英特尔**(R) PRO** 网络适配器和 **WMI** 和 **CDM Provider** 用户指南

以下信息由所提及设备的供应商提供，未经 Dell 独立的证实，并受以下限制与免责声明的制约。

限制和免责声明

本文所含的信息，包括所有说明、警告以及管制性认可和证书，均由供应商提供，未经 Dell 独立证实或测试。Dell 对因遵照或未遵照这些说明而造成的损失概不负责。

关于本文所提部件的属性、功能、速度或合格性的一切陈述和声明均由供应商而非 Dell 提供。Dell 特别指出对以上声明的准确性、完整性或可靠性无所知悉。有关以上陈述或声明的任何问题或意见应向供应商提出。

初版：*2003* 年 *10* 月

返回目录页面

简介：英特尔**(R) PRO** 网络适配器和 **WMI** 和 **CDM Provider** 用户指南

概述

欢迎使用英特尔*(R) PRO* 网络适配器和 *WMI* 和 *CDM Provider* 用户指南。此文档描述英特尔 PRO 网络适配器 WMI 和 CDM Provider 的外观。Windows Management Interface (WMI) Provider 是 Network Configuration Services (NCS) 的网络配置块，作为使用业界标准方法部署和管理所有英特尔终端计算机联网技术的一个手段。英特尔 PRO Common Diagnostic Model (CDM) Provider 是一个符合 CIM 2.5 和 WMIis 标准的上层界面 API。在下层界面，CDM Provider 将客户端界面实现为 PROSet 软件堆栈中的两层。这将保证所有的 PROSet 的数据完整性机制。

WMI 和 CDM Provider 是实现英特尔 WMI 网络类的软件组件集。这些类基于 Desktop Management Task Force （桌面管理任务组，DMTF）CIM 架构版本 2.5。

此文档不重复包含在随此产品提供的 Managed Object Format（被管理对象格式）（MOF）中的信息（例如：有关各属性的细节可在 MOF 属性描述中找到）。

本文档描述 WMI 应用程序（如英特尔 PROSet）如何使用类来配置系统网络以及 WMI 应用程序如何使用类来测试英特尔网络接口卡。读者应该熟悉 WMI API 和 WMI SDK (可从 http://www.microsoft.com/ 获取)。

返回页首

相关文档

可以参考以下文档以进一步了解 WMI 技术。

- 由 Desktop Management Task Force (DMTF) 发行的 CIM 架构版本 2.0 和 2.2。可从 http://www.dmtf.org （英文）获取。
- Microsoft Windows Management Interface （以及其他有关管理方面的信息）。可从 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_start_page.asp （英文）获取。
- DMTF 推出的 Web-Based Enterprise Management (基于 Web 的企业管理)(WBEM)。可从 http://www.dmtf.org/standards/wbem （英文）获取。
- WMI (Microsoft CIM implementation) SDK。可从 http://msdn.microsoft.com/downloads/ （英文）获取。
- DFTM 的 System Diagnostic Model White Paper （系统诊断模型白皮书）。可从 http://www.dmtf.org/standards/documents/CIM/DSP0138.pdf （英文）获取。

警告：本产品包含可被用来攻击和／或禁用计算机系统或网络的信息。应用此产品的实现必须具备 **Microsoft** 操作系统安全功能的全面知识。强烈建议开发人员和用户如果在生产环境中应用此产品的任何实现之前有任何关于安全性的问题，请务必联系 **Microsoft**。

请阅读所有限制和免责声明。

返回目录页面 返回页首

返回目录页面

WMI：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述

Web-Based Enterprise Management (WBEM) 是 Desktop Management Task Force (DMTF) 的一项倡议，旨在为大型企业系统管理员提供一个标准而经济的终端工作站管理方法。该 WBEM 倡议涉及一系列任务，从简单的工作站配置到全面的多平台企业管理。此项倡议的中心是 Common Information Model （公共信息模型）(CIM)，这是一种可扩展的数据模型，用于代表存在于典型的管理环境中的对象；也是一种 Managed Object Format（被管理对象格式）（MOF）语言，用于定义和存储模型化的数据。

Windows Management Instrumentation (WMI) 是 WBEM 倡议在 Microsoft* Windows* 平台上的实现。

WMI 包含三个主要组件：

- 内核 – 这些组件是操作系统的一部分。若要启用了 WMI 的应用程序工作，则需要这些组件，必须在安装这些组件之后才能使用 SDK 。
- SDK – SDK 包含浏览 WMI 架构、扩展架构、创建提供程序、注册和使用 WMI 的工具。它还提供对开发将使用 WMI 的应用程序有用的参考文档。SDK 作为 Microsoft Platform SDK 安装过程的一部分被安装，它在 Windows NT4 SP4 或 SP5、Windows 2000、Windows Me、Windows XP 和 Windows Server 2003 上受支持。
- 工具 – Microsoft WMI Tools 向开发人员提供构建全新一代管理应用程序和解决方案所需要的工具。它包含大量的文档和工具，以引导您完成从 WMI 访问管理数据的整个过程。

WMI 架构包含以下组件：

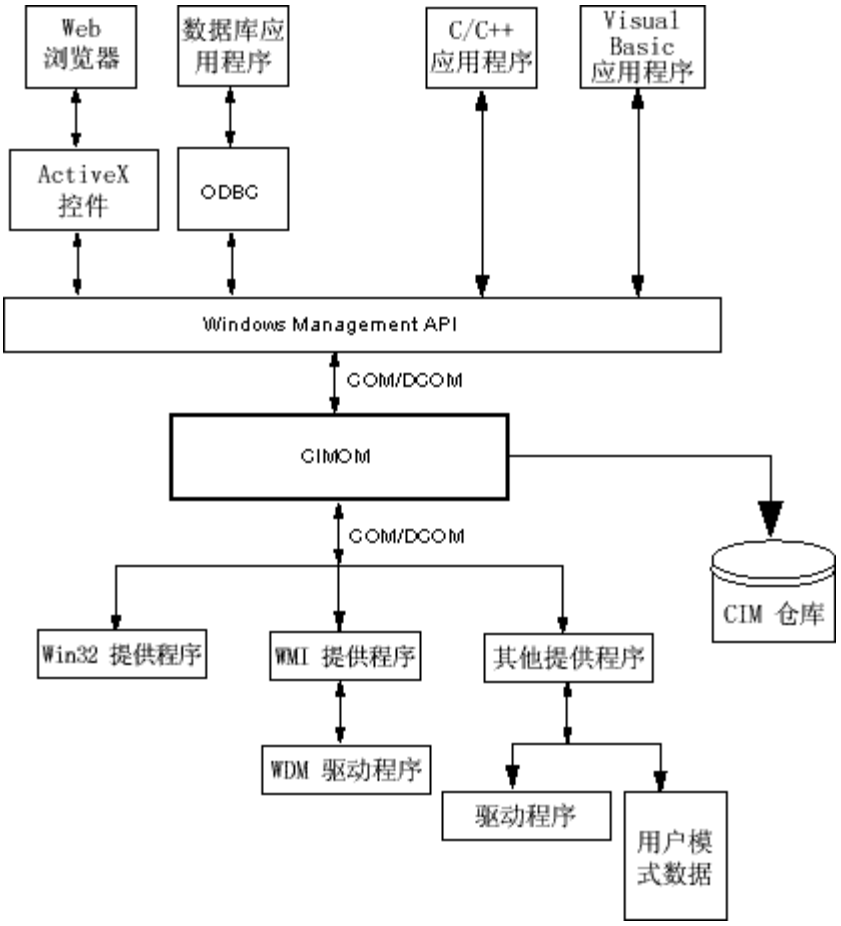
- 管理应用程序
- 被管理的对象
- 提供程序
- 管理体系结构（包含 Windows Management 和 Windows Management 库）
- Windows Management API（使用 COM/DCOM 来启用提供程序和管理应用程序与 Windows Management 体系结构的通讯）。

管理应用程序处理或显示来自被管理的对象（逻辑或物理企业组件）的数据。这些组件均使用 CIM 模型，并由应用程序通过 Windows Management 访问。提供程序使用 Windows Management API 向 Windows Management 提供来自被管理的对象的数据、处理来自应用程序的请求、并生成事件通知。

管理体系结构包含 Windows Management（用于处理管理应用程序和提供程序之间的通讯）和 Windows Management 库（用于存储数据）。Windows Management 库保存静态的管理数据。动态数据仅在提供程序发出请求时生成。可使用 MOF 语言编译器或 Windows Management API 将数据置于库中。

应用程序和提供程序通过 Windows Management 使用提供事件通知和查询处理等服务的 Windows Management API 进行通讯。

以下示图显示 WMI 体系结构组件之间的关系：



返回页首

Common Information Model （共同信息模型）（**CIM** 架构）

Common Information Model (CIM) 为被管理的环境中各类逻辑和物理对象提供稳定统一的外观。被管理的对象由面向对象的结构（如：类）代表。类包含描述数据的属性和描述行为的方法。CIM 由 DMTF 设计为不依赖于操作系统和平台。WBEM 技术包含用于 Microsoft Windows 操作系统平台的一个 CIM 扩展。请参见 DMTF 网站上的“DMTF CIM 架构”以获取更多信息。

CIM 定义三个层次的类：

- 代表应用于管理的所有方面的被管理对象的类。这些类提供用以分析和描述被管理系统的基础词汇，它们是所谓内核模型的一个部分。
- 代表应用于特定的管理方面，但是独立于特定的实现或技术的被管理的对象的类。这些类是所谓共同模型的一个部分。
- 代表作为专用技术添加到共同模型的被管理的对象的类。这些类通常应用到特定的平台，如 UNIX 或 Microsoft Win32 环境，被称为扩展模型。

所有的类都可以通过继承相连，即子类包含来自其父类的数据和方法。继承关系通常对使用它们的管理应用程序并不可见，应用程序也不要求了解继承结构。可通过包含在 WMI 工具的应用程序获取类结构（参见位于 http://www.microsoft.com 的“WMI Tools”获取更多信息）。

Windows Management 也支持关联类。关联类链接两个不同的类以建模一个用户定义的关系，对管理应用程序可见。Windows Management 定义关联类来支持系统类。第三方开发人员还可以为他们的管理环境定义关联类。

WBEM 支持架构概念来组合特定管理环境中使用的类和实例。平台 SDK 包含两个架构：CIM 架构和 Microsoft Win32 架构。CIM 架构包含 CIM 前两个层次的类定义。这些类代表作为每个管理环境（不论何种平台）一部分的被管理的对象。Win32 架构包含作为典型的 Win32 环境一部分的被管理对象的类定义。

有关 CIM 的更多信息，请访问 http://www.dmtf.org。

请阅读所有限制和免责声明。

返回目录页面 返回页首

返回目录页面

主要功能：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

NCS WMI Provider 功能

WMI Provider 的主要功能如下所示：

适配器功能

- 枚举受英特尔(R) PROSet 支持的所有物理适配器。
- 枚举已安装的适配器的设置。
- 添加／删除／更新已安装的适配器的设置。
- 获取适配器的物理设备信息。
- 获取适配器系统插槽设备信息。
- 获取适配器的 IPv4 协议设置。
- 更新和更改适配器的引导代理和关联的设置。
- 卸装适配器。

组功能

- 枚举受英特尔 PROSet 支持的组。
- 创建／删除适配器组。
- 添加／删除／更新组的设置。
- 添加／删除组的成员适配器。
- 获取组的 IPv4 协议设置。

VLAN 功能

- 枚举适配器或组上的虚拟 LAN。
- 创建／删除物理适配器或适配器组上的虚拟 LAN。
- 添加／删除／更新 VLAN 的设置。
- 获取组的 IPv4 协议设置。

事件通知功能

- 允许客户端注册。
  - 适配器状态事件。
  - 适配器配置事件。
  - 会话事件。
  - 组状态事件。
  - 组配置事件。
  - VLAN 配置事件。

返回页首

CDM Provider 功能

CDM Provider 的主要功能如下所示：

- 运行测试、停止测试和清除测试结果而不依赖于诊断测试的类型。
- 使用一般设置类应该允许以 CDM 软件自身无法预测的方式对测试进行控制。

- CDM Provider 仅用于适配器。
- 使用一般结果类解除特定结果消息对 CDM provider 代码的绑定。
- 注册表项控制 provider 的执行。
- 测试结果写入结果日志文件。

请阅读所有限制和免责声明。

返回目录页面 返回页首

返回目录页面

安装文件：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

WMI 文件

可执行文件

以下为 WMI Provider 可执行文件：

- **NcsWmiCo.exe** — 内核 provider。实现 IANet_NetService 和内核事件类。
- **NcsWmiIm.exe** — 实例和方法 provider。实现以太网适配器架构、分组架构、设置架构和 VLAN 架构。
- **NcsWmiEv.exe** — 事件 provider。实现适配器、组和 VLAN 事件。

MOF 文件

有不同的 MOF 文件可用于中性语言和特定语言数据。同时还有不同的 MOF 文件可用于 IntelNCS 和 CIMV2 名称空间。参见 语言地区和本地化和报告错误以获取更多信息。

以下为用于 IntelNCS 名称空间的 MOF 文件：

- **NcsCmLn.mof** — NCS 类所依赖的 CIM 基本类。
- **NcsCmEnu.mfl** — 美国英语版本的 CIM 基本类。
- **NcsCoLn.mof** — 内核 provider 实现的内核类。
- **NcsCoEnu.mfl** — 对内核类的美国英语文本修正。
- **NcsIaLn.mof** — 用于 IEEE 802.3 适配器、组和 VLAN 的类。
- **NcsIaEnu.mfl** — 对 802.3 内核类的美国英语文本修正。

以下为用于CIMV2 名称空间的 MOF 文件：

- **C2CmLn.mof** — NCS 类所依赖的 CIM 基本类。
- **C2CmEnu.mfl** — 美国英语版本的 CIM 基本类。
- **C2CoLn.mof** — 内核 provider 实现的内核类。
- **C2CoEnu.mfl** — 对内核类的美国英语文本修正。
- **C2IaLn.mof** — 用于 IEEE 802.3 适配器、组和 VLAN 的类。
- **C2IaEnu.mfl** — 对 802.3 内核类的美国英语文本修正。

资源文件

以下是 WMI Provider 的资源文件：

- **ENU_8023.dll** — English USA 8023 资源。
- **ENU_NWRC.dll** — 用于内核 provider 的 English USA WMI 资源。
- **ENU_NWR.dll** — 用于 8023 provider 的 English USA WMI 资源。

其他本地化的资源文件可按需加载。本地化的资源 DLL 名称的一般形式是 “`_mwr.dll`”，为本地化的语言代码（例如：FRA 是标准法语）。

返回页首

CDM Provider 文件

可执行文件

以下为 CDM Provider 可执行文件：

- **Ncsdiag.exe** CDM 诊断的主要可执行文件。它符合 Microsoft* WMI 界面规格，作为“进程外” COM 服务器访问。
- 其他可执行文件来自英特尔(R) PROSet 软件堆栈

MOF 文件

主 **.mof** 文件不随产品提供,但是被依照 Microsoft* Windows* Management Instrumentation 全球化模型编译入各基于语言的或中性语言的组件。有关更多信息，请参阅 Microsoft* WMI SDK（平台 SDK 的组件）有关 WMI 本地化的部分。请特别注意编译本地化 *MOF* 文件部分。

删除 **.mof** 文件（DNcsCdmN.mof）将删除英特尔派生的类定义，而不会删除 DMTF 定义的类，否则将危害其他现有应用程序。

此 CDM 实现的典型用法基于 CIMV2 名称空间。以下为用于 IntelNCS 名称空间的 MOF 文件：

文件名	语言类型	说明
CdIa.mof	主	英特尔 CDM 实现的类定义
CdIaLn.mof	中性语言	英特尔 CDM 实现的类定义
CdIaEnu.mfl	依赖英语语言	英特尔 CDM 实现的语言扩展
CdCm.mof	主	内核超集 CDM 类定义
CdCmLn.mof	中性语言	内核超集 CDM 类定义
CdCmEnu.mfl	依赖英语语言	内核超集 CDM 类定义的语言扩展
DNcsCdmN.mof	不适用	删除英特尔 CDM 类

以下为用于 CIMV2 名称空间的 MOF 文件：

文件名	语言类型	说明
C2Icd.mof	主	英特尔 CDM 实现的类定义
C2IcdLn.mof	中性语言	英特尔 CDM 实现的类定义
C2IcdEnu.mfl	依赖英语语言	英特尔 CDM 实现的语言扩展
C2Cd.mof	主	内核超集 CDM 类定义
C2CdLn.mof	中性语言	内核超集 CDM 类定义
C2CdEnu.mfl	依赖英语语言	内核超集 CDM 类定义的语言扩展
DNcsCdm2.mof	不适用	删除英特尔 CDM 类

