

Mit der Einführung von Windows-2000 bringt Microsoft ein neues äußerst leistungsfähiges Scripting-Interface namens Windows-Management-Instrumentation auf den Markt, das den Administrator in die Lage versetzt, nahezu allen denkbaren betriebssystemnahen Aufgaben zu automatisieren.

DIRK PELZER

Eines der Designziele von Windows-2000 war es, das Management des Betriebssystems so leistungsfähig wie möglich zu gestalten, um die immer wieder strapazierte Cost-of-Ownership besser in den Griff zu bekommen. Eines der unumgänglichen Hilfsmittel zur Erreichung dieses Ziels ist die Unterstützung einer leistungsfähigen Schnittstelle zur Automatisierung von Aufgaben und Abfrage von Systeminformationen per Skript. Windows-Management-Instrumentation (WMI) ist diese Schnittstelle und basiert auf der Web-Based-Enterprise-Management-Initiative (WBEM) und dem Common-Information-Model (CIM). WMI erlaubt dem Systemverwalter auf eine Vielzahl von Klassen für die Systemhardware, das Betriebssystem und installierte Applikationen über eine vereinheitlichte Schnittstelle zuzugreifen. Damit ist das Lesen von Einträgen des Windows-2000-Ereignisprotokolls ebenso einfach realisierbar, wie die Möglichkeit festzustellen, ob und wenn ja welcher SCSI-Controller in ein System eingebaut ist. Der Zugriff auf die bereitgestellten WMI-Klassen oder das Ermitteln einer Eigenschaft folgt dabei einem Schema, welches für alle Klassen einheitlich realisiert ist. Sobald der Administrator also verstanden hat, wie der Zugriff für eine Klasse funktioniert, ist er in der Lage, dieses Wissen auf alle anderen Klassen anzuwenden. Ziel dieses Workshops ist es, das nötige Grundwissen zu vermitteln und mit einigen Beispielen zu ergänzen, um unter der Zuhilfenahme von WMI leistungsfähige Scripts zur Systemverwaltung erstellen zu können.

WMI-Architektur

Die WMI-Architektur besteht im wesentlichen aus drei Schichten. Die erste Ebene wird durch den sogenannten »Provider« gebildet, der gewissermaßen die Rolle eines Agenten zwischen dem verwalteten System und »CIM-Object-Manager« (CIMOM) übernimmt. Die Aufgabe des Providers

besteht dabei darin, die für das Management relevanten Informationen zu ermitteln und für den CIMOM zugänglich zu machen. Wie der Provider die benötigten Daten ermittelt, bleibt ihm dabei selbst überlassen. Mit Hilfe solcher Provider werden neben Microsoft künftig zahlreiche weitere Anbieter, wie IBM, Hewlett-Packard oder Compaq Management-Informationen ihrer Hardwaregeräte und Softwareapplikationen bereit stellen. Der CIMOM agiert als Broker für Zugriffe auf gemanagte Objekte und kommuniziert mit den Providern über die Schnittstellen des Common-Object-Modells (COM). Eine weitere Aufgabe des CIMOM besteht darin, darüber Buch zu führen, welche Objektklassen verfügbar sind und welcher Provider für die Instanzen der jeweiligen Klasse verantwortlich ist. Diese Information legt er in einer Datenbank ab. Der dritte Eckpfeiler in der WMI-Architektur sind die »Consumer«, worunter man beispielsweise Management-Werkzeuge, wie Snap-Ins für die Microsoft-Management-Console (MMC) oder eben vom Administrator geschriebene Skripte versteht.

Um an Daten gemanagter Objekte zu gelangen oder Einstellungen zu manipulieren, nimmt ein Consumer Kontakt zum CIMOM auf und bittet diesem darum, die gewünschte Aktion auszuführen. Der Consumer muss dabei weder wissen, welcher Provider zuständig ist, noch wie die Daten zu ermitteln sind. Dies erledigt der CIMOM in Zusammenarbeit mit dem Provider und liefert genau die gewünschte Information zurück. Der CIMOM kann aber nicht nur passiv auf Anfragen reagieren, sondern ist auch in der Lage, sogenannte WMI-Events zu erzeugen, wenn ein bestimmtes Ereignis eingetreten ist. Dazu gehört etwa das Überschreiten eines Performance-Counters, wie der Prozessorzeit oder die Modifikation eines Registry-Keys. Sobald der betreffende Provider eine entsprechende Meldung an den CIMOM weiterleitet, informiert dieser einen registrierten Consumer über das Ereignis, damit der dann wiederum entsprechende Schritte, also etwa den Versand einer Mail oder den Aufruf einer Funk-

tion innerhalb eines Skriptes einleiten kann.

