to 0.0.0.0		
StatusInfo	uint16	This is an inherited property; refer to parent class.		
SubnetMask	string	This field has a value if the VLAN is configured to have a subnet mask. Otherwise, it will be set to 0.0.0.0		
VLANName	string	This is the name of the VLAN chosen by the user.		
VLANNumber	uint32	This is the VLAN's identifying number.		

Unsupported Properties

Description, Install Date, StartMode, and Status are not used.

Associations

Association Class	Association Partner
IANet_VLANToVLANSettingAssoc	IANet_VLANSetting

IANet_VLANSetting

Purpose

This abstract class is used to describe a settable property in a configuration. The class is derived from IANet_Setting. Instances of this class will exist for each setting on each adapter. There are several sub-classes for IANet_VLANSetting. The sub-classes correspond to the different types and ranges of values that settings can take. Each sub-class corresponds to a different style of GUI that may be used to display or change the settings.

Supported Common Properties

Each of the five IANet_VLANSetting-[String, Enum, Slider, MultiSelection, Int] classes support a similiar set of properties. To reduce reproduction of the same information, the common properties are listed here:

Name	Type	Description
Caption	string	This is an inherited property; refer to parent class.
CurrentValue	sint64	Actual value of the parameter – this is the only attribute at the user can change.
DefaultValue	sint64	The initial value of the parameter.
Description	string	This is an inherited property; refer to parent class.
ExposeLevel	uint32	This is an inherited property; refer to parent class.
Grouped	boolean	This is an inherited property; refer to parent class.
GroupId	uint16	This is an inherited property; refer to parent class.
MiniHelp	string	This is an inherited property; refer to parent class.
ParentId	string	This is an inherited property; refer to parent class.
ParentType	string	This is an inherited property; refer to parent class.
Writable	boolean	This is an inherited property; refer to parent class.

Common Methods

No methods are supported by the IANet_VLANSetting-[String, Enum, Slider, MultiSelection, Int] classes.

IANet_VLANSettingInt

Purpose

The class models a setting that takes an integer value. There are several IANet setting classes used to model integers. The differences between these classes concerns how the integer is displayed and modified by the GUI, and how validation is done by the NCS2 WMI Provider. For IANet_AdapterSettingInt, it is expected that the GUI will display an edit box with a spin control.

Instances

An instance of this class exists for each setting that should be displayed as an integer edit box. Users can neither create or remove instances.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
base	uint64	Base is the root from which an integer value may take values.

max	sint64	The maximum value the integer can take.
min	sint64	The minimum value the integer can take.
Scale	sint64	The unit of measurement to set or estimate series of marks or points at known intervals to measure value of the parameter.
step	sint64	Granularity of the integer value.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. The user can modify this property by using IWbemClassObject::Put() to change the value, then call "IWbemServices::PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue <= max, CurrentValue >= min, (CurrentValue – min) is a multiple of step

IANet_VLANSettingEnum

Purpose

The class models a enumeration setting value. For IANet_AdapterSettingEnum, it is expected that the GUI will display a list of strings which map onto a small number of enumerated values. (e.g., a drop list combo box).

Instances

An instance of this class exists for each setting that will be displayed as an enumeration.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
DescriptionMap	[ ] string	Contains what each value means.
PossibleValues	[ ] sint64	An array of possible values allowed for the Enum.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue $\in$ PossibleValues[]

IANet_VLANSettingSlider

Purpose

The class models a setting that specifically handles Slider settings. For IANet_AdapterSettingSlider, it is expected that the GUI will display a slider which will allow the user to choose the value in a graphical manner – the actual value chosen need not be displayed.

Instances

An instance of this class exists for each setting that will be displayed as a slider. Users can neither create or remove instances.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
FirstLabel	string	The label that should be displayed on the left side of the slider.
LastLabel	string	The label that should be displayed on the right side of the slider.
PossibleValues	[ ] sint64	The initial value of the parameter.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: urrentValue ∈ PossibleValues[]

IANet_VLANSettingMultiSelection

Purpose

This class models a setting whereby the user can select several options from a list of options. For IANet_AdapterSettingMultiSelection, it is expected that the GUI will display multi-selection list box which will allow the user to choose any (or no) option(s).

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
FirstLabel	string	The label that should be displayed on the left side of the slider.
LastLabel	string	The label that should be displayed on the right side of the slider.
PossibleValues	[ ] sint64	The initial value of the parameter.

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then use "PutInstance()" to update the setting. The NCS2 WMI Provider will check that: CurrentValue ∈ PossibleValues[]

IANet_VLANSettingString

Purpose

This class models a setting whereby the user can enter a free-form string value. For IANet_VLANSettingString, it is expected that the GUI will display an edit box.

Supported Properties

This class supports the following properties in addition to those listed above.

Name	Type	Description
MaxLength	uint32	The maximum length of the string

Unsupported Properties

SettingID and RequiresSession are not used.

Modifiable Properties

The "CurrentValue" attribute is the only modifiable property of this class. Modify this property by using Put() to change the value, then call "PutInstance()" to update the setting.

Diagnostic Classes

IANet_DiagTest

Purpose

IANet_DiagTest is sub classed from CIM_DiagnosticTest. The class provides a generic vehicle to run and control Diagnostic tests for an Intel® PROSet for Windows Device Manager supported ethernet adapter. The super class, CIM_DiagnosticTest, is designed to generically support the testing of any computer hardware on a CIM enabled system. Properties of the class are descriptive in nature and the mechanics of the testing are provided by the exposed methods.