注意： 本地化要求添加正确的基于语言的 **.mof** 文件。

资源文件

以下是 CDM Provider 的资源文件：

- **ENU_Diag.dll** — 用于 Diagnostic Provider 的 English USA WMI 资源。

请阅读所有限制和免责声明。

返回目录页面 返回页首

返回目录页面

安全性：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

WMI 和 CDM Provider 使用客户端冒名顶替来管理安全性。每个对 Provider 的调用都在客户端自己的安全性环境中发出，然后被传递到下面各层。如果您没有目标计算机的管理员权限，一个或所有操作将可能失败。

请阅读所有限制和免责声明。

返回目录页面 返回页首

名称空间和环境：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

CIM 类位于一个名称空间中。标准的 Microsoft* 名称空间称为 **root/cimv2**；它基于 CIM v2.2 或 **root/default**。可将 WMI 和 CDM Provider 类添加到此名称空间中。这些 Provider 基于 CIM v2.5。因为这一点，以及对象中关键字的用法的不同，Provider 的类位于一个不同的名称空间 **root/IntelNCS** 中。

WBEM 环境

环境对象向 Provider 提供不能作为参数传递给 WMI API 方法的额外信息。若要注册环境限定符，使用 **IWbemContext** 来注册环境限定符。该环境对象的界面指针作为 **IWbemServices** 方法的最后一个参数传递。

以下表格包含 Provider 使用的环境限定符（命名值）。绝大多数限定符（如 SessionHandle）仅和 Provider 的特定功能范围一同使用；而 LocaleID、MachineName 和 ApplicationName 可为所有 **IWbemServices** 调用设置。

如果没有环境被传递到 Provider，它们将使用在 **Initialize** 调用中传递到 Provider 的 LocaleID。任何环境中的读取在写入操作被执行之前，将读取现有配置。随后的读取向系统所显示的信息和在写入操作完成以后的信息一样。NULL 环境可被用于读取。

环境限定符	变异类型	说明
SessionHandle	VT_BSTR	识别应用程序的 IANet 网络类副本。应用程序在建立会话句柄之前不能更改类或其属性。请参见 IANet_NetService 类部分以了解如何建立和使用会话句柄。如果应用程序将仅从类中读取数据，则不要求此限定符。此会话句柄允许 NCS 软件管理对配置的同时多个会话，而不会导致一个用户将所有其他用户排除在外。每次会话都有一个分开的缓存，用以存储所作的任何更改。如果有多个用户在同时进行更改，则第一个应用其更改的用户将成功。 其他所有用户的缓存将都失效。
LocaleID	VT_BSTR	Microsoft 的语言地区 ID。如果应用程序向 Provider 要求本地化的文本字符串，则要求此 ID。所有的错误消息和警告将都由英语显示，除非使用要求的 LocaleID。
ApplicationName	VT_BSTR	发出调用的应用程序的名称。这是进行记录所要求的。
MachineName	VT_BSTR	连接到 Provider 的计算机名称。这是记录日志所要求的。
PreCheck	VT_BOOL	此布尔值用以告诉 Provider 以下信息：客户端在执行一项操作之前试图验证是否允许此项操作。例如：向组中添加适配器。 值： • TRUE = Provider 将不执行此项操作，但是如果此项操作不允许执行,它将返回一个错误代码以及扩展的状态。 • FALSE = Provider 将执行操作。 如果限定符丢失，其后果将和属性为 FALSE 时一样。
WarningErrorCode	VT_I4	一些操作可能要求将警告发送给用户（例如：在某些情况下向组中添加适配器可能会要求重新载入组）。WMI 不提供这一机制。如果此限定符存在，而且不为零，Provider 将在操作成功后返回 E_FAIL，但是会有一个关联的警告。客户端应该使用扩展的状态来获取警告文本。

返回目录页面

语言地区（**Locale**）和本地化：英特尔**(R) PRO** 网络适配器和 **WMI** 和 **CDM Provider** 用户指南

WMI 和 CDM Provider 的本地化有两个方面 — 本地化的 MOF 文件和本地化的属性数据。

本地化的 **MOF** 文件

Provider （提供程序）使用的所有 MOF 文件都按照 Microsoft Windows* Management Instrumentation (WMI) 全球化模式予以本地化。要实现这一点，各类的定义按照以下所列予以区分：

- 一个语言中性的版本仅包含 **.mof** 文件中的基础类定义。
- 一个特定语言的版本包含本地化信息，例如对相应的 **.mfl** 中的一个语言地区特定的属性描述。

支持的语言

中文（台湾）
中文（PRC）
丹麦语
荷兰（荷兰语）
英语（美国）
芬兰语
法语（法国）
德语（德国）
意大利语（意大利）
日语
挪威语（博克马尔语）
葡萄牙语（巴西）
西班牙语（西班牙 – 现代）
瑞典语

类存储

语言特定的类定义存储在包含语言中性基础类定义的名称空间下面的一个子名称空间内。例如，对 WMI 和 CDM Provider，在用于英语语言地区的 **root/Intelncs** 名称空间下有一个子名称空间 **ms_409** 。同样，在 **root/Intelncs** 名称空间下，每一个受支持的语言都有其子名称空间。

cimv2 名称空间中本地化的 **MOF** 支持

对 **root/cimv2** 名称空间，Provider 的类（即：IANet_ 类）均从被 WMI 添加到此名称空间的基本类派生出来。在 **root/cimv2** 名称空间下面，已预先存在一个子名称空间，后者包含基本类的语言特定类定义。IA_Net 特定语言类定义将添加到此现有子名称空间中。基于对基本类的依赖关系，MOF 的本地化仅在默认系统语言地区中进行。

运行时支持

若要检索本地化的数据，WMI 应用程序可使用 **SWbemLocator::ConnectServer** 和 **IWbemLocator::ConnectServer** 调用中的 strLocale 参数来指定语言地区。如果不指定语言地区，则使用该系统的默认语言地区。（例如：美国英语用 MS_409）。此语言地区用以在添加英语字符串时选择正确的名称空间。

此外，**IWbemServices::GetObject**、**SwbemServices.GetObject**、**IWbemServices::ExecQuery** 以及 **SWbemServices.ExecQuery** 必须指定 WBEM_FLAG_USE_AMENDED_QUALIFIERS 标志以请求基本定义和本地化的数据。在所有使用值映射生成可显示值的函数中、显示来自 MOF 文件的描述或其他修正标识符的函数中均有此要求。

返回页首

本地化的属性数据

要获取本地化的属性数据（如错误消息），Provider 需要知道每个调用的调用程序的语言地区。为保证操作正确，客户端必须将语言地区添加到为每个调用所传递的环境对象中（参见 WBEM 环境中的名称空间和环境 ）。如果 Provider 需要返回本地化的字符串，它将试图载入与客户端语言地区相应的资源 DLL。如果没有相应的资源 DLL，则 Provider 将以美国英语返回字符串。

返回目录页面 返回页首

返回目录页面

错误报告： 英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述

关于 IANet_ExtendedStatus 的这一节描述如何处理由 WMI 和 CDM Provider 生成的错误.错误对象在何时以何种方式返回将取决于该调用是同步、半同步、还是异步。在大多数情况下，发生错误时 HRESULT 设置为 WBEM_E_FAILED。但是在那阶段，尚不明了此错误是由 WMI 还是由 Provider 生成。

若要获取同步调用的错误对象，请使用 GetErrorInfo() 以获取 IErrorInfo 对象。使用 QueryInterface() 以获取包含错误消息的 IWbemClassObject。

在获取同步调用的错误对象时，IWbemClassObject 被作为最后一个 SetStatus() 调用的最后一个条目传递。获得错误对象实例之后,可以检查 __Class 属性来确定错误来源。WMI 创建 __ExtendedStatus 的一个实例，而 Provider 则为与 IANet_ classes 有关的错误创建一个 IANet_ExtendedStatus 实例。IANet_ExtendedStatus 从 __ExtendedStatus 派生出来，它包含以下错误对象限定符：

- Description - 专为当前语言地区设计的错误描述。
- File - 错误从其中生成的代码文件。
- Line - 代码文件中出错的行数。
- ParameterInfo - 出错时正在使用的类或属性。
- Operation - 出错时正在试图执行的操作。
- ProviderName - 造成错误的 Provider 的名称。
- StatusCode - 从失败的内部调用返回的代码。
- SessionHandle - 操作使用的会话句柄。
- RuleFailureReasons - 操作失败的原因。操作失败可能因技术规则的失败所致。（例如：在某些组中必须有一个管理适配器）。

返回页首

错误代码

Provider 为每一个错误代码提供一个针对特定语言地区的描述。错误代码的形式为 HRESULT，严重性设为“一”（1），设备设为 ITF。应用程序应该将以下代码作为恢复操作的基础：

- 0x80040901 -“WMI：放置属性失败”
- 0x80040902 -“WMI：无类对象”
- 0x80040903 -“WMI：创建类失败”
- 0x80040904 -“WMI：无法产生类实例”
- 0x80040905 -“WMI：无法创建安全阵列”
- 0x80040906 -“WMI：无法放置安全阵列”
- 0x80040907 -“WMI：无法向 WMI 返回对象”
- 0x80040908 -“WMI：获取属性失败”
- 0x80040909 -“WMI：获取属性时不可预知的类型”
- 0x8004090A -“WMI：类未被此 provider 采用”
- 0x8004090B -“WMI：无法解析 WQL 语句”
- 0x8004090C -“WMI：Provider 仅支持 WQL”
- 0x8004090D -“WMI：环境参数类别错误”
- 0x8004090E -“WMI：调试日志格式化错误”
- 0x8004090F -“WMI：错误对象路径”
- 0x80040910 -“WMI：无法更新设置”
- 0x80040911 - “WMI：空参数被传递到方法”

- 0x80040912 - “设置值太小。”
- 0x80040913 - “设置值太大。”
- 0x80040914 - “步骤中无设置”
- 0x80040915 - “字符串设置太长”
- 0x80040916 - “设置不是以下允许值之一”
- 0x80040917 -“WMI：找不到限定符”
- 0x80040918 -“WMI：找不到限定符集”
- 0x8004090B -“WMI：安全阵列访问失败”
- 0x8004091A -“WMI：未解决的异常”
- 0x8004091B -“WMI：操作不受此类支持”
- 0x8004091C -“WMI：意外的事件类”
- 0x8004091D -“WMI：错误事件数据”
- 0x8004091E -“WMI：操作成功，带有警告”
- 0x8004081F -“WMI：NCS 服务被中止。”

- 0x80040801 -“EAL：内部异常”
- 0x80040802 -“EAL：一般故障”
- 0x80040803 -“EAL：未初始化”
- 0x80040804 -“EAL：初始化失败。”
- 0x80040805 -“EAL：超出会话限制”
- 0x80040806 -“EAL：内存不足”
- 0x80040807 -“EAL：规则语法错误”
- 0x80040808 -“EAL：意外的列表结束”
- 0x80040809 -“EAL：规则链接错误”
- 0x8004080A -“EAL：设备创建失败”
- 0x8004080B -“EAL：找不到媒体服务”
- 0x8004080C -“EAL：找不到设备服务”
- 0x8004080D -“EAL：找不到 PCI 总线模块”
- 0x8004080E -“EAL：适配器为组成员”
- 0x8004080F -“EAL：创建规则接入点错误”
- 0x80040810 -“EAL：注册表主键错误”
- 0x80040811 -“EAL：注册表 XML 文件路径错误”
- 0x80040812 -“EAL：未知事件类”
- 0x80040813 -“EAL：未知模块 ID”
- 0x80040814 -“EAL：找不到规则服务”
- 0x80040815 -“EAL：空输入指针”
- 0x80040816 -“EAL：规则语法错误”
- 0x80040817 -“EAL：规则失败”
- 0x80040808 -“EAL：设置已组合”

- 0x80040220 -“Sync Layer：删除组失败。”
- 0x80040221 -“Sync Layer：创建 VLAN 失败。”
- 0x80040222 -“Sync Layer：删除 VLAN 失败。”
- 0x80040223 -“Sync Layer：删除适配器失败。”
- 0x80040224 -“Sync Layer：设置更改／创建／删除失败。”
- 0x80040225 -“Sync Layer：参数更改／删除失败。”
- 0x80040226 -“Sync Layer：NetConfig 子系统被锁定。 "
- 0x80040227 -“Sync Layer：正在进行系统更新。请稍后再重试。”
- 0x80040228 -“Sync Layer：适配器被锁定”
- 0x80040229 -“Sync Layer：快闪读取失败。”
- 0x8004022A -“Sync Layer：

- 0x80040210 -“Sync Layer：无效事件偏移”
- 0x80040211 -“Sync Layer：无效输入”
- 0x80040212 -“Sync Layer：无效主键”
- 0x80040213 -“Sync Layer：适配器不是组成员”
- 0x80040214 -“Sync Layer：驱动程序未加载”
- 0x80040215 -“Sync Layer：客户端冒名顶替失败”
- 0x80040216 -“Sync Layer：抓到异常”
- 0x80040217 -“Sync Layer：会话未锁定”

0x80040218 -“Sync Layer：硬件访问层不可用
- 0x80040219 -“Sync Layer：快闪不可用”
- 0x8004021A -“Sync Layer：诊断程序不受支持”
- 0x8004021B -“Sync Layer：诊断程序测试不在运行”
- 0x8004021C -“Sync Layer：Boot Agent 更新不可用”
- 0x8004021D -“Sync Layer：Boot Agent 已损坏。”
- 0x8004021E -“Sync Layer：快闪写入失败。”
- 0x8004021F - "Sync Layer：创建组失败。”
- 0x80040201 - “Sync Layer：初始化失败”
- 0x80040202 -“Sync Layer：无效初始化句柄”
- 0x80040203 -“Sync Layer：会话句柄已存在”
- 0x80040204 -“Sync Layer：会话句柄无效”
- 0x80040205 -“Sync Layer：已达到会话最大数量。”
- 0x80040206 -“Sync Layer：会话锁定句柄已存在”
- 0x80040207 -“Sync Layer：会话锁定句柄无效”
- 0x80040208 -“Sync Layer：会话已锁定”
- 0x80040209 - "Sync Layer：媒体服务模块 ID 无效”
- 0x8004020A -“Sync Layer：高级服务模块 ID 无效”
- 0x8004020B -“Sync Layer：设备服务模块 ID 无效”
- 0x8004020C -“Sync Layer：组件类型 ID 无效”
- 0x8004020D -“Sync Layer：总线接口模块 ID 无效”
- 0x8004020E -“Sync Layer：汇集窗口句柄无效”
- 0x8004020F -“Sync Layer：事件 ID 无效”

- 0x80040401 -“HAM PCI：内存映射地址无效”
- 0x80040402 -“HAM PCI：配置驱动程序加载失败”
- 0x80040403 -“HAM PCI：配置驱动程序版本不匹配”
- 0x80040404 -“HAM PCI：找不到设备插槽”
- 0x80040405 -“HAM PCI：诊断驱动程序加载失败”
- 0x80040406 -“HAM PCI：诊断程序驱动程序版本不匹配”
- 0x80040407 -“HAM PCI：诊断程序驱动程序初始化失败”
- 0x80040408 -“HAM PCI：诊断程序未初始化”
- 0x80040409 -“HAM PCI：诊断程序已初始化”
- 0x8004040A -“HAM PCI：诊断程序已在运行”
- 0x8004040B -“HAM PCI：诊断程序测试不在运行”
- 0x8004040C -“HAM PCI：诊断程序测试被中断”
- 0x8004040D -“HAM PCI：诊断程序无效测试号”
- 0x8004040E -“HAM PCI：诊断程序硬件丢失”
- 0x8004040F -“HAM PCI：诊断程序发送接收初始化失败”

- 0x80040511 -“Media Service：NDIS IO 调用失败”
- 0x80040512 -“Media Service：小型端口未加载”
- 0x8004051B -“Media Service：设备句柄无效”
- 0x8004051C -“Media Service：适配器句柄无效”
- 0x8004051D -“Media Service：组句柄无效”
- 0x8004051E -“Media Service：VLAN 句柄无效”
- 0x8004051F -“Media Service：设备丢失”
- 0x80040520 -“Media Service：设置类型无效”
- 0x80040521 -“Media Service：未知无效对象”
- 0x80040522 -“Media Service：设置句柄无效”
- 0x80040523 -“Media Service：组模式无效”
- 0x80040525 -“Media Service：设置已存在”

- 0x80042001 -“RAP：已初始化”
- 0x80042002 -“RAP：XML 文件无效”
- 0x80042003 -“RAP：XML 加载失败”
- 0x80042004 -“RAP：未初始化”
- 0x80042005 -“RAP：规则先前未提取”
- 0x80042006 -“RAP：条件计数不匹配”
- 0x80042007 -“RAP：应用结果失败”

0x80042008 -“RAP：无效规则”
- 0x80042009 -“RAP：找不到节点”
- 0x8004200A -“RAP：错误无单一节点”
- 0x8004200B -“RAP：无行动规则”
- 0x8004200C -“RAP：零条件”
- 0x8004200D -“RAP：零行动”
- 0x8004200E -“RAP：XML 译码错误”