Erste Schritte mit WMI

Als schnellster Weg zu verstehen, wie man mittels WMI an die zahllosen Objekte des Betriebssystems herankommt, eignet sich am besten ein kleines Beispiel. Dabei soll mittels WMI festgestellt werden, welche lokalen Verzeichnisse eines Systems für den Netzwerkzugriff freigegeben sind und wie gegebenenfalls der Freigabename lautet. Als Skriptsprache kommt VBScript zum Einsatz.

```
Set EnumShares=GetObject
("winmgmts:{impersonationLevel=impersonate}").
ExecQuery("SELECT * FROM
Win32_Share WHERE Name<>" ")
For each Share in EnumShares
WScript.Echo Share.Name & vbTab & Share.Path
Next
```

In dem Skript wird zunächst mit dem GetObject-Kommando ein Objekt der Klasse Win32_Share angefordert und dessen Inhalt der Variablen EnumShares zugeordnet. Dabei erfolgt eine Vorfilterung des Objekthinhalts, indem über die ExecQuery-Methode eine WQL-Abfrage mit in den Aufruf integriert wird. Innerhalb der WQL-Abfrage, die durch den Ausdruck »SELECT * FROM Win32_Share WHERE Name <> "«definiert wird, werden der Variablen EnumShares nur solche Instanzen der Klasse Win32_Share zugewiesen, die einen Namen haben. Da aber jedes freigegebene Verzeichnis einen Namen haben muss, erhält man somit alle auf dem System vorhandenen Freigaben. Allerdings werden hierbei nicht nur freigegebene Verzeichnisse, sondern auch Drucker oder der IPC\$-Share zurückgeliefert, was für das Beispiel aber nicht weiter tragisch ist. Sobald EnumShares mit Inhalt gefüllt ist, kann man daran gehen, in einer For-Schleife die interessierenden Eigenschaften herauszufiltern und auf dem Bildschirm auszugeben. In unserem Fall sind dies der Name über »Share.Name« und der dazugehörige Pfad über »Share.Path«. Ruft man das Skript auf, so erhält man beispielhaft folgende Ausgabe zurück:

```
C$ C:\
D$ D:\
E$ E:\
G$ G:\
IPC$
NETLOGON
G:\WINNT\SYSDVOL\sysvol\pelzer-consulting.de\
SCRIPTS
ADMIN$ G:\WINNT
SYSDVOL G:\WINNT\SYSDVOL\sysvol
```

Um nun zu erreichen, dass nur solche Freigaben angezeigt werden, die zu einer Festplatten-Ressource gehören, erweitert man einfach das SELECT-Statement um ein weiteres Kriterium. Für die Klasse Win32_Share lautet das in Frage kommende Kriterium Type und muss für eine Disk-Ressource den Wert 0 aufweisen, wobei die administrativen Shares, also C\$,ADMIN\$ und so weiter nicht als Disk-Ressource vom Typ 0 zählen. Das modifizierte Skript würde entsprechend folgendermaßen aussehen:

```
Set EnumShares = GetObject
("winmgmts:{impersonationLevel=impersonate}").E
xecQuery("SELECT * FROM Win32_Share
WHERE Name<>" AND Type=0")
For each Share in EnumShares
WScript.Echo Share.Name & vbTab & Share.Path
Next
```

Ruft der Administrator nun das Skript auf, erhält er nur noch folgende Ausgabe auf dem Bildschirm:

```
NETLOGON
G:\WINNT\SYSDVOL\sysvol\pelzer-consulting.de
\SCRIPTS
SYSDVOL G:\WINNT\SYSDVOL\sysvol
```

Die administrativen Freigaben und der IPC\$-Share sind darin nicht mehr enthalten. Ebenso wenig würden eventuelle vorhandene freigegebene Drucker aufgeführt werden.

Bei der in WMI integrierten Abfragesprache WQL für »WMI Query Language« handelt es sich um eine Untermenge von ANSI SQL des American National Standards Institute, das als Abfragesprache von den meisten relationalen Datenbanken unterstützt wird.

Da die vom CIMOM geführte Datenbank ebenfalls dem relationalen Modell entspricht, macht der Einsatz von WQL natürlich Sinn.