Instances

There is a one to one relationship between available diagnostic tests and instances of this class. Each test is distinguished by a Key, which is the concatenation of a diagnostic ID number, the "@" symbol, and the GUID of the referenced adapter (e.g. 1@{12345678-9ABC-DEF0-1234-123456789012}). This key value is, in one sense, redundant information, as all information to reference an adapter and test is passed as object parameters to the RunTest and other methods. Still, the instance must be consistent with parameters to the method or the NCS2 WMI Providers will reject the command. Other properties provide other description and run time information. The user cannot create or delete instances of this class.

The following table contains the diagnostic IDs which comprise the "<ID>@" part of the string. You can select which test to run on an adapter by choosing an ID from the table below and pairing it with the GUID of an adapter.

Diagnostic ID	Test Type
1	EEPROM
2	FIFO
3	REGISTER
4	INTERRUPT
17	LOOPBACK
32	LINK & DUPLEX
33	LINK & DUPLEX OFFLINE
34	CABLE
35	CABLE OFFLINE
36	PING

Supported Properties

Name	Type	Description	Values
Caption	string	This is an inherited property; refer to parent class.	
Characteristics	[ ] uint16	This is an inherited property; refer to parent class.	
Description	string	This is an inherited property; refer to parent class.	
Grouped	boolean	Some of the tests are grouped under specific categories. Grouped is true if this	

		is the case.
GroupId	uint16	Some of the tests are grouped under specific categories. This parameter specifies the ID of the group under which this test belongs.
Name	string	This is an inherited property; refer to parent class.
ResourcesUsed	[ ] uint16	This is an inherited property; refer to parent class.
Started	boolean	This is an inherited property; refer to parent class.
StartMode	string	This is an inherited property; refer to parent class.
Status	string	This is an inherited property; refer to parent class.
TestId	uint16	The test ID of the diagnostic test.

Unsupported Properties

Caption, Description, InstallDate, OtherCharacteristicDescription

Modifiable Properties

There are no user-modifiable properties for this class.

Methods

Method	Returns	Parameters	Detail
RunTest	uint32	[IN] ref : CIM_ManagedSystemElement SystemElement [IN] ref : CIM_DiagnosticSetting Setting [OUT] ref : CIM_DiagnosticResult Result	Runs a test as defined by three parameters referencing: SystemElement defines the adapter, which we are to run the test on by referring to an instance of SystemElement, which will always be the subclass IANet_EthernetAdapter. Setting defines the test to be run, and the manner in which it is run by referring to an instance of CIM_DiagnosticSetting, which will always be the subclass IANet_DiagSetting. Result defines an instance of the class CIM_DiagnosticResult, which will always be the class IANet_DiagResult.
DiscontinueTest		[IN] ref : CIM_ManagedSystemElement SystemElement [IN] ref : CIM_DiagnosticResult Result [OUT] Boolean TestingStopped	Attempts to stop a diagnostic test in progress as defined by two parameters referencing SystemElement and Result. These parameters function the same as RunTest. A third parameter TestingStopped returns a BOOLEAN value, which indicates if the command was successful in stopping the test.
ClearResults		[IN] ref : CIM_ManagedSystemElement SystemElement [OUT] [ ] String	The referenced parameter ManagedSystemElement, combined with this object's object path combine to reference instances of DiagnosticResultForMSE, which will be deleted. Also, all references of DiagnosticResult objects referenced by

ResultsNotCleared	DiagnosticResultForMSE will be deleted. Also, all instances of Diagnostic-ResultForTest, which refer to the deleted DiagnosticResult objects, will be deleted. Finally, the string array Output parameter ResultsNotCleared will list the keys of the DiagnosticResults, which could not be cleared.

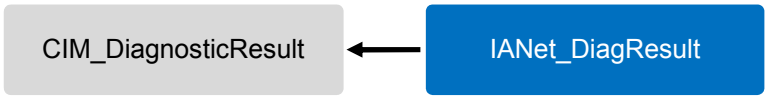
Unsupported Methods

StartService and StopService are not supported.

Associations

Association Class	Association Partner
IANet_DiagTestForTest	IANet_DiagResult
IANet_DiagSettingForTest	IANet_DiagSetting
IANet_DiagTestForMSE	IANet_PhysicalEthernetAdapter

IANet_DiagResult



Purpose

Instances of IANet_DiagResult display result data for a particular test run on a particular Adapter. Instances of this class correspond identically to instances of IANet_DiagTest and IANet_DiagSetting.

Associations

Association Class	Association Partner
IANet_DiagResultForTest	IANet_DiagTest
IANet_DiagResultForMSE	IANet_PhysicalEthernetAdapter

Instances

Instances of IANet_DiagResult correspond to results of a particular test run on a specific adapter. The format for the key is the same as IANet_DiagTest and IANet_DiagSetting. The instance is able to store any arbitrary test results as any data, which does not fit the defined properties, can be placed into the TestResults array property. Any time a new test is run on an adapter, the new instance overwrites the existing instance of test results corresponding to that adapter and test combination. The user cannot create instances or delete instances of this class. There will be one instance for each adapter and test combination.