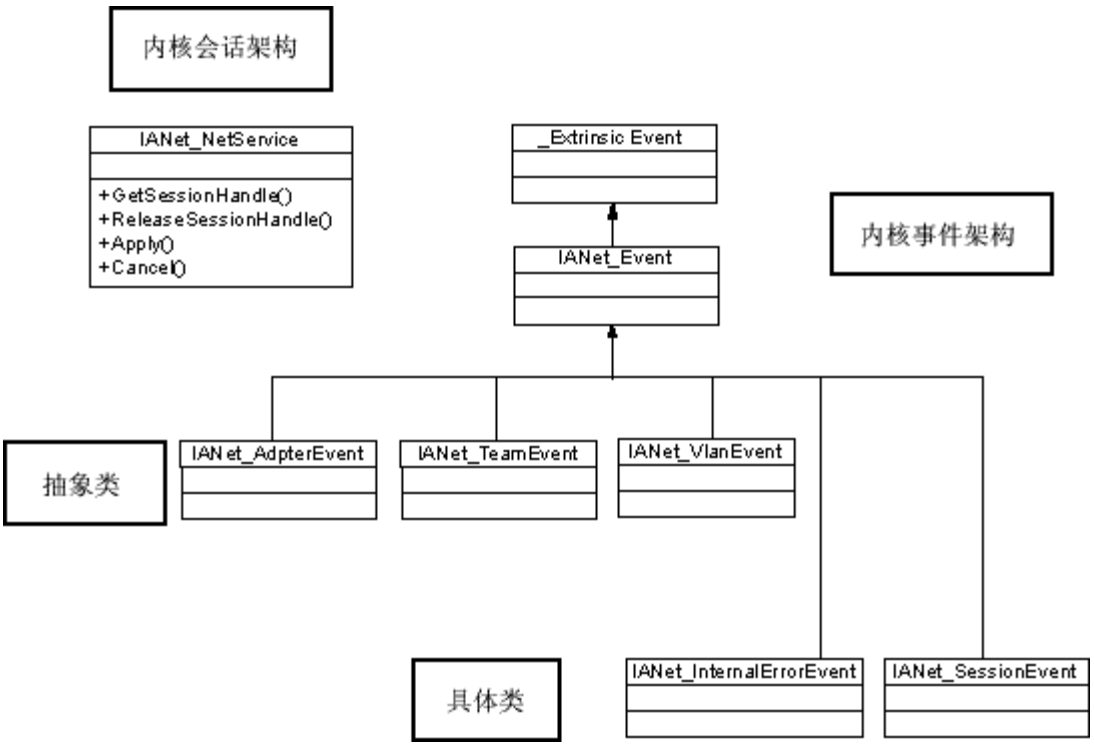
返回目录页面 返回页首

返回目录页面

内核架构： 英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述

内核架构包含 IANet_NetService 类和内核事件类。



返回页首

IANet_NetService

目的
IANet_NetService 类是 IANet_ schema 的根对象。此类使客户端能够访问执行设置所需要的会话。

实例
此对象有一个实例。客户端不应依赖用于此类的主键。而客户端应通过枚举 IANet_NetService 的所有实例以获取类的实例。

创建实例
您不能创建 IANet_NetService 的实例。

删除实例
您不能删除 IANet_NetService 的实例。

修改属性
此类中没有可供用户修改的属性。

受支持的属性
此类实现两个属性：

- Version – 包含内核 provider 的当前版本。
- InstallDate – 包含 provider 的安装日期。

方法
可用以下方法来管理会话：

- **void GetSessionHandle(([OUT] string SessionHandle, [out] uint32 ActiveSessions) —** 用于设置会话句柄字符串，应将其置于 SessionHandle 限定符的环境对象中。ActiveSessions 返回此系统活动会话的数目。这将允许客户端发出警告：可能有他方在修改网络设置。
- **void Apply([IN] string sSessionHandle, [OUT] uint32 FollowupAction);** – 应用由特定会话句柄所作的更改。返回的 uint32 参数被 WMI 和 CDM Provider 用来告诉应用程序必须重新引导方能使更改生效。这通过在 Win32_OperatingSystem 类上调用 **Reboot** 方法实现。

  值：
  1 = 要求系统重新引导
  0 = 不要求重新引导

- **void ReleaseSessionHandle ([IN] string SessionHandle)** – 在会话句柄被用过之后将其释放。此会话中的所有更改都将丢失。会话句柄在这一调用之后将失效，不能再使用。
- **void Cancel([IN] string SessionHandle);** – 取消会话。内部高速缓存将被清理；在此次调用之后所读取的所有数据将显示在当前配置中。

返回页首

内核事件

IANet_SessionEvent

目的
此事件用于通知客户端有关使用 NCS 会话 API 的信息。客户端能使用此事件来了解其他客户端是否在创建或使用会话。

触发者
客户端在创建会话、删除会话或调用 **Apply** 会话时会触发此事件。

事件数据
EventType 可取以下值中的一个：

- “New session” 表明此客户端或另一个客户端已创建一个新的会话。
- “End session” 表明客户端已经结束一次会话。此会话可由此客户端或另一个客户端结束。
- “Cache invalidated” 表明另一个客户端已对一个会话调用 **Apply**。所有其他会话均为无效，与其会话关联的缓存均已被删除。
- “Configuration changed” 表明会话的配置已被更改。

SessionHandle 包含触发事件的会话句柄。

OpenSessions 包含开放会话的数目。此数据项目对“Cache invalidated”和“Configuration changed”事件为 NULL。

IANet_InternalErrorEvent

目的
此事件用于通知客户端在事件 provider 中发生内部错误。在有些情况下，这意味着事件 provider 无法进一步报告事件。

触发者
此事件将在以下情况发生：

- 在事件 provider 从事件源获取未知事件以后
- 在提供事件的软件被关闭之后
- 在事件 provider 获取事件但事件源无法获取有关事件的进一步数据之后

事件数据
EventType 可为以下中的一个：

- “Could not get event data”发生一项事件，但是事件源无法获取有关事件的进一步数据。
- “Event source has shut down”。事件的数据源已关闭。在此示例中，事件 provider 也将关闭，而且在重新启动源并且执行新的通知查询之前将不会生成更多事件。
- “Unexpected message”。事件 provider 接收到意外的事件。

返回页首

用法示例

更改配置要求会话句柄。此会话句柄允许 NCS 软件管理对配置的同时多个访问，以防该会话将所有其他会话排除在外。每次会话都有一个分开的缓存，用以存储所作的任何更改。如果有多个会话在同时进行更改，则第一个应用其更改的会话将成功。其他所有会话将都为无效。

获取会话句柄

客户端必须在访问会话句柄之前获取 IANet_NetService 的单一实例的对象路径。调用 **IWbemServices::CreateInstanceEnum** 并传递类名称：IANet_NetService。这等于以查询 **SELECT * FROM IANet_NetService** 调用 **IWbemServices::ExecQuery**。在对配置进行更改之前，客户端必须获取一个会话句柄。使用 **GetSesssionHandle** 方法用以开始全新的会话。

客户端可以使用 **IWbemServices::ExecMethod** 对 CIM 对象执行一个方法，并需要从 IANet_NetService 实例中的 **__PATH** 属性获取该对象路径。此方法还返回当前活动会话的数目。如果客户端对 Network Configuration Service (NCS) 没有排他访问权限，则应发出警告。

在 **IWbemContext** 对象中使用会话句柄

客户端在获取会话句柄之后，必须创建一个 IWbemContext 对象。将会话句柄存储在此对象的 **SessionHandle** 限定符中。此 COM 对象的指针应该被传递到每个对 IWbemService 的调用。在发出调用以访问 IANet_NetService 对象时不要求会话句柄，因为它将句柄作为一个明确的参数。

使用会话句柄读取待决更改

如果在读取配置时传递环境中的会话句柄，则 provider 将返回该配置，就像待决升级已被应用（例如：尚未安装的适配器将丢失，而已更改的设置则返回其新值）。但是，有些对象仅在调用 **Apply** 之后才出现（例如：IANet_IPProtcolEndpoints 仅在协议被绑定到恰当的终点之后才被创建）。

结束会话句柄

更改配置之后，调用 **Apply** 方法来确认更改。这将返回一个追踪动作代码（例如：在重新引导系统之后方能使更改生效）。

在会话结束之后始终调用 **ReleaseSessionHandle** 否则所有更改将被丢弃。调用 **Cancel** 方法也会丢弃所作的全部更改。但是客户端可以继续使用会话句柄，就好像其才被创建。

为内核事件注册

应用程序应该使用 **IWbemServices::ExecNotificationQuery** 或者使用 **IWbemServices::ExecNotificationQueryAsync** 来要求事件通知。以下查询是事件通知查询的示例（因为查询的种类为数很多，此列表并非无所不包）：

- **SELECT * FROM IANet_Event —** 要求所有事件。

- **SELECT * FROM IANet_SessionEvent** — 要求所有会话事件。
- **SELECT * FROM IANet_InternalErrorEvent** — 要求所有内部事件。

请阅读所有限制和免责声明。

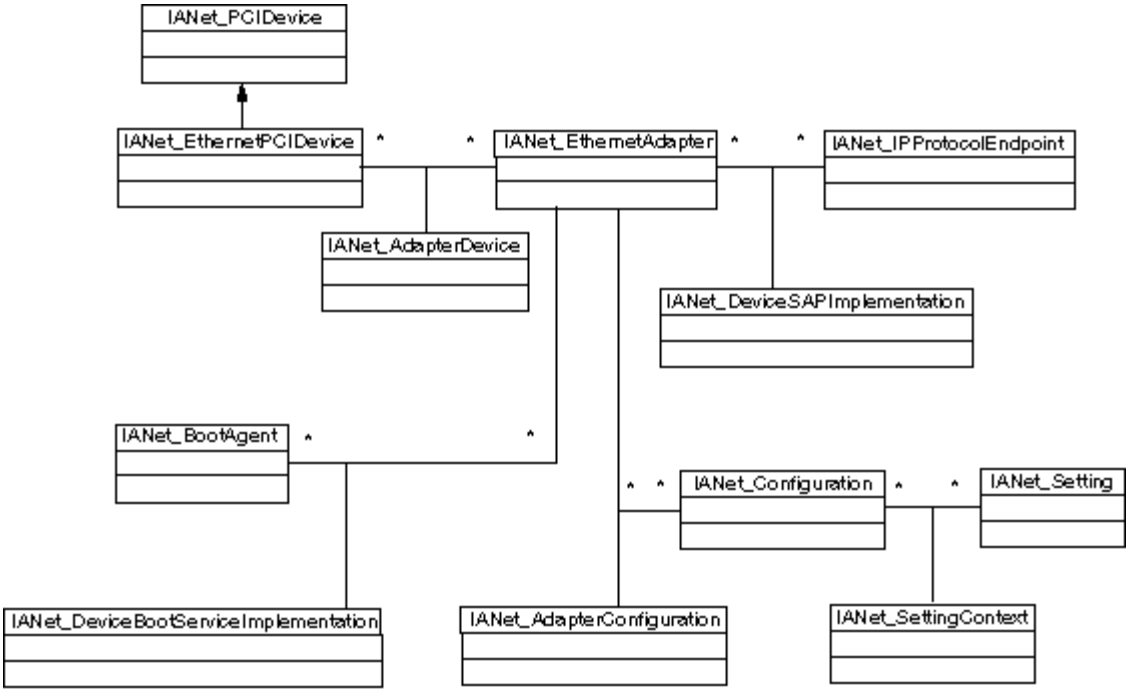
返回目录页面 返回页首

返回目录页面

以太网适配器架构： 英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述

适配器架构用于为各种可配置英特尔(R) PROSet 以太网适配器建模。此架构基于 CIM v2.5 架构。



返回页首

IANet_EthernetAdapter

目的
IANet_EthernetAdapter 定义所有安装的英特尔 PRO 网络适配器以及其他任何能用英特尔中层驱动程序组合的适配器的性能和状态。此类从在 CIMv2.5 中定义的 CIM_EthernetAdapter 超类派生出来。CIM_EthernetAdapter 从 CIM_NetworkAdapter 中派生出来；CIM_NetworkAdapter 是定义一般联网硬件概念（如：PermanentAddress、CurrentAddress、Speed of operation，等）的抽象类。

实例
此类的实例存在于以下各项中：

- 受支持的安装的英特尔 NIC
- 能加入英特尔多生产商组的非英特尔 NIC

创建的英特尔适配器组

创建实例
不能创建 IANet_EthernetAdapter 的实例。

删除实例
删除 IANet_EthernetAdapter 实例将卸装物理适配器。只有非虚拟英特尔适配器可以此方法卸装。此项操作要求会话句柄。

修改属性
在此类中没有可供用户修改的属性。

不受支持的属性
使用英特尔 PROSet 不要求以下属性，因此它们不受支持。

- AutoSense （它作为一项设置显露）
- ErrorCleared
- OtherIdentifyingInfo
- IdentifyingDescriptions
- InstallDate
- LastErrorCode
- MaxDataSize
- MaxQuiesceTime
- PowerManagementCapabilities （它作为一个方法显露）
- PowerManagementCapabilities （它作为一个方法显露）
- PowerOnHours
- ShortFramesReceived
- SymbolErrors
- TotalPowerOnHours

方法
此类的实例支持以下方法：

- **IdentifyAdapter —** 通过在适配器上闪烁指示灯（几秒钟）来识别适配器。这一方法仅对物理适配器有用。
- **HasVLANs —** 返回此适配器上的 VLAN 数。
- **IsPowerMgmtSupported —** 指出电源管理是否在适配器上受支持。
- **GetPowerUsage —** 检测适配器的总耗电量。
  0 = 正常耗电
  1 = 低耗电
- **SetPowerUsage —** 降低适配器的总耗电量。用电设置在系统重新启动时以及驱动程序重新载入时不予保存。在系统重新启动或者驱动程序重新载入时，适配器自动返回正常耗电。
- **GetPowerUsageOptions —** 检测任何可选的用电设置（例如,待命状态的用电、电池操作等）。
- **SetPowerUsageOptions —** 更改用电选项（例如，可用方法来降低待命状态和电池操作等的用电量）。
  注意： 用电设置被存储后用于后随的重新引导。
- **TestCable —** 在特定适配器上进行诊断测试。遇到故障时，此方法返回可能的问题、原因和解决方案。
- **AdvancedTestCable —** 在特定适配器上进行高级电缆测试。此测试集对 1000 Mbps 适配器可用。此方法返回各项测试的名称及其各自的结果。
  注意：如果 **SpeedDuplex** 未设置为 **Auto Negotiate**，则可能出现链接故障。在此实例中，**SpeedAndDuplexNotAutomatic**的 Out 参数为 TRUE。
- **TestLinkSpeed —** 确定适配器是否在以全速运行.如果适配器广告速度低于 1 千兆位，此方法将陈述可能的原因（例如，“链接伙伴无能力以 1000 Mbps 速度运行”）。

返回页首

IANet_IPProtocolEndpoint

目的
此类用于描述系统中一个协议终点的 IP 设置。WMI Provider 不提供任何其他类型的联网协议的信息。此类从 CIM_IPProtocolEndpoint 的抽象类派生出来。WMI Provider 仅在对由英特尔 PROSet 管理的实体关注时才提供协议信息。

Instances

对每个绑定到受英特尔支持的终点（例如，英特尔适配器、英特尔可分组的适配器、以及 VLAN 等）的 IP 协议层，都有一个 IANet_IPProtocolEndpoint 实例存在。一些分组的适配器没有自己的 IP 地址，因此也就没有 IANet_IPProtocolEndpoint 直接与它们的适配器实例关联。 IANet_IPProtocolEndpoint 仅在操作系统将协议绑定到适配器或 VLAN 之后才存在。 虽然一些适配器可能会有一个以上 IP 地址，它们将仅与一个 IP 协议终点实例关联。Provider 不支持这一高级用法，因为英特尔 PROSet 并不要求、也不使用它。

创建实例
不能创建 IANet_IPProtocolEndpoint 的实例。此实例仅在操作系统将协议绑定到终点之后才存在。

删除实例
不能删除 IANet_IPProtocolEndpoint 的实例。

修改属性
在此类中没有可供用户修改的属性。

关联
IANet_AdapterProtocolImplementation 实例用于使 IANet_EthernetAdapter 和 IANet_IPProtocolEndpoint 关联。IANet_VLANProtocolDependency 实例用于使 VLAN 和 IANet_IPProtocolEndpoint 关联。

注意： 各个组均通过代表组中虚拟适配器的适配器与终点关联。

受支持的属性
以下只读属性是英特尔 PROSet 要求的：

- Address
- AddressType
- DefaultGateway
- DHCPServerAddress
- DHCPAutoAssign
- IPVersionSupport
- SubnetMask

不受支持的属性
使用英特尔 PROSet 不要求以下属性，因此它们不受支持。

- Caption
- Description
- InstallDate
- NameFormat
- OtherTypeInformation
- ProtocolType
- Status

