E-BOOK

The Comprehensive Guide to WMI

LogicMonitor

E-BOOK

The Comprehensive Guide to WMI

This free E-Book was created for you by our team of IT experts at LogicMonitor. Get more at logicmonitor.com/resources

What is WMI?

WMI stands for *Windows Management Instrumentation*. WMI's purpose is similar to that of SNMP (*Simple Network Management Protocol*): to enable the querying and control of management information in an enterprise - but it has a significantly different architecture under the covers. Compared to SNMP, WMI provides a higher-level representation of systems, in that it supports properties, events and methods on top of classes of objects, along with a more powerful query language than SNMP supports. Of course, this also means it's a bit more complex to use, and has more overhead on the systems. It has been available in Windows since way back in the Windows 95 and Windows NT era.

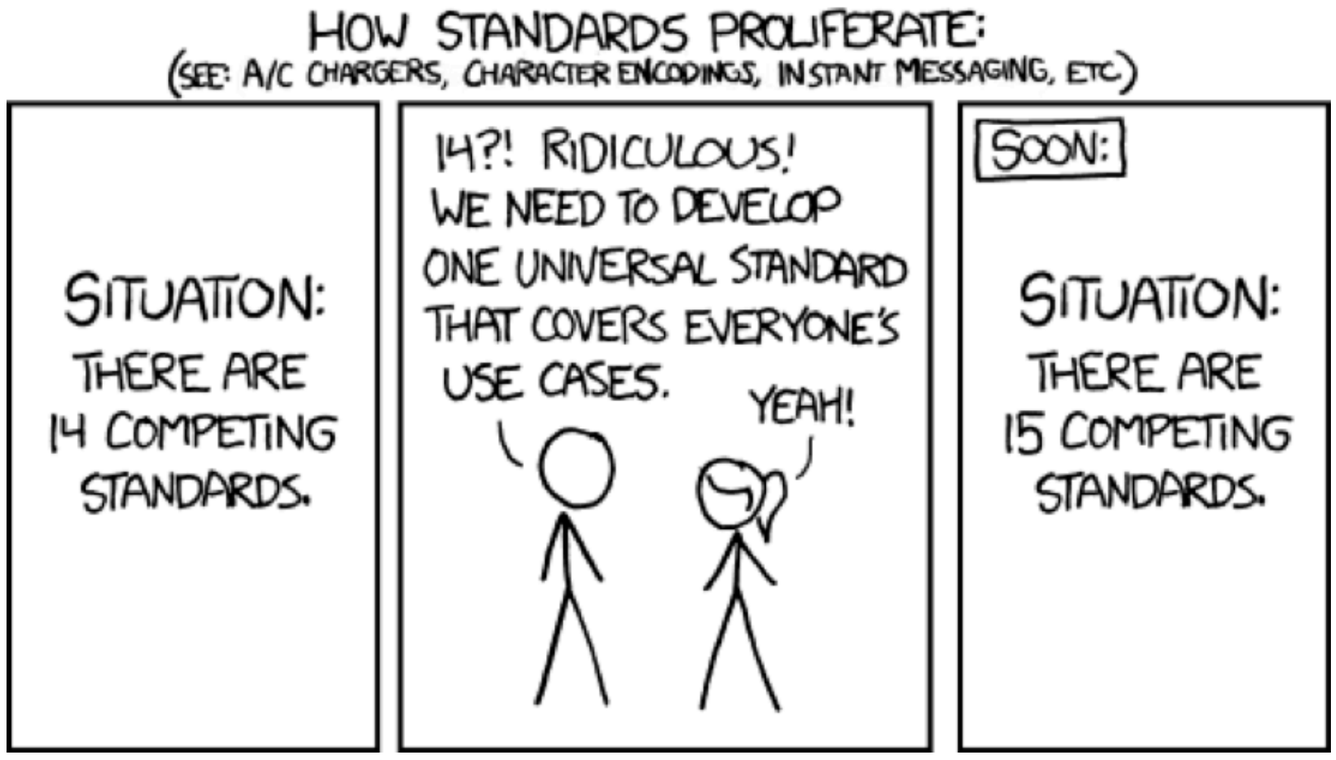
WMI's purpose is similar to that of SNMP: to enable the querying and control of management information in an enterprise.

