wmi Cookbook

Tim Golden > Python Stuff > wmi > WMI Cookbook

Introduction

These examples assume you are using the WMI module from this site. The following are examples of useful things that could be done with this module on win32 machines. It hardly scratches the surface of WMI, but that's probably as well.

The following examples, except where stated otherwise, all assume that you are connecting to the current machine. To connect to a remote machine, simply specify the remote machine name in the WMI constructor...

```
import wmi
c = wmi.WMI ("some_other_machine")
```

... and by the wonders of DCOM, all should be well.

Note also that the examples are designed to be complete and can be cut-and-pasted straight into a .py file, or even onto an open Python interpreter window (at least running under CMD on Win2000; that's how I test them). Just select the code, including the final blank line, right-click [Copy], select your Python interpreter window, and right-click.

Examples

List all running processes

```
import wmi
c = wmi.WMI ()
for process in c.Win32_Process ():
  print process.ProcessId, process.Name
```

List all running notepad processes

```
import wmi
c = wmi.WMI ()
for process in c.Win32_Process (name="notepad.exe"):
  print process.ProcessId, process.Name
```

Create and then destroy a new notepad process

```
import wmi
c = wmi.WMI ()
process_id, return_value = c.new ("Win32_Process").Create (CommandLine="notepad.exe")
for process in c.Win32_Process (ProcessId=process_id):
  print process.ProcessId, process.Name
result = process.Terminate ()
```

Show the interface for the .Create method of a Win32_Process class

Notes:

The wmi module tries to take the hard work out of WMI methods by querying the method for its in and out parameters, accepting the in parameters as Python keyword params and returning the output parameters as an tuple return value. The function which is masquerading as the WMI method has a __doc__ value which shows the input and return values.

```
import wmi
c = wmi.WMI ()
print c.new ("Win32_Process").Create
```

Show all automatic services which are not running

```
import wmi
c = wmi.WMI ()
stopped_services = c.Win32_Service (StartMode="Auto", State="Stopped")
if stopped_services:
  for s in stopped_services:
    print s.Caption, "service is not running"
else:
  print "No auto services stopped"
```

Show the percentage free space for each fixed disk

```
import wmi
c = wmi.WMI ()
for disk in c.Win32_LogicalDisk (DriveType=3):
  print disk.Caption, "%0.2f%% free" % (100.0 * long (disk.FreeSpace) / long (disk.Size)
```

Run notepad, wait until it's closed and then show its text

Notes:

This is an example of running a process and knowing when it's finished, not of manipulating text typed into Notepad. So I'm simply relying on the fact that I specify what file notepad should open and then examining the contents of that afterwards.

This one won't work as shown on a remote machine because, for security reasons, processes started on a remote machine do not have an interface (ie you can't see them on the desktop). The most likely use for this sort of technique on a remote server to run a setup.exe and then, say, reboot once it's completed.

```
import wmi
c = wmi.WMI ()
filename = r"c:\temp\temp.txt"
process = c.new ("Win32_Process")
process_id, result = process.Create (CommandLine="notepad.exe " + filename)
watcher = c.watch_for (
  notification_type="Deletion",
  wmi_class="Win32_Process",
  delay_secs=1,
  ProcessId=process_id
)
watcher ()
print "This is what you wrote:"
print open (filename).read ()
```

Watch for new print jobs

```
import wmi
c = wmi.WMI ()
print_job_watcher = c.watch_for (
  notification_type="Creation",
  wmi_class="Win32_PrintJob",
  delay_secs=1
)
while 1:
  pj = print_job_watcher ()
  print "User %s has submitted %d pages to printer %s" % \
    (pj.Owner, pj.TotalPages, pj.Name)
```

Reboot a remote machine

Notes:

To do something this drastic to a remote system, the WMI script must take RemoteShutdown privileges, which means that you must specify them in the connection moniker. The WMI constructor allows you to pass in an exact moniker, or to specify the parts of it that you need. Use help on wmi.WMI.__init__ to find out more.

```
import wmi
c = wmi.WMI (computer="other_machine", privileges=["RemoteShutdown"])
os = c.Win32_OperatingSystem (Primary=1)[0]
os.Reboot ()
```

Show the IP and MAC addresses for IP-enabled network interfaces

```
import wmi
c = wmi.WMI ()
for interface in c.Win32_NetworkAdapterConfiguration (IPEnabled=1):
  print interface.Description, interface.MACAddress
  for ip_address in interface.IPAddress:
    print ip_address
  print
```

What's running on startup and from where?

```
import wmi
c = wmi.WMI ()
for s in c.Win32_StartupCommand ():
  print "[%s] %s <%s>" % (s.Location, s.Caption, s.Command)
```

Watch for errors in the event log

```
import wmi
c = wmi.WMI (privileges=["Security"])
watcher = c.watch_for (
  notification_type="Creation",
  wmi_class="Win32_NTLogEvent",
  Type="error"
)
while 1:
  error = watcher ()
  print "Error in %s log: %s" %  (error.Logfile, error.Message)
  # send mail to sysadmin etc.
```

List registry keys

```
import _winreg
import wmi
r = wmi.Registry ()
result, names = r.EnumKey (hDefKey=_winreg.HKEY_LOCAL_MACHINE, sSubKeyName="Software")
names = list (names.Value)
for key in names:
  print key
```

Add a new registry key

```
import _winreg
import wmi
r = wmi.Registry ()
result, = r.CreateKey (hDefKey=_winreg.HKEY_LOCAL_MACHINE, sSubKeyName=r"Software\TJG")
```

Add a new registry value

```
import _winreg
import wmi
r = wmi.Registry ()
result, = r.SetStringValue (
  hDefKey=_winreg.HKEY_LOCAL_MACHINE,
  sSubKeyName=r"Software\TJG",
  sValueName="ApplicationName",
  sValue="TJG App"
)
```

Create a new IIS site

NB This has only been tested on Win2k3 / IIS6.

```
import wmi
c = wmi.WMI (namespace="MicrosoftIISv2")
#
# Could as well be achieved by doing:
#  web_server = c.IISWebService (Name="W3SVC")[0]
#
for web_server in c.IIsWebService (Name="W3SVC"):
  break

binding = c.new ("ServerBinding")
binding.IP = ""
binding.Port = "8383"
binding.Hostname = ""
result, = web_server.CreateNewSite (
  PathOfRootVirtualDir=r"c:\inetpub\wwwroot",
  ServerComment="My Web Site",
  ServerBindings= [binding.ole_object]
)
```

Show shared drives

```
import wmi
c = wmi.WMI ()
for share in c.Win32_Share ():
  print share.Name, share.Path
```

Show print jobs

```
import wmi
c = wmi.WMI ()

for printer in c.Win32_Printer ():
  print printer.Caption
  for job in c.Win32_PrintJob (DriverName=printer.DriverName):
    print "  ", job.Document
  print
```

NB This page at Microsoft is quite a good starting point for handling printer matters with WMI.

Connect to another machine as a named user

```
import wmi
connection = wmi.connect_server (server="other_machine", user="tim", password="secret")
c = wmi.WMI (wmi=connection)
```

NB You *cannot* connect to your own machine this way, no matter how hard you try to obfuscate the server name.