Monikers verschaffen nötige Rechte

Ein Aspekt, der bei der Betrachtung des Beispiel-Skripts bislang unerwähnt geblieben ist, ist der Ausdruck `winmgmts:{impersonationLevel=impersonate}`. Dabei handelt es sich um einen als »Moniker« bezeichneten Standardmechanismus aus dem Common-Object-Model. Der Moniker dient innerhalb des `GetObject`-Aufrufs zur korrekten Authentifizierung des Skripts gegenüber dem CIMOM. Über den Ausdruck `{impersonationLevel=impersonate}` wird der `GetObject`-Aufruf angewiesen, die Sicherheitseinstellungen des gerade angemeldeten Benutzers zu verwenden. Im Einzelfall kann es vor kommen, dass ein Benutzer nicht über die notwendigen Privilegien, wie zum Beispiel `Debug`, `IncreaseQuota` oder `Backup` verfügt, um einen `GetObject`-Aufruf auszuführen. In einem solchen Fall kann das Skript dahingehend erweitert werden, dass dem Aufrufer für die Ausführung des Skripts das entsprechende Privileg zugewiesen wird. Der Aufrufer muss natürlich über das Recht verfügen, Privilegien zu ändern. Der Moniker

```
winmgmts:{impersonationLevel=impersonate,
(debug)}
```

würde beispielsweise dem Aufrufer das `Debug`-Privileg zuweisen, vorausgesetzt natürlich er hat die Berechtigung, sich selbst dieses Privileg zu gewähren. Über den Moniker wird darüber hinaus noch festgelegt, auf welchem System der Aufruf erfolgen soll. Ist nicht angegeben, so erfolgt die Ausführung auf der lokalen Maschine. Möchte der Administrator aber ein anderes System ansprechen, so muss er dessen Namen angeben. Über `winmgmts:{impersonationLevel=impersonate}//w2k.pelzer-consulting.de` würde somit die Abfrage auf das System mit dem Hostnamen `w2k.pelzer-consulting.de` umgeleitet und dort bearbeitet werden. Wie das funktioniert, ist in nachfolgendem Beispiel dargestellt. Mit diesem Skript werden innerhalb einer angegebenen Domäne alle Computer daraufhin untersucht, ob auf ihnen Dienste vorhanden sind,

deren Startart auf automatisch gesetzt ist, die aber nicht gestartet sind. Sobald das Skript einen Dienst gefunden hat, auf den die Kriterien zutreffen, versucht es, diesen zu starten. Kann es den Dienst nicht starten, so gibt es die entsprechende Fehlernummer zurück. Das Skript ist so geschrieben, dass es sowohl in einer Windows-2000-, als auch einer Windows-NT-Umgebung funktioniert.

```
1 Set Domain = GetObject("WinNT://" &
  strDomain)
2 Domain.Filter = Array("Computer")
3
4 For Each Computer in Domain
5 wscript.echo "Bearbeite Comuter: " & Computer.
  Name
6 Set Services = GetObject("winmgmts:
  {impersonationLevel=impersonate}://" &
  Computer.Name).ExecQuery("select * from
  Win32_Service where State='Stopped' and
  StartMode='Auto'")
7 For Each Service in Services
8 WScript.Echo "Versuche den Dienst " & Service.
  Description & " zu starten"
9 ReturnCode = Service.StartService
10 If ReturnCode = 0 Then
11 wscript.echo "Dienst erfolgreich gestartet."
12 Else
13 wscript.echo "Dienst konnte nicht gestartet
  werden. Fehlercode: " & ReturnCode
14 End If
15 Next
16 Next
```

- Über `GetObject` wird der Variablen `Domain` eine Liste von Instanzen der in der Variablen `strDomain` angegebenen Domäne zugewiesen.
- Durch Verwendung der Methode `Filter` werden die in der Variablen `Domain` abgelegten Instanzen auf diejenigen reduziert, die die Klasse `Computer` enthalten.
- `For`-Schleife, die für jeden in der Variablen `Domain` enthaltenen `Computer` durchlaufen wird.
- Ermittlung aller Dienste für den jeweils angegebenen `Computer`, bei dem Kriterien `Startart` gleich `automatisch` und `Zustand gestoppt` erfüllt sind.
- `For`-Schleife, innerhalb derer versucht wird, jeden Dienst zu starten, auf den die im WQL-

- Statement beschriebenen Kriterien zutreffen
- 9 Versuch, den angegebenen Dienst über die Methode StartService zu starten. Das Resultat wird in der Variablen ReturnCode zurückgeliefert.
 - 10 Falls der ReturnCode gleich 0 ist, konnte der Dienst gestartet werden
 - 12 Falls der ReturnCode ungleich 0 ist, schlug der Versuch, den Dienst zu starten fehl und der ReturnCode enthält den dazugehörigen Fehler. Eine Liste der Fehlermeldungen ist in der Beschreibung der StartService-Methode der Win32-Service-Klasse im Platform SDK für Windows-2000 beschrieben.

