Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

다음 정보는 Dell사의 독자적인 확인 없이 이 문서에 언급된 장치의 공급업자가 제공하며 아래에 있는 제한 및 책임 부인의 적용을 받습니다.

- 소개
- WMI
- 기본 기능
- 설치된 파일
- 보안
- 네임스페이스 및 컨텍스트
- 로케일 및 지역화
- 오류 보고
- 코어 스키마
- 이더넷 어댑터 스키마
- 설정 스키마
- 팀 스키마
- VLAN 스키마
- 현재 구성 가져오기
- 구성 업데이트
- 이벤트 알림
- 최적화된 WQL 질의
- 진단
- IANet_DiagTest에서 메서드 실행
- CIM 클래스 요약
- 소프트웨어 라이센스
- 고객 지원

제한 및 책임 부인

지침, 주의 사항, 제한된 승인 및 인증을 포함하여 이 문서에 수록된 모든 정보는 공급업자가 제공하며 Dell사에서 독자적으로 확인 또는 테스트하지 않았습니다. Dell사는 이러한 지침을 이행하거나 이행하지 못한 결과에 대한 책임을 지지 않습니다.

이 문서에 참조된 품목의 재산권, 성능, 속도 또는 자격 요건에 관한 모든 언급이나 요청은 Dell사가 아닌 공급업자에 의한 것입니다. Dell사는 그러한 언급에 관한 정확성, 완전성 및 실증에 대한 책임을 지지 않습니다. 그러한 언급과 요청에 관한 모든 질문이나 의견이 있으면 공급업자에게 연락해야 합니다.

초판*: 2003*년 *10*월

목차 페이지로 돌아가기

소개: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요

Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서를 시작합니다. 이 문서에서는 Intel PRO Network Adapters WMI and CDM Providers의 외관에 대해 설명합니다. WMI(Windows Management Interface) Provider는 업계 표준 방식으로 모든 Intel 엔드 스테이션 네트워킹 기술을 배치 및 관리하는 수단인 NCS(Network Configuration Services)의 네트워크 구성 블록입니다. Intel PRO CDM(Common Diagnostic Model) Provider는 CIM 2.5 및 WMI 표준과 호환되는 상위 인터페이스 API입니다. 하위 인터페이스에서 CDM Provider는 PROSet 소프트웨어 스택의 하위 레이어에 클라이언트 인터페이스를 구현합니다. 따라서 데이터 무결성을 위한 모든 PROSet 메커니즘이 유지됩니다.

WMI Provider와 CDM Provider는 Intel WMI 네트워크 클래스를 구현하는 소프트웨어 구성 요소 집합입니다. 이러한 클래스는 DMTF(Desktop Management Task Force) CIM 스키마 버전 2.5를 기반으로 합니다.

이 문서에서는 이 제품과 함께 제공된 MOF(Managed Object Format) 파일에 있는 정보를 반복하지 않습니다. 예를 들어, 개별 속성의 자세한 의미는 MOF 속성 설명에서 찾을 수 있습니다.

이 문서에서는 Intel PROSet과 같은 WMI 응용 프로그램에서 시스템 네트워크를 구성하는 클래스의 사용 방법과 WMI 응용 프로그램에서 Intel 네트워크 인터페이스 카드를 테스트하는 클래스의 사용 방법에 대해 설명합니다. 이 문서를 읽기 전에 WMI API와 WMI SDK(http://www.microsoft.com/에서 구할 수 있음)에 익숙해져야 합니다.

맨 처음으로 돌아가기

관련 문서

다음 문서를 사용하여 WMI 기술에 대한 이해를 높일 수 있습니다.

- DMTF(Desktop Management Task Force)에서 게시한 CIM 스키마 버전 2.0, 2.2. http://www.dmtf.org에서 구할 수 있습니다.
- Microsoft Windows Management Interface 및 기타 관리 효율성 정보. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_start_page.asp에서 구할 수 있습니다.
- DMTF의 WBEM(Web-Based Enterprise Management) 초기 작업 정보. http://www.dmtf.org/standards/wbem에서 구할 수 있습니다.
- WMI(Microsoft CIM 구현) SDK. http://msdn.microsoft.com/downloads/에서 구할 수 있습니다.
- DFTM의 System Diagnostic Model 백서. http://www.dmtf.org/standards/documents/CIM/DSP0138.pdf에서 구할 수 있습니다.

경고: 이 제품에는 컴퓨터 시스템이나 네트워크를 공격 및/또는 사용할 수 없게 만드는 데 사용될 수 있는 정보가 포함되어 있습니다. 이 제품을 구현하기 전에 Microsoft 운영 체제 보안 기능에 대해 철저히 이해해야 하며, 개발자와 사용자는 실제 환경에서 이 제품 사용하기 전에 발생할 수 있는 보안 위험에 대해 Microsoft에 문의하는 것이 좋습니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

WMI: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
CIM(CIM 스키마)

개요

WBEM(Web-Based Enterprise Management)은 엔터프라이즈 시스템 관리자에게 엔드 스테이션을 비용 효율적으로 관리할 수 있는 표준화된 방법을 제공하기 위한 DMTF(Desktop Management Task Force) 초기 작업입니다. WBEM 초기 작업에는 단순 워크스테이션 구성부터 여러 플랫폼에 걸친 전체 엔터프라이즈 관리에 이르기까지 광범위한 작업이 포함됩니다. 이 초기 작업의 중심은 일반 관리 환경에 존재하는 객체를 표현하기 위한 확장형 데이터 모델인 CIM(Common Information Model)과 모델링된 데이터를 정의 및 저장하기 위한 MOF(Managed Object Format) 언어입니다.

WMI(Windows Management Instrumentation)는 WBEM 초기 작업을 Microsoft* Windows* 플랫폼에 맞게 구현한 것입니다.

WMI는 다음 세 가지 기본 구성 요소로 구성됩니다.

- 코어 - 이러한 구성 요소는 운영 체제에 속하고 WMI 가능 응용 프로그램이 작동하는 데 필요합니다. SDK를 사용하려면 이러한 구성 요소를 설치해야 합니다.
- SDK - SDK는 WMI 스키마를 찾아보고 스키마를 확장하고 공급자를 만들며 WMI 이벤트를 등록 및 사용하기 위한 도구를 포함합니다. WMI를 사용할 응용 프로그램 개발에 유용한 설명서도 제공합니다. SDK는 Microsoft Platform SDK를 설치하는 동안 설치되며 Windows NT4 SP4 또는 SP5, Windows 2000, Windows Me, Windows XP 및 Windows Server 2003에서 지원됩니다.
- 도구 - Microsoft WMI 도구는 개발자가 새로운 관리 응용 프로그램과 솔루션을 구축하는 데 필요한 도구를 제공하며 WMI에서 관리 데이터에 액세스하는 과정을 단계별로 안내하는 문서와 도구로 채워집니다.

WMI 아키텍처는 다음 구성 요소로 구성됩니다.

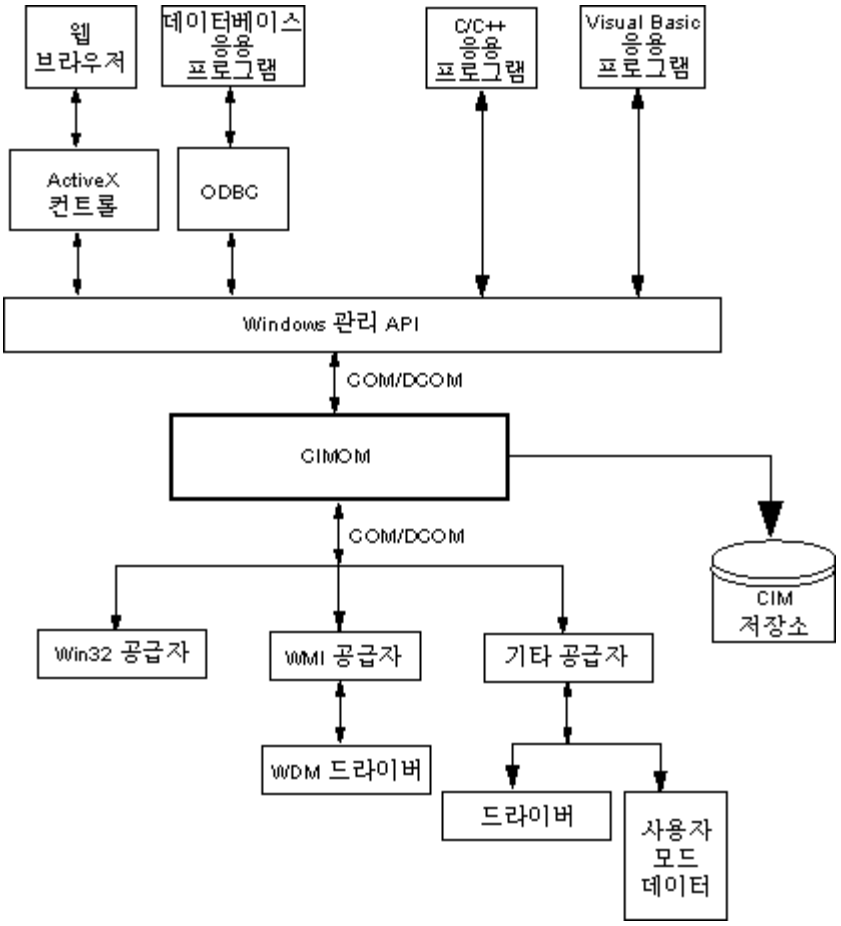
- 관리 응용 프로그램
- 관리된 객체
- 공급자
- 관리 인프라(Windows 관리와 Windows 관리 저장소로 구성됨)
- Windows 관리 API(COM/DCOM을 통해 공급자와 관리 응용 프로그램을 활성화하여 Windows 관리 인프라와 통신할 수 있도록 함)

관리 응용 프로그램은 관리된 객체(논리 엔터프라이즈 구성 요소이거나 실제 엔터프라이즈 구성 요소임)의 데이터를 처리하거나 표시합니다. 이러한 구성 요소는 CIM을 사용하여 모델링되고 응용 프로그램에서 Windows 관리를 통해 액세스됩니다. 공급자는 Windows 관리 API를 사용하여 Windows 관리에 관리된 객체의 데이터를 제공하고 응용 프로그램의 요청을 처리하며 이벤트 알림을 생성합니다.

관리 인프라는 관리 응용 프로그램과 공급자 간의 통신을 처리하기 위한 Windows 관리와 데이터를 정렬하기 위한 Windows 관리 저장소로 구성됩니다. Windows 관리 저장소는 정적 관리 데이터를 보유합니다. 동적 데이터는 공급자가 요청할 때만 생성됩니다. 데이터는 MOF 언어 컴파일러나 Windows 관리 API를 사용하여 저장소에 배치됩니다.

응용 프로그램과 공급자는 이벤트 알림 및 질의 처리와 같은 서비스를 제공하는 Windows 관리 API를 사용하여 Windows 관리를 통해 통신합니다.

다음 도표에서는 WMI 아키텍처 구성 요소의 상호 관계를 보여줍니다.



맨 처음으로 돌아가기

CIM(CIM 스키마)

CIM(Common Information Model)은 관리된 환경에 있는 모든 유형의 논리 객체와 실제 객체에 대해 일관된 통합 뷰를 제공합니다. 관리된 객체는 클래스와 같은 객체 지향 구조를 사용하여 표현됩니다. 클래스에는 데이터에 대해 설명하는 속성과 동작에 대해 설명하는 메서드가 포함됩니다. CIM은 DMTF에서 운영 체제와 플랫폼에 종속되지 않도록 설계되었습니다. WBEM 기술에는 Microsoft Windows 운영 체제 플랫폼용 CIM 확장이 포함됩니다. 자세한 내용은 DMTF 웹 사이트의 DMTF CIM 스키마를 참조하십시오.

CIM은 다음 세 수준의 클래스를 정의합니다.

- 모든 관리 영역에 적용되는 관리된 객체를 표현하는 클래스. 이러한 클래스는 관리된 시스템을 분석 및 설명하기 위한 기본 어휘를 제공하며 코어 모델에 속합니다.
- 특정 관리 영역에 적용되지만 특정 구현이나 기술에 종속되지 않는 관리된 객체를 표현하는 클래스. 이러한 클래스는 일반 모델에 속합니다.
- 일반 모델에 특정 기술을 추가한 관리된 객체를 표현하는 클래스. 이러한 클래스는 일반적으로 UNIX 또는 Microsoft Win32 환경과 같은 특정 플랫폼에 적용되며 확장 모델에 속합니다.

모든 클래스는 상속 관계(하위 클래스가 상위 클래스의 데이터와 메서드를 포함하는 관계)를 통해 상호 관련될 수 있습니다. 상속 관계는 일반적으로 상속 관계를 사용하는 관리 응용 프로그램에서도 볼 수 없고 상속 계층 구조를 알아야 하는 응용 프로그램에서도 볼 수 없습니다. 클래스 계층 구조는 WMI 도구(http://www.microsoft.com의 WMI Tools 참조)에 포함된 응용 프로그램을 사용하여 확보할 수 있습니다.

Windows 관리는 연관 클래스도 지원합니다. 연관 클래스는 서로 다른 두 클래스를 연결하여 사용자가 정의한 관계를 모델링하며 관리 응용 프로그램에서 볼 수 있습니다. Windows 관리는 연관 클래스를 정의하여 시스템 클래스를 지원합니다. 타사 개발자가 해당 관리 환경에 맞게 연관 클래스를 정의할 수도 있습니다.

WBEM은 특정 관리 환경에서 사용되는 클래스와 인스턴스를 그룹화하는 개념인 스키마를 지원합니다. Platform SDK에는 CIM 스키마와 Microsoft Win32 스키마의 두 스키마가 포함됩니다. CIM 스키마는 처음 두 수준의 CIM에 대한 클래스 정의를 포함합니다. 이러한

클래스는 플랫폼에 관계 없이 모든 관리 환경에 속한 관리된 객체를 표현합니다. Win32 스키마는 일반 Win32 환경에 속한 관리된 객체에 대한 클래스 정의를 포함합니다.

CIM에 대해 자세히 알려면 http://www.dmtf.org를 방문하십시오.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

기본 기능: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

NCS WMI Provider 기능
CDM Provider 기능

NCS WMI Provider 기능

WMI Provider의 기본 기능은 다음과 같습니다.

어댑터 기능

- Intel(R) PROSet에서 지원하는 실제 어댑터를 모두 열거합니다.
- 설치된 어댑터의 설정을 열거합니다.
- 설치된 어댑터의 설정을 추가/제거/업데이트합니다.
- 어댑터의 실제 장치 정보를 가져옵니다.
- 어댑터의 시스템 슬롯 장치 정보를 가져옵니다.
- 어댑터의 IPv4 프로토콜 설정을 가져옵니다.
- 어댑터의 Boot Agent 및 관련 설정을 업데이트하고 변경합니다.
- 어댑터를 제거합니다.

팀 기능

- Intel PROSet에서 지원하는 팀을 열거합니다.
- 어댑터 팀을 만들거나 제거합니다.
- 팀 설정을 추가/제거/업데이트합니다.
- 팀의 구성원 어댑터를 추가/제거합니다.
- 팀의 IPv4 프로토콜 설정을 가져옵니다.

VLAN 기능

- 어댑터 또는 팀의 VLAN을 열거합니다.
- 실제 어댑터 또는 어댑터 팀의 VLAN을 만들거나 제거합니다.
- VLAN 설정을 추가/제거/업데이트합니다.
- 팀의 IPv4 프로토콜 설정을 가져옵니다.

이벤트 알림 기능

- 클라이언트가 다음을 등록할 수 있도록 합니다.
  - 어댑터 상태 이벤트
  - 어댑터 구성 이벤트
  - 세션 이벤트
  - 팀 상태 이벤트
  - 팀 구성 이벤트
  - VLAN 구성 이벤트

맨 처음으로 돌아가기

CDM Provider 기능

WMI Provider의 기본 기능은 다음과 같습니다.

- 테스트를 실행, 중지하고 진단 테스트 유형에 관계 없이 테스트 결과를 지웁니다.
- 일반 설정 클래스를 사용하면 CDM 소프트웨어 자체에서 예상하지 못한 방식으로 테스트를 제어할 수 있어야 합니다.

- CDM Provider는 어댑터에서만 사용됩니다.
- 일반 결과 클래스를 사용하면 CDM 공급자 코드에서 발생한 특정 결과 메시지의 연결이 끊깁니다.
- 레지스트리 항목이 공급자 실행을 제어합니다.
- 테스트 결과가 로그 파일에 기록됩니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

설치된 파일: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

WMI 파일
CDM Provider 파일

WMI 파일

실행 파일

WMI Provider 실행 파일은 다음과 같습니다.

- **NcsWmiCo.exe** - 코어 공급자로, IANet_NetService와 코어 이벤트 클래스를 구현합니다.
- **NcsWmiIm.exe** - 인스턴스 및 메서드 공급자로, 이더넷 어댑터 스키마, 팀 구성 스키마, 설정 스키마 및 VLAN 스키마를 구현합니다.
- **NcsWmiEv.exe** - 이벤트 공급자로, 어댑터, 팀 및 VLAN 이벤트를 구현합니다.

MOF 파일

언어 중립적 데이터와 언어별 데이터에 대한 별도의 MOF 파일이 있습니다. IntelNCS 네임스페이스와 CIMV2 네임스페이스에 대한 별도의 MOF 파일도 있습니다. 자세한 내용은 로케일 및 지역화와 오류 보고를 참조하십시오.

IntelNCS 네임스페이스의 MOF 파일은 다음과 같습니다.

- **NcsCmLn.mof** - NCS 클래스가 종속된 CIM 기본 클래스입니다.
- **NcsCmEnu.mfl** - CIM 기본 클래스의 영어(미국) 버전입니다.
- **NcsCoLn.mof** - 코어 공급자가 구현하는 코어 클래스입니다.
- **NcsCoEnu.mfl** - 코어 클래스의 영어(미국) 수정본입니다.
- **NcsIaLn.mof** - IEEE 802.3 어댑터, 팀 및 VLAN의 클래스입니다.
- **NcsIaEnu.mfl** - 802.3 코어 클래스의 영어(미국) 수정본입니다.

CIMV2 네임스페이스의 MOF 파일은 다음과 같습니다.

- **C2CmLn.mof** - NCS 클래스가 종속된 CIM 기본 클래스입니다.
- **C2CmEnu.mfl** - CIM 기본 클래스의 영어(미국) 버전입니다.
- **C2CoLn.mof** - 코어 공급자가 구현하는 코어 클래스입니다.
- **C2CoEnu.mfl** - 코어 클래스의 영어(미국) 수정본입니다.
- **C2IaLn.mof** - IEEE 802.3 어댑터, 팀 및 VLAN의 클래스입니다.
- **C2IaEnu.mfl** - 802.3 코어 클래스의 영어(미국) 수정본입니다.

리소스 파일

WMI Provider의 리소스 파일은 다음과 같습니다.

- **ENU_8023.dll** - 영어(미국) 8023 리소스입니다.
- **ENU_NWRC.dll** - 코어 공급자의 영어(미국) WMI 리소스입니다.
- **ENU_NWR.dll** - 8023 공급자의 영어(미국) WMI 리소스입니다.

지역화된 다른 리소스 파일은 요구 시 로드될 수 있습니다. 지역화된 리소스 DLL의 일반 이름 패턴은 지역화 언어 코드인 `"_mwr.dll"`(예: FRA는 표준 프랑스어를 나타냄)입니다.

맨 처음으로 돌아가기

CDM Provider 파일

실행 파일

CDM Provider 실행 파일은 다음과 같습니다.

- **Ncsdiag.exe**는 CDM 진단용 기본 실행 파일로, Microsoft* WMI 인터페이스 사양을 따르며 out-of-process COM 서버로서 액세스됩니다.
- Intel(R) PROSet 소프트웨어 스택의 다른 실행 파일

MOF 파일

마스터 **.mof** 파일은 제품과 함께 제공되지 않지만 Microsoft* Windows* Management Instrumentation 전역화 모델에 따라 해당 언어 종속적 구성 요소와 언어 중립적 구성 요소로 컴파일됩니다. 자세한 내용은 WMI 지역화의 Microsoft* WMI SDK(Platform SDK 구성 요소) 장을 참조하십시오. 지역화된 *MOF* 파일 컴파일 절을 주의 깊게 살펴보십시오.

.mof 파일(DNcsCdmN.mof)을 삭제하면 Intel 파생 클래스 정의는 삭제되지만 DMTF 정의 클래스는 삭제되지 않으므로 기존의 다른 응용 프로그램에 문제가 발생할 수 있습니다.

일반적으로 이 CDM 구현은 CIMV2 네임스페이스에 기반하여 사용됩니다. IntelNCS 네임스페이스의 MOF 파일은 다음과 같습니다.

파일 이름	언어 유형	설명
CdIa.mof	마스터	CDM 구현의 클래스 정의
CdIaLn.mof	언어 중립적	CDM 구현의 클래스 정의
CdIaEnu.mfl	영어 종속적	Intel CDM 구현의 언어 확장
CdCm.mof	마스터	코어 상위 집합 CDM 클래스 정의
CdCmLn.mof	언어 중립적	코어 상위 집합 CDM 클래스 정의
CdCmEnu.mfl	영어 종속적	코어 상위 집합 CDM 클래스 정의의 언어 확장
DNcsCdmN.mof	적용되지 않음	Intel CDM 클래스 삭제

CIMV2 네임스페이스의 MOF 파일은 다음과 같습니다.