Modifiable Properties

The user cannot modify instances of this class.

Supported Properties

Name	Type	Description	Values
Caption	string	This is an inherited property; refer to parent class.	
Characteristics	[ ] uint16	This is an inherited property; refer to parent class.	
Description	string	This is an inherited property; refer to parent class.	

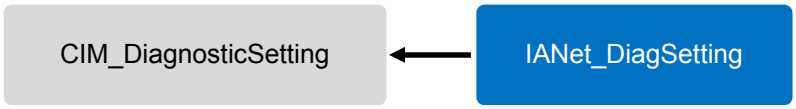
Grouped	boolean	Some of the tests are grouped under specific categories. Grouped is true if this is the case.
GroupId	uint16	Some of the tests are grouped under specific categories. This parameter specifies the ID of the group under which this test belongs.
Name	string	This is an inherited property; refer to parent class.
ResourcesUsed	[ ] uint16	This is an inherited property; refer to parent class.
Started	boolean	This is an inherited property; refer to parent class.
StartMode	string	This is an inherited property; refer to parent class.
Status	string	This is an inherited property; refer to parent class.
TestId	uint16	The test ID of the diagnostic test.

Unsupported Properties

The following properties are not supported by NCS2:

EstimatedTimeOfPerforming OtherStateDescription, HaltOnError, ReportSoftErrors, and TestWarningLevel.

IANet_DiagSetting



Purpose

Instances of IANet_DiagSetting provide specific run time diagnostic test directives. Directives used are in common to all tests and are bound to the superclass CIM_DiagnosticSetting. These include properties such as ReportSoftErrors and HaltOnError. There are no additional properties added to the subclass IANet_DiagSetting.

Associations

Association Class	Association Partner
IANet_DiagSettingForTest	IANet_DiagTest

Instances

The user cannot create instances or delete instances of this class. There will be one instance for each adapter and test combination.

Modifiable properties

UpdateInstanceAsync is implemented and can be used to set test parameters to HaltOnError, ReportSoftErrors, ReportStatusMessages, QuickMode, TestWarningLevel, and PercentOfTestCoverage.

Supported Properties

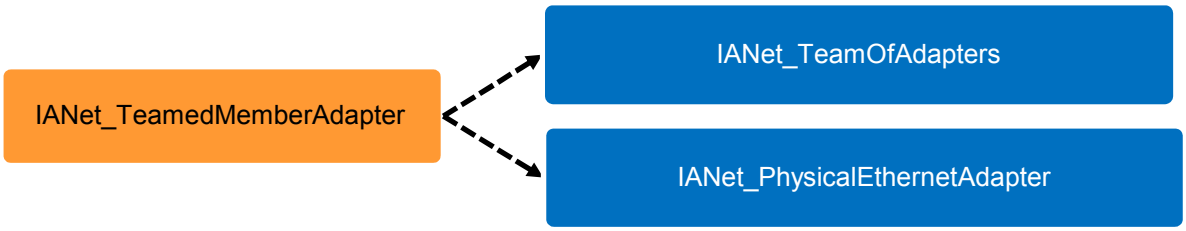
Name	Type	Description
SettingID	string	This is an inherited property; refer to parent class.

Unsupported properties

The following properties are not supported : Caption, Description.

Association Classes

IANet_TeamedMemberAdapter



Purpose

This class is used to associate the adapter with the team, determine the function of the adapter in the team, and establish that the adapter is currently active in the team. This class implements the CIM class CIM_NetworkAdapterRedundancyComponent. An instance of this class exists for each adapter that is a member of a team. To add an adapter to a team, create an instance of IANet_TeamedMemberAdapter to associate the adapter with the team. To remove an adapter from the team, remove the instance of IANet_ TeamedMemberAdapter. The adapter will no longer be part of the team and may be bound to an IP protocol endpoint after the Apply() function is called. There are no supported methods in this class.

Instances

The user cannot create instances or delete instances of this class. There will be one instance for each adapter bound to a team.

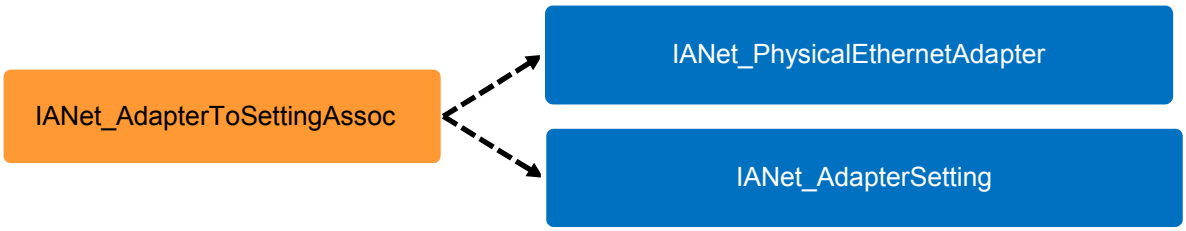
Supported Properties

Name	Type	Description	Values	
AdapterFunction	uint32	Describes how the adapter is used in the team. The AdapterFunction property of this class may be modified to describe how the adapter is used.	0 1 2 3	Unknown Primary Adapter Secondary Adapter Other
AdapterStatus	uint32	Describes the adapter's status within the team.	0 1 2 3	Unknown Active Standby InActive

Unsupported Properties

The following properties are not supported : PrimaryAdapter and ScopeOfBalancing.