WMI is the Microsoft implementation of Web-Based Enterprise Management (WBEM), which is an industry initiative for a standard technology for accessing management information in an enterprise environment. WMI uses the Common Information Model (CIM) industry standard. And as happens with so many implementations of standards – it is not at all interoperable with any other version of the standard.

In other words, WMI is an implementation of 'standards based' management, which only works on and with Windows.

E-BOOK

The Comprehensive Guide to WMI



Credit: https://xkcd.com/927/

However, as WMI has been Microsoft's main (but not exclusive! More on that later) focus for exposing information about system performance and configuration, it is the preferred way of collecting information from Windows systems for many monitoring systems. (Windows systems can run SNMP, but compared to the built-in WMI support, the built-in SNMP agent provides very little information.)

See it in Action! Querying WMI using WBEMTest

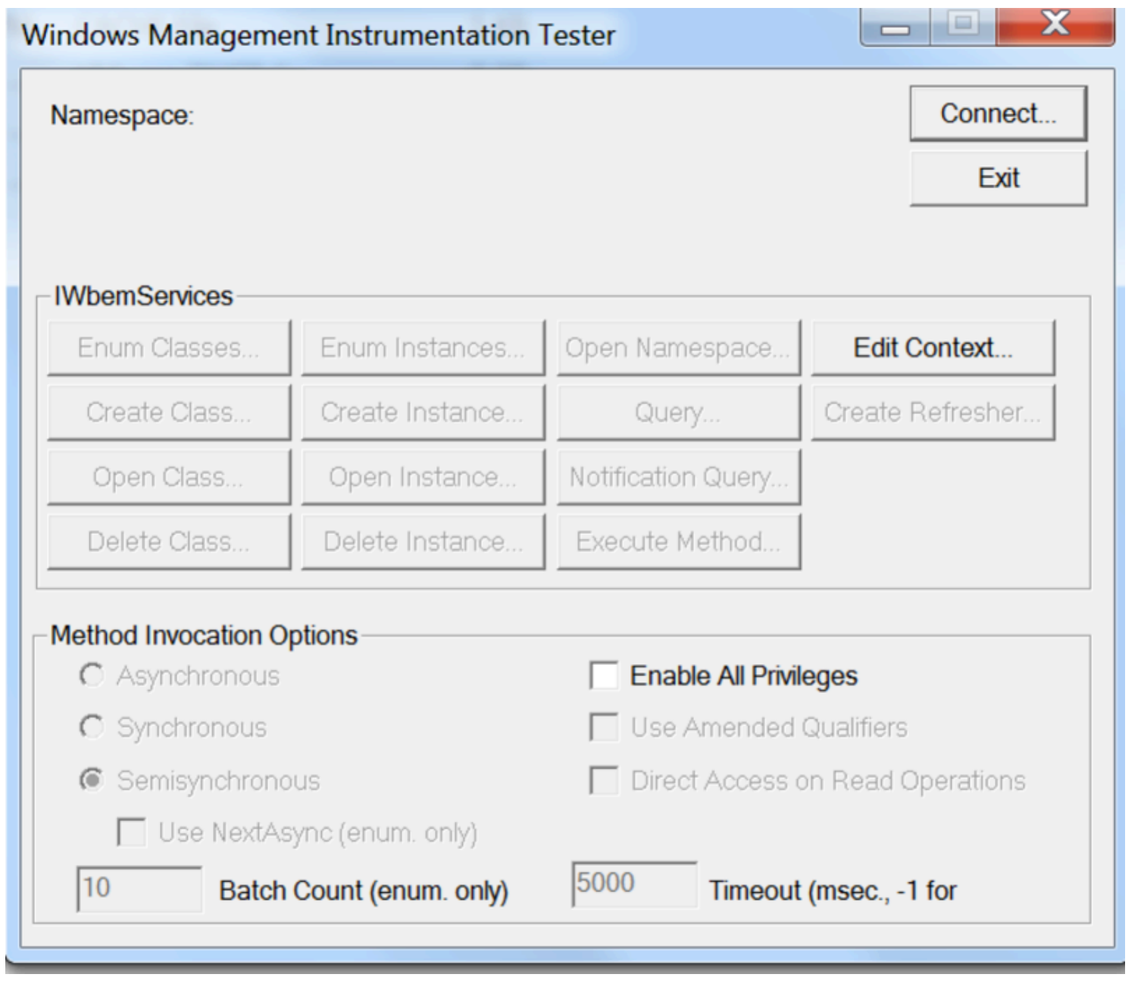
There are a variety of ways of querying WMI. We'll discuss these later, but for now, we'll use WBEMtest, which is a Microsoft tool for testing and using WMI. It has the advantage of being included with Windows.