파일 이름	언어 유형	설명
C2Icd.mof	마스터	CDM 구현의 클래스 정의
C2IcdLn.mof	언어 중립적	CDM 구현의 클래스 정의
C2IcdEnu.mfl	영어 종속적	Intel CDM 구현의 언어 확장
C2Cd.mof	마스터	코어 상위 집합 CDM 클래스 정의
C2CdLn.mof	언어 중립적	코어 상위 집합 CDM 클래스 정의
C2CdEnu.mfl	영어 종속적	코어 상위 집합 CDM 클래스 정의의 언어 확장
DNcsCdm2.mof	적용되지 않음	Intel CDM 클래스 삭제

참고**:** 지역화하려면 올바른 언어 종속적 **.mof** 파일을 추가해야 합니다.

리소스 파일

CDM Provider의 리소스 파일은 다음과 같습니다.

- **ENU_Diag.dll** - 진단 공급자의 영어(미국) WMI 리소스입니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

보안: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

WMI Provider와 CDM Provider는 클라이언트 가장을 사용하여 보안을 관리합니다. Provider로의 모든 호출은 하위 레이어로 전달된 클라이언트의 자체 보안 컨텍스트에서 수행됩니다. 대상 컴퓨터에 대한 관리자 권한이 없으면 하나의 작업이나 모든 작업이 실패할 수 있습니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

네임스페이스 및 컨텍스트: **Intel(R) PRO Network Adapters WMI and CDM Providers** 사용 설명서

CIM 클래스는 네임스페이스에 상주합니다. 표준 Microsoft* 네임스페이스는 **root/cimv2**라고 하며 CIM v2.2 또는 **root/default**를 기반으로 합니다. 이 네임스페이스에 WMI 및 CDM Provider 클래스를 추가할 수 있습니다. Provider는 CIM v2.5를 기반으로 합니다. 이것과 객체 키에서 사용되는 차이 때문에 Provider의 클래스는 별도의 네임스페이스인 **root/IntelNCS**에 있습니다.

WBEM 컨텍스트

컨텍스트 객체는 WMI API 메서드에 매개변수로 전달될 수 없는 추가 정보를 Provider에게 제공합니다. 컨텍스트 한정자를 등록하려면 **IWbemContext**를 사용하여 컨텍스트 한정자를 등록하십시오. 컨텍스트 객체의 인터페이스 포인터는 **IWbemServices** 메서드의 마지막 매개변수로 전달됩니다.

다음 표에는 Provider에서 사용하는 컨텍스트 한정자(이름이 지정된 값)가 나와 있습니다. SessionHandle과 같은 대부분의 한정자는 Provider의 특정 기능 영역과 함께만 사용되는 반면 LocaleID, MachineName 및 ApplicationName은 모든 **IWbemServices** 호출에 대해 설정될 수 있습니다.

Provider에 전달된 컨텍스트가 없으면 **Initialize** 호출에서 Provider에 전달된 LocaleID가 사용됩니다. 컨텍스트를 사용하여 수행된 읽기 작업은 쓰기 작업이 수행될 때까지 현재 구성을 읽습니다. 이후의 읽기 작업은 시스템에서 쓰기 작업이 성공한 것처럼 표시합니다. NULL 컨텍스트를 읽기에 사용할 수 있습니다.

컨텍스트 한정자	변수 유형	설명
SessionHandle	VT_BSTR	응용 프로그램의 IANet 네트워크 클래스 복사본을 식별합니다. 먼저 세션 핸들을 설정하지 않고서는 응용 프로그램에서 클래스나 해당 속성을 변경할 수 없습니다. 세션 핸들을 설정하고 사용하는 방법은 IANet_NetService 클래스에 대한 절을 참조하십시오. 응용 프로그램이 클래스에서 데이터를 읽는 경우에는 이 한정자가 필요하지 않습니다. 세션 핸들을 사용하면 NCS 소프트웨어에서 구성에 대한 여러 동시 액세스를 관리할 수 있으므로 한 사용자가 다른 모든 사용자를 잠그지 않도록 할 수 있습니다. 각 세션마다 수행된 모든 변경 사항을 저장하는 별도의 캐시가 있습니다. 여러 사용자가 동시에 변경하고 있으면 첫번째 사용자의 변경 사항이 성공적으로 적용됩니다. 다른 모든 사용자의 캐시는 무효화됩니다.
LocaleID	VT_BSTR	Microsoft의 로케일 ID입니다. 이는 응용 프로그램이 Provider에서 지역화된 텍스트 문자열을 필요로 하는 경우에 필요합니다. 필요한 LocaleID가 사용된 경우를 제외하고 모든 오류 메시지와 경고는 영어입니다.
ApplicationName	VT_BSTR	호출을 수행한 응용 프로그램의 이름입니다. 이는 로깅에 필요합니다.
MachineName	VT_BSTR	Provider에 연결 중인 컴퓨터의 이름입니다. 이는 로깅에 필요합니다.
PreCheck	VT_BOOL	이 부울 값은 Provider에게 클라이언트가 실제로 작업을 수행하기 전에 해당 작업이 허용되는지 확인 중이라고 알리는 데 사용됩니다. 예를 들어, 팀에 어댑터를 추가한다고 합시다. 값: • TRUE = Provider가 작업을 수행하지 않지만, 작업이 허용되지 않는 경우 오류 코드와 확장된 상태를 반환합니다. • FALSE = Provider가 작업을 수행합니다. 이 한정자가 누락되면 속성이 FALSE일 때와 효과가 동일합니다.
WarningErrorCode	VT_I4	일부 작업의 경우 사용자에게 경고를 보내야 할 수 있습니다. 예를 들어, 팀에 어댑터를 추가하기 위해 팀을 재로드해야 하는 경우가 있습니다. WMI는 이에 대한 메커니즘을 제공하지 않습니다. 이 한정자가 0이 아닌 상태로 존재할 경우 작업이 성공했지만 연관된 경고가 있으면 E_FAIL이 반환됩니다. 클라이언트는 확장된 상태를 사용하여 경고 텍스트를 가져와야 합니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

로케일 및 지역화: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

지역화된 MOF 파일
지역화된 속성 데이터

WMI 및 CDM Provider 지역화에는 지역화된 MOF 파일과 지역화된 속성 데이터의 두 가지 측면이 있습니다.

지역화된 MOF 파일

Provider에서 사용하는 모든 MOF 파일은 Microsoft Windows* Management Instrumentation(WMI) 전역화 모델에 따라 지역화됩니다. 이를 위해 각 클래스 정의는 다음과 같이 구분됩니다.

- **.mof** 파일의 기본 클래스 정의만 포함하는 언어 중립적 버전
- 해당 **.mfl** 파일의 로케일별 속성 설명과 같은 지역화된 정보를 포함하는 언어별 버전

지원되는 언어

중국어(대만)
중국어(중국)
덴마크어
네덜란드어(네덜란드)
영어(미국)
핀란드어
프랑스어(프랑스)
독일어(독일)
이탈리아어(이탈리아)
일본어
노르웨이어(복말)
포르투갈어(브라질)
스페인어(스페인-현대)
스웨덴어

클래스 저장

언어별 클래스 정의는 언어 중립적 기본 클래스 정의를 포함하는 네임스페이스 아래의 하위 네임스페이스에 저장됩니다. 예를 들어, WMI and CDM Provider의 경우 **root/IntelIncs** 네임스페이스 아래에 **ms_409**라는 영어 로케일용 하위 네임스페이스가 있습니다. 마찬가지로, 지원되는 각 언어용 하위 네임스페이스도 **root/IntelIncs** 네임스페이스 아래에 있습니다.

cimv2 네임스페이스에서 지역화된 **MOF** 지원

root/cimv2 네임스페이스의 경우 Provider의 클래스(예: IANet_ classes)는 WMI가 이 네임스페이스에 추가한 기본 클래스에서 파생됩니다. 기본 클래스의 언어별 클래스 정의가 있는 하위 네임스페이스는 **root/cimv2** 네임스페이스 아래에 이미 있습니다. IA_Net 언어별 클래스 정의는 이 기존 하위 네임스페이스에 추가됩니다. 기본 클래스에 대한 이 종속성 때문에 MOF 지역화는 기본 시스템 로케일에서만 수행됩니다.

런타임 지원

지역화된 데이터를 검색하기 위해 WMI 응용 프로그램에서 strLocale 매개변수를 **SWbemLocator::ConnectServer** 및 **IWbemLocator::ConnectServer** 호출에 사용하여 로케일을 지정할 수 있습니다. 로케일이 지정되지 않은 경우 해당 시스템의 기본 로케일이 사용됩니다. 예: 영어(미국)의 경우 MS_409가 사용됩니다. 이 로케일은 영어 문자열에서 추가할 때 올바른 네임스페이스를 선

택하는 데 사용됩니다.

또한 기본 정의와 함께 지역화된 데이터를 요청하려면 **IWbemServices::GetObject**, **SwbemServices.GetObject**, **IWbemServices::ExecQuery** 및 **SWbemServices.ExecQuery**에서 WBEM_FLAG_USE_AMENDED_QUALIFIERS 플래그를 지정해야 합니다. 이는 MOF 파일에서 값 맵, 표시 설명 또는 기타 수정된 한정자를 사용하여 표시 가능한 값을 생성하는 모든 기능에 필요합니다.

맨 처음으로 돌아가기

지역화된 속성 데이터

오류 메시지와 같은 지역화된 속성 데이터를 가져오려면 Provider가 모든 호출에 대한 호출자 로케일을 알아야 합니다. 이것이 올바르게 작동하려면 클라이언트가 모든 호출에 대해 전달되는 컨텍스트 객체에 로케일을 추가해야 합니다(WBEM 컨텍스트의 네임스페이스 및 컨텍스트 참조). 지역화 가능 문자열을 반환해야 할 경우 Provider는 클라이언트 로케일에 해당 리소스 DLL을 로드하려고 합니다. 해당 리소스 DLL이 없으면 영어(미국) 문자열이 반환됩니다.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

오류 보고: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
오류 코드

개요

IANet_ExtendedStatus에 대한 이 절에서는 WMI Provider 및 CDM Provider에서 생성된 오류를 처리하는 방법에 대해 설명합니다. 오류 객체의 반환 방식과 반환 시기는 해당 호출이 동기 호출인지, 반동기 호출인지 아니면 비동기 호출인지에 따라 다릅니다. 대부분의 경우 오류가 발생하면 HRESULT가 WBEM_E_FAILED로 설정됩니다. 하지만 이 시점에서는 WMI가 오류를 생성했는지 아니면 Provider가 오류를 생성했는지 알 수 없습니다.

동기 호출의 오류 객체를 가져오려면 GetErrorInfo()를 사용하여 IErrorInfo 객체를 가져오십시오. QueryInterface()를 사용하여 오류 정보를 포함하는 IWbemClassObject를 가져오십시오.

비동기 호출의 오류 객체를 가져오기 위해 IWbemClassObject가 마지막 SetStatus() 호출의 마지막 항목으로 다시 전달됩니다. 오류 객체 인스턴스를 가져온 후에는 __Class 속성을 검사하여 오류의 발생 원인을 확인할 수 있습니다. WMI는 __ExtendedStatus 인스턴스를 만들고, Provider는 IANet_ 클래스에 관련된 오류의 IANet_ExtendedStatus 인스턴스를 만듭니다. IANet_ExtendedStatus는 __ExtendedStatus에서 파생되고 다음 오류 객체 한정자를 포함합니다.

- Description - 현재 로케일에 맞게 조정된 오류 설명입니다.
- File - 오류가 생성된 코드 파일입니다.
- Line - 오류가 있는 코드 파일의 행 수입니다.
- ParameterInfo - 오류 발생 시 활용되고 있던 클래스 또는 속성입니다.
- Operation - 오류 발생 시 시도되고 있던 작업입니다.
- ProviderName - 오류를 발생시킨 Provider 이름입니다.
- StatusCode - 실패한 내부 호출에서 반환된 코드입니다.
- SessionHandle - 작업에 사용된 세션 핸들입니다.
- RuleFailureReasons - 작업 실패 이유입니다. 기술 규칙이 실패했기 때문에 작업이 실패할 수 있습니다. 예를 들어, 특정 팀에 관리 어댑터가 있어야 합니다.

맨 처음으로 돌아가기

오류 코드

모든 오류 코드의 경우 Provider는 로케일에 맞게 사용자 정의된 설명을 제공합니다. 오류 코드의 형식은 심각도가 1로, 기능이 ITF로 설정된 HRESULT입니다. 응용 프로그램에서 다음 코드를 복구 작업에 사용할 수 있습니다.

- 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: Providers only support 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: Qualifer 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 NCS Service has been stopped."

- 0x80040801 - "EAL: Internal exception"
- 0x80040802 - "EAL: General failure"
- 0x80040803 - "EAL: Not initialized"
- 0x80040804 - "EAL: Failed to initialize."
- 0x80040805 - "EAL: Session limits exceeded"
- 0x80040806 - "EAL: Out of memory"
- 0x80040807 - "EAL: Rule syntax error"
- 0x80040808 - "EAL: Unexpected end of list"
- 0x80040809 - "EAL: Rule link error"
- 0x8004080A - "EAL: Device Creation Failed"
- 0x8004080B - "EAL: Media service not found"
- 0x8004080C - "EAL: Device service not found"
- 0x8004080D - "EAL: PCI bus module not found"
- 0x8004080E - "EAL: Adapter is a member of a team"
- 0x8004080F - "EAL: Rule Access Point creation error"
- 0x80040810 - "EAL: Registry key error"
- 0x80040811 - "EAL: Registry XML file path error"
- 0x80040812 - "EAL: Unknown event class"
- 0x80040813 - "EAL: Unknown module id"
- 0x80040814 - "EAL: Rule service not found"
- 0x80040815 - "EAL: NULL input pointer"
- 0x80040816 - "EAL: Rule grammar error"
- 0x80040817 - "EAL: Rule failed"
- 0x80040818 - "EAL: Setting is already grouped"

- 0x80040220 - "Sync Layer: Team removal failed."
- 0x80040221 - "Sync Layer: Vlan creation failed."
- 0x80040222 - "Sync Layer: Vlan removal failed."
- 0x80040223 - "Sync Layer: Adapter removal failed."
- 0x80040224 - "Sync Layer: Setting Change/Creation/Removal failed."
- 0x80040225 - "Sync Layer: Parameter Change/Removal failed."
- 0x80040226 - "Sync Layer: NetConfig subsystem locked. "
- 0x80040227 - "Sync Layer: System Update In Progress. Please try again later."
- 0x80040228 - "Sync Layer: Adapter is Locked"
- 0x80040229 - "Sync Layer: Flash read failed."
- 0x8004022A - "Sync Layer:"

- 0x80040210 - "Sync Layer: Invalid event offset"
- 0x80040211 - "Sync Layer: Invalid input"
- 0x80040212 - "Sync Layer: Invalid key"
- 0x80040213 - "Sync Layer: Adapter not team member"
- 0x80040214 - "Sync Layer: Driver not loaded"

0x80040215 - "Sync Layer: Client impersonation failed"
- 0x80040216 - "Sync Layer: Caught exception"
- 0x80040217 - "Sync Layer: Session not locked"
- 0x80040218 - "Sync Layer: Hardware access layer is not available"
- 0x80040219 - "Sync Layer: Flash not available"
- 0x8004021A - "Sync Layer: Diagnostics not supported"
- 0x8004021B - "Sync Layer: Diagnostic test not running"
- 0x8004021C - "Sync Layer: Boot Agent update not available"
- 0x8004021D - "Sync Layer: Boot Agent corrupted."
- 0x8004021E - "Sync Layer: Flash write failed."
- 0x8004021F - "Sync Layer: Team creation failed."
- 0x80040201 - "Sync Layer: Initialization failed"
- 0x80040202 - "Sync Layer: Invalid initialization handle"
- 0x80040203 - "Sync Layer: Session handle already exists"
- 0x80040204 - "Sync Layer: Invalid session handle"
- 0x80040205 - "Sync Layer: The maximum number of sessions has been reached."
- 0x80040206 - "Sync Layer: The session lock handle already exists"
- 0x80040207 - "Sync Layer: Invalid session lock handle"
- 0x80040208 - "Sync Layer: Session already locked"
- 0x80040209 - "Sync Layer: Invalid media service module Id"
- 0x8004020A - "Sync Layer: Invalid Advanced Service Module Id"
- 0x8004020B - "Sync Layer: Invalid device service module Id"
- 0x8004020C - "Sync Layer: Invalid component type Id"
- 0x8004020D - "Sync Layer: Invalid bus interface module Id"
- 0x8004020E - "Sync Layer: Invalid sink window handle"
- 0x8004020F - "Sync Layer: Invalid event Id"

- 0x80040401 - "HAM PCI: Invalid memory map address"
- 0x80040402 - "HAM PCI: Configuration driver failed to load"
- 0x80040403 - "HAM PCI: Configuration driver version mismatch"
- 0x80040404 - "HAM PCI: Device slot not found"
- 0x80040405 - "HAM PCI: Diagnostic driver failed to load"
- 0x80040406 - "HAM PCI: Diagnostic driver version mismatch"
- 0x80040407 - "HAM PCI: Diagnostic driver initialization failed"
- 0x80040408 - "HAM PCI: Diagnostics not initialized"
- 0x80040409 - "HAM PCI: Diagnostics already initialized"
- 0x8004040A - "HAM PCI: Diagnostic test already running"
- 0x8004040B - "HAM PCI: Diagnostic test not running"
- 0x8004040C - "HAM PCI: Diagnostic test terminated"
- 0x8004040D - "HAM PCI: Diagnostic Invalid test number"
- 0x8004040E - "HAM PCI: Diagnostic hardware missing"
- 0x8004040F - "HAM PCI: Diagnostic send receive initialization failed"

- 0x80040511 - "Media Service: NDIS IO call failed"
- 0x80040512 - "Media Service: Miniport not loaded"
- 0x8004051B - "Media Service: Invalid device handle"
- 0x8004051C - "Media Service: Invalid adapter handle"
- 0x8004051D - "Media Service: Invalid team handle"
- 0x8004051E - "Media Service: Invalid VLAN handle"
- 0x8004051F - "Media Service: Device missing"
- 0x80040520 - "Media Service: Invalid setting type"
- 0x80040521 - "Media Service: Unknown invalid object"
- 0x80040522 - "Media Service: Invalid Setting Handle"
- 0x80040523 - "Media Service: Invalid Team Mode"
- 0x80040525 - "Media Service: Setting Already Exists"

- 0x80042001 - "RAP: Already initialized"
- 0x80042002 - "RAP: Invalid XML file"
- 0x80042003 - "RAP: XML load error"
- 0x80042004 - "RAP: Not initialized"

0x80042005 - "RAP: Rule not extracted before"
- 0x80042006 - "RAP: Conditions count mismatch"
- 0x80042007 - "RAP: Results apply error"
- 0x80042008 - "RAP: Invalid rule"
- 0x80042009 - "RAP: Node not found"
- 0x8004200A - "RAP: Error no single node"
- 0x8004200B - "RAP: No action rule"
- 0x8004200C - "RAP: Zero condition"
- 0x8004200D - "RAP: Zero action"
- 0x8004200E - "RAP: XML Decode error"

목차 페이지로 돌아가기 맨 처음으로 돌아가기

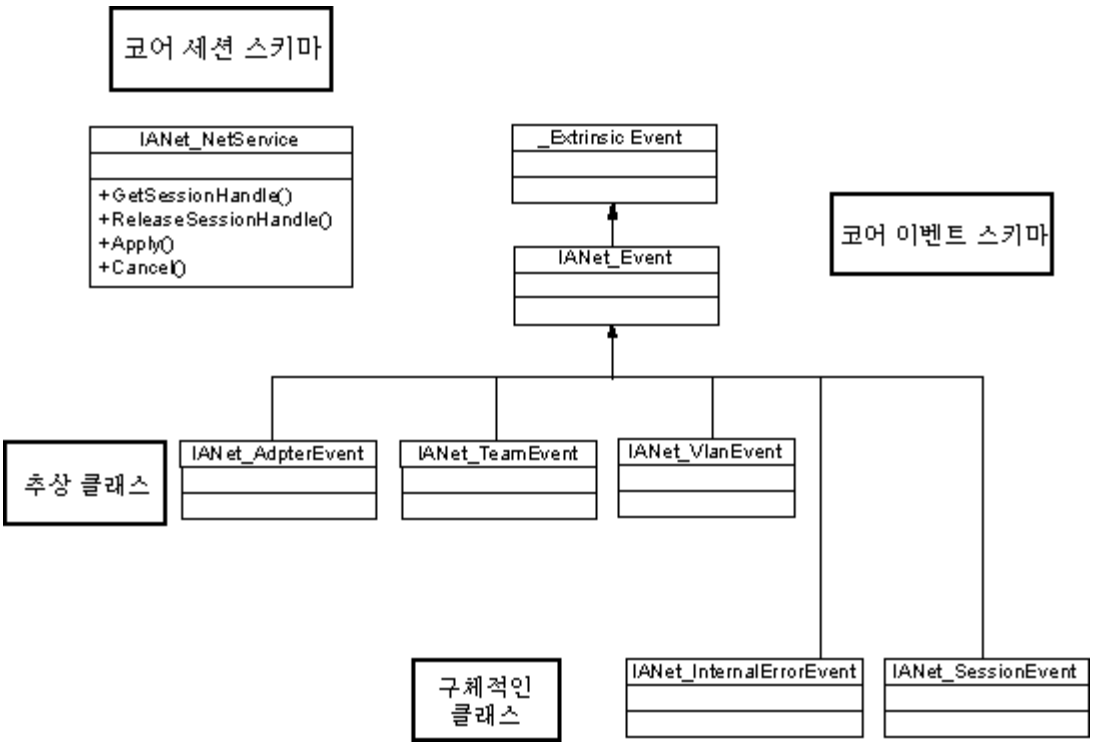
목차 페이지로 돌아가기

코어 스키마: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
IANet_NetService
코어 이벤트
사례

개요

코어 스키마는 IANet_NetService 클래스와 코어 이벤트 클래스로 구성됩니다.