方法
无。

返回页首

IANet_BootAgent

目的
此类用于捕获有关适配器网络引导性能的信息（例如，受一些英特尔适配器支持的 PXE Boot Agent 的设置等）。此类从 CIM_BootService 派生出来。

实例
对每个支持引导代理功能的适配器（即使该引导代理当时未被安装），都有一个 IANet_BootAgent 实例存在。

创建实例
不能创建 IANet_BootAgent 的实例。此类仅在适配器支持引导代理功能的情况下才存在。

删除实例
不能删除 IANet_BootAgent 的实例。

修改属性
在此类中没有可供用户修改的属性。

关联
IANet_DeviceBootServiceImplementation 的实例用于使 IANet_EthernetAdapter 和 IANet_BootAgent 关联，如果受适配器支持的话。

受支持的属性
以下只读属性是英特尔 PROSet 要求的：

- InvalidImageSignature
- Version
- UpdateAvailable
- FlashImageType

不受支持的属性
英特尔 PROSet 不要求以下属性，因此它们不受支持。

- Caption
- Description
- InstallDate
- Started
- StartMode
- Status

方法
此类的以下方法能用于更新 NIC 的快闪 ROM：

uint32 ProgramFlash([IN, ValueMap {"0","1"} , Values {"Check Version", "Write Flash"}:Amended uint32 Action, [IN] uint8 NewFlashData[], [OUT] string strErrorMessage);	此方法用于更新 NIC 上的快闪 ROM。这将使 NIC 在更新快闪的时候停止与网络的通讯。
uint32 ReadFlash([OUT] uint8 FlashData[]);	此方法读取 NIC 上的快闪 ROM。

返回页首

IANet_PCIDevice

目的
此类用于描述系统中一个网络设备的 PCI 设备属性。此类从 CIM_PCIDevice 派生出来。

实例
对系统中的每个作为网络设备的 PCI 卡，都有一个此类的实例存在。对 IA64，只有作为受英特尔 PROSet 支持的适配器的 PCI 设备才有实例。

创建实例
不能创建 IANet_PCIDevice 的实例。

删除实例
不能删除 IANet_PCIDevice 的实例。

修改属性
在此类中没有可供用户修改的属性。

关联
请参见 IANet_EthernetPCIDevice 以了解类关联。

方法
此类没有受支持的方法。

不受支持的属性
以下属性不受 WMI 支持：

- AdditionalAvailabitlity
- Capabilities
- CapabilityDescriptions
- Caption
- DeviceSelectTiming
- ErrorCleared
- ErrorDescription
- IdentifyingDescription
- InstallDate
- LastErrorCode
- MaxNumberController
- MaxQuiesceTime
- Name
- OtherIdentifyingInfo
- PowerManagementCapabilities
- PowerManagementSupported
- PowerOnHours
- ProtocolDescription
- ProtocolSupported
- SelfTestEnabled
- TimeOfLastReset
- TotalPowerOnHours

返回页首

IANet_EthernetPCIDevice

目的
此类用于描述受英特尔 PROSet 支持的以太网适配器的 PCI 设备属性。这是 IANet_PCIDevice 的一个子类。此类包含某些仅知用于英特尔 PROSet 支持的 PCI 设备的额外属性。

实例
对每个作为受英特尔 PROSet 支持的以太网适配器的 PCI 卡，都有一个此类的实例存在。

创建实例
不能创建 IANet_EthernetPCIDevice 的实例。

删除实例
不能删除 IANet_ EthernetPCIDevice 的实例。

修改属性
在此类中没有可供用户修改的属性。

关联
instance IANet_AdapterDevice 的实例用于使 IANet_PCIDevice 和 IANet_EthernetAdapter 关联。虚拟适配器（即，被创建以代表组的适配器）没有关联的 IANet_PCIDevice。

不受支持的属性
以下属性不受 WMI 支持：

- AdditionalAvailabitlity
- Capabilities
- CapabilityDescriptions
- Caption
- DeviceSelectTiming
- ErrorCleared
- ErrorDescription
- IdentifyingDescription
- InstallDate
- LastErrorCode
- MaxNumberController
- MaxQuiesceTime
- Name
- OtherIdentifyingInfo
- PowerManagementCapabilities
- PowerManagementSupported
- PowerOnHours
- ProtocolDescription
- ProtocolSupported
- SelfTestEnabled
- Status
- StatusInfo
- TimeOfLastReset
- TotalPowerOnHours

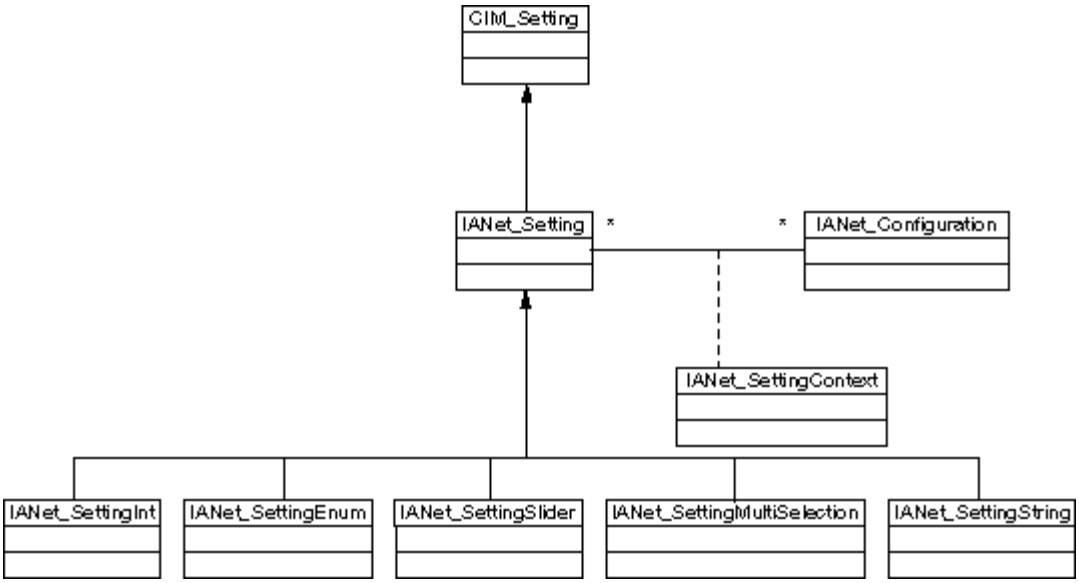
方法
此类没有受支持的方法。

返回目录页面 返回页首

返回目录页面

设置架构：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述



返回页首

IANet_Configuration

目的
此类用于组合一集 IANet_Setting 实例。此类从 CIM_Configuration 派生出来。

实例
每个适配器、VLAN、或组都可以有若干关联的 IANet_Configuration 实例（每个配置与该适配器的一个不同的使用情形相配）。

对本次 WMI 和 CDM Provider 发行，每个适配器、VLAN、或组仅有一个 IANet_Configuration 实例。

创建实例
不能创建 IANet_Configuration 的实例。

删除实例
不能删除 IANet_Configuration 的实例。

修改属性
在此类中没有可供用户修改的属性。

关联
IANet_AdapterConfiguration 实例将使每个适配器（IANet_EthernetAdapter）与其配置相关联。IANet_VLANConfiguration 实例使每个 VLAN（IANet_VLAN）与其配置相关联。IANet_BootAgentConfiguration 实例使每个引导代理（IANet_BootAgent）与其配置相关联。

方法
此类没有受支持的方法。

不受支持的属性
无。

返回页首

IANet_Setting

目的
此抽象类用于描述配置中一个可设置的属性。此类从 CIM_Setting 派生出来。

实例
对每个适配器、VLAN、或组的每个设置，都各有一个此类的实例存在。各个配置的设置不可共享。

IANet_Setting 有几个不同的子类。子类与该设置的不同类型和取值范围相应。每个子类都和一个可用来显示或更改设置的不同类型的 GUI 相应。

创建实例
不能创建 IANet_Setting 的实例。

删除实例
不能删除 IANet_Setting 的实例。

修改属性
此抽象类带有不可修改的属性，但是子类有可修改的属性（参见下文）。

关联
每个 IANet_Setting 的实例都使用 IANet_SettingContext 实例与 IANet_Configuration 实例关联。

方法
此类没有受支持的方法。若要更改一个设置，修改要求的属性，并调用 **PutInstance**。

不受支持的属性
不用 SettingID。

返回页首

IANet_SettingInt

目的
此类为取整数值的设置提供模型。由若干个 IANet 设置类被用于为整数提供模型。这些类之间的不同之处与 GUI 显示和修改整数的方式以及 Provider 验证的方式有关。对 IANet_SettingInt，应该预期 GUI 以旋转控件显示编辑框。

实例
对每个应作为整数编辑框显示的设置，都有此类的一个实例存在。

创建实例
不能创建此类的实例。

删除实例
不能删除此类的实例。

修改属性
“CurrentValue” 属性是此类中唯一可以修改的属性。可以通过使用 **IWbemClassObject::Put()** 修改其值来修改此属性，然后调用 **IWbemServices::PutInstance()** 来更新设置。Provider 将检查：

CurrentValue <= **max**
CurrentValue > = **min**
(**CurrentValue** - **min**) 是 **Step** 的倍数，

此处 **max**、**min**、**CurrentValue** 和 **Step** 均为 IANet_SettingInt 的属性。

关联
每个 IANet_SettingInt 的实例都使用 IANet_SettingContext 实例与 IANet_Configuration 实例关联。

不受支持的属性
不用 SettingID。

方法
此类没有受支持的方法。若要更改一个设置，修改要求的属性，并调用 **PutInstance**。

返回页首

IANet_SettingEnum

目的
此类为取整数值的设置提供模型。由若干个 IANet 设置类被用于为整数提供模型。这些类之间的不同之处与 GUI 显示和修改整数的方式以及 Provider 验证的方式有关。对 IANet_SettingEnum，应期待 GUI 将显示一个字符串列表，这些字符串映射到少数几个枚举值（例如，下拉组合列表框）。

实例
对每个将作为 enum 显示的设置，都有一个此实例存在。

创建实例
不能创建此类的实例。

删除实例
不能删除此类的实例。

修改属性
CurrentValue 属性是此类中唯一可以修改的属性。可以通过使用 **Put()** 修改其值来修改此属性，然后调用 **PutInstance()** 来更新设置。Provider 将检查**CurrentValue ∈ PossibleValues[]**

关联
每个 IANet_SettingEnum 的实例都使用 IANet_SettingContext 实例与 IANet_Configuration 实例关联。

不受支持的属性
不用 SettingID。

方法
此类没有受支持的方法。若要更改一个设置，修改要求的属性，并调用 **PutInstance**。

返回页首

IANet_SettingSlider

目的
此类为取整数值的设置提供模型。由若干个 IANet 设置类被用于为整数提供模型。这些类之间的不同之处与 GUI 显示和修改整数的方

式以及 Provider 验证的方式有关。对 IANet_SettingSlider，应期待 GUI 将显示一个允许您以图形方式选择值的滑杆（被选的具体值不一定要显示）。

实例
对每个将作为滑杆显示的设置，都有一个此实例存在。

创建实例
不能创建此类的实例。

删除实例
不能删除此类的实例。

修改属性
CurrentValue 属性是此类中唯一可以修改的属性。可以通过使用 **Put()** 修改其值来修改此属性，然后调用 **PutInstance()** 来更新设置。Provider 将检查**CurrentValue Є PossibleValues[]**

关联
每个 IANet_SettingSlider 的实例都使用 IANet_SettingContext 实例与 IANet_Configuration 实例关联。

不受支持的属性
不用 SettingID。

方法
此类没有受支持的方法。若要更改一个设置，修改要求的属性，并调用 **PutInstance**。

返回页首

IANet_SettingMultiSelection

目的
此类为您可从选项表选择若干选项的设置提供模型。对 IANet_SettingMultiSelection，应期待 GUI 将显示允许您选择（或者不选择）任何选项的多项选择列表框。

实例
对每个将作为多项选择显示的设置，都有一个此实例存在。

创建实例
不能创建此类的实例。

删除实例
不能删除此类的实例。

修改属性
CurrentValue 属性是此类中唯一可以修改的属性。可以通过使用 **Put()** 修改其值来修改此属性，然后使用 **PutInstance()** 来更新设置。Provider 将检查**CurrentValue Є PossibleValues[]**。

关联
每个 IANet_SettingMultiSelection 的实例都使用 IANet_SettingContext 实例与 IANet_Configuration 实例关联。

不受支持的属性
不用 SettingID。

方法
此类没有受支持的方法。若要更改一个设置，修改要求的属性，并调用 **PutInstance**。

返回页首

IANet_SettingString

目的

此类为一个您能在其中输入一个无格式字符串值的设置提供模型。对 IANet_SettingMultiSelection，应期待 GUI 将显示一个编辑框。

实例
对每个应作为编辑框显示的设置，都有一个此实例存在。

创建实例
不能创建此类的实例。

删除实例
不能删除此类的实例。

修改属性
CurrentValue 属性是此类中唯一可以修改的属性。可以通过使用 **Put()** 修改其值来修改此属性，然后调用 **PutInstance()** 来更新设置。

关联
每个 IANet_SettingMultiSelection 的实例都使用 IANet_SettingString 实例与 IANet_ElementConfiguration 实例关联。

方法
此类没有受支持的方法。

不受支持的属性
不用 SettingID。

方法
此类没有受支持的方法。若要更改一个设置，修改要求的属性，然后调用 **PutInstance**。

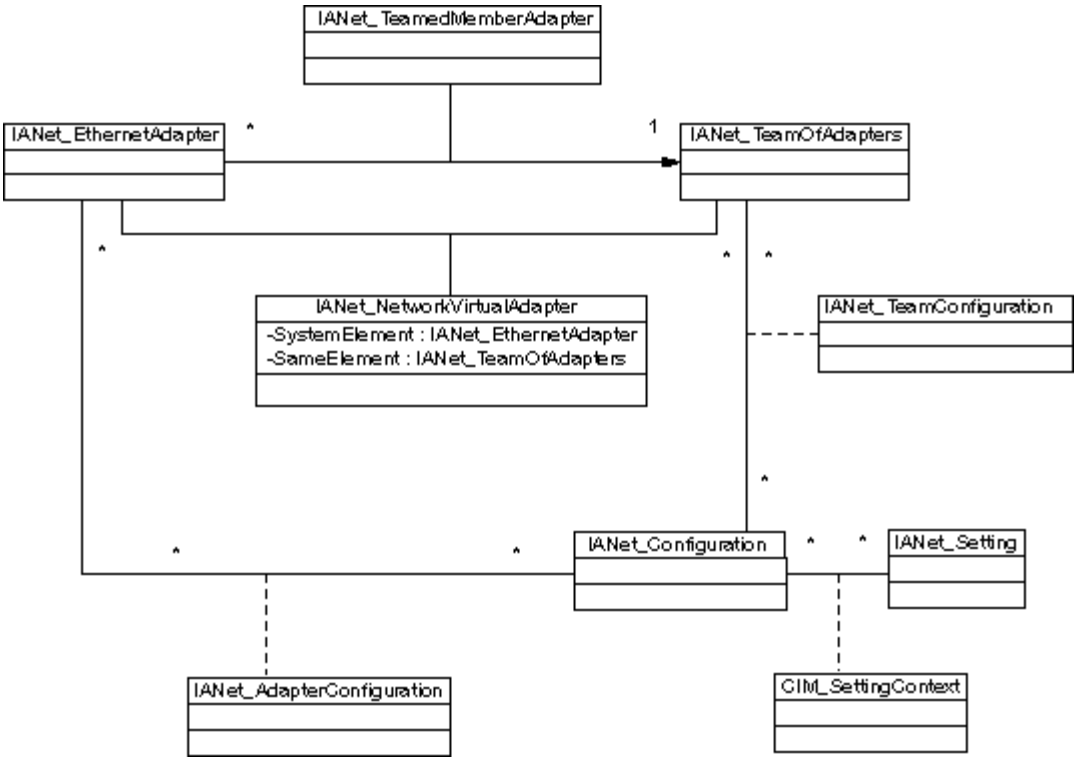
返回目录页面 返回页首

返回目录页面

组架构：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述

组架构描述以太网适配器如何结合成组。



返回页首

IANet_TeamOfAdapters

目的
此类实现 CIM_RedundancyGroup 类。此类所含的成员描述组的类型、组中适配器的数目、以及组中能容纳的最多适配器数目。

实例
每个英特尔组均有一个此类的实例。

创建实例
若要创建一个空的组，创建 IANet_TeamOfAdapters 的一个实例。必须先设置正确的 **TeamingMode**，然后再调用 **IWbemServices::PutInstance()** 在提供程序中创建对象。提供程序将返回一个包含新对象的对象路径的字符串。