WBEMTest can connect to the local computer and to remote computers – but connecting to the local computer avoids any of the many possible issues that can interfere with remote WMI. (Again – more on that later.)

This free E-Book was created for you by our team of IT experts at LogicMonitor. Get more at logicmonitor.com/resources

The Comprehensive Guide to WMI

Run WBEMTest (from the Start menu), and you'll see a simple screen:



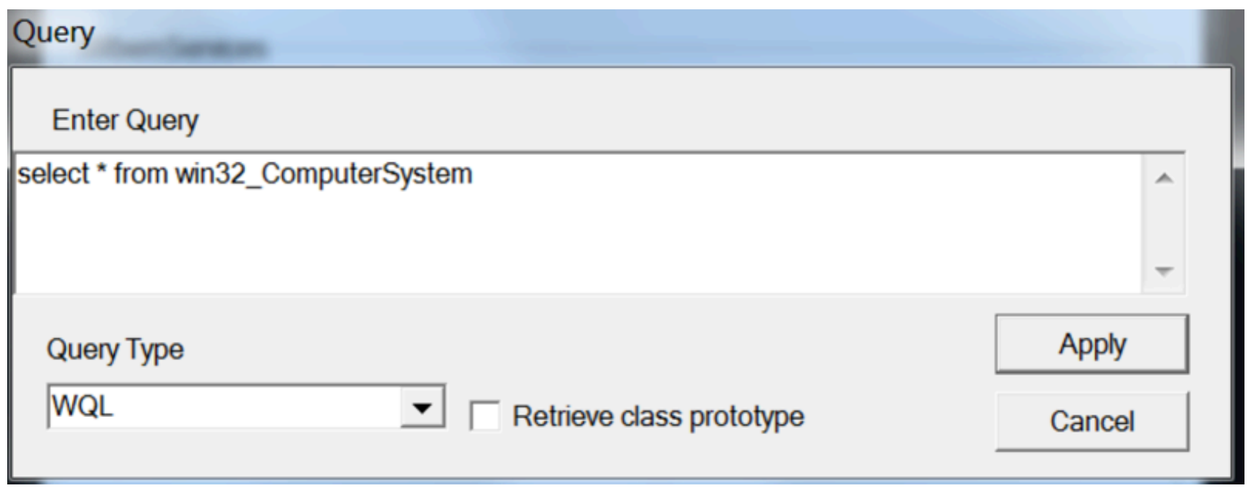
Click Connect, and then Connect again on the next form to display. The defaults are appropriate for connecting to the local system.

Now that we are connected, the form will have the buttons enabled. Clicking the "Query" button will allow us to enter a WQL (Windows Query Language) query, which is how WMI data is queried.

Enter the query:

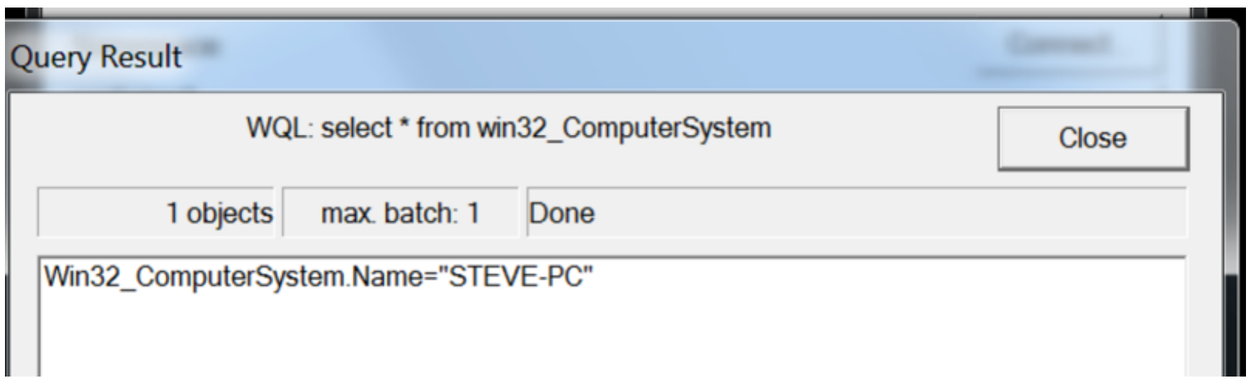
Select * from win32_computerSystem

and click apply:



The Comprehensive Guide to WMI

A window should return, with one line, listing the name of your computer:



Double click the result line, and a new window will open, listing all the properties (and methods) of the returned object.

There will be properties for the Manufacturer, the BootUpState, Domain, TotalPhysicalMemory, and so on.

In this case, we queried the class Win32_ComputerSystem, in the Namespace cimv2. The namespace was specified when we connected – cimv2 is the default namespace, and where almost all Windows performance and configuration data is found. Some specialized software, with different security requirements, may use a different namespace.

Above, we used a very simple query to show information about the computer itself - but you can also query information for things with multiple occurrences, and use conditions to refine the results.

For example:

select * from Win32_NetworkAdapter where AdapterType like "%Ethernet%"

will return information about ethernet network adapters, but not wireless or ATM adapaters.

The full definition of WQL options are available on **this MSDN page**.

E-BOOK

The Comprehensive Guide to WMI

WQL is similar to standard SQL (Structured Query Language), used by many popular databases. If you're familiar with SQL, the comparison table below may help:

Concept	SQL	WMI
Individual items	rows	instances
Characteristics	columns	properties
Containers of columns and rows	tables	classes
Containers of tables	databases	namespaces
Program code that functions on data	stored procedures	methods

What can you query with WMI?

The answer is easy: almost anything. WMI has classes for almost anything you may want to know about a computer – hardware information, performance information, software information – and in many cases there are classes added for installed software, so you can even monitor things like Exchange mailboxes. The hard thing is in knowing how to get the information.

Unlike SNMP, WMI does not have the concept of a published MIB (*Management Information Base*, detailing all the SNMP queries that are supported). Some classes are well documented, but many are not documented at all, or only mentioned in passing on occasional blogs.

E-BOOK

The Comprehensive Guide to WMI

WMI information is in a hierarchical tree under namespaces. Some of the namespaces have hundreds of classes in them and each class often has dozens of properties. This can make finding the namespace and class that has the information you are looking for the hardest part of using WMI.

A good place to start for finding the right WMI classes is **this MSDN page**. While this is a great resource for WMI information, it does not mean that the information you want will be easy to find. For instance – to determine that a system's BootUpState (for example "Normal Boot", as opposed to "Fail-safe boot") can be found in the Win32_computerSystem name class we used above, you'd have to navigate to the WMI providers page, down into the CIMWin32 section, then to Win32 Provider, then Operating System Classes, and finally to the Win32_ComputerSystem page, for a listing of all the properties and methods of that class. Often it's easier to use your favorite search engine to search for what you want, by searching for a phrase like **WMI boot state**. (Ironically, the first Google result for that search is the Win32_computerSystem class page, but that result doesn't show on the first page of Bing…)

There are, of course, things that are not exposed via WMI. Some aspects of Exchange can only be monitored via powershell scripts (such as Get-MailboxDatabaseCopyStatus – there is no equivalent WMI class.) The core OS is well instrumented via WMI, but the extra packages (Exchange, SQL Server, Clustering, etc) seem as if they sometimes added on WMI support as an afterthought, and forgot to expose some important things. In general, though, most things you would want to query are exposed by WMI.

E-BOOK

The Comprehensive Guide to WMI

This free E-Book was created for you by our team of IT experts at LogicMonitor. Get more at logicmonitor.com/resources

Why would you use WMI? Because SCRIPTS!

If you are monitoring your Windows servers with an automated platform like LogicMonitor, most likely it is using WMI to query a good chunk of the information it is collecting about your servers.