맨 처음으로 돌아가기

IANet_NetService

용도
IANet_NetService 클래스는 IANet_ schema의 루트 객체입니다. 클라이언트는 이 클래스를 사용하여, 설정을 수행하는 데 필요한 세션에 액세스할 수 있습니다.

인스턴스
이 객체의 인스턴스가 하나 있습니다. 클라이언트는 이 클래스에 사용되는 키에 의존하는 대신에 모든 IANet_NetService 인스턴스를 열거하여 클래스 인스턴스를 가져와야 합니다.

인스턴스 만들기
IANet_NetService 인스턴스를 만들 수 없습니다.

인스턴스 제거

IANet_NetService 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

지원되는 속성
이 클래스는 다음 두 속성을 구현합니다.

- Version - 코어 공급자의 현재 버전을 포함합니다.
- InstallDate - 공급자가 설치된 날짜를 포함합니다.

메서드
다음 메서드를 사용하여 세션을 관리할 수 있습니다.

- **void GetSessionHandle(([OUT] string SessionHandle, [out] uint32 ActiveSessions)** - SessionHandle 한정자의 컨텍스트 객체에 배치되어야 하는 세션 핸들 문자열을 설정하는 데 사용됩니다. ActiveSessions는 이 시스템의 활성 세션 수를 반환합니다. 따라서 클라이언트가 다른 사용자가 네트워크 설정을 수정하고 있음을 경고할 수 있습니다.
- **void Apply([IN] string sSessionHandle, [OUT] uint32 FollowupAction);** - 특정 세션 핸들로 변경된 사항을 적용합니다. 반환된 uint32 인수는 WMI Provider 및 CDM Provider가 변경 사항 적용 전에 서버를 재부팅하도록 응용 프로그램에 지시하는 데 사용됩니다. 이 작업은 Win32_OperatingSystem 클래스에서 **Reboot** 메서드를 호출하여 수행할 수 있습니다.

  값:
  1 = 시스템 재부팅이 필요함
  0 = 재부팅이 필요하지 않음

- **void ReleaseSessionHandle ([IN] string SessionHandle)** - 세션 핸들이 사용된 후 해당 세션 핸들을 해제합니다. 이 세션으로 수행된 모든 변경 사항이 손실됩니다. 이 호출 후에는 세션 핸들이 더 이상 유효하지 않고 더 이상 사용될 수 없습니다.
- **void Cancel([IN] string SessionHandle);** - 세션을 취소합니다. 내부 캐시가 비워지며 이 호출 뒤에 읽은 모든 데이터가 현재 구성을 표시합니다.

맨 처음으로 돌아가기

코어 이벤트

IANet_SessionEvent

용도
이 이벤트는 클라이언트에게 NCS 세션 API의 사용을 알리는 데 사용됩니다. 클라이언트는 이 이벤트를 사용하여 다른 클라이언트가 세션을 만들거나 사용하고 있는지에 대한 알림을 받을 수 있습니다.

트리거
이 이벤트는 클라이언트가 세션을 만들거나 세션을 삭제하거나 세션의 **Apply**를 호출할 때 트리거됩니다.

이벤트 데이터
EventType은 다음 값 중 하나를 가질 수 있습니다.

- "New session"은 이 클라이언트나 다른 클라이언트가 새 세션을 만들었음을 나타냅니다.
- "End session"은 클라이언트가 세션을 종료했음을 나타냅니다. 이 클라이언트나 다른 클라이언트가 세션을 종료할 수 있습니다.
- "Cache invalidated"는 다른 클라이언트가 세션에 대해 **Apply**를 호출했음을 나타냅니다. 다른 모든 세션이 무효화되고 해당 세션과 연관된 캐시가 삭제되었습니다.
- "Configuration changed"는 세션의 구성이 변경되었음을 나타냅니다.

SessionHandle은 이벤트를 트리거한 세션 핸들을 포함합니다.

OpenSessions는 열린 세션 수를 포함합니다. 이 데이터 항목은 "Cache invalidated" 및 "Configuration changed" 이벤트의 경우 NULL입니다.

IANet_InternalErrorEvent

용도
이 이벤트는 클라이언트에게 이벤트 공급자 내부 오류가 발생했음을 알리는 데 사용됩니다. 이벤트 공급자가 더 이상의 이벤트를 보고할 수 없음을 의미할 수도 있습니다.

트리거
이 이벤트는 다음과 같은 경우에 발생합니다.

- 이벤트 공급자가 이벤트 소스에서 알 수 없는 이벤트를 가져온 후
- 이벤트를 제공하는 소프트웨어가 종료된 후
- 이벤트 공급자가 이벤트를 가져온 후 이벤트 소스가 더 이상의 이벤트 데이터를 가져올 수 없는 경우

Event Data
EventType은 다음 중 하나일 수 있습니다.

- "Could not get event data". 이벤트가 발생했지만 이벤트 소스가 더 이상의 이벤트 데이터를 가져올 수 없습니다.
- "Event source has shut down". 이벤트의 데이터 소스가 종료되었습니다. 이 경우 이벤트 공급자도 종료되며 소스가 재시작되고 새 알림 질의가 수행될 때까지 더 이상의 이벤트가 생성되지 않습니다.
- "Unexpected message". 이벤트 공급자가 예상치 못한 이벤트를 수신했습니다.

맨 처음으로 돌아가기

사례

구성을 변경하려면 세션 핸들이 필요합니다. 세션 핸들을 사용하면 NCS 소프트웨어에서 구성에 대한 여러 동시 액세스를 관리할 수 있으므로 세션이 다른 모든 세션을 잠그지 않도록 할 수 있습니다. 각 세션마다 수행된 모든 변경 사항을 저장하는 별도의 캐시가 있습니다. 여러 세션에서 동시에 변경하는 경우에는 첫번째 세션의 변경 사항이 성공적으로 적용됩니다. 다른 모든 세션 캐시는 무효화됩니다.

세션 핸들 가져오기

클라이언트는 세션 핸들에 액세스하기 전에 단일 IANet_NetService 인스턴스의 객체 경로를 가져와야 합니다. **IWbemServices::CreateInstanceEnum**을 호출하고 IANet_NetService 클래스 이름을 전달하십시오. 이는 **SELECT * FROM IANet_NetService** 질의를 사용하여 **IWbemServices::ExecQuery**를 호출하는 것과 같습니다. 구성을 변경하기 전에 클라이언트가 세션 핸들을 가져와야 합니다. 새 세션을 시작하려면 **GetSesssionHandle** 메서드를 사용하십시오.

클라이언트는 **IWbemServices::ExecMethod**를 사용하여 CIM 객체에 대해 메서드를 실행할 수 있고 IANet_NetService 인스턴스의 **__PATH** 속성에서 객체 경로를 필요로 합니다. 이 메서드는 현재 활성화된 세션 수도 반환합니다. NCS(Network Configuration Service)에 대한 단독 액세스가 없으면 클라이언트가 경고를 실행해야 할 수 있습니다.

IWbemContext 객체에서 세션 핸들 사용

세션 핸들을 가져온 후 클라이언트는 IWbemContext 객체를 만들어야 합니다. 이 객체의 **SessionHandle** 한정자에 세션 핸들을 저장하십시오. 이 COM 객체에 대한 포인터는 IWbemServices로의 모든 호출에 전달되어야 합니다. IANet_NetService 객체에 액세스하기 위한 호출을 수행할 때는 핸들이 명시적 인수로 사용되므로 세션 핸들이 필요하지 않습니다.

세션 핸들을 사용하여 보류 중인 변경 사항 읽기

구성을 읽을 때 컨텍스트에서 세션 핸들을 전달하면 보류 중인 업데이트가 적용된 것처럼 구성이 반환됩니다. 예를 들어, 설치되지 않은 어댑터가 누락되고 변경된 설정이 새 값을 반환합니다. 하지만 일부 객체는 **Apply**가 호출될 때까지 나타나지 않습니다. 예를 들어, 프로토콜이 해당 미니포트에 바인드될 때까지 IANet_IPProtcolEndpoints가 만들어지지 않습니다.

세션 핸들로 마무리

구성을 변경한 후 **Apply** 메서드를 호출하여 변경 사항을 커밋하십시오. 그러면 후속 작업 코드가 반환될 수 있습니다. 예를 들어, 변경 사항을 적용하려면 먼저 시스템을 재부팅하십시오.

항상 세션이 마무리된 후에 **ReleaseSessionHandle**을 호출하십시오. 그렇지 않으면 변경한 사항이 모두 무시됩니다. **Cancel** 메서드를 호출해도 변경한 사항이 모두 무시되지만, 이 경우에는 클라이언트가 세션 핸들을 방금 전에 만든 것처럼 계속 사용할 수 있습니다.

코어 이벤트 등록

응용 프로그램에서는 **IWbemServices::ExecNotificationQuery** 또는 **IWbemServices:: ExecNotificationQueryAsync**를 사용하여 이벤트 알림을 요청해야 합니다. 다음 질의는 이벤트 알림 질의를 보여주는 예입니다. 이 목록에 가능한 모든 질의가 포함되는 것은 아닙니다.

- **SELECT * FROM IANet_Event** - 모든 이벤트를 요청합니다.
- **SELECT * FROM IANet_SessionEvent** - 모든 세션 이벤트를 요청합니다.
- **SELECT * FROM IANet_InternalErrorEvent** - 모든 내부 이벤트를 요청합니다.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

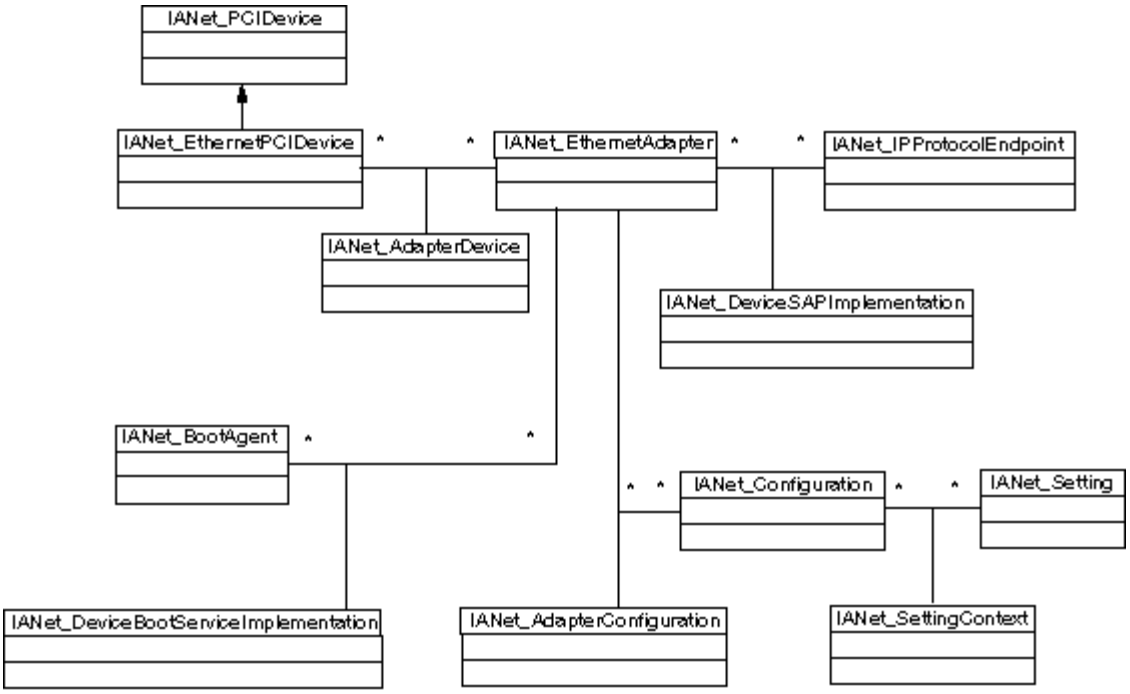
목차 페이지로 돌아가기

이더넷 어댑터 스키마: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
IANet_EthernetAdapter
IANet_IPProtocolEndpoint
IANet_BootAgent
IANet_PCIDevice
IANet_EthernetPCIDevice

개요

어댑터 스키마는 구성 가능한 여러 Intel(R) PROSet Ethernet Adapters를 모델링하는 데 사용됩니다. 이 스키마는 CIM v2.5 스키마를 기반으로 합니다.



맨 처음으로 돌아가기

IANet_EthernetAdapter

용도
IANet_EthernetAdapter는 설치된 모든 Intel PRO Network Adapters의 기능과 상태뿐 아니라 Intel 중간 드라이버를 사용하여 팀으로 구성될 수 있는 다른 모든 어댑터의 기능과 상태도 정의합니다. 클래스는 CIMv2.5에서 정의한 CIM_EthernetAdapter 수퍼클래스에서 파생됩니다. CIM_EthernetAdapter는 PermanentAddress, CurrentAddress, Speed of operation 등과 같은 일반 네트워킹 하드웨어 개념을 정의하는 추상 클래스인 CIM_NetworkAdapter에서 파생됩니다.

인스턴스
이 클래스의 인스턴스는 다음 각 경우마다 존재합니다.

- 지원/설치된 Intel NIC

Intel 다중 공급업체 팀에 참여할 수 있는 비Intel NIC
- 만들어진 Intel 어댑터 팀

인스턴스 만들기
IANet_EthernetAdapter 인스턴스는 만들 수 없습니다.

인스턴스 제거
IANet_EthernetAdapter 인스턴스를 삭제하면 실제 어댑터가 제거됩니다. 비가상 Intel 어댑터만 이 방식으로 제거할 수 있습니다. 이 작업에는 세션 핸들이 필요합니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

지원되지 않는 속성
다음 속성은 Intel PROSet에 필요하지 않으므로 지원되지 않습니다.

- AutoSense(설정으로 노출됨)
- ErrorCleared
- OtherIdentifyingInfo
- IdentifyingDescriptions
- InstallDate
- LastErrorCode
- MaxDataSize
- MaxQuiesceTime
- PowerManagementCapabilities(메서드로 노출됨)
- PowerManagementSupported(메서드로 노출됨)
- PowerOnHours
- ShortFramesReceived
- SymbolErrors
- TotalPowerOnHours

메서드
이 클래스 인스턴스는 다음 메서드를 지원합니다.

- **IdentifyAdapter** - 어댑터의 표시등을 몇 초간 깜박여서 어댑터를 식별합니다. 이 메서드는 실제 어댑터에 대해서만 작동합니다.
- **HasVLANs** - 이 어댑터의 VLAN 수를 반환합니다.
- **IsPowerMgmtSupported** - 전원 관리가 어댑터에서 지원되는지 여부를 나타냅니다.
- **GetPowerUsage** - 어댑터의 전체 전원 사용량을 감지합니다.
  0 = 일반 전원
  1 = 저전원
- **SetPowerUsage** - 어댑터의 전체 전원 사용량을 줄입니다. 시스템을 재시작하거나 드라이버를 재로드하면 전원 사용량 설정이 보존되지 않습니다. 시스템이 재시작되거나 드라이버가 재로드되면 자동으로 어댑터가 일반 전원 소비로 돌아갑니다.
- **GetPowerUsageOptions** - 선택적 전원 사용량 설정(예: 대기 시 전원 사용량, 배터리 작동 시 전원 사용량 등)을 감지합니다.
- **SetPowerUsageOptions** - 전원 사용량 옵션을 변경(예: 이 메서드를 사용하여 대기 시 전원 사용량, 배터리 작동 시 전원 사용량 등을 줄일 수 있음)합니다.
  참고**:** 전원 사용량 설정이 저장되고 이후 재부팅에 사용됩니다.
- **TestCable** - 특정 어댑터에 대한 진단 테스트를 수행합니다. 실패 시 이 메서드는 가능한 문제, 원인 및 해결 방법을 반환합니다.
- **AdvancedTestCable** - 특정 어댑터에 대한 고급 케이블 테스트를 수행합니다. 이 테스트 모음은 1000Mbps 어댑터와 함께 사용할 수 있습니다. 메서드는 테스트 이름과 해당 결과를 반환합니다.
  참고**: SpeedDuplex**가 **Auto Negotiate**로 설정되어 있지 않으면 링크 실패가 발생할 수 있습니다. 이 경우 **SpeedAndDuplexNotAutomatic** 출력 매개변수는 TRUE입니다.
- **TestLinkSpeed** - 어댑터가 최대 속도로 실행되고 있는지 여부를 확인합니다. 어댑터가 1기가비트 미만이라고 알리는 경우 메서드가 가능한 이유(예: "Link partner is not capable of running at 1000 Mbps")를 반환합니다.

맨 처음으로 돌아가기

IANet_IPProtocolEndpoint

용도
이 클래스는 시스템에 있는 프로토콜 종점의 IP 설정을 설명하는 데 사용됩니다. WMI Provider는 다른 유형의 네트워킹 프로토콜에 대한 정보는 제공하지 않습니다. 클래스는 CIM_IPProtocolEndpoint 추상 클래스에서 파생됩니다. WMI Provider는 Intel PROSet에서 관리하는 엔티티와 관련된 경우에만 프로토콜 정보를 제공합니다.

인스턴스
Intel에서 지원하는 종점(즉, Intel 어댑터, Intel 팀 가능 어댑터 및 VLAN)에 대한 각 IP 프로토콜 스택 바인딩마다 IANet_IPProtocolEndpoint 인스턴스가 존재합니다. 일부 팀 구성된 어댑터에는 자체 IP 주소가 없으므로 해당 어댑터 인스턴스와 직접 연관된 IANet_IPProtocolEndpoint가 없습니다. IANet_IPProtocolEndpoint는 운영 체제가 어댑터 또는 VLAN에 프로토콜을 바인드한 후에만 존재합니다. 일부 어댑터에 둘 이상의 IP 주소가 있을 수 있지만 이러한 어댑터는 하나의 IP 프로토콜 종점 인스턴스와만 연관됩니다. 이 고급 사용이 Intel PROSet에 필요하거나 사용되지 않으므로 Provider는 이 고급 사용을 지원하지 않습니다.

인스턴스 만들기
IANet_IPProtocolEndpoint 인스턴스는 만들 수 없습니다. 인스턴스는 운영 체제가 종점에 프로토콜을 바인드한 경우에만 존재합니다.

인스턴스 제거
IANet_IPProtocolEndpoint 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

연관
IANet_AdapterProtocolImplementation 인스턴스는 IANet_EthernetAdapter를 IANet_IPProtocolEndpoint와 연관시키는 데 사용됩니다. IANet_VLANProtocolDependency 인스턴스는 VLAN을 IANet_IPProtocolEndpoint와 연관시키는 데 사용됩니다.

참고**:** 팀은 해당 팀의 가상 어댑터를 나타내는 어댑터를 통해 종점과 연관됩니다.

지원되는 속성
다음 읽기 전용 속성은 Intel PROSet에 필요합니다.

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

지원되지 않는 속성
다음 속성은 Intel PROSet에 필요하지 않으므로 지원되지 않습니다.

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

메서드
없음

맨 처음으로 돌아가기

IANet_BootAgent

용도
이 클래스는 어댑터의 네트워크 부트 기능(일부 Intel 어댑터에서 지원하는 PXE Boot Agent의 설정)에 대한 정보를 캡처하는 데 사용됩니다. 이 클래스는 CIM_BootService에서 파생됩니다.

인스턴스

IANet_BootAgent 인스턴스는 Boot Agent가 현재 설치되어 있지 않더라도 Boot Agent 기능을 지원하는 각 어댑터마다 존재합니다.

인스턴스 만들기
IANet_BootAgent 인스턴스는 만들 수 없습니다. 인스턴스는 어댑터가 Boot Agent 기능을 지원하는 경우에만 존재합니다.

인스턴스 제거
IANet_BootAgent 인스턴스는 제거할 수 없습니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

연관
IANet_DeviceBootServiceImplementation 인스턴스는 어댑터에서 지원하는 경우 IANet_EthernetAdapter를 IANet_BootAgent와 연관시키는 데 사용됩니다.

지원되는 속성
다음 읽기 전용 속성은 Intel PROSet에 필요합니다.