删除实例
相应地，若要删除一个组，删除 IANet_TeamOfAdapters 的实例。提供程序将删除与组成员的关联，以及该组的虚拟适配器和设置。

修改属性
使用 **Put()** 来更改 **TeamingMode** 属性值，然后调用 **PutInstance()** 来更新组。

关联
组中的每个适配器都使用 IANet_TeamMemberAdapter 实例与该组的 IANet_TeamOfAdapters 实例关联。组的虚拟适配器使用 IA_NetNetworkVirtualAdapter 的实例与此类关联。

方法
此类的实例支持以下方法：

TestSwitchConfiguration — 测试交换器配置以确保组能与交换器正常工作。此测试可用于检查链接伙伴（即适配器与其链接的一个设备，如另一个适配器、集线器、交换器等）是否支持所选的适配器分组模式。例如，如果适配器是“链接聚合”组的成员，此项测试可验证连接到该适配器的链接伙伴支持“链接聚合”。

返回页首

IANet_TeamedMemberAdapter

目的
此类用于将适配器与组关联，并确定组中适配器的功能，以及确认配器当前在组中活动。此类实现 CIM 类 CIM_NetworkAdapterRedundancyComponent。

实例
每个作为组成员的适配器都有一个此类的实例。

创建实例
若要将适配器添加到组，创建一个 IANet_TeamedMemberAdapter 实例以将适配器与该组关联。

删除实例
若要从组中删除适配器，删除 IANet_ TeamedMemberAdapter 实例。适配器将不再是该组的一部分，并可在调用 **Apply()** 之后与一个 IP 协议终点绑定。

修改属性
可以修改此类的 **AdapterFunction** 属性以描述此适配器在组中如何使用。

关联
这是一个关联类。

方法
此类没有受支持的方法。

返回页首

IANet_NetworkVirtualAdapter

目的
此类用于将组的 IANet_TeamOfAdapters 与代表组中虚拟适配器的 IANet_EthernetAdapter 关联。此类实现 CIM 类 CIM_ CIM_NetworkVirtualAdapter。

实例
每个绑定到虚拟适配器的英特尔组都有此类的一个实例。

创建实例
不能创建此类的实例。若要创建一个组，创建 IANet_TeamOfAdapters 的一个实例。此类仅在有效会话的环境中调用 **IANet_NetService .Apply()** 并创建了 IANet_EthernetAdapter 实例后才会存在。

删除实例
不能删除此类的实例。

关联

这是一个关联类。

方法
此类没有受支持的方法。

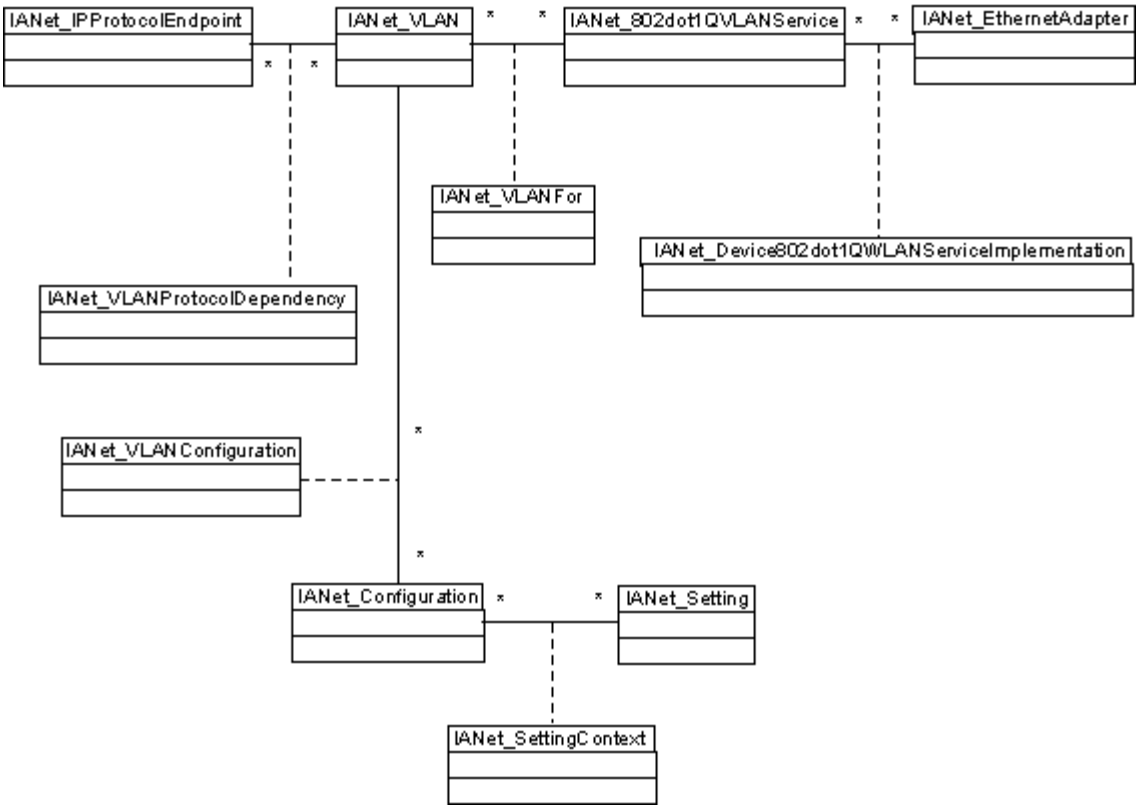
请阅读所有限制和免责声明。

返回目录页面 返回页首

返回目录页面

VLAN 架构：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述



返回页首

IANet_802dot1QVLANService

目的
此类用于保存网络适配器的 IEEE 802.1Q 属性。此类实现 CIM 类 CIM_802dot1QVLANService。

实例
每个支持 IEEE 802.1Q 的组和适配器都有一个此类的实例。每个适配器仅可有一个 IANet_802dot1QVLANService。有些组，如果多生产商容错组等，不支持此项服务。

异常情况
不带有 VLAN 的组将没有 VLAN 服务，除非在有效会话的环境中予以枚举。对组来说，802.3QvlanService 实例仅在以下情形中出现：

- 如果组已有 VLAN

- 如果组没有 VLAN，而您在枚举此类时在环境中使用了会话句柄。

创建实例
不能创建此类的实例。如果一个适配器没有一个实例与其关联，则它不支持此项服务。

删除实例
不能删除此类的实例。

修改属性
在此类中没有可修改的属性。

关联
此类的每个实例将使用一个 IANet_DeviceServiceImplementation 实例与 IANet_EthernetAdapter 关联。

每个 IANet_802dot1QVLANService 的实例能支持若干个 VLAN；每个 VLAN 将使用 IANet_VLANFor 关联与此实例关联。

方法
uint16 CreateVLAN([in] uint32 VLANNumber, [in] string Name, [out] IANet_VLAN REF VLANpath); — 用于在适配器或组上创建一个 VLAN。客户端必须提供 VLAN 号码以及 VLAN 名称，并将获取新创建的 VLAN 的对象路径。

返回页首

IANet_VLAN

目的
此类保存每个英特尔 VLAN 的信息。此类实现 CIM_VLAN。

实例
每个英特尔 VLAN 均有一个此类的实例。

创建实例
若要创建一个 VLAN，在恰当的 IANet_802dot1QVLANService 实例上调用 **CreateVLAN** 。

删除实例
可以通过删除此类的一个实例来删除相应的 VLAN。

修改属性
可以修改 VLANNumber 和 Caption（标题）属性。

关联
每个实例均与 IANet_802dot1QVLANService 的一个实例关联，因此对一个 IANet_EthernetAdapter 实例，使用 IANet_VLANFor 类。

每个实例均可与若干 IANet_Configuration 实例关联，以组合 VLAN 的一套设置。对此 Provider 发布，每个 VLAN 仅有一个 IANet_Configuration 对象。

每个实例均可与 IANet_IPProtocolEndpoint 的一个实例关联，为使用 IANet_VLANProtocolDependency 类的 VLAN 提供 IP 设置。

方法
无

返回目录页面 返回页首

返回目录页面

获取当前配置： 英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

客户端读取当前配置不需要会话句柄。客户端能使用 NULL 环境，但是所有错误消息将都以受管理计算机的默认语言返回。在以下表格中，包含在{ }中的条目为对象路径.这些路径都假定从先前的 WQL 查询中获取。客户端不应该在不进行查询的情况下就构建对象路径。每个对象的 **__PATH** 属性包含该对象的对象路径。

在以下所有使用示例中，**IWbemServices::ExecQuery** 或者 **IWbemServices::ExecQueryAsync** 方法都用于执行 WQL 查询。

获取物理适配器

任务	**WQL** 查询	结果类	注释
枚举所有适配器	SELECT * FROM IANet_EthernetAdapter	IANet_EthernetAdapter	返回所有 IANet_EthernetAdapters。 这等于 IWbemServices::CreateInstanceEnumAsync。
确定适配器是否虚拟	ASSOCIATORS OF {适配器路径} WHERE AssocClass = IANet_NetworkVirtualAdapter	IANet_TeamOfAdapters	如果查询结果中没有类，则此适配器为真正的适配器。
确定适配器是否为虚幻适配器	ASSOCIATORS OF {适配器路径} WHERE ResultClass = IANet_EthernetPCIDevice	IANet_EthernetPCIDevice	如果适配器不是虚拟适配器，而且此查询未返回对象，则此适配器为虚幻适配器。

适配器的主类是 IANet_EthernetAdapter。此类用于物理适配器和虚幻适配器二者，而且客户端需要知道如何加以区分。

返回页首

获取 **PCI** 设备

主要的类别为 IANet_EthernetPCIDevice、IANet_PCIDevice、和 IANet_AdapterDevice（用以将一个适配器与其设备联系起来的关联类）。

在此示例中，关联类不包含任何数据，即其本身不具备值。IANet_EthernetPCIDevice 继承于 IANet_PCIDevice，并且包含专门针对以太网适配器 PCI 设备的额外属性。

任务	**WQL** 查询	结果类	注释
枚举网络 PCI 设备	SELECT * FROM IANet_PCIDevice	IANet_PCIDevice	查询可能返回英特尔(R) PROSet 不予配置的调制解调器和其他设备。 这等于

			IWbemServices::CreateInstanceEnumAsync。
枚举英特尔(R) PROSet 配置的适配器所使用的所有 PCI 设备。	SELECT * FROM IANet_EthernetPCIDevice	IANet_EthernetPCIDevice	此查询仅返回英特尔(R) PROSet 所管理的 PCI 设备。 这等于 IWbemServices::CreateInstanceEnumAsync。
确定适配器是否已安装。	不适用	IANet_EthernetPCIDevice	检查 IANet_EthernetPCIDevice 的“Availability”属性。如果等于 10 - “Not Installed（未安装）”，则设备尚未安装。注意：在此阶段对于此设备的信息的了解很有限。
获取与适配器关联的 PCI 设备	ASSOCIATORS OF { IANet_EthernetAdapter 路径} WHERE ResultClass = IANet_EthernetPCIDevice	IANet_EthernetPCIDevice	如果不返回对象，则适配器是一个虚拟适配器，或是一个虚幻适配器。
获取与 PCI 设备关联的适配器	ASSOCIATORS OF {以太网 PCI 设备路径} WHERE ResultClass = IANet_EthernetAdapter	IANet_EthernetAdapter	此查询对 Provider 不是最佳选择 – 客户机以适配器启动则更理想。

返回页首

获取适配器设置

此设置对象并不直接与适配器关联。如 CIM 标准所规定，它们与一个配置对象关联，而该对象则与适配器关联。

涉及架构这一部分的类
为：IANet_EthernetAdapter、IANet_Configuration、IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANetSettingSlider。

关联类 IANet_AdapterConfiguration 和 IANet_SettingContext 不包含任何真正的数据 – 它们的作用就象在设置和其父对象之间的胶水。

任务	**WQL** 查询	结果类	说明
获取适配器的配置对象	ASSOCIATORS OF { IANet_EthernetAdapter 路径} WHERE ResultClass = IANet_Configuration	IANet_Configuration	返回一个对象，即使没有设置，但配置对象始终存在。此对象的对象路径将在下一次查询中使用。
获取与适配器关联的设置	ASSOCIATORS OF {IANet_Configuration 路径} WHERE AssocClass = IANet_SettingContext	IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANet_SettingSlider 的混合体	返回与适配器关联的所有类别。客户端应该使用 __CLASS 属性来确定各个设置的类型。

返回页首

获取组配置

分组架构中的主要类为 IANet_EthernetAdapter、IANet_TeamOfAdapters、IANet_NetworkVirtualAdapter 和 IANet_TeamedMemberAdapter。

此架构的一个挑战就是对每个物理适配器和虚拟适配器都有一个 IANet_EthernetAdapter 实例。客户端必须能够区分一个组的虚拟适配器和此组的成员适配器。

关联类 IANet_NetworkVirtualAdapter 不包含有用的数据 - 客户端感兴趣的仅为此关联的终点。IANet_TeamedMemberAdapter 不包含有关成员适配器在组中是如何使用的这种有用的数据。

任务	**WQL** 查询	结果类	说明
枚举所有组	SELECT * FROM IANet_TeamOfAdapters	IANet_TeamOfAdapters	每个组都有一个 IANet_TeamOfAdapters 的实例。 这等于 IWbemServices::CreateInstanceEnumAsync。
获取组的虚拟适配器	ASSOCIATORS OF {IANet_TeamOfAdapters 路径} WHERE AssocClass = IANet_NetworkVirtualAdapter	IANet_EthernetAdapter	仅返回组中虚拟适配器的适配器对象。如果组已被创建，但是“Apply（应用）”尚未调用，则此适配器不会存在。（参见以下有关更新配置的部分）。
枚举组的成员适配器	ASSOCIATORS OF {IANet_TeamOfAdapters 路径} WHERE AssocClass = IANet_TeamedMemberAdapter	IANet_EthernetAdapter	返回位于组中的适配器，但不描述各适配器的担任的角色。
确定组中一个适配器的角色	REFERENCES OF { IANet_EthernetAdapter 路径} WHERE ResultClass = IANet_TeamedMemberAdapter	IANet_TeamedMemberAdapter	此类包含有关成员适配器与组的关系，以及其当前在组中的状态。

返回页首

获取组设置

此设置对象并不直接与组关联。如 CIM 标准所规定，它们与一个配置对象关联，而该对象则与该组的虚拟 IANet_EthernetAdapter 关联。同一个配置对象还与该组的 IANet_TeamOfAdapters 对象关联。

涉及架构这一部分的类为： IANet_EthernetAdapter、IANet_TeamOfAdapters、IANet_Configuration、IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANetSettingSlider.