The tool will have all the WMI discovery, queries and filtering taken care of for you. Nonetheless, understanding WMI is great if you want to extend the monitoring.

But WMI is also great for system administration apart from monitoring. Being able to query your own computer's boot-state is nice – but not terribly useful. Being able to run a script that uses WMI to query all the computers on your network, to see if any have booted into fail-safe mode and need attention, and being able to update drivers on those that do – well, that is more interesting.

WMI is easy to call from programming and scripting languages, which means it's easy to perform queries and reports on all the computers you are responsible for.

Below is a simple script using **VBscript**:

```
' Lines that begin with this single accent are just comments

' set computer to current computer

strComputer = "."

' connect to WMI in the namespace called “CIMV2”

Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\CIMV2")

' run the WMI query

Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_BIOS")

'Loops thru each BIOS Object and gets the value and shows the value

For Each objItem in colItems

    Wscript.Echo "SerialNumber: " & objItem.SerialNumber

Next

'end
```

The Comprehensive Guide to WMI

If you save this file (make sure you use ANSI encoding) with a .vbs extension, and run it, you will get a dialog pop up showing the serial number of the BIOS.

You can achieve similar results using **Powershell** commands to access WMI:

Get-WmiObject win32_BIOS | Select SerialNumber

There are also tools such as "**Scriptomatic**" which not only help in exploring WMI namespaces and the WMI classes within those Namespaces - it also can create script/code in various languages to extract the WMI information.

There are also many sites with pre-built WMI scripts, such as **this one** that uses WMI to perform an inventory report on all computers found in the AD tree.

WMI to remote computers

There are several requirements to making WMI work on remote computers:

- Network access for WMI (which uses DCOM) needs to be allowed by any firewalls on or between the computers.
- User access and credentials are needed for remote WMI access.

Firewalls and WMI:

Because WMI uses DCOM (Distributed Component Object Model) by default to communicate between computers, it doesn't use a single port – so it's not easy to allow through firewalls. DCOM uses TCP port 135 to initiate connections, but then dynamically chosen ports are used to actually transfer data.

The Comprehensive Guide to WMI

The actual port numbers are different depending on the version of Windows. On older versions of Windows (i.e. Windows 2000, Windows XP and Windows Server 2003) the dynamically chosen ports are in the range of 1025 to 5000. On newer versions of Windows (Vista, Windows Server 2008 and later) ports are in the range of 49152 to 65535. See this **Microsoft article** for details. This makes it hard to run WMI through firewalls without opening up a wide range of ports.

The Microsoft built in firewall can deal with the dynamic ports, but by default will block WMI. To enable remote WMI access while using the Windows Firewall:

- You can use a netsh firewall command at the command prompt to allow for remote administration. The following command enables this feature:

netsh firewall set service RemoteAdmin enable or (depending on your version of Windows)
netsh advfirewall firewall set rule group="windows management instrumentation (wmi)" new enable=yes

- You can use either the Group Policy editor (Gpedit.msc) or a script to enable the *Windows Firewall: Allow remote administration* exception. To use the Group Policy editor, use the following steps in the Group Policy editor to enable "Allow Remote Administration":

  a. Under the **Local Computer Policy** heading, double click **Computer Configuration**.

  b. Double-click **Administrative Templates**, **Network, Network Connections**, and then **Windows Firewall**.

  c. If the computer is in the domain, then double-click **Domain Profile**; otherwise, double-click **Standard Profile**.

d. Click **Windows Firewall: Allow remote administration exception**.

e. On the **Action** menu, select **Properties**.

f. Click **Enable**, and then click **OK**.

There are other approaches to making WMI more firewall friendly – such as limiting the range of ports that DCOM will use via dcomcnfg, or configuring the WMI service to run in standalone mode with a fixed port via WinMgmt.exe /StandAloneHost - but, as in most systems administration tasks, the less things that are changed from default, the less problems you are likely to run into, so we won't explore these methods.

WMI User Credentials:

Connecting to the WMI service on the local system avoids any issues with user credentials (indeed, it refuses to use them), and will always connect with the account of the user running the WMI query. When connecting to a remote server, however, you can use different credentials to connect.