- InvalidImageSignature
- Version
- UpdateAvailable
- FlashImageType

지원되지 않는 속성
다음 속성은 Intel PROSet에 필요하지 않으므로 지원되지 않습니다.

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

메서드
이 클래스의 다음 메서드를 사용하여 NIC에서 플래시 ROM을 업데이트할 수 있습니다.

uint32 ProgramFlash([IN, ValueMap {"0","1"} , Values {"Check Version", "Write Flash"}: Amended] uint32 Action, [IN] uint8 NewFlashData[], [OUT] string strErrorMessage);	이 메서드는 NIC에서 플래시 ROM을 업데이트하는 데 사용됩니다. 이 메서드를 사용하면 플래시가 업데이트되는 동안 NIC와 네트워크 간의 통신이 중지됩니다.
uint32 ReadFlash([OUT] uint8 FlashData[]);	이 메서드는 NIC에서 플래시 ROM을 읽습니다.

맨 처음으로 돌아가기

IANet_PCIDevice

용도
이 클래스는 시스템에 있는 네트워크 장치의 PCI 장치 속성을 설명하는 데 사용됩니다. 이 클래스는 CIM_PCIDevice에서 파생됩니다.

인스턴스
이 클래스의 인스턴스는 시스템에 있는 네트워크 장치인 각 PCI 카드마다 존재합니다. IA64의 경우 Intel PROSet에서 지원하는 어댑터인 PCI 장치에만 인스턴스가 있습니다.

인스턴스 만들기
IANet_PCIDevice 인스턴스는 만들 수 없습니다.

인스턴스 제거
IANet_PCIDevice 인스턴스는 제거할 수 없습니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

연관
IANet_EthernetPCIDevice의 클래스 연관을 참조하십시오.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다.

지원되지 않는 메서드
다음 속성은 WMI Provider에서 지원하지 않습니다.

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

맨 처음으로 돌아가기

IANet_EthernetPCIDevice

용도
이 클래스는 Intel PROSet에서 지원하는 이더넷 어댑터의 PCI 장치 속성을 설명하는 데 사용됩니다. 이 클래스는 IANet_PCIDevice의 하위 클래스입니다. 이 클래스는 Intel PROSet에서 지원하는 PCI 장치에 대해서만 알려진 몇 가지 추가 속성을 포함합니다.

인스턴스
이 클래스의 인스턴스는 Intel PROSet에서 지원하는 이더넷 어댑터인 각 PCI 카드마다 존재합니다.

인스턴스 만들기
IANet_EthernetPCIDevice 인스턴스는 만들 수 없습니다.

인스턴스 제거
IANet_ EthernetPCIDevice 인스턴스는 제거할 수 없습니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

연관
IANet_AdapterDevice 인스턴스는 IANet_PCIDevice를 IANet_EthernetAdapter와 연관시키는 데 사용됩니다. 가상 어댑터(즉, 팀을 나타내기 위해 만든 어댑터)에는 연관된 IANet_PCIDevice가 없습니다.

지원되지 않는 메서드
다음 속성은 WMI Provider에서 지원하지 않습니다.

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

메서드
이 클래스에 대해 지원되는 메서드가 없습니다.

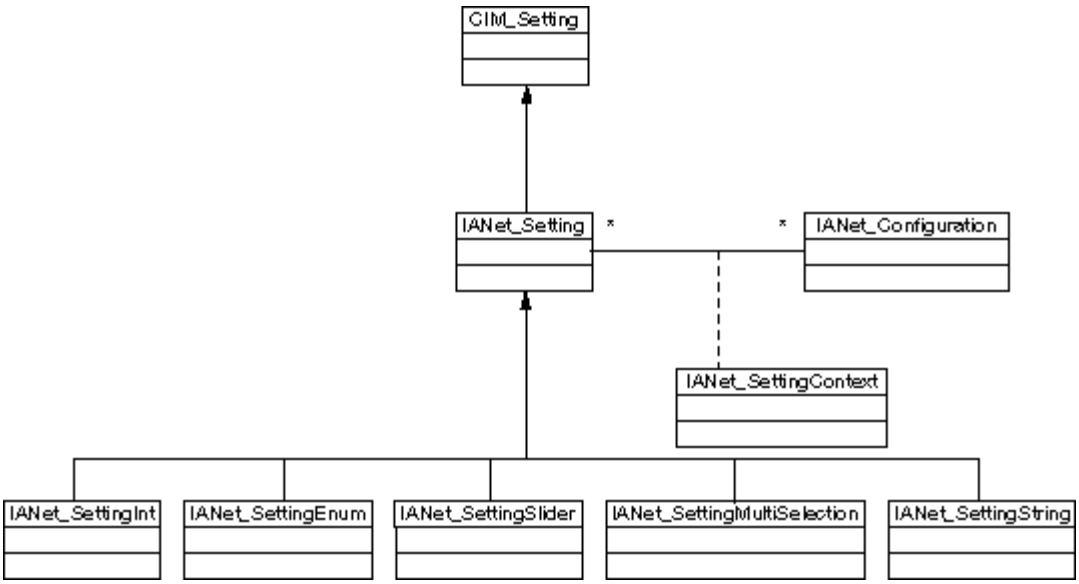
목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

설정 스키마: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
IANet_Configuration
IANet_Setting
IANet_SettingInt
IANet_SettingEnum
IANet_SettingSlider
IANet_SettingMultiSelection
IANet_SettingString

개요



맨 처음으로 돌아가기

IANet_Configuration

용도
이 클래스는 IANet_Setting 인스턴스의 컬렉션을 그룹화하는 데 사용됩니다. 이 클래스는 CIM_Configuration에서 파생됩니다.

인스턴스
각 어댑터, VLAN 또는 팀마다 연관된 IANet_Configuration 인스턴스를 여러 개씩 가질 수 있습니다(각 구성마다 다른 어댑터 사용 시나리오에 해당됨).

이 WMI and CDM Providers 릴리스의 경우에는 어댑터, VLAN 또는 팀마다 IANet_Configuration이 하나씩만 있습니다.

인스턴스 만들기
IANet_Configuration 인스턴스를 만들 수 없습니다.

인스턴스 제거
IANet_Configuration 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

연관
IANet_AdapterConfiguration 인스턴스는 각 어댑터(IANet_EthernetAdapter)를 해당 구성과 연관시키기 위해 존재합니다.
IANet_VLANConfiguration 인스턴스는 각 VLAN(IANet_VLAN)을 해당 구성과 연관시키기 위해 존재합니다.
IANet_BootAgentConfiguration 인스턴스는 각 Boot Agent(IANet_BootAgent)를 해당 구성과 연관시키기 위해 존재합니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다.

지원되지 않는 속성
없음

맨 처음으로 돌아가기

IANet_Setting

용도
이 추상 클래스는 구성에서 설정 가능한 속성에 대해 설명하는 데 사용됩니다. 이 클래스는 CIM_Setting에서 파생됩니다.

인스턴스
각 어댑터, VLAN 또는 팀의 각 설정마다 이 클래스의 별도 인스턴스가 존재합니다. 설정은 구성 간에 공유되지 않습니다.

IANet_Setting의 여러 하위 클래스가 있습니다. 이러한 하위 클래스는 설정이 가질 수 있는 값의 여러 유형과 범위에 대응합니다. 각 하위 클래스는 설정을 표시하거나 변경하는 데 사용될 수 있는 GUI의 여러 스타일에 대응합니다.

인스턴스 만들기
IANet_Setting 인스턴스를 만들 수 없습니다.

인스턴스 제거
IANet_Setting 인스턴스를 제거할 수 없습니다.

속성 수정
이 추상 클래스에는 수정할 수 있는 속성이 없지만, 하위 클래스에는 수정할 수 있는 속성이 있습니다(아래 내용 참조).

연관
각 IANet_Setting 인스턴스는 IANet_SettingContext 인스턴스를 사용하여 IANet_Configuration 인스턴스와 연관됩니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다. 설정을 변경하려면 필수 속성을 수정하고 **PutInstance**를 호출하십시오.

지원되지 않는 속성
SettingID는 사용되지 않습니다.

맨 처음으로 돌아가기

IANet_SettingInt

용도
이 클래스는 정수 값을 가지는 설정을 모델링합니다. 정수를 모델링하는 데 사용되는 여러 IANet 설정 클래스가 있습니다. 이러한 클래스 간의 차이는 GUI에서 정수를 표시하고 수정하는 방식 및 Provider에서 유효성을 검사하는 방식과 관련됩니다. IANet_SettingInt의 경우 GUI에 스핀 컨트롤이 있는 편집 상자가 표시됩니다.

인스턴스
이 클래스의 인스턴스는 정수 편집 상자로 표시되어야 하는 각 설정마다 존재합니다.

인스턴스 만들기
이 클래스의 인스턴스를 만들 수 없습니다.

인스턴스 제거
이 클래스의 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스의 수정 가능한 속성은 "CurrentValue" 속성뿐입니다. **IWbemClassObject::Put()**을 사용하여 값을 변경한 다음 **IWbemServices::PutInstance()**를 호출하여 설정을 업데이트함으로써 이 속성을 수정할 수 있습니다. Provider는 다음 사항을 검사합니다.

CurrentValue <= max
CurrentValue > = min
(**CurrentValue** - **min**)은 **Step**의 배수임

여기서 **max**, **min**, **CurrentValue** 및 **Step**은 모두 IANet_SettingInt의 속성입니다.

연관
각 IANet_SettingInt 인스턴스는 IANet_SettingContext 인스턴스를 사용하여 IANet_Configuration 인스턴스와 연관됩니다.

지원되지 않는 속성
SettingID는 사용되지 않습니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다. 설정을 변경하려면 필수 속성을 수정하고 **PutInstance**를 호출하십시오.

맨 처음으로 돌아가기

IANet_SettingEnum

용도
이 클래스는 정수 값을 가지는 설정을 모델링합니다. 정수를 모델링하는 데 사용되는 IANet 설정 클래스는 여러 가지가 있습니다. 이러한 클래스 간의 차이는 GUI에서 정수를 표시하고 수정하는 방식 및 Provider에서 유효성을 검사하는 방식과 관련됩니다. IANet_SettingEnum의 경우 GUI에 적은 수의 열거된 값으로 매핑되는 문자열 목록(예: 드롭 목록 콤보 상자)이 표시됩니다.

인스턴스
이 클래스의 인스턴스는 열거로 표시될 각 설정마다 존재합니다.

인스턴스 만들기
이 클래스의 인스턴스를 만들 수 없습니다.

인스턴스 제거
이 클래스의 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스의 수정 가능한 속성은 **CurrentValue** 속성뿐입니다. **Put()**을 사용하여 값을 변경한 다음 **PutInstance()**를 호출하여 설정을 업데이트함으로써 이 속성을 수정하십시오. Provider는 **CurrentValue Є PossibleValues[]**를 검사합니다.

연관
각 IANet_SettingEnum 인스턴스는 IANet_SettingContext 인스턴스를 사용하여 IANet_Configuration 인스턴스와 연관됩니다.

지원되지 않는 속성
SettingID는 사용되지 않습니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다. 설정을 변경하려면 필수 속성을 수정하고 **PutInstance**를 호출하십시오.

맨 처음으로 돌아가기

IANet_SettingSlider

용도
이 클래스는 정수 값을 가지는 설정을 모델링합니다. 정수를 모델링하는 데 사용되는 IANet 설정 클래스는 여러 가지가 있습니다. 이러한 클래스 간의 차이는 GUI에서 정수를 표시하고 수정하는 방식 및 Provider에서 유효성을 검사하는 방식과 관련됩니다. IANet_SettingSlider의 경우 GUI에 그래픽 방식으로 값을 선택할 수 있는 슬라이더가 표시됩니다(선택된 실제 값이 표시될 필요가 없음).

인스턴스
이 클래스의 인스턴스는 슬라이더로 표시될 각 설정마다 존재합니다.

인스턴스 만들기
이 클래스의 인스턴스를 만들 수 없습니다.

인스턴스 제거
이 클래스의 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스의 수정 가능한 속성은 **CurrentValue** 속성뿐입니다. **Put()**을 사용하여 값을 변경한 다음 **PutInstance()**를 호출하여 설정을 업데이트함으로써 이 속성을 수정하십시오. Provider는 **CurrentValue Є PossibleValues[]**를 검사합니다.

연관
각 IANet_SettingSlider 인스턴스는 IANet_SettingContext 인스턴스를 사용하여 IANet_Configuration 인스턴스와 연관됩니다.

지원되지 않는 속성
SettingID는 사용되지 않습니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다. 설정을 변경하려면 필수 속성을 수정하고 **PutInstance**를 호출하십시오.

맨 처음으로 돌아가기

IANet_SettingMultiSelection

용도
이 클래스는 옵션 목록에서 여러 옵션을 선택할 수 있게 해주는 설정을 모델링합니다. IANet_SettingMultiSelection의 경우 GUI에 임의 옵션을 선택하거나 아무 옵션도 선택하지 않을 수 있는 다중 선택 목록 상자가 표시됩니다.

인스턴스
이 클래스의 인스턴스는 다중 선택으로 표시될 각 설정마다 존재합니다.

인스턴스 만들기
이 클래스의 인스턴스를 만들 수 없습니다.

인스턴스 제거
이 클래스의 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스의 수정 가능한 속성은 **CurrentValue** 속성뿐입니다. **Put()**을 사용하여 값을 변경한 다음 **PutInstance()**를 사용하여 설정을 업데이트함으로써 이 속성을 수정하십시오. Provider는 **CurrentValue Є PossibleValues[]**를 검사합니다.

연관
각 IANet_SettingMultiSelection 인스턴스는 IANet_SettingContext 인스턴스를 사용하여 IANet_Configuration 인스턴스와 연관됩니다.

지원되지 않는 속성
SettingID는 사용되지 않습니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다. 설정을 변경하려면 필수 속성을 수정하고 **PutInstance**를 호출하십시오.

맨 처음으로 돌아가기

IANet_SettingString

용도
이 클래스는 자유 형식 문자열 값을 입력할 수 있게 해주는 설정을 모델링합니다. IANet_SettingMultiSelection의 경우 GUI에 편집 상자가 표시됩니다.

인스턴스
이 클래스의 인스턴스는 편집 상자로 표시될 각 설정마다 존재합니다.

인스턴스 만들기
이 클래스의 인스턴스를 만들 수 없습니다.

인스턴스 제거
이 클래스의 인스턴스를 제거할 수 없습니다.

속성 수정
이 클래스의 수정 가능한 속성은 **CurrentValue** 속성뿐입니다. **Put()**을 사용하여 값을 변경한 다음 **PutInstance()**를 호출하여 설정을 업데이트함으로써 이 속성을 수정하십시오.

연관
각 IANet_SettingMultiSelection 인스턴스는 IANet_SettingString 인스턴스를 사용하여 IANet_ElementConfiguration 인스턴스와 연관됩니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다.

지원되지 않는 속성
SettingID는 사용되지 않습니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다. 설정을 변경하려면 필수 속성을 수정한 다음 **PutInstance**를 호출하십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

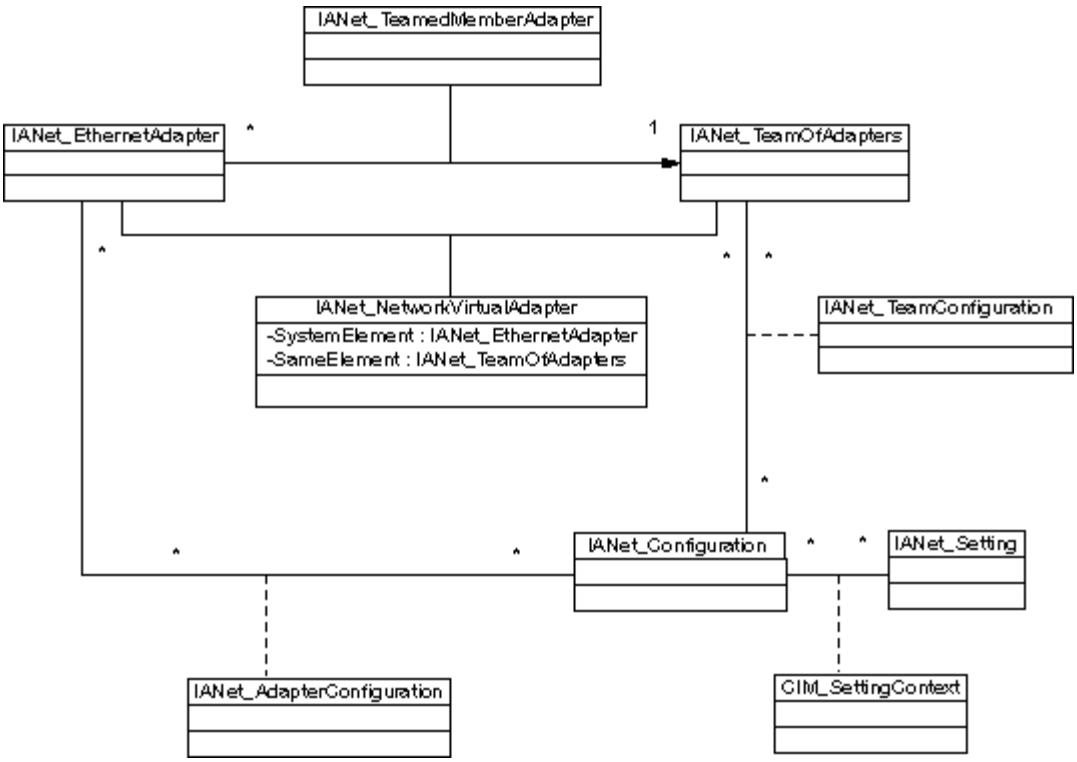
목차 페이지로 돌아가기

팀 스키마: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
IANet_TeamOfAdapters
IANet_TeamedMemberAdapter
IANet_NetworkVirtualAdapter

개요

팀 스키마에서는 이더넷 어댑터를 팀으로 그룹화하는 방법에 대해 설명합니다.



맨 처음으로 돌아가기

IANet_TeamOfAdapters

용도
이 클래스는 CIM_RedundancyGroup 클래스를 구현합니다. 이 클래스에는 팀 유형, 팀 내의 어댑터 수, 팀에 포함될 수 있는 최대 어댑터 수에 대해 설명하는 구성원이 있습니다.

인스턴스
각 Intel 팀마다 이 클래스의 인스턴스가 있습니다.

인스턴스 만들기
빈 팀을 만들려면 IANet_TeamOfAdapters 인스턴스를 만드십시오. **IWbemServices::PutInstance()**를 호출하여 Provider에 객체를 만들기 전에 올바른 **TeamingMode**를 설정해야 합니다. Provider는 새 객체의 객체 경로를 포함하는 문자열을 반환합니다.

인스턴스 제거
팀을 제거하려면 IANet_TeamOfAdapters 인스턴스를 삭제하십시오. Provider는 팀 구성원에 대한 연관을 삭제하고 팀의 가상 어댑터 및 설정을 삭제합니다.

속성 수정
Put()을 사용하여 **TeamingMode** 속성 값을 변경한 다음 **PutInstance()**를 호출하여 팀을 갱신하십시오.

연관
팀의 각 어댑터는 IANet_TeamMemberAdapter 인스턴스를 사용하여 해당 팀의 IANet_TeamOfAdapters 인스턴스와 연관됩니다. 팀의 가상 어댑터는 IA_NetNetworkVirtualAdapter 인스턴스를 사용하여 이 클래스와 연관됩니다.

메서드
이 클래스 인스턴스는 다음 메서드를 지원합니다.

TestSwitchConfiguration - 스위치 구성을 테스트하여 팀이 스위치와 함께 올바르게 작동하는지 확인합니다. 이 테스트를 사용하여 링크 대상(즉, 다른 어댑터, 허브, 스위치 등과 같이 어댑터가 연결된 장치)이 선택된 어댑터 팀 구성 모드를 지원하는지 검사할 수 있습니다. 예를 들어, 어댑터가 Link Aggregration 팀의 구성원인 경우 이 테스트를 사용하여 어댑터에 연결된 링크 대상이 Link Aggregation을 지원하는지 확인할 수 있습니다.

맨 처음으로 돌아가기

IANet_TeamedMemberAdapter

용도
이 클래스는 어댑터를 팀과 연관시키는 데 사용되고 팀 내의 어댑터 기능을 확인하며 어댑터가 팀에서 현재 활성화되도록 설정합니다. 이 클래스는 CIM 클래스 CIM_NetworkAdapterRedundancyComponent를 구현합니다.

인스턴스
이 클래스의 인스턴스는 팀 구성원인 각 어댑터마다 존재합니다.

인스턴스 만들기
팀에 어댑터를 추가하려면 IANet_TeamedMemberAdapter 인스턴스를 만들어 어댑터를 팀과 연관시키십시오.

인스턴스 제거
팀에서 어댑터를 제거하려면 IANet_ TeamedMemberAdapter 인스턴스를 제거하십시오. 어댑터는 더 이상 팀에 속하지 않으며 **Apply()** 함수가 호출된 후 IP 프로토콜 종점에 바인드될 수 있습니다.

속성 수정
이 클래스의 **AdapterFunction** 속성을 수정하여 어댑터가 팀 내에서 사용되는 방법에 대해 설명할 수 있습니다.