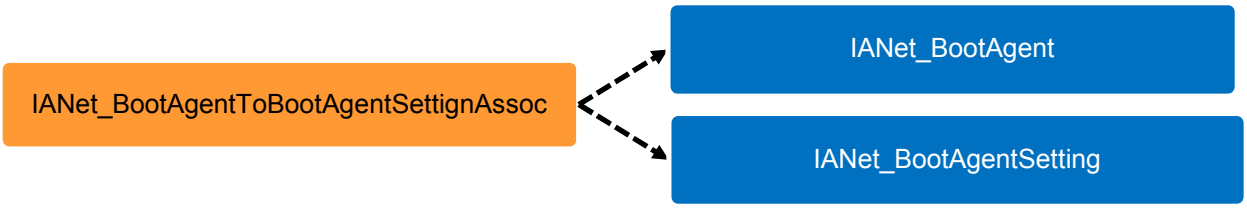
IANet_AdapterToSettingAssoc



Purpose

This class associates Intel network cards with their respective settings.

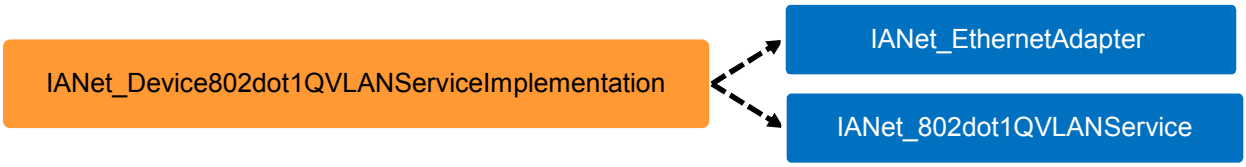
IANet_BootAgentToBootAgentSettingAssoc



Purpose

This class is used to group a collection of IANet_BootAgentSetting instances.

IANet_Device802dot1QVLANServiceImplementation



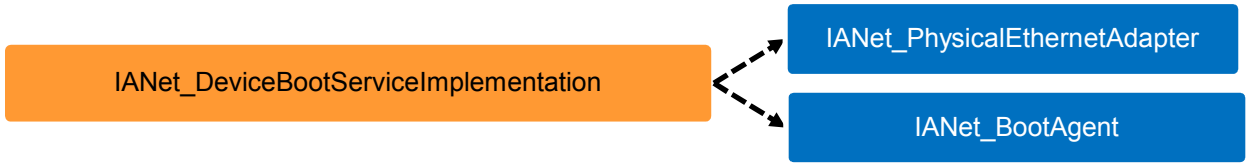
Purpose

This class is used to group a collection of IANet_Device802dot1QVLANServiceImplementation instances. This is a service class through which users can initiate VLAN addition or removal.

Instances

There will be once instance of this class for every adapter or team present. The user cannot create instances or delete instances of this class.

IANet_DeviceBootServiceImplementation



Purpose

This class is used to group a collection of IANet_Device802dot1QVLANServiceImplementation instances.

Instances

There will be once instance of this class for every adapter. The user cannot create instances or delete instances of this class.

IANet_DiagResultForMSE

Purpose

This class relates diagnostic test results to the ManagedSystemElement that was tested.

Instances

There will be one instance of this class for every diagnostic result. Diagnostic tests must be executed before instances of this class will exist.

IANet_DiagResultForTest

Purpose

This class is an implementation of the DiagnosticResultForTest class.

Instances

There will be one instance of this class for every diagnostic result. Diagnostic tests must be executed before instances of this class will exist.

IANet_DiagSettingForTest

Purpose

This class is an implementation of the DiagnosticSettingForTest class.

Instances

There will be one instance of this class for every diagnostic test.

IANet_DiagTestForMSE

Purpose

This class is an implementation of the DiagnosticTestForMSE class.

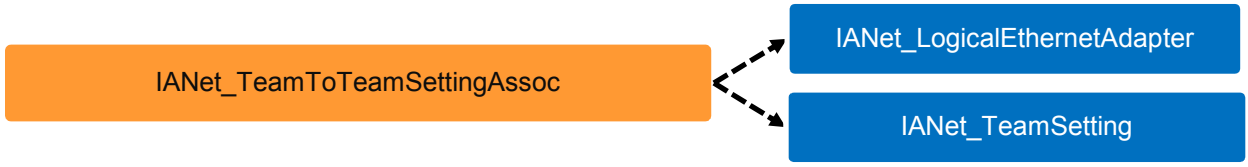
Instances

There will be one instance of this class for every diagnostic test.

Unsupported Properties

The following properties are not supported : EstimatedTimeOFPerforming

IANet_TeamToTeamSettingAssoc



Purpose

This class associates teams with their respective settings.

Instances

There will be one instance of this class for every team setting.

IANet_VLANFor

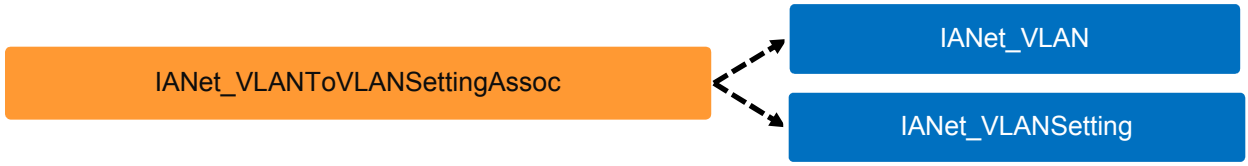
Purpose

This class is used by the IANet_802dot1QVLANService.