关联类 IANet_AdapterConfiguration 和 IANet_SettingContext 不包含任何真正的数据 — 它们的作用就象在设置和其父对象之间的胶水。这与适配器设置情况完全相同。

任务	**WQL** 查询	结果类	说明
获取组的配置对象，从虚拟适	ASSOCIATORS OF { IANet_EthernetAdapter 路径} WHERE ResultClass = IANet_Configuration	IANet_Configuration	返回一个对象。即使没有设置，但配置对象始终存在。此对象的

配器开始			对象路径将在下一次查询中使用。
获取组的配置对象，从适配器组开始	ASSOCIATORS OF {IANet_TeamOfAdapters 路径} WHERE ResultClass = IANet_Configuration		
获取与适配器关联的设置	ASSOCIATORS OF {IANet_Configuration 路径} WHERE AssocClass = IANet_SettingContext	IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANet_SettingSlider 的混合体	返回与适配器关联的所有类。客户端应该使用 __CLASS 属性来确定各个设置的类型。

返回页首

获取 **VLAN** 配置

支持 VLAN 的每个适配器都有一个 IANet_802dot1QVLANService 与其关联，使用关联类 IANet_Device802do1QVVLANServiceImplementation。如果一个适配器没有一个这一类的实例与其关联，则它不支持 VLAN。

每个 VLAN 都由 IANet_VLAN 的一个实例代表。VLAN 并不直接与适配器关联 - 它与适配器的 IANet_802dot1QVLANService 关联。

关联类 IANet_VLANFor 用于将每个 VLAN 实例与正确的 IANet_802dot1QVLANService 关联。此类不包含有关用户的有用数据。

任务	**WQL** 查询	结果类	说明
获取与适配器关联的 802.1q VLAN 服务对象	ASSOCIATORS OF {IANet_EthernetAdapter 路径} WHERE ResultClass = IANet_802dot1QVLANService	IANet_802dot1QVLANService	返回一个或零个对象。
获取适配器上的 VLAN	ASSOCIATORS OF {IANet_802dot1QVLANService 路径} WHERE ResultClass = IANet_VLAN	IANet_VLAN	如果未安装 VLAN，则无法返回对象。

返回页首

获取 **VLAN** 设置

此设置对象并不直接与 VLAN 关联。如 CIM 标准所规定，它们与一个配置对象关联，而该配置对象则与 VLAN 的 IANet_VLAN 对象关联。

涉及架构这一部分的类
为：IANet_VLAN、IANet_Configuration、IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANetSettingSlider。

关联类 IANet_VLANConfiguration 和 IANet_SettingContext 不包含任何真正的数据 — 它们的作用就象在设置和其父对象之间的检胶水。这

与适配器设置情况完全相同。

任务	**WQL** 查询	结果类	说明
获取 VLAN 的配置对象	ASSOCIATORS OF {IANet_VLAN 路径} WHERE ResultClass = IANet_Configuration	IANet_Configuration	返回一个对象，即使没有设置，但配置对象始终存在。此对象的对象路径将在下一次查询中使用。
获取与 VLAN 关联的设置	ASSOCIATORS OF {IANet_Configuration 路径} WHERE AssocClass = IANet_SettingContext	IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANet_SettingSlider 的混合体	返回与 VLAN 关联的所有类。客户端应该使用 __CLASS 属性来确定各个设置的类型。

返回页首

获取 **IP** 协议信息

Provider 提供有关与适配器、VLAN 和组关联的 IP 协议终点的一些有限的信息。不支持其他协议。

包含协议信息的主要类是 ANet_IPProtocolEndpoint。有两种关联类：IANet_VLANProtocolDependency 和 IANet_AdapterProtocolImplementation。要获取组的 IP 终点，先获取组的虚拟 IANet_EthernetAdapter，即：该 IP 终点与此实例相关联。

任务	**WQL** 查询	结果类	说明
获取与适配器关联的 IP 协议终点	ASSOCIATORS OF {IANet_EthernetAdapter 路径} WHERE ResultClass = IANet_IPProtocolEndpoint	IANet_IPProtocolEndpoint	虽然一些适配器可能会有一个以上 IP 地址，它们将仅与一个 IP 协议终点实例关联。
获取与 VLAN 关联的 IP 协议终点	ASSOCIATORS OF {IANet_VLAN 路径} WHERE ResultClass = IANet_IPProtocolEndpoint	IANet_IPProtocolEndpoint	会有一个 IP 协议终点与 VLAN 关联。

返回页首

获取 **Boot Agent** 信息

每个能在快闪 ROM 中支持引导代理的适配器都有一个 IANet_BootAgent 实例使用 IANet_DeviceBootServiceImplementation 关联类与其关联。

任务	**WQL** 查询	结果类	说明
获取与	ASSOCIATORS OF	IANet_BootAgent	以下只读属性提供引导 ROM 映像上有关此适配器的信息。

Boot Agent 关联的设置	{IANet_EthernetAdapter 路径} WHERE ResultClass = IANet_BootAgent		InvalidImageSignature、Version、UpdateAvailable、FlashImageType

返回页首

获取 **Boot Agent** 设置

此设置对象并不直接与引导代理关联。如 CIM 标准所规定，它们与一个配置对象关联，而该对象则与引导代理关联。

涉及架构这一部分的类别：IANet_BootAgent、IANet_Configuration、IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANetSettingSlider。

关联类 IANet_BootAgentConfiguration 和 IANet_SettingContext 不包含任何真正的数据 — 它们的作用就象在设置和其父对象之间的胶水。

任务	**WQL** 查询	结果类	说明
获取引导代理的配置对象	ASSOCIATORS OF {IANet_BootAgent 路径} WHERE ResultClass = IANet_Configuration	IANet_Configuration	返回一个对象。即使没有设置，但配置对象始终存在。此对象的对象路径将在下一次查询中使用。
获取与引导代理关联的设置	ASSOCIATORS OF {IANet_Configuration 路径} WHERE AssocClass = IANet_SettingContext	IANet_SettingInt、IANet_SettingString、IANet_SettingEnum、IANet_SettingMultiSelection 和 IANet_SettingSlider 的混合体	返回与适配器关联的所有类。客户端应该使用 __CLASS 属性来确定各个设置的类型。

返回目录页面 返回页首

返回目录页面

更新配置：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

概述

在大多数情况下，要更新配置，客户端应用程序需要从 IANet_NetService 获取一个会话句柄，并将其存储在 IWbemContext 环境对象中。对配置的更改只有在对 IANet_NetService 调用了 **Apply** 方法之后才生效。对此要求有一些例外：

- 对引导代理类的更改会立即生效而不须会话句柄。
- 某些方法调用（例如：识别适配器）将导致在调用 **Apply** 之前执行一项操作。

对某些操作，可使用环境中的 PreCheck 限定符来检查是否允许某项操作。这将允许用户界面在要求的情况下丢弃某些控制或菜单条目。

返回页首

更改适配器、组或 **VLAN** 设置

更改适配器、组或 VLAN：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要更改适配器、VLAN 或组设置，客户端必须先获取要更改的设置的对象路径。最好的方法是枚举对象上的设置，并存储该设置的 **__PATH** 属性（参见上文）。

客户端更新设置的最简单方法是：

□□□ 从 WMI 获取设置对象的实例。
□□□ 修改 CurrentValue 属性（使用 **IWbemClassObject::Put()**）。
□□□ 调用 **IWbemServices::PutInstance()** 将修改的实例传递回给 WMI Provider。**PutInstance** 必须以 WBEM_FLAG_UPDATE_ONLY 标志调用。

WMI Provider 将验证 CurrentValue；如果验证失败，则返回 WBEM_E_FAIL。失败的确切原因将在 IANet_ExtendedStatus 对象的 Description 属性中返回。

针对设置的描述包括：

- 整数设置值小于允许的最低值。

- 整数设置值大于允许的最高值。
- 整数设置值不是允许的步骤之一。
- 字符串设置长度大于最大允许值。
- 设置值不是允许的值之一。

在 IANet_SettingEnum、IANet_SettingSlider 或 IANet_SettingMultiSelection 的当前值不是允许的值之一的情况下，返回最后一个描述。

客户端能更改的唯一设置属性是 CurrentValue。WMI Provider 将忽略对任何其他值的更改。

此设置类没有受支持的方法。要更改一个设置，修改 CurrentValue 属性，调用 **PutInstance**。

返回页首

创建新（空）组

创建新组：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

若要创建新组，创建一个 IANet_TeamOfAdapters 实例（即：使用 **IWbemServices::GetObject()** 以获取 IANet_TeamOfAdapters 的类对象，然后使用 **IWbemServices::SpawnInstance()** 来创建此对象的实例）。

然后，使用 **IWbemClassObject::Put** 将实例中的 TeamMode 属性设置为想要的组类型（例如：AFT）。最后，调用 **IWbemServices::PutInstance()** 来创建组，传递 WBEM_FLAG_CREATE_ONLY 标志。

新组的对象路径存储在调用完成之后被传递回来的 **IWbemCallResultObject** 中。**IWbemCallResult::GetResultString** 方法将获取一个新的对象路径。

如果此动作失败，客户端将检查 IANet_ExtendedStatus 以了解失败的原因。

组的虚拟 IANet_EthernetAdapter 和 IANet_IPProtocolEndpoint 类在调用 **Apply** 之后将不再可用。可使用与新 IANet_TeamOfAdapters 关联的 IANet_Configuration 对象获取组的设置。

返回页首

向组中添加适配器

向组中添加适配器：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要向组中添加适配器，创建一个 IANet_TeamedMemberAdapter 实例（即：使用 **IWbemServices::GetObject()** 获取 IANet_TeamedMemberAdapter 的类对象，然后使用 **IWbemServices::SpawnInstance()** 来创建此对象的实例）。

对象中的以下属性必须使用 IWbemClassObject::Put(): 来设置：

- GroupComponent 必须设为要将适配器添加其上的 IANet_TeamOfAdapters 的完整对象路径。
- PartComponent 必须设为要添加到组的 IANet_EthernetAdapter 的完整对象路径。

还可以设置组中适配器的优先级别。最后，调用 **IWbemServices::PutInstance()** 将适配器添加到组，传递 WBEM_FLAG_CREATE_ONLY 标志。如果此项动作失败，检查 IANet_ExtendedStatus 以了解错误代码。

返回页首

从组中删除适配器

从组中删除适配器：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要从组中删除适配器，使用 **IWbemServices::DeleteInstance()** 删除将适配器与组关联的 IANet_TeamedMemberAdapter 实例。如果此项动作失败，检查 IANet_ExtendedStatus 以了解错误代码。

返回页首

删除组

删除组：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要删除组，使用 **IWbemServices::DeleteInstance()** 删除 IANet_TeamOfAdapters 实例。如果此项动作失败，检查 IANet_ExtendedStatus 以了解错误代码。

返回页首

更改组模式

更改组模式：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要更改组模式，获取该组的 IANet_TeamOfAdapters 实例（例如：运用使用该组对象路径的 **IWbemServices::GetObject**）。然后，使用 **IWbemClassObject::Put** 来更改该组的 TeamMode 属性。最后，调用 **IWbemClassObject::PutInstance** 来指示 WMI Provider 更新组模式，传递 WBEM_FLAG_UPDATE_ONLY 标志。如果此项动作失败，检查 IANet_ExtendedStatus 以了解错误代码。

返回页首

更改组中适配器的优先级

更改组中适配器的优先级：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要更改适配器优先级，客户机应该首先获取适配器的 IANet_TeamedMemberAdapter 实例。（例如：运用使用对象路径的 **IWbemServices::GetObject** ）。然后，客户端可使用 **IWbemClassObject::Put** 来更改适配器的 AdapterFunction 属性。最后，客户端需要调用 **IWbemClassObject::PutInstance** 指示 WMI Provider 更新适配器优先级。如果此项动作失败，客户端应该检查 IANet_ExtendedStatus 以了解错误代码。

返回页首

卸装适配器

卸装适配器：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

若要卸装适配器，调用 **IWbemServices::DeleteInstance**，传递要卸装的适配器的对象路径。

返回页首

创建 **VLAN**

创建 VLAN：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要创建 VLAN，对要将 VLAN 添加到其上的适配器的 IANet_802dot1QVLANService 调用 **CreateVLAN** 方法。必须将下列参数传递到该方法：

- VLANNumber，这是 VLAN 的数目。（范围为 1- 4094）
- Name，这是用以识别 VLAN 的用户可定义名称。

此函数将在 Out 参数 VLANpath 中返回新创建的 VLAN 的对象路径。如果此项动作失败，检查 IANet_ExtendedStatus 以了解错误代码。

返回页首

更改 **VLAN** 的属性

更改 VLAN 的属性：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

客户端能更改 VLAN 的 VLANNumber 和 VLANName 属性。要更改适配器的优先级，首先获取适配器的 IANet_VLAN 实例（例如，运用使用对象路径的 **IWbemServices::GetObject** ）。

然后，将 VLANNumber 或 VLANName 更改到所需的值。最后，调用 **IWbemClassObject::PutInstance** 来指示 WMI Provider 更新属性，传递 WBEM_FLAG_UPDATE_ONLY 标志。如果此项动作失败，检查 IANet_ExtendedStatus 以了解错误代码。

返回页首

删除 **VLAN**

删除 VLAN：

- 要求会话句柄。
- PreCheck 可用。
- 在执行操作之前要求调用 **Apply**。

要删除 VLAN，调用 **IWbemServices::DeleteInstance**，传递要删除的 VLAN 的对象路径。

更新 **Boot Agent**

更新 Boot Agent：

- 不要求会话句柄。
- PreCheck 不可用。
- 不要求在执行操作之前调用 **Apply** 。

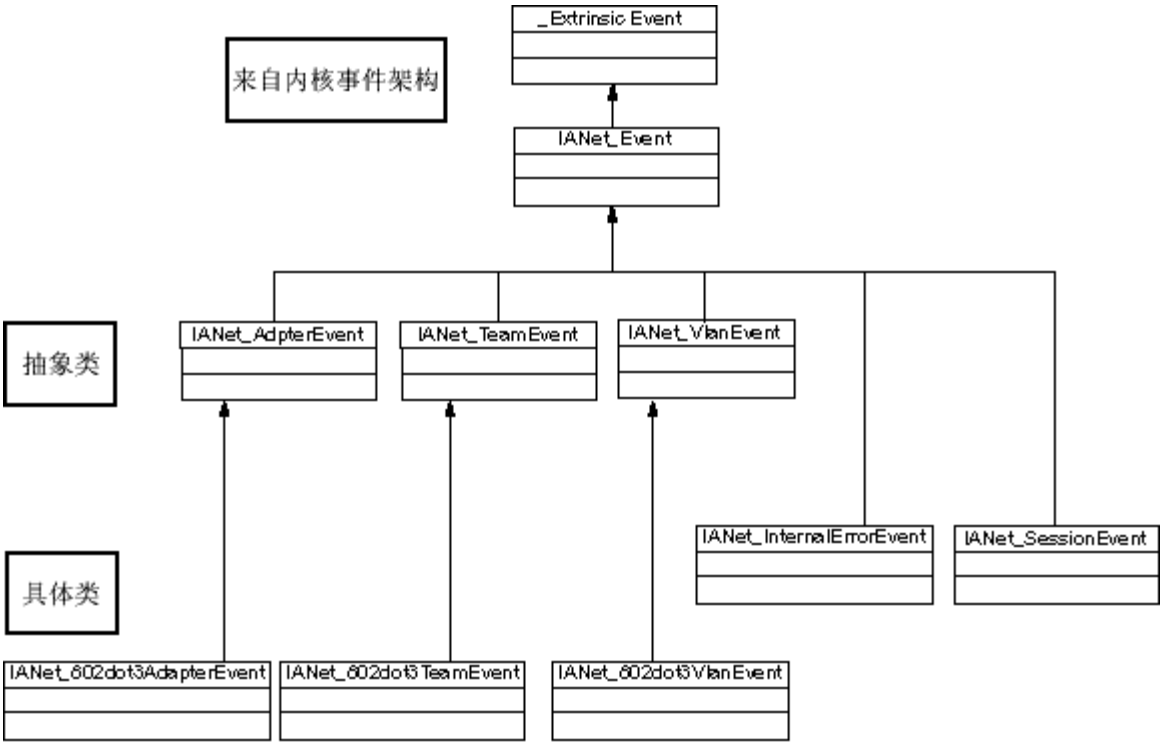
客户端能使用方法调用来更新 Boot Agent 映像。要读取／写入快闪映像，首先获取适配器的 IANet_BootAgent 实例（例如，运用使用对象路径的 **IWbemServices::GetObject**）。

然后，执行 ReadFlash() 来读取现有快闪引导 ROM 映像或者 ProgramFlash() 来更新快闪引导 ROM 映像。如果此项动作失败，检查 IANet_ExtendedStatus 以了解错误代码。

任务	**WMI** 方法	结果	说明
更新或插入适配器的引导 ROM 映像。	uint32 ProgramFlash([IN, ValueMap {"0","1"}, Values {"Check Version", "Write Flash"}:Amended uint32 Action, [IN] uint8 NewFlashData[], [OUT] string strErrorMessage);	如果指定了“检查版本”动作，要是 NewFlashData[] 中正被更新的引导 ROM 映像比 NIC 中的引导 ROM 映像时间更早，此方法将返回一个警告消息。 如果指定“写入”动作，这将以 NewFlashData[] 更新 NIC 上的 FLASH ROM。	此方法用于更新 NIC 上的快闪 ROM。这将使 NIC 在更新快闪的时候停止与网络的通讯。
读取引导 ROM 映像	uint32 ReadFlash([OUT] uint8 FlashData[]);	FlashData[] 包含 NIC 上的快闪 ROM 映像。	此方法读取 NIC 上可被保存入文件的快闪 ROM。

返回目录页面

事件通知：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南



返回页首

具体事件类

IANet_802dot3AdapterEvent

目的
此事件通知客户端有关适配器状态或配置的更改。

触发者
适配器状态更改或您更改了适配器设置并调用了 **Apply** 便会发生此事件。

事件数据
AdapterPath 包含导致事件的适配器的对象路径。

IANet_802dot3TeamEvent

目的
此事件通知客户端有关组状态或配置的更改。

触发者
此事件将在以下情况发生：

- 适配器状态更改。
- 更改组设置并调用了 **Apply**。
- 组配置被更改，而且您调用了 **Apply**。

事件数据
TeamPath 包含导致事件的组的对象路径。

IANet_802dot3VlanEvent

目的
此事件通知客户端有关 VLAN 状态或配置的更改。

触发者
此事件将在以下情况发生：

- VLAN 状态更改。
- 更改 VLAN 设置，并调用了 **Apply**。
- VLAN 配置被更改，而且您调用了 **Apply**。

事件数据
VlanPath 包含导致事件的 VLAN 的对象路径。

返回页首

注册事件

应用程序应该使用 **IWbemServices::ExecNotificationQuery** 或 **IWbemServices::ExecNotificationQueryAsync** 来要求事件通知。以下查询是事件通知查询的示例。因为查询的种类为数很多，此列表并非无所不包