연관
연관 클래스입니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다.

맨 처음으로 돌아가기

IANet_NetworkVirtualAdapter

용도
이 클래스는 팀의 IANet_TeamOfAdapters를 해당 팀의 가상 어댑터를 나타내는 IANet_EthernetAdapter와 연관시키는 데 사용됩니다. 이 클래스는 CIM 클래스 CIM_ CIM_NetworkVirtualAdapter를 구현합니다.

인스턴스
이 클래스의 인스턴스는 가상 어댑터에 바인드된 각 Intel 팀마다 존재합니다.

인스턴스 만들기
이 클래스의 인스턴스를 만들 수 없습니다. 팀을 만들려면 IANet_TeamOfAdapters 인스턴스를 만드십시오. 이 클래스는 유효한 세션의 컨텍스트 내에서 **IANet_NetService .Apply()**를 호출하고 IANet_EthernetAdapter 인스턴스가 만들어진 후에야 존재합니다.

인스턴스 제거
이 클래스의 인스턴스를 삭제할 수 없습니다.

연관
연관 클래스입니다.

메서드
이 클래스에 대해 지원되는 메서드가 없습니다.

제한 및 책임 부인을 자세히 읽어보십시오.

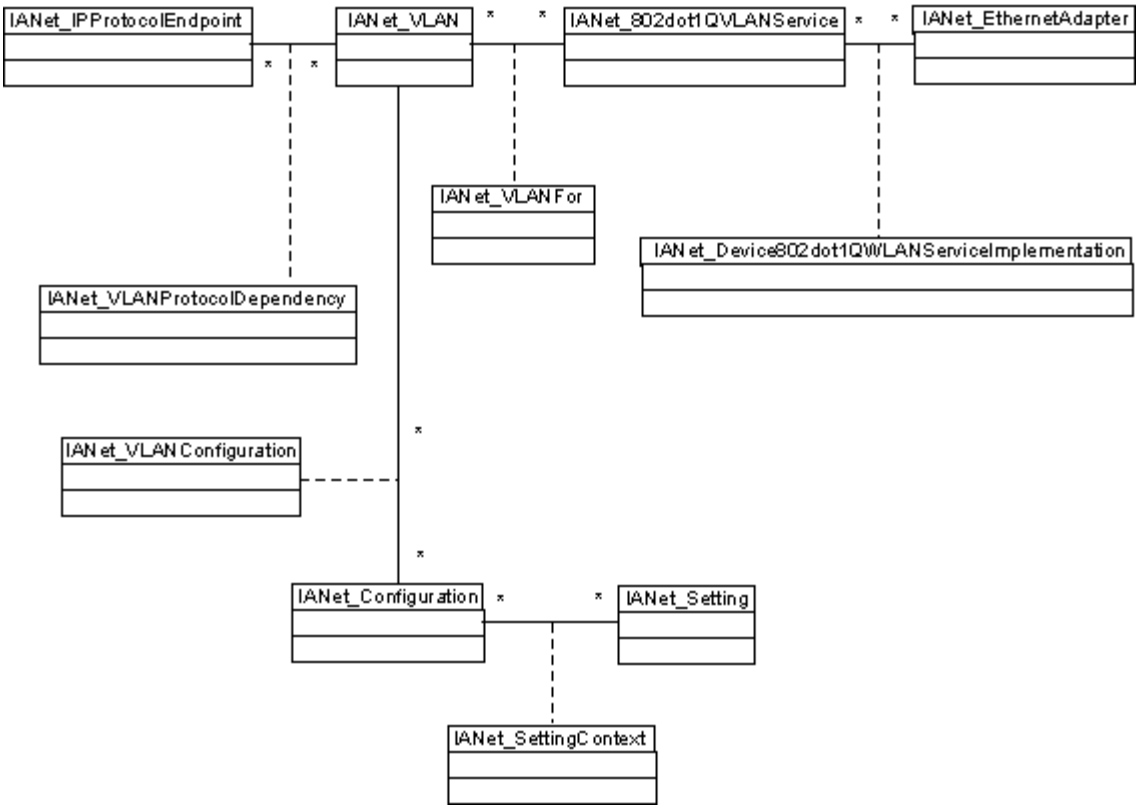
목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

VLAN 스키마: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
IANet_802dot1QVLANService
IANet_VLAN

개요



맨 처음으로 돌아가기

IANet_802dot1QVLANService

용도
이 클래스는 네트워크 어댑터의 IEEE 802.1Q 속성을 보유하는 데 사용됩니다. 이 클래스는 CIM 클래스 CIM_802dot1QVLANService를 구현합니다.

인스턴스
이 클래스의 인스턴스는 IEEE 802.1Q를 지원하는 각 어댑터 또는 팀마다 존재합니다. 각 어댑터는 IANet_802dot1QVLANService를 하나씩만 가질 수 있습니다. 다중 공급업체 FAT 팀과 같은 일부 팀은 이 서비스를 지원하지 않습니다.

장애
유효한 세션의 컨텍스트 내에서 열거한 경우를 제외하고 VLAN이 없는 팀에는 VLAN 서비스도 없습니다. 팀의 경우 인스턴스는 다음과 같은 상황에서만 나타납니다

- 팀에 이미 VLAN이 있는 경우
- 팀에 VLAN이 없고 이 클래스를 열거하는 동안 컨텍스트에서 세션 핸들을 사용하는 경우

인스턴스 만들기
이 클래스의 인스턴스를 만들 수 없습니다. 어댑터와 연관된 인스턴스가 없으면 해당 어댑터가 이 서비스를 지원하지 않습니다.

인스턴스 제거
이 클래스의 인스턴스를 삭제할 수 없습니다.

속성 수정
이 클래스에는 수정할 수 있는 속성이 없습니다.

연관
이 클래스의 각 인스턴스는 IANet_DeviceServiceImplementation을 사용하여 하나의 IANet_EthernetAdapter와 연관됩니다.

IANet_802dot1QVLANService의 각 인스턴스는 여러 VLAN을 지원할 수 있습니다. 각 VLAN은 IANet_VLANFor 연관을 사용하여 인스턴스와 연관됩니다.

메서드
uint16 CreateVLAN([in] uint32 VLANNumber, [in] string Name, [out] IANet_VLAN REF VLANpath); - 어댑터나 팀에 VLAN을 만드는 데 사용됩니다. 클라이언트는 VLAN 번호와 VLAN 이름을 제공해야 하며 새로 만든 VLAN의 객체 경로를 가져옵니다.

맨 처음으로 돌아가기

IANet_VLAN

용도
이 클래스는 각 Intel VLAN에 대한 정보를 보유합니다. 이 클래스는 CIM_VLAN을 구현합니다.

인스턴스
이 클래스의 인스턴스는 각 Intel VLAN마다 존재합니다.

인스턴스 만들기
VLAN을 만들려면 해당 IANet_802dot1QVLANService 인스턴스에서 **CreateVLAN**을 호출하십시오.

인스턴스 제거
이 클래스의 인스턴스를 제거하여 해당 VLAN을 제거할 수 있습니다.

속성 수정
VLANNumber 및 Caption 속성을 수정할 수 있습니다.

연관
각 인스턴스는 IANet_802dot1QVLANService와 연관되므로 IANet_VLANFor 클래스를 사용하여 하나의 IANet_EthernetAdapter 인스턴스와 연관됩니다.

각 인스턴스가 여러 IANet_Configuration 인스턴스와 연관되어 일련의 VLAN 설정을 그룹화할 수 있습니다. 이 Provider 릴리스의 경우 VLAN마다 IANet_Configuration 객체가 하나씩만 있습니다.

각 인스턴스가 하나의 IANet_IPProtocolEndpoint와 연관되어 IANet_VLANProtocolDependency 클래스를 통해 VLAN의 IP 설정을 제공할 수 있습니다.

메서드
없음

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

현재 구성 가져오기: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

실제 어댑터 가져오기
PCI 장치 가져오기
어댑터 설정 가져오기
팀 구성 가져오기
팀 설정 가져오기
VLAN 구성 가져오기
VLAN 설정 가져오기
IP 프로토콜 정보 가져오기
Boot Agent 정보 가져오기
Boot Agent 설정 가져오기

클라이언트는 세션 핸들을 가져오지 않고도 현재 구성을 읽을 수 있습니다. 클라이언트가 NULL 컨텍스트를 사용할 수 있지만, 이 경우 오류 메시지가 관리 대상 컴퓨터의 기본 언어로 표시됩니다. 다음 표에서 { } 안에 있는 항목은 객체 경로로, 이전 WQL 질의에서 가져온 것으로 간주됩니다. 클라이언트는 질의를 수행하지 않고도 객체 경로를 구성할 수 있습니다. 모든 객체의 **__PATH** 속성은 해당 객체의 객체 경로를 포함합니다.

다음 모든 사례에서는 **IWbemServices::ExecQuery** 또는 **IWbemServices::ExecQueryAsync** 메서드를 사용하여 WQL 질의를 실행합니다.

실제 어댑터 가져오기

작업	**WQL** 질의	결과 클래스	설명
모든 어댑터 열거	SELECT * FROM IANet_EthernetAdapter	IANet_EthernetAdapter	모든 IANet_EthernetAdapters를 반환합니다. IWbemServices::CreateInstanceEnumAsync와 같습니다.
가상 어댑터인지 확인	ASSOCIATORS OF {어댑터 경로} WHERE AssocClass = IANet_NetworkVirtualAdapter	IANet_TeamOfAdapters	질의가 아무 클래스도 반환하지 않으면 실제 어댑터입니다.
팬텀 어댑터인지 확인	ASSOCIATORS OF {어댑터 경로} WHERE ResultClass = IANet_EthernetPCIDevice	IANet_EthernetPCIDevice	가상 어댑터가 아니고 이 질의가 아무 객체도 반환하지 않으면 팬텀 어댑터입니다.

어댑터의 기본 클래스는 IANet_EthernetAdapter입니다. 이 클래스가 실제 어댑터와 가상 어댑터 모두에 사용되므로 클라이언트는 실제 어댑터와 가상 어댑터를 구분하는 방법을 알아야 합니다.

맨 처음으로 돌아가기

PCI 장치 가져오기

기본 클래스는 IANet_EthernetPCIDevice, IANet_PCIDevice 및 IANet_AdapterDevice(어댑터를 해당 장치와 연관시키는 연관 클래스)입니다.

이 경우 연관 클래스는 아무 데이터도 포함하지 않습니다. 즉, 자체적으로 값을 갖지 않습니다. IANet_EthernetPCIDevice는 IANet_PCIDevice에서 상속하고 이더넷 어댑터인 PCI 장치에만 적용되는 추가 속성을 포함합니다.

작업	**WQL** 질의	결과 클래스	설명
네트워크 PCI 장치 열거	SELECT * FROM IANet_PCIDevice	IANet_PCIDevice	질의는 모뎀과 Intel(R) PROSet이 구성하지 않는 다른 장치를 반환할 수 있습니다. IWbemServices::CreateInstanceEnumAsync와 같습니다.
Intel(R) PROSet 구성 어댑터에서 사용하는 모든 PCI 장치 열거	SELECT * FROM IANet_EthernetPCIDevice	IANet_EthernetPCIDevice	질의는 Intel(R) PROSet에서 관리하는 PCI 장치만 반환합니다. IWbemServices::CreateInstanceEnumAsync와 같습니다.
어댑터가 설치되었는지 확인	해당 없음	IANet_EthernetPCIDevice	IANet_EthernetPCIDevice의 'Availability' 속성을 검사합니다. 10 - 'Not Installed'와 같으면 장치가 설치되지 않은 것입니다. 참고: 이 단계에서는 장치에 대해 알려진 정보가 적습니다.
어댑터와 연관된 PCI 장치 가져오기	ASSOCIATORS OF { IANet_EthernetAdapter 경로} WHERE ResultClass = IANet_EthernetPCIDevice	IANet_EthernetPCIDevice	아무 객체도 반환하지 않으면 가상 어댑터이거나 팬텀 어댑터입니다.
PCI 장치와 연관된 어댑터 가져오기	ASSOCIATORS OF {이더넷 PCI 장치 경로} WHERE ResultClass = IANet_EthernetAdapter	IANet_EthernetAdapter	이 질의는 Provider용으로 최적화되지 않으므로 클라이언트는 어댑터부터 시작하는 것이 좋습니다.

맨 처음으로 돌아가기

어댑터 설정 가져오기

설정 객체는 어댑터와 직접 연관되지 않고 CIM 표준에 따라 어댑터와 연관된 구성 객체와 연관됩니다.

이 스키마 부분과 관련된 클래스는 IANet_EthernetAdapter, IANet_Configuration, IANet_SettingInt, IANet_SettingString, IANet_SettingEnum, IANet_SettingMultiSelection 및 IANetSettingSlider입니다.

IANet_AdapterConfiguration 및 IANet_SettingContext 연관 클래스는 실제 데이터를 포함하지 않고 설정과 해당 상위 객체를 연결하는 역할을 합니다.

작업	**WQL** 질의	결과 클래스	설명
어댑터의 구성 객체 가져오기	ASSOCIATORS OF {IANet_EthernetAdapter 경로} WHERE ResultClass = IANet_Configuration	IANet_Configuration	정확히 하나의 객체를 반환합니다. 아무 설정이 없어도 항상 하나의 구성 객체가 존재합니다. 이 객체의 객체 경로는 다음 질의에서 사용됩니다.
어댑터와 연관된 설정 가져오기	ASSOCIATORS OF {IANet_Configuration 경로} WHERE AssocClass = IANet_SettingContext	IANet_SettingInt, IANet_SettingString, IANet_SettingEnum, IANet_SettingMultiSelection 및 IANet_SettingSlider의 조합	어댑터와 연관된 모든 설정 클래스를 반환합니다. 클라이언트는 __CLASS 속성을 사용하여 각 설정의 해당 유형을 확인해야 합니다.

맨 처음으로 돌아가기

팀 구성 가져오기

팀 구성 스키마의 기본 클래스는 IANet_EthernetAdapter, IANet_TeamOfAdapters, IANet_NetworkVirtualAdapter 및 IANet_TeamedMemberAdapter입니다.

이 스키마의 문제는 IANet_EthernetAdapter 인스턴스가 각 실제 어댑터와 각 가상 어댑터마다 존재한다는 점입니다. 클라이언트는 팀의 가상 어댑터와 구성원 어댑터를 구분할 수 있어야 합니다.

IANet_NetworkVirtualAdapter 연관 클래스는 유용한 데이터를 포함하지 않습니다. 실제로 클라이언트는 이 연관의 종점에만 관심을 가집니다. IANet_TeamedMemberAdapter는 구성원 어댑터가 팀 내에서 사용되는 방법에 대한 유용한 데이터를 포함하지 않습니다.

작업	**WQL** 질의	결과 클래스	설명
모든 팀 열거	SELECT * FROM IANet_TeamOfAdapters	IANet_TeamOfAdapters	각 팀당 하나씩 IANet_TeamOfAdapters 인스턴스가 있습니다. IWbemServices::CreateInstanceEnumAsync와 같습니다.
팀의 가상 어댑터 가져오기	ASSOCIATORS OF {IANet_TeamOfAdapters 경로} WHERE AssocClass = IANet_NetworkVirtualAdapter	IANet_EthernetAdapter	팀에 있는 가상 어댑터의 어댑터 객체만 반환합니다. 팀이 만들어졌지만 Apply가 호출되지 않은 경우에는 이 어댑터가 없습니다. 구성 업데이트에 대한 자세한 내용은 아래를 참조하십시오.
팀의 구성원 어댑터 열거	ASSOCIATORS OF {IANet_TeamOfAdapters 경로} WHERE AssocClass = IANet_TeamedMemberAdapter	IANet_EthernetAdapter	팀에 있는 어댑터를 반환하지만 해당 어댑터의 역할에 대해서는 설명하지 않습니다.
팀에 있는 어댑터의 역할 확인	REFERENCES OF {IANet_EthernetAdapter 경로} WHERE ResultClass = IANet_TeamedMemberAdapter	IANet_TeamedMemberAdapter	이 클래스는 구성원 어댑터가 팀에 어떻게 관련되는지와 팀 내에서의 현재 상태에 대한 정보를 포함합니다.

맨 처음으로 돌아가기

팀 설정 가져오기

설정 객체는 팀과 직접 연관되지 않고 CIM 표준에 따라 팀의 가상 IANet_EthernetAdapter와 연관된 구성 객체와 연관됩니다. 동일한 구성 객체 또한 팀의 IANet_TeamOfAdapters 객체와 연관됩니다.

이 스키마 부분과 관련된 클래스는 IANet_EthernetAdapter, IANet_TeamOfAdapters, IANet_Configuration, IANet_SettingInt, IANet_SettingString, IANet_SettingEnum, IANet_SettingMultiSelection 및 IANetSettingSlider입니다.

IANet_AdapterConfiguration 및 IANet_SettingContext 연관 클래스는 실제 데이터를 포함하지 않고 설정과 해당 상위 객체를 연결하는 역할을 합니다. 이 사례는 어댑터 설정 사례와 정확히 동일합니다.

작업	**WQL** 질의	결과 클래스	설명
팀의 구성 객체 가져오기, 가상 어댑터로 시작	ASSOCIATORS OF {IANet_EthernetAdapter 경로} WHERE ResultClass = IANet_Configuration	IANet_Configuration	정확히 하나의 객체를 반환합니다. 아무 설정이 없어도 항상 하나의 구성 객체가 존재합니다. 이 객체의 객체 경로는 다음 질의에서 사용됩니다.
팀의 구성 객체 가져오기, 어댑터 팀으로	ASSOCIATORS OF {IANet_TeamOfAdapters 경로} WHERE ResultClass =		

시작	IANet_Configuration		
어댑터와 연관된 설정 가져오기	ASSOCIATORS OF {IANet_Configuration 경로} WHERE AssocClass = IANet_SettingContext	IANet_SettingInt, IANet_SettingString, IANet_SettingEnum, IANet_SettingMultiSelection 및 IANet_SettingSlider의 조합	어댑터와 연관된 모든 설정 클래스를 반환합니다. 클라이언트는 __CLASS 속성을 사용하여 각 설정의 해당 유형을 확인해야 합니다.

맨 처음으로 돌아가기

VLAN 구성 가져오기

VLAN을 지원하는 각 어댑터는 IANet_Device802do1QVVLANServiceImplementation 연관 클래스를 사용하는 IANet_802dot1QVLANService와 연관됩니다. 어댑터와 연관된 이 클래스의 인스턴스가 없으면 해당 어댑터가 VLAN을 지원하지 않는 것입니다.

각 VLAN은 IANet_VLAN 인스턴스로 표현됩니다. VLAN은 어댑터와 직접 연관되지 않고 어댑터의 IANet_802dot1QVLANService와 연관됩니다.

IANet_VLANFor 연관 클래스는 각 VLAN 인스턴스를 올바른 IANet_802dot1QVLANService와 연관시키는 데 사용됩니다. 이 클래스는 사용자에 대한 유용한 데이터를 포함하지 않습니다.

작업	**WQL** 질의	결과 클래스	설명
어댑터와 연관된 802.1q VLAN 서비스 객체 가져오기	ASSOCIATORS OF {IANet_EthernetAdapter 경로} WHERE ResultClass = IANet_802dot1QVLANService	IANet_802dot1QVLANService	하나의 객체를 반환하거나 아무 객체도 반환하지 않습니다.
어댑터의 VLAN 가져오기	ASSOCIATORS OF {IANet_802dot1QVLANService 경로} WHERE ResultClass = IANet_VLAN	IANet_VLAN	설치된 VLAN이 없으면 아무 객체도 반환하지 않을 수 있습니다.

맨 처음으로 돌아가기

VLAN 설정 가져오기

설정 객체는 VLAN과 직접 연관되지 않고 CIM 표준에 따라 VLAN의 IANet_VLAN 객체와 연관된 구성 객체와 연관됩니다.

이 스키마 부분과 관련된 클래스는 IANet_VLAN, IANet_Configuration, IANet_SettingInt, IANet_SettingString, IANet_SettingEnum, IANet_SettingMultiSelection 및 IANetSettingSlider입니다.

IANet_VLANConfiguration 및 IANet_SettingContext 연관 클래스는 실제 데이터를 포함하지 않고 설정과 해당 상위 객체를 연결하는 역할을 합니다. 이 사례는 어댑터 설정 사례와 정확히 동일합니다.

작업	**WQL** 질의	결과 클래스	설명
VLAN의 구성 객체 가져오기	ASSOCIATORS OF {IANet_VLAN 경로} WHERE ResultClass = IANet_Configuration	IANet_Configuration	정확히 하나의 객체를 반환합니다. 아무 설정이 없어도 항상 하나의 구성 객체가 존재합니다. 이 객체의 객체 경로는 다음 질의에서 사용됩니다.
VLAN과	ASSOCIATORS OF	IANet_SettingInt, IANet_SettingString,	VLAN과 연관된 모든 설정 클래스를 반환합니

연관된 설정 가져오기	{IANet_Configuration 경로} WHERE AssocClass = IANet_SettingContext	IANet_SettingEnum, IANet_SettingMultiSelection 및 IANet_SettingSlider의 조합	다. 클라이언트는 __CLASS 속성을 사용하여 각 설정의 유형을 확인해야 합니다.