Instances

There will be one instance of this class for every VLAN.

IANet_VLANToVLANSettingAssoc



Purpose

This class associates VLANs with their respective settings.

Instances

There will be one instance of this class for every VLAN.

CIMv2 Class Definitions

Intel PROSet for Windows Device Manager adds some classes to the root\CIMv2 namespace. In some cases, these classes are duplicates of similar named classes in the root\IntelNCS2 namespace. There are also some classes which might maintain the same name but vary in functionality. The information below captures the differences between the root\IntelNCS2 definition of the class and the root\CIMv2 definition.

IANet_DiagSetting

This class is the same as the definition for it in the root\IntelNCS2 namespace.

IANet_DiagSettingForTest

This class is the same as the definition for it in the root\IntelNCS2 namespace.

IANet_EthernetAdapter

This class is present in the root\CIMv2 namespace is inherited by the IANet_PhysicalEthernetAdapter class. Since this class defines properties needed by its descendants, its use has been carried over to this namespace. The class definition in this namespace is no different than the root\IntelNCS2 definition; please refer to that section of the document for class detail.

IANet_PhysicalEthernetAdapter

Since the IANet_PhysicalEthernetAdapter class installs into the root\CIMv2 namespace, additional parent class properties will be present which are not shown in the same class for the root\IntelNCS2 namespace. The reason different properties appear is due to different CIM specifications being used.

Additional Unsupported Properties

These additional properties are defined by Microsoft based on the CIM version 2.5 schema and are *not supported* by instances of IANet_PhysicalEthernetAdapter: ConfigManagerErrorCode, ConfigManagerUserConfig, PnPDeviceID.

Associations

IANet_PhysicalEthernetAdapter has fewer associations in this namespace:

Association Class	Association Partner
IANet_DiagTestForMSE	IANet_DiagTest
IANet_DiagResultForMSE	IANet_DiagResult

IANet_DiagTest

The following properties are present for this class in the root\IntelNCS2 namespace but are missing in the root\CIMv2 namespace: Grouped, GroupId. Also, the .VendorID property is present and used in the root\CIMv2 namespace (it will evaluate to "Intel Corp.").

IANet_DiagResult

The following properties are present for this class in the root\IntelNCS2 namespace but are missing in the root\CIMv2 namespace: ConnectionTest_DHCPServerAddresses, ConnectionTest_DNSAddresses,

ConnectionTest_GatewayAddresses, ConnectionTest_NetworkAddresses, ConnectionTest_WINSAddresses, TestResultsIds, TestResultsAttr.

IANet_DiagResultForMSE

This class is the same as the definition for it in the root\IntelNCS2 namespace.

IANet_DiagResultForTest

This class is the same as the definition for it in the root\IntelNCS2 namespace.

Appendix

This section contains specific information to help users working with the NCS2 architecture.

Related Documents

CIM schema version 2.0, 2.2 published by Distributed Management Task Force (DMTF), http://www.dmtf.org.

Microsoft Windows Management Instrumentation (and other manageability information) http://www.microsoft.com/hwdev/WMI/.

Web-based Enterprise Management (WBEM) initiative by DMTF http://www.dmtf.org/wbem/index.html.

Terminology

Term	Explanation
ANS	Advanced Networking Services (ANS) teaming is a feature of the Intel® Advanced Networking Services component that lets you group multiple adapters in a system into a team.
API	An Application Programming Interface exposed by a library or system for service requests.
CIM	Common Information Model; a standard for describing computers and services.
CIMOM	CIM Object Manager; part of Windows Management.
COM	Component Object Model; a Microsoft platform for inter-process communication and object creation.
DMTF	Distributed Management Task Force; a standards organization for the IT industry.
GUI	Graphical User Interface; refers to the user interface layer.
MOF	Managed Object Format; a file extension of a special file format used in Windows management.
NCS2	Network Configuration Services 2.0 - the architecture used in Intel® PROSet for Windows Device Manager
VLAN	Virtual LAN; a method for creating logical networks within a physical network.
WBEM	Web Based Enterprise Management; technologies to unify distributed computing environments.
WMI	Windows Management Instrumentation; Microsoft's implementation of the CIM standard for Windows.

Working Examples

Getting the Current Configuration

The client does not need to get a client handle to read the current configuration. Clients can use a NULL context, however, any error messages will be returned in the default language for the managed machine. In the following tables, items enclosed in { } are object paths. These paths are assumed to have been obtained from previous WQL queries. The client should never need to construct an object path without doing a query. The __PATH attribute of every object contains the object path for that object. In all the following use cases, the methods IWbemServices::ExecQuery or IWbemServices::ExecQueryAsync are used to execute WQL queries.

Getting the Physical Adapters

The main class for adapters is IANet_PhysicalEthernetAdapter. This class is used for both physical and virtual adapters, and the client needs to know how to distinguish between them.

Task	WQL Query	Result Class	Comment

Enumerate all adapters	SELECT * FROM IANet_EthernetAdapter	IANet_EthernetAdapter	Returns all IANet_EthernetAdapters. This is equivalent to IWbemServices::CreateInstanceEnumAsync.
Determine if adapter is virtual	ASSOCIATORS OF {adapter path} WHERE AssocClass = IANet_NetworkVirtualAdapter	IANet_TeamOfAdapters	If the query results in no classes then the adapter is a real adapter.