- **SELECT * FROM IANet_Event** — 用来请求所有事件。
- **SELECT * FROM IANet_AdapterEvent** — 用来请求所有适配器事件。
- **SELECT * FROM IANet_TeamEvent** — 用来请求所有组事件。
- **SELECT * FROM IANet_SessionEvent** — 用来请求所有会话事件。
- **SELECT * FROM IANet_VlanEvent** — 用来请求所有 VLAN 事件。
- **SELECT * FROM IANet_InternalErrorEvent** — 用来请求所有内部事件。
- **SELECT * FROM IANet_AdapterEvent WHERE AdapterPath={IANet_EthernetAdapter object path}** — 用来请求特定适配器的适配器事件。

返回目录页面 返回页首

返回目录页面

优化的 **WQL** 查询：英特尔**(R) PRO** 网络适配器和 **WMI** 和 **CDM Provider** 用户指南

概述

WMI Provider 已被优化，允许应用程序使用查询获取设置。WMI Provider 能识别以下查询，并仅返回匹配的对象。所有其他的查询都将使 WMI Provider 获取所有对象的所有设置，而 CIMOM 则在它们到达应用程序之前对其过滤。如果有几个适配器、组和 VLAN，这将导致在检索所要求的数据时有几秒钟的延迟。

返回页首

获取特定适配器、**VLAN** 或组的设置

以下查询将仅获取特定适配器、VLAN 或组的设置。WQL 不允许在 WHERE 子句中有任何额外的子句。

ASSOCIATORS OF {IANet_Configuration 路径**} WHERE AssocClass = IANet_SettingContext**

返回页首

获取一个设置

以下查询可用以获取一个对象的单个设置而不必查询以获取其全部设置：

SELECT * FROM [设置类**] WHERE ParentId="[**设备 **ID]" AND ParentType="[**类型**]" AND Caption="[**设置名称**]"**

注**:**

- 类必须是真正的设置类，而不是基础类（如：IANet_SettingInt）。
- 允许的 ParentTypes 为“NIC”、“Team”、“VLAN”或“BootAgent”。
- ParentId 是独特地定义带有该设置（对适配器，为 DeviceId）的对象的 GUID 。
- 不推荐使用此方法来获取与对象关联的设置。首选方法是使用关联。但是，WQL 不支持要求的复杂查询（即：WQL 不支持 ASSOCIATORS OF {IANet_Configuration 路径} WHERE AssocClass = IANet_SettingContext AND Caption="[设置名称]")

请阅读所有限制和免责声明。

返回目录页面 返回页首

返回目录页面

诊断程序： 英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

诊断类

IANet_DiagTest

目的
IANet_DiagTest 是来自 CIM_DiagnosticTest 的子类。此类提供一个通用手段来运行和控制对 Intel(R) PROSet 支持的以太网适配器进行的诊断测试。CIM_DiagnosticTest 是一个超类，旨在向在启用了 CIM 的系统上对任何计算机硬件进行测试提供通用支持。该类的属性在本质上是描述性的，而且测试机制由显露的方法提供。

实例
主键是“Name”，在此 provider 中，它是在引用的适配器 GUID 上的测试数字索引的结合（例如：1@{12345678-9ABC-DEF0-1234-123456789012}）。此主键的值在某种意义上是一项多余的信息，因为所有引用一个适配器和测试的信息都被作为对象参数被传递到 RunTest 和其他方法。但是，实例必须与传递至方法的参数一致，否则 provider 将拒绝该项命令。标题属性提供此实例将要运行的测试的名称。其他属性提供其他描述和运行时信息。

创建实例
不能创建 IANet_DiagTest 的实例。

删除实例
不能删除 IANet_DiagTest 的实例。

修改属性
在此类中没有可供用户修改的属性。

关联

- IANet_DiagTestForMSE 实例使 IANet_DiagTest 和 IANet_ManagedSystemElement 关联。IANet_ManagedSystemElement 将是 IANet_EthernetAdapter 的实例。
- IANet_DiagResultForTest 实例使 IANet_DiagTest 和 IANet_DiagnosticResult 实例相关联。
- IANetDiagSettingForTest 实例使 IANet_DiagTest 和 ANet_DiagSetting 相关联。

不受支持的属性
Install Date, OtherCharacteristicDescription

方法
此类的实例支持以下方法：

- RunTest — 运行用三个参数定义的测试，这三个参数引用：
  - SystemElement — 定义要测试的适配器，方法是引用 SystemElement 的一个实例；它将始终为 IANet_EthernetAdapter 子类。
  - Setting — 定义要运行的测试及其运行方式，方法是引用 CIM_DiagnosticSetting 的一个实例，它将始终为 IANet_DiagSetting 子类。
  - DiagnosticResult — 定义 CIM_DiagnosticResult 类的实例；它将始终为 IANet_DiagResult 类。
- DiscontinueTest — 试图中止由引用 CIM_ManagedSystemElement 和 CIM_DiagnosticResult 的两个参数定义的，正在进行的测试。这些参数的功能与 RunTest 相同。第三个参数 TestingStopped 返回一个 BOOLEAN 值，表明该命令是否成功地中止了测试。

- ClearResults — 使用以下参数清除测试结果：
  - SystemElement
  - ResultsNotCleared

  引用的参数 ManagedSystemElement 与该对象的对象路径结合起来，引用将被删除的 DiagnosticResultForMSE 实例。同时，DiagnosticResultForMSE 引用的所有 DiagnosticResult 对象引用也将被删除。而且，引用被删除的 DiagnosticResult 对象的 DiagnosticResultForTest 的所有实例也将被删除。最后，字符串数组 Output 参数 ResultsNotCleared 将列出不能被删除的 DiagnosticResults 的主键。

类层次结构

对 CimV2。不列出未使用的属性和方法。

- CIM_ManagedElement:
  Caption
  Description
- CIM_ManagedSystemElement:
  Install Date
  Name
  Status
- CIM_LogicalElement
- CIM_Service:
  - 主键
    - Name (字符串)
  - 属性
    - Caption (字符串)
    - CreationClassName (字符串)
    - Description (字符串)
    - Started (布尔值)
    - StartMode (字符串)
    - Status (字符串)
    - SystemCreationClass (字符串)
    - SystemName (字符串)
- CIM_DiagnosticTest：
  - 属性
    - Characteristics (uint16 数组)
    - IsInUse (布尔值)
    - ResourcesUsed (uint16 数组)
  - 方法
    - RunTest
    - ClearResults
    - DiscontinueTest

在 **WbemTest** 中执行 **RunTest** 和其他方法

来自 MOF 文件的 RunTest 方法如下：

```
uint32
RunTest([IN] CIM_ManagedSystemElement ref SystemElement,
[IN] CIM_DiagnosticSetting ref Setting,
[OUT] CIM_DiagnosticResult ref Result);
```

前两个参数为 In 参数。必须获取被引用的两个对象的对象路径。还必须获取 IANet_DiagTest 对象的对象路径，它将导出 RunTest 对象。

□□□ 从主 WBEM 测试对话框，单击 **Connect**（连接）。
□□□ 输入合适的服务器\名称空间。Namespaces IntelNCS 和 CimV2 均受支持。
□□□ 单击 WBEM 测试的**Enum Instances**（枚举实例）按钮并输入 `IANet_DiagTest`。
□□□ 双击 IANet_DiagTest 的实例。名称的形式为 X@[AdapterGUID}，其中 X 是测试名称，AdapterGUID 是适配器名称，与 IANet_EthernetAdapter 的“Name”键相同。
□□□ 以下是检索的对象路径示例：
`\\MYCOMPUTER\root\Cimv2:IANet_DiagTest.Name="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`
□□□ 保存对象路径。

□□□ 单击 WBEM 测试的**Enum Instances**（枚举实例） 按钮并输入 IANet_EthernetAdapter。
□□□ 双击要测试的适配器。
□□□ 以下是检索的对象路径示例:
\\MYCOMPUTER\root\cimv2:IANet_EthernetAdapter.DeviceID="{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"
□□□□ 保存对象路径。
□□□□ 单击 WBEM 测试的**Enum Instances**（枚举实例） 按钮并输入 IANet_DiagSetting。
□□□□ 双击代表适配器／测试组合的设置。
□□□□ 以下是检索的对象路径示例:
\\MYCOMPUTER\root\cimv2:IANet_DiagSetting.SettingID="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"
□□□□ 保存对象路径。
□□□□ 从主 WBEM 测试对话框，单击 **Execute Method**（执行方法）。
□□□□ 将 IANet_DiagTest 对象路径粘贴到对话框中。单击 **OK(**确定**)**。
□□□□ 在方法下面的下拉框中选择测试。
□□□□ 单击 **Edit In Parameters**（编辑 **In** 参数）按钮。
□□□□ 对 RunTest，Setting 和 SystemElement 都是 In 参数。粘贴先前保存的“Setting（设置）” 和“Adapter（适配器）” 对象路径并关闭。
□□□□ 单击 **Execute**（执行）按钮。
□□□□ 枚举 IANet_DiagResult 类，其方式与 In 参数相同。
□□□□ 如需要，检查选择的结果对象。

IANet_DiagSetting

目的
IANet_DiagSetting 实例提供特定的运行时诊断测试指令。使用的指令对所有测试均通用，并绑定至超类 CIM_DiagnosticSetting。这些包括属性，如 ReportSoftErrors 和 HaltOnError。没有额外属性被添加到子类 IANet_DiagSetting。

创建实例
不能创建此类的实例。

删除实例
不能删除此类的实例。

修改属性
UpdateInstanceAsync 已被实现，可以用来将测试参数设置为“Halt On Error（出错即停顿）”、“Report Soft Errors（报告软错误）”、“Report Status Messages（报告状态消息）”、“Quick Mode（快速模式）”、“Test Warning Level（测试警告级别）”以及“Percent Of Test Coverage（测试覆盖百分比）”。

关联
IANetDiagSettingForTest 实例使 IANet_DiagTest 和 ANet_DiagSetting 相关联。

不受支持的属性
以下属性不受 NCS 支持:

- Caption
- Description

方法
无

类层次结构

对 CimV2。不列出未使用的属性和方法。

- CIM_ManagedElement
- CIM_Setting:
  - Properties
  - SettingID
  - Methods（均不受支持）
  - VerifyOKToApplyToMSE
  - ApplyToMSE

VerifyOKToApplyToCollection
    - ApplyToCollection
    - VerifyOKToApplyIncrementalChangeToMSE
    - ApplyIncrementalChangesToMSE
    - ApplyIncrementalChangeToCollection
- CIM_DiagnosticSetting:
    - 主键
        - SettingID (字符串)
    - 属性
        - TestWarningLevel (uint16)
        - ReportSoftErrors (布尔值)
        - ReportStatusMessages (布尔值)
        - HaltOnError(布尔值)
        - QuickMode (布尔值)
        - PercentOfTestCoverage (uint8)

IANet_DiagResult

目的
IANet_DiagResult 显示在特定适配器上运行的特定测试的结果数据。此类的实例跟 IANet_DiagTest 和 IANet_DiagSetting 实例一一对应。

实例
IANet_DiagResult 的实例和在特定适配器上运行的特定测试的结果相对应。主键格式跟 IANet_DiagTest 和 IANet_DiagSetting 相同。此实例能将任何随意测试结果储存为任何数据（不符合定义的属性），可被置入 TestResults 数组属性。任何时候在一个适配器上运行一个新测试，其新实例将覆盖与此适配器和测试组合相应的测试结果的现有实例。

创建实例
不能创建此类的实例。

删除实例
不能删除此类的实例。

修改属性
不能修改此类的实例。

关联
IANet_DiagResultForTest 实例使 IANet_DiagTest 和 IANet_DiagnosticResult 实例相关联。

不受支持的属性
以下属性不受 NCS 支持：

- EstimatedTimeOfPerforming
- HaltOnError
- OtherStateDescription
- ReportSoftErrors
- estWarningLevel

方法
无

类层次结构

对 CimV2。不列出未使用的属性和方法。

- CIM_DiagnosticResult:
    - 主键
        - DiagnosticCreationClassName (字符串)
        - DiagnosticName (字符串)
        - ExecutionID (字符串)
        - DiagSystemCreationClassName (字符串)
        - DiagSystemName (字符串)

    - 属性
        - TimeStamp (字符串)
        - IsPackage (布尔值)
        - TestStartTime (datetime)
        - TestCompletionTime (datetime)
        - TestState (uint16)
        - TestResults (字符串)
        - PercentComplete (uint8)
- IANet_DiagResult

返回页首

注册表条目

以下条目在安装时被输入 **HKLM\Software\Intel\NETWORK_SERVICES\NCS\NcsDiag** 下的注册表。这些表项和值控制诊断测试的执行，其定义如下。

以下表格包含主键的值、其类型，以及对其用法的简单描述：

值	类型	默认值	用法
检查时间	REG_DWORD	2 秒	检查对发送方或响应方测试的完成情况的间隔时间长短。
启用	REG_DWORD	0	“启用” (1) 或“禁用” (0) 结果日志文件。
FileAppend	REG_DWORD	1	将结果文件“附加”(1) 到现有日志结果文件。如果为 0，则删除现有文件。
LogFileName	REG_SZ	NcsDiag.log	日志结果文件的名称。
MaxFileSize	REG_DWORD	0x10000	日志结果文件的最大大小。
MaxPktsRcvd	REG_DWORD	200	在“快速模式”中（从 IANet_DiagSetting），如果接收的数据包数目大于此值，则发送／接收测试将中断。
TimeoutSndRsp	REG_DWORD	100	如果测试时间（以秒计）超出此值，则退出测试。

返回页首

记录日志

“结果”日志

“结果”日志主要显示信息，此信息还可从 IANet_DiagResult 对象获取。所不同的是从 CIM 浏览器获取的信息将仅显示在特定适配器上进行的特定测试的最后一个结果。后随的测试结果覆盖先前的测试结果。“结果”日志比较容易设置，而且可以查看某一项测试的多次运行结果。

启用“结果”日志

要启用结果日志：

□□□ 在注册表项 **HKLM\Software\Intel\NETWORK_SERVICES\NCS\NCSDiag** 中，将 Enable 值设为 1。
□□□ 将 LogFileName 值设为您喜欢的日志文件名称，或者接受默认值 **NcsDiag.log**。
□□□ 日志文件可在 InstalledDir 值指明的目录中找到。

返回页首

关联类

关联类	首选项属性／值	首选项属性／值
IANet_DiagTestForMSE	Antecedent = IANet_DiagTest	Dependent = IANet_EthernetAdapter
IANet_DiagResultForTest	DiagnosticResult = IANet_DiagResult	DiagnosticTest = IANet_DiagTest
IANet_DiagSettingForTest	Element = IANet_DiagTest	Setting = IANet_DiagSetting
IANet_DiagResultForTest	DiagnosticResult = IANetDiagResult	DiagnosticTest = IANet_DiagTest

返回页首

测试

被实施的测试可在一台或两台计算机上运行。有关测试的详细描述不属本文档的范围，因为 CDM provider 的主旨是作为一个不必依赖于具体测试细节的通用的测试手段。但是，代码中仍然还有一些内在的依赖性，本节将予以解释。

单个适配器测试

以下测试在单个适配器上运行，不须和其他任何适配器进行交互。

- EEPROM
- 控制寄存器
- MAC 回送
- PHY 循环
- 链接

这些测试的错误消息来自向较低堆栈层次发出的调用返回的 HRESULT 错误代码。错误代码作为错误代码存储在内部，以下两种情况将导致它们被转换为错误消息：IANet_DiagResult 对象因枚举而被反引用；管理应用程序接收到对象调用指令。

要求两个适配器的测试

发送方和响应方测试相互依赖，其中一个适配器（发送方）将数据包发送到另一个适配器（响应方），后者将数据包送回给发送方，至此完成一个循环。这些是相同的测试，都可以从英特尔(R) PROSet 运行。但是， 英特尔(R) PROSet 不使用 CDM，也不允许在同一个机器上同时运行两个测试。CDM 允许同时在同一个机器上运行不同的测试。

发送方响应方测试

发送方／响应方要求两个英特尔适配器，一个为发送方，一个为响应方。这是唯一基于第二个线程运行的测试；该线程保持运行直至测试按照完成标准得以完成或者被主线程中止。完成标准是基于测试时间长度所定的超时，或是基于接收到的数据包数目。这两项值均来自注册表。如果启用了“Quick”模式，则测试仅能基于接收到的数据包数目结束。“Quick”模式是 IANet_DiagSetting 类的属性,因此能对适配器进行个别设置。CDM 响应方对 PROSet 响应方作出响应，反之亦然。

发送方／响应方测试返回两类错误值。首先，可能从较低层次返回一个 Error Code （错误代码）(HRESULT) 。其次，在测试运行时（除非测试被返回的错误代码先期中断），测试线程将返回中级测试统计数据，然后返回最终测试统计数据，包含以下各项：

- Link Status（链接状态）
- Using Auto-Negotiation（使用自动协商）
- Collisions（碰撞）
- Packets Received（接收的数据包）
- Packets Received Total（接收的数据包总数）
- Packets Sent（发送的数据包）
- Transmit Ok（传输成功）
- Receive Oks（接收成功）
- Transmit Errors（传输错误）
- Receive Errors（接收错误）
- Collisions（碰撞）