The simplest case is where the account used to connect to WMI on the remote computer is a domain account in the Administrators group (not necessarily a domain Admin.) Because of User Account Control, the account running the WMI query must be in the Administrators group on the local computer to have the ability to run with elevated rights. This means that the user you connect as must be a domain account in the administrators group, on the computer you are querying WMI on.

E-BOOK

The Comprehensive Guide to WMI

This free E-Book was created for you by our team of IT experts at LogicMonitor. Get more at logicmonitor.com/resources

If you are using an account other than your own to connect to remote servers via scripts, or with a monitoring tool such as LogicMonitor, it is recommended that you set “password does not expire” on this user account – otherwise when the password expires, it may require changing many scripts or services.

Troubleshooting WMI

WMI is great when it works, but when it doesn’t – it can be frustrating. Generally the first step is ensuring that WMI works locally, using WBEMTest while logged in to the computer itself. This eliminates any firewalls and username password/domain issues.

But even then, you will sometimes run into oddities – in which case these are some things to check.

WMI Services & Dependencies

All of the following services should be running and set to an "Automatic" startup type for WMI monitoring to work correctly:

- DCOM Server Process Launcher
- Remote Procedure Call (RPC)
- RPC Endpoint Mapper
- Windows Management Instrumentation

And the following service(s) may be set to a "Manual" startup type:

- WMI Performance Adapter

E-BOOK

The Comprehensive Guide to WMI

Weird Data from WMI

Sometimes you'll find that while some WMI queries work fine, some WMI data cannot be retrieved – even for objects that do in fact exist. You may retrieve errors such as "Empty result set", or the permission error 0x80041003 on some objects, but not others.

In this case, **rebuilding the Performance Counter library** may be necessary. Why? Because .. well… Erm, the registry……. Just try it. It does actually fix issues.

If WBEMTest works locally, but remote WMI does not – you most likely have either a firewall issue, or are passing in incorrect user credentials. For further WMI troubleshooting advice – see this **Microsoft page**.

What next?

Well, having read this far, you'll most likely be using WMI either in a monitoring tool, where you may want to add in some extra classes to get some specific information the monitoring doesn't provide out-of-the-box, or you'll be using scripting (most likely Powershell) to manage and script systems administration tasks.

For example, you should be able to easily write a quick powershell to get free space information on all drives on a remote computer (or all your computers!):

```
get-wmiobject -class Win32_Volume -computername atl-fs-01 |    Select-Object name,freespace,driveletter | Sort-Object name
```

And here's the kind of data you get back:

```
NAME                                   FreeSpace DriveLetter
----                                   --------- -----------
\\?\Volume{56fde643-619...              74199040
C:\                                  89016860672 C:
D:\                                        48676 D:
E:\                                              E:
F:\                                              F:
```

LogicMonitor

This free E-Book was created for you by our team of IT experts at LogicMonitor. Get more at logicmonitor.com/resources

E-BOOK

The Comprehensive Guide to WMI

But be advised – once you dive into this world of automated monitoring and management – there is a lot to keep up with.

For example, Microsoft is now using Windows Management Infrastructure (with the helpfully distinguishing acronym MI), which is backward compatible with Windows Management Interface – but is more compliant with the current management standards; can be run without DCOM, and is easier for software companies to write providers for. For similar reasons, powershell is deprecating the common Get-WMiObject Cmdlet – it has been superseded by Get CimInstance. (But Get-CimInstance uses a different protocol by default, which doesn't work with 2003 servers – but can be configured to use DCOM for backward compatibility… No one said this was easy!)

If you always run the same level of servers everywhere, it's less complicated, but with a mix of older and newer operating systems to support – the mix of tools that will work on each will require you to expand your bag of tricks all the time.

Summary

WMI is a very powerful way to retrieve information about systems and their performance. Knowing your way around WMI can help you improve the monitoring of your systems, and help you automate many common systems administration tasks. Indeed – an installation option on Windows 2016 is the Nano Server – which has no GUI, and has all management performed on it remotely via WMI and powershell. WMI is the de facto foundation of Microsoft's management strategy. The more you are familiar with it, the better systems administrator you can be. And helping you and your team improve is what LogicMonitor is about.

LogicMonitor

LogicMonitor.com