맨 처음으로 돌아가기

IP 프로토콜 정보 가져오기

Provider는 어댑터, VLAN 및 팀과 연관된 IP 프로토콜 종점에 대해 제한된 몇 가지 정보를 제공합니다. 다른 프로토콜은 지원되지 않습니다.

프로토콜 정보를 포함하는 기본 클래스는 IANet_IPProtocolEndpoint입니다. 연관 클래스에는 IANet_VLANProtocolDependency와 IANet_AdapterProtocolImplementation의 두 가지가 있습니다. 팀의 IP 종점을 가져오려면 먼저 팀의 가상 IANet_EthernetAdapter를 가져오십시오. 즉, IP 종점은 이 인스턴스와 연관됩니다.

작업	**WQL** 질의	결과 클래스	설명
어댑터와 연관된 IP 프로토콜 종점 가져오기	ASSOCIATORS OF {IANet_EthernetAdapter 경로} WHERE ResultClass = IANet_IPProtocolEndpoint	IANet_IPProtocolEndpoint	일부 어댑터에 둘 이상의 IP 주소가 있을 수 있지만 이러한 어댑터는 하나의 IP 프로토콜 종점 인스턴스와만 연관됩니다.
VLAN과 연관된 IP 프로토콜 종점 가져오기	ASSOCIATORS OF {IANet_VLAN 경로} WHERE ResultClass = IANet_IPProtocolEndpoint	IANet_IPProtocolEndpoint	VLAN과 연관된 하나의 IP 프로토콜 종점이 있습니다.

맨 처음으로 돌아가기

Boot Agent 정보 가져오기

플래시 ROM에서 Boot Agent를 지원할 수 있는 각 어댑터는 IANet_DeviceBootServiceImplementation 연관 클래스를 사용하는 IANet_BootAgent 인스턴스와 연관됩니다.

작업	**WQL** 질의	결과 클래스	설명
어댑터와 연관된 Boot Agent 가져오기	ASSOCIATORS OF {IANet_EthernetAdapter 경로} WHERE ResultClass = IANet_BootAgent	IANet_BootAgent	다음 읽기 전용 속성은 이 어댑터의 부트 ROM 이미지에 대한 정보를 제공합니다. InvalidImageSignature, Version, UpdateAvailable, FlashImageType

맨 처음으로 돌아가기

Boot Agent 설정 가져오기

설정 객체는 Boot Agent와 직접 연관되지 않고 CIM 표준에 따라 Boot Agent와 연관된 구성 객체와 연관됩니다.

이 스키마 부분과 관련된 클래스는 IANet_BootAgent, IANet_Configuration, IANet_SettingInt, IANet_SettingString, IANet_SettingEnum, IANet_SettingMultiSelection 및 IANetSettingSlider입니다.

IANet_BootAgentConfiguration 및 IANet_SettingContext 연관 클래스는 실제 데이터를 포함하지 않고 설정과 해당 상위 객체를 연결하는 역할을 합니다.

작업	**WQL** 질의	결과 클래스	설명
Boot Agent의 구성 객체 가져오기	ASSOCIATORS OF { IANet_BootAgent 경로} WHERE ResultClass = IANet_Configuration	IANet_Configuration	정확히 하나의 객체를 반환합니다. 아무 설정이 없어도 항상 하나의 구성 객체가 존재합니다. 이 객체의 객체 경로는 다음 질의에서 사용됩니다.
Boot Agent와 연관된 설정 가져오기	ASSOCIATORS OF {IANet_Configuration 경로} WHERE AssocClass = IANet_SettingContext	IANet_SettingInt, IANet_SettingString, IANet_SettingEnum, IANet_SettingMultiSelection 및 IANet_SettingSlider의 조합	어댑터와 연관된 모든 설정 클래스를 반환합니다. 클라이언트는 __CLASS 속성을 사용하여 각 설정의 해당 유형을 확인해야 합니다.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

구성 업데이트: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

개요
어댑터, 팀 또는 VLAN 설정 변경
새 (빈) 팀 만들기
팀에 어댑터 추가
팀에서 어댑터 제거
팀 삭제
팀 모드 변경
팀 내의 어댑터 우선 순위 변경
어댑터 제거
VLAN 만들기
VLAN 속성 변경
VLAN 삭제
Boot Agent 업데이트

개요

대부분의 경우 구성을 업데이트하려면 클라이언트 응용 프로그램에서 IANet_NetService 클래스의 세션 핸들을 가져와서 IWbemContext 컨텍스트 객체에 저장해야 합니다. 구성 변경은 IANet_NetService에서 **Apply** 메서드가 호출될 때만 발생합니다. 이 요구 사항에는 다음 몇 가지 예외가 따릅니다.

- Boot Agent 클래스의 변경 사항은 변경하자마자 적용되며 세션 핸들은 필요하지 않습니다.
- 특정 메서드 호출(예: 어댑터 식별)은 **Apply** 호출 전에 작업을 실행시킵니다.

일부 작업의 경우 컨텍스트에서 PreCheck 한정자를 사용하여 해당 작업이 허용되는지 확인할 수 있습니다. 이는 필요한 경우 사용자 인터페이스에서 특정 컨트롤이나 메뉴 항목을 비활성화할 수 있도록 하기 위한 것입니다.

맨 처음으로 돌아가기

어댑터, 팀 또는 **VLAN** 설정 변경

어댑터, 팀 또는 VLAN을 변경할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

어댑터, VLAN 또는 팀 설정을 변경하려면 먼저 클라이언트가 변경할 설정의 객체 경로를 가져와야 합니다. 객체의 설정을 열거한 다음 해당 설정의 **__PATH** 속성을 저장하는 것이 이 작업에 가장 좋은 방법입니다(위 내용 참조).

클라이언트가 가장 쉽게 설정을 업데이트할 수 있는 방법은 다음과 같습니다.

□□□ WMI에서 설정 객체의 인스턴스를 가져옵니다.
□□□ CurrentValue 속성을 수정합니다(**IWbemClassObject::Put()** 사용).
□□□ **IWbemServices::PutInstance()**를 호출하여 수정된 인스턴스를 WMI Provider에 다시 전달합니다. **PutInstance**는 WBEM_FLAG_UPDATE_ONLY 플래그를 사용하여 호출해야 합니다.

WMI Provider는 CurrentValue를 검증하고 검증이 실패한 경우 WBEM_E_FAIL을 반환합니다. 정확한 실패 이유는 IANet_ExtendedStatus 객체의 Description 속성에서 반환됩니다.

설정별 설명은 다음을 포함합니다.

- 정수 설정값이 최소값보다 작습니다.
- 정수 설정값이 최대값보다 큽니다.
- 허용된 간격의 정수 설정값이 아닙니다.
- 문자열 설정 길이가 최대값보다 큽니다.
- 허용되지 않은 설정값입니다.

마지막 설명은 IANet_SettingEnum, IANet_SettingSlider 또는 IANet_SettingMultiSelection의 현재 값이 허용되는 값 중 하나가 아닐 때 반환됩니다.

클라이언트가 변경할 수 있는 설정의 속성은 CurrentValue뿐입니다. WMI Provider는 다른 모든 값의 변경 사항을 무시합니다.

설정 클래스에 대해 지원되는 메서드가 없습니다. 설정을 변경하려면 CurrentValue 속성을 수정하고 **PutInstance**를 호출하십시오.

맨 처음으로 돌아가기

새 (빈) 팀 만들기

새 팀을 만들 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

새 팀을 만들려면 IANet_TeamOfAdapters 인스턴스를 만드십시오. 즉, **IWbemServices::GetObject()**를 사용하여 IANet_TeamOfAdapters의 클래스 객체를 가져온 다음 **IWbemServices::SpawnInstance()**를 사용하여 이 객체의 인스턴스를 만드십시오.

그런 다음 **IWbemClassObject::Put**을 사용하여 인스턴스의 TeamMode 속성을 원하는 팀 유형(예: AFT)으로 설정하십시오. 마지막으로 **IWbemServices::PutInstance()**를 호출하여 팀을 만들면서 WBEM_FLAG_CREATE_ONLY 플래그를 전달하십시오.

새 팀의 객체 경로는 호출 완료 시 다시 전달되는 **IWbemCallResultObject**에 저장됩니다. **IWbemCallResult::GetResultString** 메서드는 새 객체 경로를 가져옵니다.

이 작업이 실패한 경우 클라이언트는 IANet_ExtendedStatus를 검사하여 실패 이유를 확인해야 합니다.

팀의 가상 IANet_EthernetAdapter 및 IANet_IPProtocolEndpoint 클래스는 **Apply**를 호출한 후에야 사용할 수 있습니다. 새 IANet_TeamOfAdapters와 연관된 IANet_Configuration 객체를 사용하여 팀 설정에 액세스할 수 있습니다.

맨 처음으로 돌아가기

팀에 어댑터 추가

팀에 어댑터를 추가할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

팀에 어댑터를 추가하려면 IANet_TeamedMemberAdapter 인스턴스를 만드십시오. 즉, **IWbemServices::GetObject()**를 사용하여 IANet_TeamedMemberAdapter의 클래스 객체를 가져온 다음 **IWbemServices::SpawnInstance()**를 사용하여 이 객체의 인스턴스를 만드십시오.

IWbemClassObject::Put()을 사용하여 객체의 다음 속성을 설정해야 합니다.

- GroupComponent를 어댑터가 추가될 IANet_TeamOfAdapters의 전체 객체 경로로 설정해야 합니다.
- PartComponent를 팀에 추가될 IANet_EthernetAdapter의 전체 객체 경로로 설정해야 합니다.

팀 내의 어댑터 우선 순위도 설정할 수 있습니다. 마지막으로, **IWbemServices::PutInstance()**를 호출하여 팀에 어댑터를 추가하여 WBEM_FLAG_CREATE_ONLY 플래그를 전달하십시오. 이 작업이 실패한 경우 IANet_ExtendedStatus에서 오류 코드를 검사하십시

오.

맨 처음으로 돌아가기

팀에서 어댑터 제거

팀에서 어댑터를 제거할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

팀에서 어댑터를 제거하려면 **IWbemServices::DeleteInstance()**를 사용하여 어댑터를 팀에 연관시키는 IANet_TeamedMemberAdapter 인스턴스를 삭제하십시오. 이 작업이 실패한 경우 IANet_ExtendedStatus에서 오류 코드를 검사하십시오.

맨 처음으로 돌아가기

팀 삭제

팀을 삭제할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

팀을 삭제하려면 **IWbemServices::DeleteInstance()**를 사용하여 IANet_TeamOfAdapters 인스턴스를 삭제하십시오. 이 작업이 실패한 경우 IANet_ExtendedStatus에서 오류 코드를 검사하십시오.

맨 처음으로 돌아가기

팀 모드 변경

팀 모드를 변경할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

팀 모드를 변경하려면 팀의 IANet_TeamOfAdapters 인스턴스를 가져오십시오. 예를 들어, 팀의 객체 경로를 사용하는 **IWbemServices::GetObject**를 사용하십시오. 그런 다음 **IWbemClassObject::Put**을 사용하여 팀의 TeamMode 속성을 변경하십시오. 마지막으로, **IWbemClassObject:: PutInstance**를 호출하여 팀 모드를 변경하여 WBEM_FLAG_UPDATE_ONLY 플래그를 전달하십시오. 이 작업이 실패한 경우 IANet_ExtendedStatus에서 오류 코드를 검사하십시오.

맨 처음으로 돌아가기

팀 내의 어댑터 우선 순위 변경

팀 내의 어댑터 우선 순위를 변경할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

어댑터 우선 순위를 변경하려면 먼저 클라이언트가 어댑터의 IANet_TeamedMemberAdapter 인스턴스를 가져와야 합니다. 예를 들어, 객체 경로를 사용하는 **IWbemServices::GetObject**를 사용하십시오. 그런 다음 클라이언트가 **IWbemClassObject::Put**을 사용하여

어댑터의 AdapterFunction 속성을 변경할 수 있습니다. 마지막으로 클라이언트가 **IWbemClassObject:: PutInstance**를 호출하여 어댑터 우선 순위를 업데이트해야 합니다. 이 작업이 실패한 경우 클라이언트는 IANet_ExtendedStatus에서 오류 코드를 검사해야 합니다.

맨 처음으로 돌아가기

어댑터 제거

어댑터를 제거할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

어댑터를 제거하려면 **IWbemServices::DeleteInstance**를 호출하여 제거할 어댑터의 객체 경로를 전달하십시오.

맨 처음으로 돌아가기

VLAN 만들기

VLAN을 만들 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

VLAN을 만들려면 VLAN이 추가될 어댑터의 IANet_802dot1QVLANService에서 **CreateVLAN** 메서드를 호출하십시오. 다음 인수를 메서드에 전달해야 합니다.

- VLANNumber - VLAN 번호로, 범위는 1- 4094입니다.
- Name - VLAN을 식별하는 이름으로, 사용자가 정의할 수 있습니다.

이 함수는 출력 매개변수 VLANpath에서 새로 만든 VLAN의 객체 경로를 반환합니다. 이 작업이 실패한 경우 IANet_ExtendedStatus에서 오류 코드를 검사하십시오.

맨 처음으로 돌아가기

VLAN 속성 변경

VLAN 속성을 변경할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

클라이언트는 VLAN의 VLANNumber 및 VLANName 속성을 변경할 수 있습니다. 어댑터 우선 순위를 변경하려면 먼저 어댑터의 IANet_VLAN 인스턴스를 가져오십시오. 예를 들어, 객체 경로를 사용하는 **IWbemServices::GetObject**를 사용하십시오.

그런 다음 VLANNumber 또는 VLANName을 원하는 값으로 변경하십시오. 마지막으로 **IWbemClassObject:: PutInstance**를 호출하여 속성을 업데이트하여 WBEM_FLAG_UPDATE_ONLY 플래그를 전달하십시오. 이 작업이 실패한 경우 IANet_ExtendedStatus에서 오류 코드를 검사하십시오.

맨 처음으로 돌아가기

VLAN 삭제

VLAN을 삭제할 경우

- 세션 핸들이 필요합니다.
- PreCheck를 사용할 수 있습니다.
- 작업 실행 전에 **Apply**를 호출해야 합니다.

VLAN을 삭제하려면 **IWbemServices::DeleteInstance**를 호출하여 삭제할 VLAN의 객체 경로를 전달하십시오.

맨 처음으로 돌아가기

Boot Agent 업데이트

Boot Agent를 업데이트할 경우

- 세션 핸들이 필요하지 않습니다.
- PreCheck를 사용할 수 없습니다.
- 작업 실행 전에 **Apply**를 호출하지 않아도 됩니다.

클라이언트는 메서드 호출을 사용하여 Boot Agent 이미지를 업데이트할 수 있습니다. 플래시 이미지를 읽거나 쓰려면 먼저 어댑터의 IANet_BootAgent 인스턴스를 가져오십시오. 예를 들어, 객체 경로를 사용하는 **IWbemServices::GetObject**를 사용하십시오.

그런 다음 ReadFlash()를 실행하여 기존 플래시 부트 ROM 이미지를 읽거나 ProgramFlash()를 실행하여 플래시 부트 ROM 이미지를 업데이트하십시오. 이 작업이 실패한 경우 IANet_ExtendedStatus에서 오류 코드를 검사하십시오.

작업	**WMI** 메서드	결과	설명
어댑터의 부트 ROM 이미지 업데이트 또는 삽입	uint32 ProgramFlash([IN, ValueMap {"0","1"} , Values {"Check Version", "Write Flash"}: Amended] uint32 Action, [IN] uint8 NewFlashData[], [OUT] string strErrorMessage);	'버전 검사' 작업이 지정된 경우 NewFlashData[]에서 업데이트 중인 부트 ROM 이미지가 이미 NIC에 있는 이미지보다 오래된 것이면 경고 메시지가 반환됩니다. '쓰기' 작업이 지정된 경우에는 NIC에 있는 플래시 ROM이 NewFlashData[]로 업데이트됩니다.	이 메서드는 NIC에서 플래시 ROM을 업데이트하는 데 사용됩니다. 이 메서드를 사용하면 플래시가 업데이트되는 동안 NIC와 네트워크 간의 통신이 중지됩니다.
부트 ROM 이미지 읽기	uint32 ReadFlash([OUT] uint8 FlashData[]);	FlashData[]는 NIC에 있는 플래시 ROM 이미지를 포함합니다.	이 메서드는 NIC에 있는 플래시 ROM(파일에 저장할 수 있음)을 읽습니다.

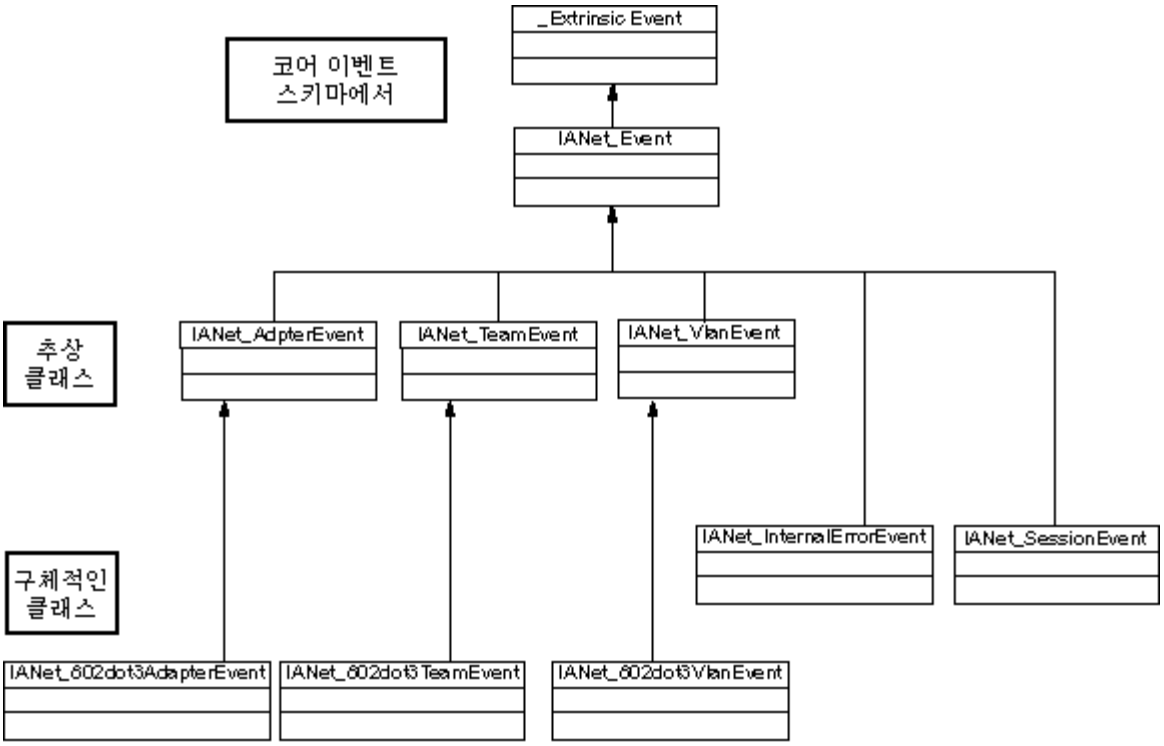
제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

이벤트 알림: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

구체적인 이벤트 클래스
이벤트 등록



맨 처음으로 돌아가기

구체적인 이벤트 클래스

IANet_802dot3AdapterEvent

용도
이 이벤트는 클라이언트에게 어댑터의 상태 또는 구성 변경을 알립니다.

트리거
이 이벤트는 어댑터 상태가 변경된 후에 발생하거나 어댑터 설정을 변경하고 **Apply**를 호출한 후에 발생합니다.

이벤트 데이터
AdapterPath는 이벤트를 발생시킨 어댑터의 객체 경로를 포함합니다.

IANet_802dot3TeamEvent

용도
이 이벤트는 클라이언트에게 팀의 상태 또는 구성 변경을 알립니다.

트리거
이 이벤트는 다음과 같은 경우에 발생합니다.

- 어댑터 상태가 변경된 후
- 팀 설정을 변경하고 **Apply**를 호출한 후
- 팀 구성을 변경하고 **Apply**를 호출한 후

이벤트 데이터
TeamPath는 이벤트를 발생시킨 팀의 객체 경로를 포함합니다.

IANet_802dot3VlanEvent

용도
이 이벤트는 클라이언트에게 VLAN의 상태 또는 구성 변경을 알립니다.

트리거
이 이벤트는 다음과 같은 경우에 발생합니다.

- VLAN 상태가 변경된 후
- VLAN 설정을 변경하고 **Apply**를 호출한 후
- VLAN 구성을 변경하고 **Apply**를 호출한 후

이벤트 데이터
VlanPath는 이벤트를 발생시킨 VLAN의 객체 경로를 포함합니다.

맨 처음으로 돌아가기

이벤트 등록

응용 프로그램은 **IWbemServices:: ExecNotificationQuery** 또는 **IWbemServices:: ExecNotificationQueryAsync**를 사용하여 이벤트 알림을 요청해야 합니다. 다음 질의는 이벤트 알림 질의를 보여주는 예입니다. 이 목록이 가능한 모든 질의를 포함하는 것은 아닙니다.