Mit WMI auf Systemereignisse reagieren

Wie bereits erwähnt, liegt eine Stärke von WMI in der Möglichkeit, auf bestimmte Systemereignisse zu reagieren. Solche Systemereignisse können beispielsweise das Schreiben eines Event-Log-Eintrages, die Änderung eines Registry-Parameters oder die Überschreitung eines Schwellwertes für einen Performance-Counter sein. Sobald ein dertartiges Ereignis eintritt, ist WMI in der Lage, ein WMI-Event zu kreieren und damit einen Event-Consumer etwa in Form eines Skriptes zu benachrichtigen. Das nachfolgende Skript veranschaulicht die Vorgehensweise in einem solchen Fall. Dabei soll eine Meldung auf den Bildschirm geschrieben werden, wenn die CPU-Auslastung des betrachteten Systems größer als zehn Prozent ist. Als Besonderheit in diesem Skript wird gezeigt, wie man ein WMI-Event erzeugen kann, obwohl für die Klasse eigentlich keine Eventunterstützung existiert.

- 1 Set PerfCounter = GetObject("winmgmts:
{impersonationLevel=impersonate}").ExecNotificationQuery ("SELECT * FROM __InstanceModificationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32_Processor' AND TargetInstance.LoadPercentage > 10")
- 2
- 3 WScript.Echo "Warte auf auf CPU-Schwellwert-
überschreitung"

- 4 WScript.Echo ""
- 5 do
- 6 Set NTEvent = PerfCounter.NextEvent
- 7 WScript.Echo "Schwellwertüberschreitung für: "
& NTEvent.TargetInstance.DeviceID
- 8 WScript.Echo "Gemessene Last: " & NTEvent.
TargetInstance.LoadPercentage
- 9 loop

Wie schon bei den vorangegangenen Skripten, so wird auch in diesem Fall zunächst mit GetObject ein Objekt angefordert und dessen Inhalt der Variablen PerfCounter zugewiesen. Der inzwischen bekannte Moniker winmgmts:{impersonationLevel=impersonate} weist dabei darauf hin, dass das Skript lokal mit den Standardrechten des angemeldeten Benutzers ausgeführt werden soll. Die Methode »ExecNotificationQuery« weist den CIMON an, ein WMI-Event zu generieren, sobald das innerhalb des WQL-Ausdrucks angegebene SELECT-Statement erfüllt ist. Die verwendete Klasse »__InstanceModificationEvent« zeigt an, dass bei einer Modifikation der Eigenschaft »LoadPercentage« der Klasse »Win32_Processor« ein WMI-Event erzeugt und an den Consumer, also in diesem Fall das aufrufende Skript geschickt werden soll. Um die Zahl der gemeldeten Ereignisse einzuschränken, soll nur dann ein Event erzeugt werden, wenn die CPU-Last größer als zehn Prozent ist, was durch »TargetInstance.LoadPercentage > 10« gewährleistet ist. Über den Ausdruck WITHIN 5 wird WMI mitgeteilt, dass es nur alle fünf Sekunden prüfen soll, ob sich eine Änderung des gesuchten Wertes ergeben hat. Als Intervalldauer sind übrigens auch Werte unter einer Sekunde, also zum Beispiel 0,1 darstellbar. Allerdings sollte man die Intervalldauer nicht zu kurz setzen, da sich WMI sonst unter Umständen aus Performancegründen weigert, den Wert anzunehmen. Im Rahmen einer Do-Schleife wartet das Skript mit dem Befehl PerfCounter.NextEvent auf neue WMI-Events und gibt bei deren Eintreffen den Bezeichner der CPU aus, für die das Ereignis generiert wurde und zusätzlich den gemessenen Wert für die CPU-Auslastung.