Getting the Team Configuration

The main classes in the teaming schema are IANet_LogicalEthernetAdapter, IANet_TeamOfAdapters, IANet_NetworkVirtualAdapter and IANet_TeamedMemberAdapter in the root\IntelNCS2 namespace. The association class IANet_NetworkVirtualAdapter contains no useful data – clients are really only interested in the endpoints of this association. IANet_TeamedMemberAdapter does contain useful data about how the member adapter is used within the team.

Task	WQL Queries	Result Class	Comments
Enumerate all teams	SELECT * FROM IANet_TeamOfAdapters	IANet_TeamOfAdapters	There is one instance of IANet_TeamOfAdapters for each team. This is equivalent to IWbemServices::CreateInstanceEnumAsync.
Get the virtual adapter for a team	ASSOCIATORS OF {IANet_TeamOfAdapters path} WHERE AssocClass = IANet_NetworkVirtualAdapter	IANet_LogicalEthernetAdapter	Returns only the adapter object for the virtual adapter in the team. This adapter will not exist if the team has been created but Apply has not been called. (see below on updating the configuration).
Enumerate the team's member adapters	ASSOCIATORS OF {IANet_TeamOfAdapters path} WHERE AssocClass = IANet_TeamedMemberAdapter	IANet_PhysicalEthernetAdapter	Returns the adapters which are in the team, but does not describe what role the adapter plays.
Determine an adapter's role in a team	REFERENCES OF {IANet_PhysicalEthernetAdapter path} WHERE ResultClass = IANet_TeamedMemberAdapter	IANet_TeamedMemberAdapter	The class contains information about how the member adapter relates to the team and its current status within the team.

Getting the VLAN configuration

Any adapter or team supporting VLANs has an IANet_802dot1QVLANService associated with it, using the association class IANet_Device802do1QVVLANServiceImplementation. If an adapter or team does not have an instance of this class associated with it, then it does not support VLANs. Each VLAN is represented by an instance of IANet_VLAN in the root\IntelNCS2 namespace. IANet_VLAN does not have a direct association – it is associated with the corresponding IANet_802dot1QVLANService for the adapter or team. The association class IANet_VLANFor is used to associate each VLAN instance with the correct ANet_802dot1QVLANService.

Task	WQL Queries	Result Class	Comments
Get the 802.1q VLAN service object associated with an adapter	ASSOCIATORS OF {IANet_EthernetAdapter path} WHERE ResultClass = IANet_802dot1QVLANService	IANet_802dot1QVLANService	Returns one or no object(s).
Get the VLANs on an adapter	ASSOCIATORS OF {IANet_802dot1QVLANService path} WHERE ResultClass = IANet_VLAN	IANet_VLAN	This can return no objects if there are no VLANs installed.

Getting the Boot Agent Information

Each adapter that can support a boot agent in flash ROM will have an IANet_BootAgent instance associated with it using the IANet_DeviceBootServiceImplementation association class.

Task	WQL Queries	Result Class	Comments
Get the Boot Agent associated with an adapter	ASSOCIATORS OF {path of IANet_EthernetAdapter} WHERE ResultClass = IANet_BootAgent	IANet_BootAgent	The following read only properties provide information on the boot ROM image for this adapter: InvalidImageSignature, Version, UpdateAvailable, FlashImageType

Updating the configuration

In most cases, to update the configuration, the client application will need to get a client handle from the IANet_NetService class and store this handle in an IWbemContext context object. Changes to the configuration will only occur when the "Apply" method on the IANet_NetService is called.

Changing the adapter, team or VLAN settings

To change an adapter, VLAN or team setting, the client must first get the object path of the setting that it will change. This is best done by enumerating the settings on the object and storing the __PATH attribute of the setting (see above).

The easiest way for the client to update a setting, is to:

1) Get an instance of the setting object from WMI,
2) Modify the CurrentValue attribute (using IWbemClassObject::Put())
3) Call IWbemServices::PutInstance() to pass the modified instance back to the NCS2 WMI Providers. PutInstance must be called with the flag WBEM_FLAG_UPDATE_ONLY.

The NCS2 WMI Providers will validate CurrentValue and return WBEM_E_FAIL if the validation failed. The exact reason for the failure will be returned in the Description attribute of the IANet_ExtendedStatus object.

Setting specific descriptions include:

- The integer setting value was less than the minimum allowed
- The integer setting value was greater than the maximum allowed

- The integer setting value is not one of the allowable steps
- The length of the string setting is bigger than the maximum allowed
- The setting value is not one of the allowable values

The last description is returned whenever the current value for IANet_SettingEnum, IANet_SettingSlider or IANet_SettingMultiSelection is not one of the allowable values.

The only attribute for a setting that the client can change is CurrentValue. The NCS2 WMI Providers will ignore changes made to any of the other values. There are no supported methods on the setting class. To make changes to a setting modify the CurrentValue property, then call PutInstance.