- **SELECT * FROM IANet_Event** - 모든 이벤트를 요청하는 데 사용됩니다.
- **SELECT * FROM IANet_AdapterEvent** - 모든 어댑터 이벤트를 요청하는 데 사용됩니다.
- **SELECT * FROM IANet_TeamEvent** - 모든 팀 이벤트를 요청하는 데 사용됩니다.
- **SELECT * FROM IANet_SessionEvent** - 모든 세션 이벤트를 요청하는 데 사용됩니다.
- **SELECT * FROM IANet_VlanEvent** - 모든 VLAN 이벤트를 요청하는 데 사용됩니다.
- **SELECT * FROM IANet_InternalErrorEvent** - 모든 내부 이벤트를 요청하는 데 사용됩니다.
- **SELECT * FROM IANet_AdapterEvent WHERE AdapterPath={IANet_EthernetAdapter** 객체 경로**}** - 특정 어댑터의 어댑터 이벤트를 요청하는 데 사용됩니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

최적화된 **WQL** 질의**: Intel(R) PRO Network Adapters WMI and CDM Providers** 사용 설명서

개요
특정 어댑터, VLAN 또는 팀 설정 가져오기
단일 설정 가져오기

개요

WMI Provider는 응용 프로그램에서 질의를 사용하여 설정을 가져올 수 있도록 최적화되어 있습니다. WMI Provider는 다음 질의를 인식할 수 있으며 일치하는 객체만 반환합니다. 다른 모든 질의를 사용하면 WMI Provider는 모든 객체의 모든 설정을 가져오고 CIMOM은 응용 프로그램에 도달하기 전에 이러한 설정을 필터링합니다. 여러 어댑터, 팀 및 VLAN을 사용할 경우 이렇게 하면 필요한 데이터를 검색할 때 몇 초 정도의 지연이 발생할 수 있습니다.

맨 처음으로 돌아가기

특정 어댑터**, VLAN** 또는 팀 설정 가져오기

다음 질의는 특정 어댑터, VLAN 또는 팀의 설정만 가져옵니다. WQL은 WHERE 절에서 추가 섹션을 허용하지 않습니다.

ASSOCIATORS OF {IANet_Configuration 경로} WHERE AssocClass = IANet_SettingContext

맨 처음으로 돌아가기

단일 설정 가져오기

다음 질의를 사용하여 모든 설정을 가져오도록 질의하지 않은 채 객체의 단일 설정을 가져올 수 있습니다.

SELECT * FROM [SETTING CLASS] WHERE ParentId="[Device ID]" AND ParentType="[type]" AND Caption="[SETTING NAME]"

참고**:**

- 클래스는 기본 클래스가 아닌 정확한 설정 클래스여야 합니다(예: IANet_SettingInt).
- 적용 가능한 ParentType은 "NIC", "Team", "VLAN" 또는 "BootAgent"입니다.
- ParentId는 해당 설정이 있는 객체를 고유하게 식별하는 GUID입니다(어댑터의 경우 DeviceId).
- 이 메서드는 객체에 대해 연관된 설정을 가져오는 데 권장되는 메서드가 아닙니다. 권장되는 메서드는 use associations입니다. 하지만 WQL은 필수 복합 질의를 지원하지 않습니다. 즉, WQL은 ASSOCIATORS OF {IANet_Configuration 경로} WHERE AssocClass = IANet_SettingContext AND Caption="[SETTING NAME]"을 지원하지 않습니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

진단: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

진단 클래스
레지스트리 항목
로깅
연관 클래스
테스트

진단 클래스

IANet_DiagTest

용도
IANet_DiagTest는 CIM_DiagnosticTest의 하위 클래스입니다. 이 클래스는 Intel(R) PROSet에서 지원하는 이더넷 어댑터의 진단 테스트를 실행하고 제어하는 일반적인 전달 수단을 제공합니다. CIM_DiagnosticTest 수퍼클래스는 일반적으로 CIM 가능 시스템에 있는 모든 컴퓨터 하드웨어의 테스트를 지원합니다. 클래스 속성은 본래 설명적이며 테스트 메커니즘은 노출된 메서드를 통해 제공됩니다.

인스턴스
키는 이름이며, 이 공급자의 경우에는 참조된 어댑터의 GUID에 있는 테스트의 숫자 인덱스를 연결한 것입니다(예: 1@{12345678-9ABC-DEF0-1234-123456789012}). 이 키 값은 어댑터와 테스트를 참조하는 모든 정보가 RunTest와 다른 메서드에 객체 매개변수로 전달되므로 여분의 정보로 간주될 수도 있습니다. 인스턴스는 메서드에 대한 매개변수와의 일관성을 유지해야 합니다. 그렇지 않으면 공급자가 명령을 거부합니다. 캡션 속성은 이 인스턴스가 실행될 테스트의 이름을 제공합니다. 다른 속성은 기타 설명과 런타임 정보를 제공합니다.

인스턴스 만들기
IANet_DiagTest 인스턴스는 만들 수 없습니다.

인스턴스 삭제
IANet_DiagTest 인스턴스는 삭제할 수 없습니다.

속성 수정
이 클래스에는 사용자가 수정할 수 있는 속성이 없습니다.

연관

- IANet_DiagTestForMSE 인스턴스는 IANet_DiagTest를 IANet_ManagedSystemElement와 연관시킵니다. IANet_ManagedSystemElement는 IANet_EthernetAdapter 인스턴스가 됩니다.
- IANet_DiagResultForTest 인스턴스는 IANet_DiagTest를 IANet_DiagnosticResult 인스턴스와 연관시킵니다.
- IANetDiagSettingForTest 인스턴스는 IANet_DiagTest를 IANet_DiagSetting과 연관시킵니다.

지원되지 않는 속성
Install Date, OtherCharacteristicDescription

메서드
이 클래스는 다음 메서드를 지원합니다.

- RunTest - 다음을 참조하는 세 매개변수에서 정의한 대로 테스트를 실행합니다.
  - SystemElement - 항상 IANet_EthernetAdapter 하위 클래스가 될 SystemElement의 인스턴스를 참조하여 테스트를 실행할 어댑터를 정의합니다.
  - Setting - 항상 IANet_DiagSetting 하위 클래스가 될 CIM_DiagnosticSetting의 인스턴스를 참조하여 실행할 테스트와 해당 실행 방식을 정의합니다.
  - DiagnosticResult - 항상 IANet_DiagResult 클래스가 될 CIM_DiagnosticResult 클래스의 인스턴스를 정의합니다.
- DiscontinueTest - CIM_ManagedSystemElement와 CIM_DiagnosticResult를 참조하는 두 매개변수에서 정의한 대로 진행 중인 진단 테스트를 중지하려고 합니다. 이러한 매개변수는 RunTest와 동일한 기능을 합니다. 세번째 매개변수 TestingStopped는

명령이 테스트를 성공적으로 중지했는지 여부를 나타내는 부울 값을 반환합니다.
- ClearResults - 다음 매개변수를 사용하여 테스트 결과를 지웁니다.
  - SystemElement
  - ResultsNotCleared

  참조된 매개변수 ManagedSystemElement는 이 객체의 객체 경로와 결합하여 삭제될 DiagnosticResultForMSE의 인스턴스를 참조합니다. DiagnosticResultForMSE가 참조하는 DiagnosticResult 객체의 모든 인스턴스가 삭제되며 삭제된 DiagnosticResult 객체를 참조하는 DiagnosticResultForTest의 모든 인스턴스도 삭제됩니다. 마지막으로 문자열 배열 출력 매개변수 ResultsNotCleared는 지울 수 없는 DiagnosticResults의 키를 나열합니다.

클래스 계층

CimV2의 경우. 사용되지 않는 속성과 메서드는 나와 있지 않습니다.

- CIM_ManagedElement:
  Caption
  Description
- CIM_ManagedSystemElement:
  Install Date
  Name
  Status
- CIM_LogicalElement
- CIM_Service:
  - 키
    - Name(문자열)
  - 속성
    - Caption(문자열)
    - CreationClassName(문자열)
    - Description(문자열)
    - Started(부울)
    - StartMode(문자열)
    - Status(문자열)
    - SystemCreationClass(문자열)
    - SystemName(문자열)
- CIM_DiagnosticTest:
  - 속성
    - Characteristics(uint16 배열)
    - IsInUse(부울)
    - ResourcesUsed(uint16 배열)
  - 메서드
    - RunTest
    - ClearResults
    - DiscontinueTest

WbemTest에서 **RunTest** 및 기타 메서드 실행

MOF 파일에 있는 RunTest 메서드는 다음과 같습니다.

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

처음 두 매개변수는 입력 매개변수입니다. 참조된 두 객체의 객체 경로를 가져와야 합니다. RunTest 객체를 엑스포트 중인 IANet_DiagTest 객체의 객체 경로도 가져와야 합니다.

□□□ 기본 WBEM 테스트 대화 상자에서 **Connect**를 누릅니다.
□□□ 해당 '서버\네임스페이스'를 입력합니다. IntelNCS 및 CimV2 네임스페이스가 지원됩니다.
□□□ WBEM 테스트의 **Enum Instances** 단추를 누르고 `IANet_DiagTest`를 입력합니다.
□□□ IANet_DiagTest 인스턴스를 두 번 누릅니다. 이름 형식은 X@[AdapterGUID}입니다. 여기서 X는 테스트 이름이고, AdapterGUID는 IANet_EthernetAdapter의 이름 키와 동일한 어댑터 이름입니다.
□□□ 다음은 검색된 객체 경로의 예입니다.
`\\MYCOMPUTER\root\Cimv2:IANet_DiagTest.Name="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`

□□□ 객체 경로를 저장합니다.
□□□ WBEM 테스트의 **Enum Instances** 단추를 누르고 IANet_EthernetAdapter를 입력합니다.
□□□ 테스트할 어댑터를 두 번 누릅니다.
□□□ 다음은 검색된 객체 경로의 예입니다.
\\MYCOMPUTER\root\cimv2:IANet_EthernetAdapter.DeviceID="{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"
□□□□ 객체 경로를 저장합니다.
□□□□ WBEM 테스트의 **Enum Instances** 단추를 누르고 IANet_DiagSetting을 입력합니다.
□□□□ 어댑터/테스트 조합을 나타내는 설정을 두 번 누릅니다.
□□□□ 다음은 검색된 객체 경로의 예입니다.
\\MYCOMPUTER\root\cimv2:IANet_DiagSetting.SettingID="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"
□□□□ 객체 경로를 저장합니다.
□□□□ 기본 WBEM 테스트 대화 상자에서 **Execute Method**를 누릅니다.
□□□□ IANet_DiagTest 객체 경로를 대화 상자에 붙여 넣습니다. **OK**를 클릭합니다.
□□□□ 메서드 아래의 드롭다운 상자에서 테스트를 선택합니다.
□□□□ **Edit In Parameters** 단추를 누릅니다.
□□□□ RunTest의 경우 Setting 및 SystemElement는 입력 매개변수입니다. 앞서 저장한 Setting 및 Adapter 객체 경로를 붙여 넣고 닫습니다.
□□□□ **Execute** 단추를 누릅니다.
□□□□ 입력 매개변수와 동일한 방식으로 IANet_DiagResult 클래스를 열거합니다.
□□□□ 필요에 맞게 선택된 결과 객체를 검사합니다.

IANet_DiagSetting

용도
IANet_DiagSetting 인스턴스는 특정 런타임 진단 테스트 지시어를 제공합니다. 사용된 지시어는 모든 테스트에 공통되고 CIM_DiagnosticSetting 수퍼클래스에 바인드됩니다. 여기에는 ReportSoftErrors 및 HaltOnError와 같은 속성이 포함됩니다. IANet_DiagSetting 수퍼클래스에 다른 속성이 추가되지는 않습니다.

인스턴스 만들기
이 클래스의 인스턴스는 만들 수 없습니다.

인스턴스 삭제
이 클래스의 인스턴스는 삭제할 수 없습니다.

속성 수정
UpdateInstanceAsync가 구현되며 테스트 매개변수를 "Halt On Error", "Report Soft Errors", "Report Status Messages", "Quick Mode", "Test Warning Level" 및 "Percent Of Test Coverage"로 설정하는 데 사용될 수 있습니다.

연관
IANetDiagSettingForTest 인스턴스는 IANet_DiagTest를 IANet_DiagSetting과 연관시킵니다.

지원되지 않는 속성
다음 속성은 NCS에서 지원되지 않습니다.

- Caption
- Description

메서드
없음

클래스 계층

CimV2의 경우. 사용되지 않는 속성과 메서드는 나와 있지 않습니다.

- CIM_ManagedElement
- CIM_Setting:
    - Properties
    - SettingID
    - Methods(아무것도 지원되지 않음)
    - VerifyOKToApplyToMSE

ApplyToMSE
    - VerifyOKToApplyToCollection
    - ApplyToCollection
    - VerifyOKToApplyIncrementalChangeToMSE
    - ApplyIncrementalChangesToMSE
    - ApplyIncrementalChangeToCollection
- CIM_DiagnosticSetting:
    - 키
        - SettingID(문자열)
    - 속성
        - TestWarningLevel(uint16)
        - ReportSoftErrors(부울)
        - ReportStatusMessages(부울)
        - HaltOnError(부울)
        - QuickMode(부울)
        - PercentOfTestCoverage(uint8)

IANet_DiagResult

용도
IANet_DiagResult 인스턴스는 특정 어댑터에 대해 실행된 특정 테스트의 결과 데이터를 표시합니다. 이 클래스의 인스턴스는 IANet_DiagTest 및 IANet_DiagSetting 인스턴스와 동일하게 일치합니다.

인스턴스
IANet_DiagResult 인스턴스는 특정 어댑터에 대해 실행된 특정 테스트의 결과와 일치합니다. 키 형식은 IANet_DiagTest 및 IANet_DiagSetting과 동일합니다. 정의된 속성과 맞지 않는 데이터가 TestResults Array 속성에 포함될 수 있으므로 인스턴스가 임의 테스트 결과를 저장할 수 있습니다. 언제든 어댑터에 대해 새 테스트가 실행되면 새 인스턴스가 그 어댑터/테스트 조합에 해당하는 테스트 결과의 기존 인스턴스를 덮어씁니다.

인스턴스 만들기
이 클래스의 인스턴스는 만들 수 없습니다.

인스턴스 삭제
이 클래스의 인스턴스는 삭제할 수 없습니다.

속성 수정
이 클래스의 인스턴스는 수정할 수 없습니다.

연관
IANet_DiagResultForTest 인스턴스는 IANet_DiagTest를 IANet_DiagnosticResult 인스턴스와 연관시킵니다.

지원되지 않는 속성
다음 속성은 NCS에서 지원하지 않습니다.

- EstimatedTimeOfPerforming
- HaltOnError
- OtherStateDescription
- ReportSoftErrors
- TestWarningLevel

메서드
없음

클래스 계층

CimV2의 경우. 사용되지 않는 속성과 메서드는 나와 있지 않습니다.

- CIM_DiagnosticResult:
    - 키
        - DiagnosticCreationClassName(문자열)
        - DiagnosticName(문자열)
        - ExecutionID(문자열)

        - DiagSystemCreationClassName(문자열)
        - DiagSystemName(문자열)
    - 속성
        - TimeStamp(문자열)
        - IsPackage(부울)
        - TestStartTime(날짜 시간)
        - TestCompletionTime(날짜 시간)
        - TestState(uint16)
        - TestResults(문자열)
        - PercentComplete(uint8)
- IANet_DiagResult

맨 처음으로 돌아가기

레지스트리 항목

다음 항목은 설치하는 동안 **HKLM\Software\Intel\NETWORK_SERVICES\NCS\NcsDiag** 아래의 레지스트리에 입력됩니다. 진단 테스트 실행을 제어하는 이러한 키와 값이 아래에 정의되어 있습니다.

다음 표에는 키의 값, 유형 및 간략한 사용법 설명이 나와 있습니다.

값	유형	기본	사용
Check Time	REG_DWORD	2 seconds	송신기 또는 응답기 테스트의 완료 검사 간격
Enable	REG_DWORD	0	결과 로그 파일 사용 활성화(1) 또는 결과 로그 파일 사용 비활성화(0)
FileAppend	REG_DWORD	1	기존 결과 로그 파일에 결과 로그 파일 첨부(1) 또는 기존 파일 삭제(0)
LogFileName	REG_SZ	NcsDiag.log	결과 로그 파일 이름
MaxFileSize	REG_DWORD	0x10000	최대 결과 로그 파일 크기
MaxPktsRcvd	REG_DWORD	200	빠른 모드(IANet_DiagSetting에서)에서는 수신 패킷 수가 이 값보다 클 때 송/수신 테스트가 종료됩니다.
TimeoutSndRsp	REG_DWORD	100	테스트 지속 시간(초)이 이 값을 초과할 때 테스트가 종료됩니다.

맨 처음으로 돌아가기

로깅

결과 로그

결과 로그는 기본적으로 IANet_DiagResult 객체에서도 가져올 수 있는 정보를 표시합니다. 차이는 CIM 브라우저에서 가져온 정보는 특정 어댑터에 대한 특정 테스트의 최근 결과만 표시한다는 점입니다. 이후 테스트는 이전 테스트 결과를 덮어씁니다. 결과 로그를 사용하면 나중에 실행할 테스트를 보다 쉽게 설정 및 검토할 수 있습니다.

결과 로그 활성화

결과 로그를 활성화하려면

□□□ **HKLM\Software\Intel\NETWORK_SERVICES\NCS\NCSDiag** 레지스트리 키에서 Enable 값을 1로 설정합니다.
□□□ LogFileName 값을 선호하는 로그 파일 이름으로 설정하거나 기본값 **NcsDiag.log**를 그대로 둡니다.
□□□ 로그 파일은 InstalledDir 값이 나타내는 디렉토리에서 찾을 수 있습니다.

맨 처음으로 돌아가기

연관 클래스

연관 클래스	참조 속성/값	참조 속성/값
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

맨 처음으로 돌아가기

테스트

구현된 테스트는 단일 컴퓨터에서 실행되거나 두 대의 컴퓨터에서 실행될 수 있습니다. CDM Provider가 특정 테스트에 종속되지 않는 일반 테스트 전달체(Vehicle)이기 때문에 테스트에 대한 자세한 설명은 이 문서의 범위를 벗어납니다. 하지만 이 절에서 설명한 것처럼 코드는 일부 종속성이 내장되어 있습니다.

단일 어댑터 테스트

다음 테스트는 단일 어댑터에 대해 실행되고 다른 어댑터와의 상호 작용이 필요하지 않습니다.

- EEPROM
- 제어 레지스터
- MAC 루프백
- PHY 루프백
- 링크

이러한 테스트에서 발생하는 모든 오류 메시지는 하위 스택 레이어에 대한 호출에서 반환된 HRESULT 오류 코드로 인한 것입니다. 오류 코드는 내부에서 오류 코드로 저장되며 IANet_DiagResult 객체가 열거에 의해 참조 해제되거나 객체 호출이 관리 응용 프로그램에서 수신될 때까지 오류 메시지로 변환되지 않습니다.

두 대의 어댑터가 필요한 테스트

송신기 및 응답기 테스트는 상호 의존적 테스트입니다. 즉, 한 어댑터(송신기)가 다른 어댑터(응답기)에 패킷을 보내면 응답기가 송신기에게 패킷을 돌려보내므로 루프가 형성됩니다. 이는 동일한 테스트로, Intel(R) PROSet에서 실행할 수 있습니다. 하지만 Intel PROSet에서는 CDM을 사용하지 않으며 한 컴퓨터에서 두 테스트를 동시에 실행할 수 없습니다. CDM을 사용하면 한 컴퓨터에서 여러 테스트를 동시에 실행할 수 있습니다.

송신기/응답기 테스트

송신기/응답기 테스트에는 송신기와 응답기 역할을 할 인텔 어댑터가 하나씩 필요합니다. 완료 조건에 따라 테스트가 완료되거나 주 스레드가 테스트를 중지할 때까지 계속 실행되는 보조 스레드에 기반하여 실행되는 테스트는 이 테스트뿐입니다. 완료 조건은 테스트 시간이나 수신 패킷 수에 기반한 시간 초과입니다. 이러한 값은 둘 다 레지스트리에서 가져옵니다. 빠른 모드가 활성화된 경우에는 수신 패킷 수에 기반해서만 테스트가 종료될 수 있습니다. 빠른 모드는 IANet_DiagSetting 클래스의 속성이므로 어댑터마다 설정될 수 있습니다. CDM 응답기는 PROSet 응답기에 응답하고, PROSet 응답기는 CDM 응답기에 응답합니다.

두 유형의 오류 값이 송신기/응답기 테스트에서 반환됩니다. 첫째, 오류 코드(HRESULT)가 하위 레이어에서 반환될 수 있습니다. 둘째, 테스트가 실행되는 동안 반환된 오류 코드로 인해 테스트가 중간에 종료되는 경우가 아니면 테스트 스레드가 중간 테스트 통계를 반환한 후 최종 테스트 통계를 반환합니다. 최종 테스트 통계에는 다음 사항이 포함됩니다.