WMI gestattet auf einfache Weise die Überwachung von Ereignissen, wie etwa das Überschreiten einer bestimmten CPU-Auslastung

```
G:\WINNT\system32\cmd.exe - cscript wmi_cpu_perf.vbs
E:\vbs>cscript wmi_cpu_perf.vbs
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights reserved.

Warte auf auf CPU-Schwellwertüberschreitung

Schwellwertrüberschreitung für: CPU0
Gemessene Last: 92
Schwellwertrüberschreitung für: CPU0
Gemessene Last: 93
Schwellwertrüberschreitung für: CPU0
Gemessene Last: 95
Schwellwertrüberschreitung für: CPU0
Gemessene Last: 94
Schwellwertrüberschreitung für: CPU0
Gemessene Last: 93
```

Fazit

In diesem Workshop konnten die zahlreichen Funktionen, die WMI bietet nur oberflächlich gestreift werden. Insbesondere die Installation von Applikationen mit dem Windows-Installer verdient besondere Aufmerksamkeit, da hierdurch die Verteilung von Softwarepaketen deutlich erleichtert wird. Als weiterführende Lektüre und für Skriptentwicklung unentbehrliche Referenz sei auf den WMI-Teil des Microsoft Platform-SDK verwiesen, das im Internet unter msdn.microsoft.com zum Download oder als HTML-Ressource zur Verfügung steht.

Weiterführende Ressourcen zum Thema Scripting und Management mit Windows-2000 und Windows-NT

<http://msdn.microsoft.com/scripting/>
http://msdn.microsoft.com/library/psdk/wmisdk/wmi_start_5kth.htm
http://msdn.microsoft.com/library/psdk/adsi/dsstartpage_1rg3.htm
<http://www.15seconds.com/>
<http://wsh.glazier.co.nz>
<http://cwashington.netreach.net>
<http://www.dmtf.org>

Windows-2000-Provider

Die folgenden Provider sind Bestandteil von Windows-2000:

Win32-Provider

Stellt Informationen über das Betriebssystem, Peripheriegeräte und Dateisysteme sowie Sicherheitsinformationen bereit

Event-Log

Erlaubt das Auslesen, Konfigurieren und Sichern der Event-Logs.

Registry-Provider

Ermöglicht das Anlegen, Lesen und Schreiben von Registry-Keys. Performance-Counter-Provider stellt alle im System zur Verfügung stehenden Performance-Counter zum Auslesen bereit.

Active Directory-Provider

Fungiert als Gateway für den Zugriff auf sämtliche Daten des Active Directory Service.

SNMP-Provider

Fungiert als Gateway für Objekte, auf normalerweise per SNMP zugegriffen wird. Auf die MIB-Objekte ist sowohl lesender als auch schreibender

Zugriff möglich.

Windows-Installer-Provider

Erlaubt die vollständige Kontrolle über den Windows-Installer und liefert Informationen über alle mit dem Windows-Installer installierten Anwendungen. Ermöglicht die vollständige Installation von Applikationen.

View-Provider

Erlaubt den Aufbau neuer Klassen aus den bestehenden, mit denen nur die Informationen dargestellt werden, die für eine bestimmte Aufgabe von Interesse sind.

Windows-Driver-Modell-Provider

Liefert detaillierte Angaben über Treiber aus den Bereichen Eingabegeräte, Speicher, Netzwerkkarten und Kommunikationsschnittstellen.

Common Information Model (CIM)

Das Common-Information-Model beschreibt, wie Managed-Objects dargestellt werden. Dabei kommt ein objektorientierter Ansatz zum tragen, der die Objekte durch Klassen und Instanzen realisiert. Auf diese Weise erfolgt eine durchgängige und vereinheitlichte Darstellung aller Arten von logischen und physikalischen Objekten in einer gemanagten Umgebung.

Distributed Management Task Force (DMTF)

Die DMTF ist ein Gremium, das mit verschiedenen Herstellern aus der IT-Branche, wie Microsoft, Compaq oder Sun zusammenarbeitet, um einheitliche Standards für das Management von Desktop-, Unternehmens- und Internetumgebungen zu definieren.

Web-Based-Enterprise-Management (WBEM)

Initiative der DMTF zur Bereitstellung einer Standardlösung für das Management von Computersystemen.

Windows-Management-Instrumentation (WMI)

Microsofts Implementierung von WBEM für Windows-NT/2000, sowie Windows-95 und 98.

Managed Object

Hard- oder Softwarekomponente, die als Instanz einer WMI-Klasse dargestellt wird.

Zur Person

DIPL. ING. DIRK PELZER arbeitet als freier Consultant und Journalist in München. Er betreibt ein Storage Labor für verschiedene namhafte Fachzeitschriften. Zudem beschäftigt er sich mit Speichernetzen und Hochverfügbarkeit.