Creating a new team

Adapter teams can be created by utilizing classes and methods in the root\IntelNCS2 namespace:

1) Create an instance of IANet_TeamOfAdapters (i.e., use IWbemServices::GetObject() to get a class object for IANet_TeamOfAdapters, and then use IWbemServices::SpawnInstance() to create an instance of this object).
2) Use IWbemClassObject::Put to set the TeamMode attribute in the instance to be the desired team type (e.g., AFT).
3) Finally, call IWbemServices::PutInstance() to create the team, passing the flag WBEM_FLAG_CREATE_ONLY.
4) The object path for the new team is stored in the IWbemCallResultObject that is passed back to the user when the call has completed. The method IWbemCallResult::GetResultString will get the new object path. If this action fails, the client should check the IANet_ExtendedStatus to get the failure reasons.

Adding an adapter to a team

1) Create an instance of IANet_TeamedMemberAdapter (i.e., use IWbemServices::GetObject() to get a class object for IANet_TeamedMemberAdapter, and then use IWbemServices::SpawnInstance() to create an instance of this object).
2) The following properties in the object must be set using IWbemClassObject::Put() :
3) GroupComponent must be set to be the full object path of the IANet_TeamOfAdapter which the adapter is to be added
4) PartComponent must be set to be the full object path of the IANet_EthernetAdapter that is to be added to the team.
5) Priority of the adapter in the team (optional)
6) Finally, call IWbemServices::PutInstance() to add the adapter to the team, passing the flag WBEM_FLAG_CREATE_ONLY. If this action fails, check IANet_ExtendedStatus for the error code.

Removing an adapter from a team

To remove an adapter from a team, delete the IANet_TeamedMemberAdapter instance that associates the adapter to the team using IWbemServices::DeleteInstance(). If this action fails, check IANet_ExtendedStatus for the error code.

Deleting a team

To delete a team, delete the IANet_TeamOfAdapters instance using IWbemServices::DeleteInstance()

If this action fails, check IANet_ExtendedStatus to get the error code.

Changing the mode of a team

The client can change modes of a team, however, this action is limited to the root\IntelNCS2 namespace.

1) Get the instance of IANet_TeamOfAdapters for the team (e.g., use IWbemServices::GetObject using the object path of the team).
2) Use IWbemClassObject::Put to change the TeamMode attribute for the team.
3) Finally, call IWbemClassObject:: PutInstance to tell NCS2 WMI Provider to update the team mode, passing the flag WBEM_FLAG_UPDATE_ONLY. If this action fails, check IANet_ExtendedStatus to get the error code.

Changing an adapter's priority within a team

1) Get the instance of IANet_TeamedMemberAdapter for the adapter. (e.g. use IWbemServices::GetObject using the object path).
2) Use IWbemClassObject::Put to change the AdapterFunction attribute for the adapter.
3) Finally the client needs to call IWbemClassObject:: PutInstance to tell the NCS2 WMI Provider to update adapter's priority. If this action fails the client should check the IANet_ExtendedStatus for the error code.

Uninstalling an adapter

To uninstall an adapter, call IWbemServices::DeleteInstance passing the object path of the adapter to uninstall.

Creating a VLAN

The client can create VLANs on adapters or teams; this operation is limited to the root\IntelNCS2 namespace.

To create a VLAN

1) Call the CreateVLAN method on the IANet_802dot1QVLANService for the device (adapter or team) to which the VLAN is to be added. The following arguments must be passed to the method:
2) VLANNumber the number of the VLAN. (Range 1- 4094)
3) Name a user definable name to identify the VLAN.
4) The function will return the object path of the newly created VLAN in the out parameter VLANpath. If this action fails, check IANet_ExtendedStatus for the error code.

Changing the Properties of a VLAN

The client can change the VLANNumber and VLANName properties for a VLAN. This operation is only supported in the Intel\NCS2 namespace.

To change the priority of an adapter

1) Get the instance of IANet_VLAN for the adapter (e.g. use IWbemServices::GetObject using the object path).
2) Change VLANNumber or VLANName to the desired values.
3) Call IWbemClassObject:: PutInstance to tell the NCS2 WMI Provider to update the properties, passing the flag WBEM_FLAG_UPDATE_ONLY. If this action fails, check the IANet_ExtendedStatus for the error code.

Deleting a VLAN

To delete a VLAN, call IWbemServices::DeleteInstance passing the object path of the VLAN to delete.

Updating the Boot Agent

The client can update the Boot Agent Image by using methods calls.

To read/write flash image

1) Get the instance of IANet_BootAgent for the adapter (e.g., use IWbemServices::GetObject using the object path).

2) Execute ReadFlash() to read the existing flash boot ROM image or ProgramFlash() to update the flash boot ROM image. If this action fails, check the IANet_ExtendedStatus for the error code.

Run Diagnostics from WBEMTest

Here is the RunTest method, from the MOF file:

"uint32 RunTest([IN] CIM_ManagedSystemElement ref SystemElement,

[IN] CIM_DiagnosticSetting ref Setting,

[OUT] CIM_DiagnosticResult ref Result);"

The first two parameters are IN parameters. You must get the object path of both objects referenced. You must also get the object path of the IANet_DiagTest object, which is exporting the RunTest object.