- 연결 상태
- 자동 협상 사용
- 충돌
- 수신 패킷 수

- 총 수신 패킷 수
- 송신 패킷 수
- 전송 성공
- 수신 성공
- 전송 오류 수
- 수신 오류 수
- 충돌
- 진단 단계

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

IANet_DiagTest에서 메서드 실행: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

WbemTest에서 RunTest 및 기타 메서드 실행

MOF 파일에 있는 RunTest 메서드는 다음과 같습니다.

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

처음 두 매개변수는 입력 매개변수입니다. 참조된 두 객체의 객체 경로를 가져와야 합니다. RunTest 객체를 엑스포트 중인 IANet_DiagTest 객체의 객체 경로도 가져와야 합니다.

□□□ 기본 WBEM 테스트 대화 상자에서 **Connect**를 누릅니다.
□□□ 해당 '서버\네임스페이스'를 입력합니다. IntelNCS 및 CimV2 네임스페이스가 지원됩니다.
□□□ WBEM 테스트의 **Enum Instances** 단추를 누르고 `IANet_DiagTest`를 입력합니다.
□□□ IANet_DiagTest 인스턴스를 두 번 누릅니다. 이름 형식은 X@[AdapterGUID}입니다. 여기서 X는 테스트 이름이고, AdapterGUID는 IANet_EthernetAdapter의 이름 키와 동일한 어댑터 이름입니다.
□□□ 다음은 검색된 객체 경로의 예입니다.
`\\MYCOMPUTER\root\Cimv2:IANet_DiagTest.Name="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`
□□□ 객체 경로를 저장합니다.
□□□ WBEM 테스트의 **Enum Instances** 단추를 누르고 `IANet_EthernetAdapter`를 입력합니다.
□□□ 테스트할 어댑터를 두 번 누릅니다.
□□□ 다음은 검색된 객체 경로의 예입니다.
`\\MYCOMPUTER\root\cimv2:IANet_EthernetAdapter.DeviceID="{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`
□□□□ 객체 경로를 저장합니다.
□□□□ WBEM 테스트의 **Enum Instances** 단추를 누르고 `IANet_DiagSetting`을 입력합니다.
□□□□ 어댑터/테스트 조합을 나타내는 설정을 두 번 누릅니다.
□□□□ 다음은 검색된 객체 경로의 예입니다.
`\\MYCOMPUTER\root\cimv2:IANet_DiagSetting.SettingID="1@{4A0CDABE-F6C3-45D0-B60D-F6E7BAFA2C2C}"`
□□□□ 객체 경로를 저장합니다.
□□□□ 기본 WBEM 테스트 대화 상자에서 **Execute Method**를 누릅니다.
□□□□ IANet_DiagTest 객체 경로를 대화 상자에 붙여 넣고 **OK**를 클릭합니다.
□□□□ 메서드 아래의 드롭다운 상자에서 테스트를 선택합니다.
□□□□ **Edit In Parameters** 단추를 누릅니다.
□□□□ RunTest의 경우 Setting과 SystemElement는 입력 매개변수입니다. 앞서 저장한 Setting 및 Adapter 객체 경로를 붙여 넣고 닫습니다.
□□□□ **Execute** 단추를 누릅니다.
□□□□ 입력 매개변수와 동일한 방식으로 IANet_DiagResult 클래스를 열거합니다.
□□□□ 필요에 맞게 선택된 결과 객체를 검사합니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

CIM 클래스 요약: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

클래스	생성 가능 여부	삭제 가능 여부	구현된 메서드	설정 가능한 속성	지원되지 않는 속성	인스턴스 수	관련된 연관 클래스
IANet_802dot1QVLANService	N	N	CreateVLAN	GRVPEnabled, JoinTime	Description, Install Date, Started, StartMode, Status	VLAN을 지원하는 각 팀 또는 어댑터당 하나씩	IANet_Device802dot1QVLANServiceImplementation, IANet_VLANFor
IANet_AdapterConfiguration	N	N	없음	없음	없음	각 어댑터당 하나씩	이 클래스는 IANet_EthernetAdapter를 IANet_Configuration과 연관시킵니다.
IANet_AdapterDevice	N	N	없음	없음	없음	각 비팬텀 어댑터당 하나씩	이 클래스는 IANet_EthernetAdapter를 IANet_EthernetPCIDevice와 연관시킵니다.
IANet_AdapterProtocolImplementation	N	N	없음	없음	없음	어댑터에 바인드된 각 IP 프로토콜 종점당 하나씩	이 클래스는 IANet_EthernetAdapter를 IANet_IPProtocolEndpoint와 연관시킵니다.
IANet_BootAgent	N	N	ProgramFlash ReadFlash	없음	Caption, Description, InstallDate, Started, StartMode, Status	Boot Agent 기능을 지원하는 각 어댑터당 하나씩	IANet_DeviceBootServiceImplementation, IANet_BootAgentConfiguration
IANet_BootAgentConfiguration	N	N	없음	없음	없음	각 Boot Agent당 하나씩	이 클래스는 IANet_BootAgent를 IANet_Configuration과 연관시킵니다.
IANet_Configuration	N	N	없음	없음	없음	각 어댑터, VLAN 및 팀당 하나씩	IANet_AdapterConfiguration, IANet_VLANConfiguration, IANet_SettingContext
IANet_Device802dot1QVLANServiceImplementation	N	N	없음	없음	없음	VLAN을 지원하는 각 어댑터 또는 팀당 하나씩	이 클래스는 IANet_EthernetAdapter를 IANet_802dot1QVLANService와 연관시킵니다.
IANet_DiagTest	N	N	RunTest, DiscontinueTest, ClearResults	없음	InstallDate, OtherCharacteristicsDescription	각 어댑터/테스트 조합당 하나씩	IANet_DiagTestForMSE, IANet_DiagResultForTest, IANet_DiagSettingForTest
IANet_DiagSetting	N	N	없음	HaltOnError, ReportSoftErrors, ReportStatusMessages, QuickMode, PercentOfTestCoverage,	Caption, Description	각 어댑터/테스트 조합당 하나씩	IANet_DiagSettingForTest

				TestWarningLevel			
IANet_DiagResult	N	N	없음	없음	EstimatedTimeOfPerforming, HaltOnError, OtherStateDescription, ReportSoftErrors, TestWarningLevel	각 어댑터/테스트 조합당 하나씩	IANet_DiagResultForTest, IANet_DiagResultForMSEM
IANet_EthernetAdapter	N	Y	IdentifyAdapter HasVLANs IsPowerMgmtSupported GetPowerUsage SetPowerUsage GetPowerUsageOptions SetPowerUsageOptions TestCable AdvancedTestCable TestLinkSpeed	없음	AutoSense - (설정으로 노출됨), ErrorCleared, OtherIdentifyingInfo, IdentifyingDescriptions, InstallDate, LastErrorCode, MaxDataSize, MaxQuiesceTime, PowerManagementCapabilities - (메서드로 노출됨), PowerManagementSupported - (메서드로 노출됨), PowerOnHours, ShortFramesReceived, SymbolErrors, TotalPowerOnHours	Intel(R) PROSet을 지원하는 설치된 각 어댑터, 팬 텀 어댑터 및 팀당 하나씩	IANet_AdapterProtocolImplementation, IANet_AdapterDevice, IANet_AdapterConfiguration, IANet_TeamedMemberAdapter, IANet_NetworkVirtualAdapter, IANet_Device802dot1QVLANServiceImplementation, IANet_DeviceBootServiceImplementation
IANet_EthernetPCIDevice	N	N	없음	없음	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	Intel PROSet을 지원하는 이더넷 어댑터인 각 PCI 카드당 하나씩	IANet_AdapterDevice
IANet_IPProtocolEndpoint	N	N	없음	없음	Caption, Description, InstallDate, NameFormat, OtherTypeInformation, Status	인텔 지원 종점에 대한 각 IP 프로토콜 스택 바인딩당 하나씩	IANet_AdapterProtocolImplementation, IANet_VLANProtocolDependency
IANet_NetService	N	N	GetSessionHandle, Apply, ReleaseSessionHandle, Cancel	없음	Caption, Description, Install Date, Started, Start Mode, Status	정확히 하나만	없음
IANet_NetworkVirtualAdapter	N	N	없음	없음	없음	각 팀당 하나씩	이 클래스는 IANet_TeamOfAdapters를 IANet_EthernetAdapter와 연관시킵니다.
IANet_PCIDevice	N	N	없음	없음	AdditionalAvailability, Capabilities, CapabilityDescriptions, Caption, DeviceSelectTiming,	시스템의 네트워크 장치인 각 PCI 카드	

					ErrorCleared, ErrorDescription, IdentifyingDescription, InstallDate, LastErrorCode, MaxNumberController, MaxQuiesceTime, Name, OtherIdentifyingInfo, PowerManagementCapabilities, PowerManagementSupported, PowerOnHours, ProtocolDescription, ProtocolSupported, SelfTestEnabled, TimeOfLastReset, TotalPowerOnHours	당 하나씩	
IANet_Setting	N	N	없음	없음	SettingID	추상 클래스	IANet_SettingContext
IANet_SettingContext	N	N	없음	없음	없음	각 설정당 하나씩	이 클래스는 IANet_Setting을 IANet_Configuration과 연관시킵니다.
IANet_SettingInt	N	N	없음	CurrentValue	SettingID	각 정수 설정당 하나씩	IANet_SettingContext
IANet_SettingMultiSelection	N	N	없음	CurrentValue	SettingID	각 다중 선택 설정당 하나씩	IANet_SettingContext
IANet_SettingSlider	N	N	없음	CurrentValue	SettingID	각 슬라이더 설정당 하나씩	IANet_SettingContext
IANet_SettingString	N	N	없음	CurrentValue	SettingID	각 문자열 설정당 하나씩	IANet_SettingContext
IANet_TeamedMemberAdapter	Y	Y	없음	AdapterFunction	PrimaryAdapter, ScopeOfBalancing	팀에 있는 모든 어댑터당 하나씩	이 클래스는 IANet_TeamOfAdapters를 IANet_EthernetAdapter와 연관시킵니다.
IANet_TeamOfAdapters	Y	Y	TestSwitchConfiguration	TeamingMode	Install Date, Status	각 팀당 하나씩	IANet_NetworkVirtualAdapter, IANet_TeamedMemberAdapter
IANet_VLAN	N	Y	없음	VLANNumber, Caption	Description, Install Date, StartMode, Status	각 VLAN당 하나씩	IANet_VLANFor
IANet_VLANConfiguration	N	N	없음	없음	없음	각 VLAN당 하나씩	이 클래스는 IANet_VLAN을 IANet_Configuration과 연관시킵니다.
IANet_VLANFor	N	N	없음	없음	없음	각 VLAN당 하나씩	이 클래스는 IANet_VLAN을 IANet_802dot1QVLANService와 연관시킵니다.
IANet_VLANProtocolDependency	N	N	없음	없음	없음	각 VLAN당 하나씩	이 클래스는 IANet_VLAN을 IANet_IPProtocolEndpoint와 연관시킵니다.

목차 페이지로 돌아가기

Intel 소프트웨어 라이센스 계약서(최종, 라이센스): Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

중요 - 복사, 설치 또는 사용 전에 반드시 읽어보시기 바랍니다.

아래 계약서 내용을 주의 깊게 읽어본 후 이 소프트웨어와 모든 관련 자료(이하 "소프트웨어")를 사용하거나 로드하십시오. 소프트웨어를 로드하거나 사용하는 것은 곧 아래의 계약서 내용에 동의하는 것입니다. 아래 계약서 내용에 동의하지 않으면 소프트웨어를 설치하거나 사용하지 마십시오.

라이센스:

- 네트워크 관리자에게는 "사이트 라이센스"가 제공됩니다.
- 최종 사용자에게는 "단일 사용자 라이센스"가 제공됩니다.
- OEM(Original Equipment Manufacturer)에게는 "OEM 라이센스"가 제공됩니다.

사이트 라이센스. 다음 조건을 따를 경우 네트워크 관리자는 해당 조직에서 사용할 목적으로 조직 내의 컴퓨터에 소프트웨어를 복사하고 적절한 개수의 소프트웨어 백업본을 만들 수 있습니다.

□□□ 이 소프트웨어는 Intel 구성요소 제품과 함께 사용할 경우에 한해 라이센스가 부여되며 타사 구성요소 제품과 함께 사용할 경우에는 라이센스가 부여되지 않습니다.
□□□ 이 계약서에서 허용하는 경우가 아니면 소프트웨어의 어떠한 부분도 복사, 수정, 대여, 판매, 유통 또는 양도할 수 없으며, 사용자는 소프트웨어의 무단 복사 행위를 하지 않을 것에 동의해야 합니다.
□□□ 소프트웨어를 리버스 엔지니어링, 디컴파일 또는 디스어셈블할 수 없습니다.
□□□ 다른 사용자에게 하위 라이센스를 부여하거나 두 명 이상의 사용자가 동시에 소프트웨어를 사용하도록 허용할 수 없습니다.
□□□ 소프트웨어에는 이 라이센스 계약서의 내용뿐 아니라 다른 관련 라이센스 계약 내용이 적용될 수도 있습니다.

단일 사용자 라이센스. 다음 조건을 따를 경우 최종 사용자는 상용이 아닌 개인 용도로 사용할 목적으로 자신의 컴퓨터 한 대에 소프트웨어를 복사하고 한 개의 소프트웨어 백업본을 만들 수 있습니다.

□□□ 이 소프트웨어는 Intel 구성요소 제품과 함께 사용할 경우에 한해 라이센스가 부여되며 타사 구성요소 제품과 함께 사용할 경우에는 라이센스가 부여되지 않습니다.
□□□ 이 계약서에서 허용하는 경우가 아니면 소프트웨어의 어떠한 부분도 복사, 수정, 대여, 판매, 유통 또는 양도할 수 없으며, 사용자는 소프트웨어의 무단 복사 행위를 하지 않을 것에 동의해야 합니다.
□□□ 소프트웨어를 리버스 엔지니어링, 디컴파일 또는 디스어셈블할 수 없습니다.
□□□ 다른 사용자에게 하위 라이센스를 부여하거나 두 명 이상의 사용자가 동시에 소프트웨어를 사용하도록 허용할 수 없습니다.
□□□ 소프트웨어에는 이 라이센스 계약서의 내용뿐 아니라 다른 관련 라이센스 계약 내용이 적용될 수도 있습니다.

OEM 라이센스. 다음 조건을 따를 경우 OEM은 해당 제품의 필수 요소로, 해당 제품에 통합된 요소로 또는 해당 제품의 기존 최종 사용자를 위한 독립형 소프트웨어 유지 관리 업데이트(다른 독립형 제품은 제외)로 소프트웨어를 재생 및 배포할 수 있습니다.

□□□ 이 소프트웨어는 Intel 구성요소 제품과 함께 사용할 경우에 한해 라이센스가 부여되며 타사 구성요소 제품과 함께 사용할 경우에는 라이센스가 부여되지 않습니다.
□□□ 이 계약서에서 허용하는 경우가 아니면 소프트웨어의 어떠한 부분도 복사, 수정, 대여, 판매, 유통 또는 양도할 수 없으며, 사용자는 소프트웨어의 무단 복사 행위를 하지 않을 것에 동의해야 합니다.
□□□ 소프트웨어를 리버스 엔지니어링, 디컴파일 또는 디스어셈블할 수 없습니다.
□□□ 서면 계약서에 의거해서만 고객에게 소프트웨어를 배포할 수 있습니다. 이러한 라이센스 계약서는 "BTS(Break-The-Seal)" 라이센스 계약서일 수 있습니다. 적어도 이러한 라이센스는 소프트웨어에 대한 Intel 소유권을 보호해야 합니다.
□□□ 소프트웨어에는 이 라이센스 계약서의 내용뿐 아니라 다른 관련 라이센스 계약 내용이 적용될 수도 있습니다.

다른 권리는 없습니다. 이 계약서에서 명시적으로 부여한 경우를 제외하고 Intel은 Intel이 소유하거나 제어하는 독점 정보, 특허권, 저작권, 배치 설계, 상표권, 거래 기밀 또는 기타 지적 재산권에 대해 어떠한 명시적 또는 묵시적 권리/라이센스도 부여하지 않습니다.

소프트웨어 소유권 및 저작권. 모든 소프트웨어 사본에 대한 타이틀은 Intel 또는 해당 공급자가 소유합니다. 소프트웨어에 대한 저작권은 Intel 또는 해당 공급자가 소유하며 소프트웨어는 미국, 대한민국 및 기타 국가의 저작권법과 국제 협약 규정의 보호를 받습니다. 소프트웨어에서 저작권 통지 부분을 제거하면 안됩니다. Intel은 언제든지 예고 없이 소프트웨어나 소프트웨어에서 참조하는 항목을 변경할 수 있지만, 소프트웨어 지원이나 업데이트 의무는 갖지 않습니다. 명시적으로 부여된 경우를 제외하고 Intel은 Intel 특허권, 저작권, 상표권 또는 기타 지적 재산권에 대한 어떠한 명시적 또는 묵시적 권리도 부여하지 않습니다. 양도인이 모든 소프트웨어 사본을 폐

기하고 양수인이 이 라이센스 계약서를 준수하겠다고 동의한 경우에 한해 소프트웨어를 양도할 수 있습니다.

제한적 미디어 보증. Intel에서 물리적 미디어에 담긴 상태로 소프트웨어를 전달한 경우, Intel은 수령일로부터 90일 동안 미디어의 재료에 물리적인 결함이 없음을 보증합니다. 이러한 결함이 발견되면 Intel에 미디어를 반송하여 미디어를 교환하거나 다른 방법을 통해 해당 소프트웨어를 전달받을 수 있습니다.

기타 보증의 제외. 위에 규정된 경우를 제외하고 소프트웨어는 상품성, 비침해 또는 특정 목적에의 적합성에 대한 보증을 포함하여 어떠한 명시적 또는 묵시적 보증도 없이 "있는 그대로" 제공됩니다. Intel은 정보, 텍스트, 그래픽, 링크 또는 이 소프트웨어에 포함된 기타 항목의 정확성이나 완벽성에 대하여 어떠한 보증이나 책임도 배제합니다.

책임의 제한. Intel 또는 해당 공급자는 소프트웨어의 사용 또는 사용할 수 없음으로 인한 모든 손해(영업 이익 손실, 영업 중단 또는 정보 손실을 포함하되 이에 제한되지 않음)에 대하여 어떠한 경우에도 책임을 지지 않으며, 이는 Intel이 그와 같은 손해의 가능성을 사전에 알고 있었다 하더라도 마찬가지입니다. 일부 주/지방/관할지에서는 묵시적 보증이나 파생적 또는 부수적 손해에 대한 책임을 배제하거나 제한하는 행위를 허용하지 않으므로, 위 제한이 사용자에게 적용되지 않을 수도 있습니다. 귀하는 주/지방/관할지에 따라 다른 법적 권리를 보유할 수도 있습니다.

계약의 종결. 본 계약 내용을 위반할 경우 Intel은 언제든지 계약을 해지할 수 있습니다. 계약이 해지되면 사용자는 곧바로 소프트웨어를 파기하거나 모든 소프트웨어 사본을 Intel로 반송해야 합니다.

관련 법률. 관련 법률 및 국제 상품 판매에 관한 UN 협약에 상충되는 경우를 제외하고, 이 계약으로 인해 제기되는 모든 소송은 캘리포니아주 법률이 적용됩니다. 관련 수출법과 규정을 위반해서 소프트웨어를 수출하면 안됩니다. Intel은 Intel 대표자의 서명날인이 없는 다른 모든 계약에 대하여 어떠한 의무도 지지 않습니다.

정부 기관의 제한된 권리. 소프트웨어에는 "제한된 권리"만 부여됩니다. 정부 기관에서의 사용, 복제 또는 공개는 FAR52.227-14, DFAR252.227-7013 또는 후속 규정을 따릅니다. 정부 기관에서의 소프트웨어 사용은 소프트웨어에 포함된 Intel의 소유권을 전제로 합니다. 계약자 또는 제조자는 Intel입니다.

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기 맨 처음으로 돌아가기

목차 페이지로 돌아가기

지원: Intel(R) PRO Network Adapters WMI and CDM Providers 사용 설명서

웹 및 인터넷 사이트

http://www.dell.com

고객 지원 센터 기술자

이 설명서의 문제 해결 방법으로 문제가 해결되지 않으면 Dell Computer Corporation에 기술 지원을 요청하십시오(시스템 설명서의 "도움 요청" 항목 참조).

연락하기 전에 준비할 사항

컴퓨터에서 소프트웨어를 실행하고 제품 설명서를 준비해야 합니다.

기술 담당자가 다음 정보를 요구할 수 있습니다.

- 주소와 전화 번호
- 문의하는 제품의 이름과 모델 번호
- 제품의 일련 번호와 서비스 태그
- 제품을 작동하기 위해 사용하는 소프트웨어의 이름과 버전 번호
- 사용하는 운영 체제의 이름과 버전 번호
- 컴퓨터 종류(제조업체 및 모델 번호)
- 컴퓨터에 설치된 확장 보드나 추가 기능 카드
- 컴퓨터의 메모리 크기

제한 및 책임 부인을 자세히 읽어보십시오.

목차 페이지로 돌아가기