Diagnostic Phase（诊断程序阶段）

请阅读所有限制和免责声明。

返回目录页面 返回页首

在 **IANet_DiagTest** 中执行方法：英特尔**(R) PRO** 网络适配器和 **WMI** 和 **CDM Provider** 用户指南

在 **WbemTest** 中执行 **RunTest** 和其他方法

来自 MOF 文件的 RunTest 方法如下：

```
uint32
RunTest([IN] CIM_ManagedSystemElement ref SystemElement,
[IN] CIM_DiagnosticSetting ref Setting,
[OUT] CIM_DiagnosticResult ref Result);
```

前两个参数为 In 参数。必须获取被引用的两个对象的对象路径。还必须获取 IANet_DiagTest 对象的对象路径，它将导出 RunTest 对象。

□□□ 从 WBEM 测试主对话框，单击 **Connect**（连接）。
□□□ 输入合适的服务器\名称空间。名称空间 IntelNCS 和 CimV2 均受支持。
□□□ 单击 WBEM 测试的 **Enum Instances**（枚举实例）按钮并输入 `IANet_DiagTest`。
□□□ 双击 IANet_DiagTest 的实例。名称的形式为 X@[AdapterGUID}，其中 X 是测试名称，AdapterGUID 是适配器名称，与 IANet_EthernetAdapter 的“Name”关键字相同。
□□□ 以下是检索的对象路径示例：
`\\MYCOMPUTER\root\Cimv2:IANet_DiagTest.Name="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`
□□□ 保存对象路径。
□□□ 单击 WBEM 测试的 **Enum Instances**（枚举实例） 按钮并输入 `IANet_EthernetAdapter`。
□□□ 双击要测试的适配器。
□□□ 以下是检索的对象路径示例：
`\\MYCOMPUTER\root\cimv2:IANet_EthernetAdapter.DeviceID="{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`
□□□□ 保存对象路径。
□□□□ 单击 WBEM 测试的 **Enum Instances**（枚举实例） 按钮并输入 `IANet_DiagSetting`。
□□□□ 双击代表适配器／测试组合的设置。
□□□□ 以下是检索的对象路径示例：
`\\MYCOMPUTER\root\cimv2:IANet_DiagSetting.SettingID="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`
□□□□ 保存对象路径。
□□□□ 从主 WBEM 测试对话框，单击 **Execute Method**（执行方法）。
□□□□ 将 IANet_DiagTest 对象路径粘贴到对话框中。单击 **OK**（确定）。
□□□□ 在方法下面的下拉框中选择测试。
□□□□ 单击 **Edit In Parameters**（编辑 **In** 参数）按钮。
□□□□ 对 RunTest，Setting 和 SystemElement 都是 In 参数。粘贴先前保存的“Setting（设置）” 和“Adapter（适配器）” 对象路径并关闭。
□□□□ 单击 **Execute**（执行）按钮。
□□□□ 枚举 IANet_DiagResult 类，其方式与 In 参数相同。
□□□□ 如需要，检查选择的结果对象。

CIM 类摘要： 英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

类	能否创建?	能否删除?	实现方法	可设定的属性	不受支持的属性	实例计数	有连系的关联类
IANet_802dot1QVLANService	否	否	CreateVLAN	GRVPEnabled，JoinTime	Description, Install Date, Started, StartMode, Status	每个支持 VLAN 的组或适配器一个实例	IANet_Device802dot1QVLANServiceImplementation，IANet_VLANFor
IANet_AdapterConfiguration	否	否	无	无	无	每个适配器一个实例	此类使 IANet_EthernetAdapter 和 IANet_Configuration 相关联。
IANet_AdapterDevice	否	否	无	无	无	每个非虚幻适配器一个实例	此类使 IANet_EthernetAdapter 和 ANet_EthernetPCIDevice 相关联。
IANet_AdapterProtocolImplementation	否	否	无	无	无	每个绑定至适配器的 IP 协议终点一个实例	此类使 IANet_EthernetAdapter 和 IANet_IPProtocolEndpoint 相关联。
IANet_BootAgent	否	否	ProgramFlash ReadFlash	无	Caption, Description, InstallDate, Started, StartMode, Status	每个支持引导代理能力的适配器一个实例	IANet_DeviceBootServiceImplementation， IANet_BootAgentConfiguration。
IANet_BootAgentConfiguration	否	否	无	无	无	每个引导代理一个实例。	此类使 IANet_BootAgent 和 IANet_Configuration 相关联。
IANet_Configuration	否	否	无	无	无	每个适配器、VLAN 和组一个实例	IANet_AdapterConfiguration，IANet_VLANConfiguration, IANet_SettingContext。
IANet_Device802dot1QVLANServiceImplementation	否	否	无	无	无	每个支持 VLAN 的适配器或组一个实例	此类使 IANet_EthernetAdapter 和 IANet_802dot1QVLANService 相关联。
IANet_DiagTest	否	否	RunTest，DiscontinueTest，ClearResults	无	InstallDate，OtherCharacteristicsDescription	每个适配器／测试组合一个实例	IANet_DiagTestForMSE, IANet_DiagResultForTest, IANet_DiagSettingForTest
IANet_DiagSetting	否	否	无	HaltOnError, ReportSoftErrors, ReportStatusMessages, QuickMode, PercentOfTestCoverage, TestWarningLevel	Caption, Description	每个适配器／测试组合一个实例	IANet_DiagSettingForTest
IANet_DiagResult	否	否	无	无	EstimatedTimeOfPerforming，HaltOnError，OtherStateDescription，ReportSoftErrors，TestWarningLevel	每个适配器／测试组合一个实例	IANet_DiagResultForTest, IANet_DiagResultForMSEM
IANet_EthernetAdapter	否	是	IdentifyAdapter HasVLANs IsPowerMgmtSupported GetPowerUsage SetPowerUsage GetPowerUsageOptions SetPowerUsageOptions TestCable AdvancedTestCable TestLinkSpeed	无	AutoSense-（此属性显露为一种设置），ErrorCleared, OtherIdentifyingInfo, IdentifyingDescriptions, InstallDate, LastErrorCode, MaxDataSize, MaxQuiesceTime, PowerManagementCapabilities-（此属性显露为一种设置），PowerManagementSupported-（此属性显露为一种设置），PowerOnHours， ShortFramesReceived, SymbolErrors, TotalPowerOnHours	每个受英特尔(R) PROSet 支持的安装的适配器、虚幻适配器和组一个实例	IANet_AdapterProtocolImplementation, IANet_AdapterDevice, IANet_AdapterConfiguration, IANet_TeamedMemberAdapter, IANet_NetworkVirtualAdapter, IANet_Device802dot1QVLANServiceImplementation, IANet_DeviceBootServiceImplementation
IANet_EthernetPCIDevice	否	否	无	无	AdditionalAvailability, Capabilities, CapabilityDescriptions, Caption, Description, DeviceSelectTiming, ErrorCleared, ErrorDescription, IdentifyingDescription, InstallDate, LastErrorCode, MaxNumberController, MaxQuiesceTime, Name, OtherIdentifyingInfo, PowerManagementCapabilities, PowerManagementSupported, PowerOnHours, ProtocolDescription, ProtocolSupported, SelfTestEnabled, Status, StatusInfo, TimeOfLastReset, TotalPowerOnHours	每个受英特尔 PROSet 支持的以太网适配器 PCI 卡一个实例	IANet_AdapterDevice
IANet_IPProtocolEndpoint	否	否	无	无	Caption, Description, InstallDate, NameFormat, OtherTypeInformation, Status	每个绑定到英特尔支持的终点的 IP 协议堆栈一个实例	IANet_AdapterProtocolImplementation, IANet_VLANProtocolDependency
IANet_NetService	否	否	GetSessionHandle, Apply, ReleaseSessionHandle, Cancel	无	Caption, Description, Install Date, Started, Start Mode, Status	一个。	无
IANet_NetworkVirtualAdapter	否	否	无	无	无	每个组一个实例	此类使 IANet_TeamOfAdapters 和 IANet_EthernetAdapter 相关联。

IANet_PCIDevice	否	否	无	无	AdditionalAvailability, Capabilities, CapabilityDescriptions, Caption, DeviceSelectTiming, ErrorCleared, ErrorDescription, IdentifyingDescription, InstallDate, LastErrorCode, MaxNumberController, MaxQuiesceTime, Name, OtherIdentifyingInfo, PowerManagementCapabilities, PowerManagementSupported, PowerOnHours, ProtocolDescription, ProtocolSupported, SelfTestEnabled, TimeOfLastReset, TotalPowerOnHours	系统中每个作为网络设备的 PCI 卡一个实例	
IANet_Setting	否	否	无	无	SettingID	这是一个抽象类。	IANet_SettingContext
IANet_SettingContext	否	否	无	无	无	每项设置一个实例	此类使 IANet_Setting 和 IANet_Configuration 相关联。
IANet_SettingInt	否	否	无	CurrentValue	SettingID	每个整数设置一个实例	IANet_SettingContext
IANet_SettingMultiSelection	否	否	无	CurrentValue	SettingID	每项多重选择设置一个实例	IANet_SettingContext
IANet_SettingSlider	否	否	无	CurrentValue	SettingID	每项滑块标尺设置一个实例	IANet_SettingContext
IANet_SettingString	否	否	无	CurrentValue	SettingID	每项字符串设置一个实例	IANet_SettingContext
IANet_TeamedMemberAdapter	是	是	无	AdapterFunction	PrimaryAdapter, ScopeOfBalancing	组中每个适配器一个实例	此类使 IANet_TeamOfAdapters 和 IANet_EthernetAdapter 相关联。
IANet_TeamOfAdapters	是	是	TestSwitchConfiguration	TeamingMode	Install Date, Status	每个组一个实例	IANet_NetworkVirtualAdapter, IANet_TeamedMemberAdapter
IANet_VLAN	否	是	无	VLANNumber, Caption	Description, Install Date, StartMode, Status	每个 VLAN 一个实例	IANet_VLANFor
IANet_VLANConfiguration	否	否	无	无	无	每个 VLAN 一个实例	此类使 IANet_VLAN 和 IANet_Configuration 相关联
IANet_VLANFor	否	否	无	无	无	每个 VLAN 一个实例	此类使 IANet_VLAN 和 IANet_802dot1QVLANService 相关联。
IANet_VLANProtocolDependency	否	否	无	无	无	每个 VLAN 一个实例。	此类使 IANet_VLAN 和 IANet_IPProtocolEndpoint 相关联。

返回目录页面 返回页首

返回目录页面

英特尔软件许可证协议（最终许可证）英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

重要 - 在复制、安装或使用前请先阅读本协议

在认真阅读以下条款之前不得使用或装载本软件及其相关材料（统称“软件”）。装载或使用本软件表示已您同意本协议的条款。如果您不同意本协议的条款，请勿安装或使用本软件。

许可证：

- 如果您是网络管理员，下面的“站点许可证”将适用于您。
- 如果您是最终用户，下面的“单一用户许可证”将适用于您。
- 如您是原始设备制造商（OEM），“OEM 许可证”将适用您。

站点许可证。 您可以将本软件复制到贵组织的计算机上以供组织使用，您也可以制作合理数目的本软件备份，条件是：

□□□ 本软件的使用授权仅限于配合英特尔组件产品的使用。配合非英特尔组件产品使用本软件在此不获授权。
□□□ 除本协议规定之外，您不得复制、修改、出租、出售、分发或转让本软件的任何部分，您并且同意防止他人未经授权而拷贝本软件。
□□□ 您不得逆向工程、反编译或反汇编本软件。
□□□ 您不会转授权或允许由一个以上的用户同时使用本软件。
□□□ 本软件可能包括此处所载之外的条款提供的部分，这些部分需遵从其相应的许可证条款。

单一用户许可证。 您可以将本软件复制到单台计算机上供个人使用，您也可以制作本软件的一个备份，条件是：

□□□ 本软件的使用授权仅限于配合英特尔组件产品的使用。配合非英特尔组件产品使用本软件在此不获授权。
□□□ 除本协议规定之外，您不得复制、修改、出租、出售、分发或转让本软件的任何部分，您并且同意防止他人未经授权而拷贝本软件。
□□□ 您不得逆向工程、反编译或反汇编本软件。
□□□ 您不会转授权或允许由一个以上的用户同时使用本软件。
□□□ 本软件可能包括此处所载之外的条款提供的部分，这些部分需遵从其相应的许可证条款。

OEM 许可证。 您仅可以将本软件作为您的产品不可分割的一部分或结合到您的产品中的一部分，或作为向你的产品用户提供的独立的软件维护性更新进行复制和发布（排除任何其他独立产品），受以下条件限制：

□□□ 本软件的使用授权仅限于配合英特尔组件产品的使用。配合非英特尔组件产品使用本软件在此不获授权。
□□□ 除本协议规定之外，您不得复制、修改、出租、出售、分发或转让本软件的任何部分，您并且同意防止他人未经授权而拷贝本软件。
□□□ 您不得逆向工程、反编译或反汇编本软件。
□□□ 您仅可以在符合书面许可证协议的情况下将软件发布给您的客户。该许可证协议可以是一个“开封”许可证协议。在最低限度，该许可证应该保证英特尔对软件的拥有权。
□□□ 本软件可能包括此处所载之外的条款提供的部分，这些部分需遵从其相应的许可证条款。

无其他权利。 除了在此协议中明确提供的许可，英特尔不以明确或暗示的方式向您提供任何有关英特尔拥有或控制的专属信息，或专利、版权、掩码操作、商标、业界机密，或其他知识产权的权利或许可证。

软件的所有权和版权本软件所有副本的所有权归英特尔或其供应商所有。本软件具有版权，并受到美国和其它国家法律以及国际条约条款的保护。您不得从本软件上删掉任何版权说明。英特尔可能在未作通知的情况下，在任何时间更改“软件”或在此引用的项目，但没有支持或更新“软件”的义务。除非另有明确说明，否则英特尔不授予从属于英特尔专利、版权、商标或其它知识产权的任何明确或隐含权利。只有在受让方同意完全受本条款的约束并且不保留本软件的任何副本时，您才可以转让本软件。

有限的媒体质量担保如果本软件由英特尔以实物媒体的形式提供，英特尔保证该媒体在英特尔提供九十天的时期内免于材料实物上的瑕疵。如果发现此类缺陷，请将媒体退回英特尔，以便英特尔选择替换“软件”或另外交付“软件”。

没有其它担保。除上述保证之外，此软件“按原样”提供而无任何种类的任何明确或隐含保证，包括商业性、不侵权或适用于特定目的的保证。英特尔公司对本软件中包括的任何信息、文字、图形、链接或其它项目的准确性或完整性不作担保，也不承担责任。

责任的限定。在任何情况下，对于由于使用此软件或不能使用此软件而引起的任何损害（包括但不限于利润损失、业务中断或信息丢失），即使英特尔已被通知发生这类损害的可能性，英特尔或其供应商概不负责。有些法律管辖区禁止排除或限制对隐含担保或间接性、偶发性损失的赔偿责任，因此上述限制可能对您不适用。您可能还具有因管辖区而各异的其它法律权利。

本协议的终止。如果您不遵守本协议下的条款，英特尔可随时终止本协议。协议终止时，您必须立即销毁本软件或将软件的所有副本退还英特尔公司。

适用的法律。 本协议引起的索赔应由加利福尼亚州法律管辖，但不包括它的法律冲突规则和《联合国货物销售合同公约》 (the United Nations Convention on Contracts for the Sale of Goods) 的规则。您不得违反有关出口法规而将本软件出口至国外。英特尔不对任何其它协议负责，除非它们是由英特尔授权代表所签署的书面形式。

政府机构的有限权利。 软件附带有“有限权利”。政府使用、复制或泄密应受在 FAR52.227-14 和 DFAR252.227-7013 等及后续条款当中规定的限制制约。政府使用本软件须确认英特尔在此处的专属权利。订约人或生产厂商为英特尔。

请阅读所有限制和免责声明。

返回目录页面 返回页首

返回目录页面

支持：英特尔(R) PRO 网络适配器和 WMI 和 CDM Provider 用户指南

Web 和 Internet 站点

http://www.dell.com

客户支持技术人员

如果本文档中的故障检修步骤无法解决问题，请与 Dell Computer Corporation 联系以获得技术帮助（参阅系统说明文档中的“获得帮助”部分）。

在您致电之前...

请坐在正在运行该软件的计算机前面并备妥产品的说明文档。

技术人员可能会请您提供以下信息：

- 您的地址和电话号码
- 您要求支持的产品名称和型号
- 产品的序列号和服务标签
- 您操作产品所用的软件名称和版本号
- 您使用的操作系统名称和版本号
- 计算机类型（制造商和型号）
- 计算机中的扩展板或添加式插卡
- 计算机的内存容量

请阅读所有限制和免责声明。

返回目录页面