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

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

Note

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.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

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.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

Note

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.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.Win32_PrintJob.watch_for (
  notification_type="Creation",
  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

Note

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
# other_machine = "machine name of your choice"
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

Note

This example and the ones below use the convenience function `Registry()` which was added to the wmi package in its early days. It’s exactly equivalent to:

```
import wmi
r = wmi.WMI (namespace="DEFAULT").StdRegProv
```

```
import _winreg
import wmi

r = wmi.Registry ()
result, names = r.EnumKey (
  hDefKey=_winreg.HKEY_LOCAL_MACHINE,
  sSubKeyName="Software"
)
```

```
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

```
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

Note

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

```
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
```

Show disk partitions

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

for physical_disk in c.Win32_DiskDrive ():
  for partition in physical_disk.associators
("Win32_DiskDriveToDiskPartition"):
    for logical_disk in partition.associators
("Win32_LogicalDiskToPartition"):
      print physical_disk.Caption, partition.Caption, logical_disk.Caption
```

Install a product

Note

Example is after a post by Roger Upole to the python-win32 mailing list

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

c.Win32_Product.Install (
  PackageLocation="c:/temp/python-2.4.2.msi",
  AllUsers=False
)
```

Connect to another machine as a named user

Note

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

```
import wmi

#
# Using wmi module before 1.0rc3
#
connection = wmi.connect_server (
  server="other_machine",
  user="tim",
  password="secret"
)
c = wmi.WMI (wmi=connection)

#
# Using wmi module at least 1.0rc3
#
c = wmi.WMI (
  computer="other_machine",
  user="tim",
  password="secret"
)
```

Show a method’s signature

```
import wmi
c = wmi.WMI ()
for opsys in c.Win32_OperatingSystem ():
  break

print opsys.Reboot
print opsys.Shutdown
```

Schedule a job

Note

The WMI ScheduledJob class correponds to the AT Windows service (controlled through the “at” command). As far as I know, it is not related to the Scheduled Tasks mechanism, controlled by a control panel applet.

```
import os
import wmi

c = wmi.WMI ()
one_minutes_time = datetime.datetime.now () + datetime.timedelta (minutes=1)
job_id, result = c.Win32_ScheduledJob.Create (
  Command=r"cmd.exe /c dir /b c:\ > c:\\temp.txt",
  StartTime=wmi.from_time (one_minutes_time)
)
print job_id

for line in os.popen ("at"):
  print line
```

Run a process minimised

Note

Thanks to Keith Veleba for providing the question and code which prompted this example

```
import wmi

SW_SHOWMINIMIZED = 1

c = wmi.WMI ()
startup = c.Win32_ProcessStartup.new (ShowWindow=SW_SHOWMINIMIZED)
pid, result = c.Win32_Process.Create (
  CommandLine="notepad.exe",
  ProcessStartupInformation=startup
)
print pid
```

Find Drive Types

```python
import wmi

DRIVE_TYPES = {
  0 : "Unknown",
  1 : "No Root Directory",
  2 : "Removable Disk",
  3 : "Local Disk",
  4 : "Network Drive",
  5 : "Compact Disc",
  6 : "RAM Disk"
}

c = wmi.WMI ()
for drive in c.Win32_LogicalDisk ():
  print drive.Caption, DRIVE_TYPES[drive.DriveType]
```

List Namespaces

```python
import wmi

def enumerate_namespaces (namespace=u"root", level=0):
  print level * "  ", namespace.split ("/")[-1]
  c = wmi.WMI (namespace=namespace)
  for subnamespace in c.__NAMESPACE ():
    enumerate_namespaces (namespace + "/" + subnamespace.Name, level + 1)

enumerate_namespaces ()
```

Use WMI in a thread

Note

Note the use of pythoncom.Co(Un)initialize. WMI is a COM-based technology, so to use it in a thread, you must init the COM threading model. This applies also if you’re running in a service, for example, which is implicitly threaded.

```
import pythoncom
import wmi
import threading
import time

class Info (threading.Thread):
  def __init__ (self):
    threading.Thread.__init__ (self)
  def run (self):
    print 'In Another Thread...'
    pythoncom.CoInitialize ()
    try:
      c = wmi.WMI ()
      for i in range (5):
        for process in c.Win32_Process ():
          print process.ProcessId, process.Name
        time.sleep (2)
    finally:
      pythoncom.CoUninitialize ()

if __name__ == '__main__':
  print 'In Main Thread'
  c = wmi.WMI ()
  for process in c.Win32_Process ():
    print process.ProcessId, process.Name
  Info ().start ()
```

Monitor multiple machines for power events

This is a demonstration of extrinsic events, threading and remote monitoring... all in one small package! The idea is that the power subsystem generates extrinsic events via its WMI provider whenever a machine enters or leaves suspend mode. Extrinsic events are useful because WMI doesn't have to poll for them so you shouldn't miss any. The multiple machines was just a practical example of using threads.

Note

Note the use of CoInitialize and CoUninitialize in the thread control code. Note also the simplified use of `_wmi_class.watch_for()` which will work for intrinsic and extrinsic events transparently.

```
import pythoncom
import wmi
import threading
import Queue

class Server (threading.Thread):

  def __init__ (self, results, server, user, password):
    threading.Thread.__init__ (self)
    self.results = results
    self.server = server
    self.user = user
    self.password = password
    self.setDaemon (True)

  def run (self):
    pythoncom.CoInitialize ()
    try:
      #
      # If you don't want to use explicit logons, remove
      # the user= and password= params here and ensure
      # that the user running *this* script has sufficient
      # privs on the remote machines.
      #
      c = wmi.WMI (self.server, user=self.user, password=self.password)
      power_watcher = c.Win32_PowerManagementEvent.watch_for ()
      while True:
        self.results.put ((self.server, power_watcher ()))
    finally:
      pythoncom.CoUninitialize ()

#
# Obviously, change these to match the machines
# in your network which probably won't be named
# after Harry Potter characters. And which hopefully
# use a less obvious admin password.
#
servers = [
  ("goyle", "administrator", "secret"),
  ("malfoy", "administrator", "secret")
]
if __name__ == '__main__':
  power_events = Queue.Queue ()
  for server, user, password in servers:
    print "Watching for", server
    Server (power_events, server, user, password).start ()

  while True:
    server, power_event = power_events.get ()
    print server, "=>", power_event.EventType
```

Find the current wallpaper

```
import wmi
import win32api
import win32con

c = wmi.WMI ()
full_username = win32api.GetUserNameEx (win32con.NameSamCompatible)
for desktop in c.Win32_Desktop (Name=full_username):
  print \
    desktop.Wallpaper or "[No Wallpaper]", \
    desktop.WallpaperStretched, desktop.WallpaperTiled
```