1) From the main WBEM test dialog box:
2) Click "Connect".
3) Enter the appropriate Server\Namespace. Namespaces IntelNCS2 and CimV2 are supported.
4) Click the "Enum Instances" button of WBEM test and enter "IANet_DiagTest". Enable recursive queries.
5) Double click the desired instance of IANet_DiagTest. The name will be in the form X@[AdapterGUID}, where X is the test name and AdapterGUID will be the Adapter Name, same as the Name key of the IANet_EthernetAdapter. The following is an example of the object path retrieved: \\MYCOMPUTER\root\Cimv2:IANet_DiagTest.Name="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"
6) Save the object path.
7) Click the "Enum Instances" button of WBEM test and enter "IANet_EthernetAdapter"
8) Double click on the desired adapter, to be tested. Following is an example of the object path retrieved: \\MYCOMPUTER\root\cimv2:IANet_EthernetAdapter.DeviceID="{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"
9) Save the object path.
10) Click the "Enum Instances" button of WBEM test and enter "IANet_DiagSetting"
11) Double click on the setting which represents the desired adapter/test combination. Following is an example of the object path retrieved: \\MYCOMPUTER\root\cimv2:IANet_DiagSetting.SettingID="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"
12) Save the object path.
13) From the main WBEM test dialog box, click "Execute Method"
14) Paste the IANet_DiagTest object path into the dialog box. Click OK
15) Select the desired test in the drop down box under method.
16) Click the "Edit In Parameters" button.
17) For RunTest, Setting and SystemElement are the in parameters, paste the previously saved Setting and Adapter object paths. Close.
18) Click the execute button.
19) Enumerate the IANet_DiagResult class, in the same manner as the In parameters were.
20) Examine the selected result object as needed.

Addendum to NCS2 Architecture

Note #	Area	Description
1	Method return values	Notes regarding WMI methods offered by the NCS2 architecture.
2	iSCSI	How to get and set iSCSI adapter states and settings.
3	Permissions	Permissions required for set and get operations.
4	Context for CIMv2 namespace	How to obtain client ID locks for the CIMv2 namespace.
5	Diagnostic results time limit	There is a time limit on how long diagnostic results are available.

1) Method return values

When calling a method from the NCS2 architecture a numeric value will be returned. If the function call was successful, a value of 0 will be passed back. A return value of 0 only indicates the function was able to be called – it has no bearing on whether the function call had the expected outcome. In most cases, you can expect a value of 0 to be returned, but will need to use other means to determining whether the requested operation was successful.

2) iSCSI

The iSCSI settings of an Intel network card can be manipulated through the WMI interface. Only operating systems which support iSCSI will have these settings available

Getting iSCSI Status

To retrieve the iSCSI status of a network card, enumerate instances of the IANet_BootAgent_iSCSI_Adapters class. In this class, look at the iSCSI_Status parameter. This will indicate the current state of your iSCSI enabled adapter.

Setting iSCSI Status

The iSCSI state of the adapter is manipulated through the SetiSCSI_Status method of the IANet_BootAgent_iSCSI_Adapters class. This only controls whether the adapter can be set to a Primary, Secondary or Disabled state.

Manipulating iSCSI Parameters

There are several settings applicable to iSCSI which can be manipulated through WMI. To locate these parameters, use the table below to find the name of the class, the type of parameter, and guidelines for setting them. Each iSCSI enabled adapter will have its own settings.

Setting Caption	Class	Notes
Authentication	IANet_BootAgentSettingEnum	0 (Disable CHAP) or 1 (Enable CHAP)
BootLUN	IANet_BootAgentSettingInt	The iSCSI Boot LUN
ChapPassword	IANet_BootAgentSettingString	A string value no longer than 16 characters
ChapUserName	IANet_BootAgentSettingString	A string value no longer than 16 characters
InitiatorDHCP	IANet_BootAgentSettingEnum	This requires an string formatted as an IP address
InitiatorGateway	IANet_BootAgentSettingString	This requires an string formatted as an IP address
InitiatorIPAddress	IANet_BootAgentSettingString	This requires an string formatted as an IP address
InitiatorName	IANet_BootAgentSettingString	A string value no longer than 255 characters
InitiatorSubnetMask	IANet_BootAgentSettingString	This requires an string formatted as an IP address

TargetDHCP	IANet_BootAgentSettingEnum	This requires an string formatted as an IP address
TargetIPAddress	IANet_BootAgentSettingString	This requires an string formatted as an IP address
TargetName	IANet_BootAgentSettingString	A string value no longer than 255 characters
TargetPort	IANet_BootAgentSettingInt	An integer between 0 and 65535

3) Permissions

Software changes require Administrator rights on the operating system. Any level of permissions with WMI access rights can make queries to retrieve information. This applies to local and remote access.

Windows Vista requires Administrator rights which can be obtained by logging in as the Administrator account or elevating permissions for an Administrator group member.

4) Context for CIMv2 namespace

Making any changes requires a software lock ID, which can only be obtained through the IANet_NetService class in the root\IntelNCS2 namespace. It is possible to facilitate setting changes in the CIMv2 namespace but the client access lock get and apply operations will need to occur in the root\IntelNCS2 namespace.

5) Diagnostic results time limit

The NCS2 providers will automatically terminate within a few minutes of not being used. When this event occurs, all diagnostic results will be lost.

6) Remote Desktop Limitations

When connecting to a computer remotely with Remote Desktop Protocol and no Administrator locally logged in, use the "/console" option. The other workaround is to make sure a local Administrator account is logged in when initiating remote desktop access. This makes sure the WMI layer can authenticate with the local permissions.