



## Chapter 9

# Other Computer Management Tasks

**In this chapter:**

|  |     |
|--|-----|
| Change Local Account Passwords . . . . . | 125 |
| Clear Print Queues . . . . .             | 129 |
| Print Test Pages . . . . .               | 133 |
| List Running Processes . . . . .         | 137 |
| Start a Process . . . . .                | 140 |
| End a Process . . . . .                  | 144 |
| List Page Files . . . . .                | 148 |
| Modify Page File Settings . . . . .      | 152 |
| Shut Down or Restart Computers . . . . . | 156 |
| Modify Boot.ini Files . . . . .          | 160 |

Administrators are often required to make minor changes to client computers in their environments—minor, of course, until they must be made on a large number of computers. Some of these changes include adjustments to security practices such as modifying local account passwords. Other changes result from troubleshooting efforts and maintenance, such as managing client and server page files.

Tasks automated in this chapter include:

- Changing local account passwords
- Clearing print queues
- Printing test pages on print queues
- Listing running processes
- Starting processes
- Ending processes
- Inventorying page file settings
- Modifying page file settings
- Shutting down and restarting computers
- Modifying Boot.ini files

# Change Local Account Passwords



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9\ChangeLocalPassword\ChangeLocalPassword.wsf`.

| Operating System                      | Supported? | Prerequisites   |
|---------------------------------------|------------|---|
| Microsoft® Windows® 2000 family       | Yes        | <ul style="list-style-type: none"> <li>■ Microsoft Windows Script Host (WSH) 5.6 or later</li> </ul>  |
| Microsoft Windows XP Professional     | Yes        | <ul style="list-style-type: none"> <li>■ Windows Management Instrumentation (WMI)</li> </ul>  |
| Microsoft Windows Server™ 2003 family | Yes        | <ul style="list-style-type: none"> <li>■ Administrative permission on targeted computers</li> <li>■ Network connectivity to each remote computer</li> </ul> |

## Description

One of the best yet most overlooked security practices in a Windows® environment is regularly changing local account passwords, especially for sensitive built-in accounts like Administrator. Domain-based password policies can be used to force a change on a periodic basis, but because local accounts like Administrator are rarely used on many client computers, the password change is never triggered. (Password policies can require a change only when the account is actually used to log on to the computer.) As a result, these local passwords often go unchanged for long periods of time, increasing the likelihood that they will be compromised.

## Performing This Task Manually

Windows provides several means for changing passwords on a single computer. You can use the Computer Management console, for example, to change the password on any local user account. This console can also connect to remote computers and manage their local accounts. From the command line, you can use the `NET USER` command to modify a password. For example, to change the local Administrator account password, you would use this code:

```
NET USER Administrator NewPassword
```

However, the Windows operating system does not provide a built-in means of changing local accounts on multiple computers at once. (In fact, the primary purpose of a domain is to avoid the need to constantly manage local user accounts on multiple computers.)

## Example

The `ChangeLocalPassword.wsf` script allows you to target one computer or multiple computers. If you want to use it to target a single remote computer named `ServerA` and change the password for the `Administrator` account, you would use this code:

```
ChangeLocalPassword.wsf /computer:ServerA /user:Administrator /password:NewPassword
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named `C:\Computers.txt`, you would use this syntax:

```
ChangeLocalPassword.wsf /list:C:\Computers.txt /user:Administrator  
/password:NewPassword
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named `West`, you would use the following syntax:

```
ChangeLocalPassword.wsf /container:west /user:Administrator /password:NewPassword
```

Note that the `/container` argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain that the computer running the script belongs to. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
ChangeLocalPassword.wsf /container:west /recurse /user:Administrator  
/password:NewPassword
```

This is an excellent way to ensure that a consistent local user account password is used across multiple client or server computers. Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

This script can be executed as command-line utilities. Set `CScript.exe` to be your default script processor, as described in Chapter 3, “Working with VBScript.”

|                              |  |
|------------------------------|--|
| <code>/list:path</code>      | One and only one of these is required by the script. Use <code>/list</code> to target a list of computers contained within a text file. Use <code>/computer</code> to target a single computer. Use <code>/container</code> to target an organizational unit within Microsoft Active Directory®. |
| <code>/computer:name</code>  | Note that all computer names must be actual computer names; aliases such as <i>localhost</i> , as well as IP addresses, will not work.   |
| <code>/container:name</code> |  |
| <code>/recurse</code>        | When used with <code>/container</code> , also targets computers contained within nested OUs.   |

|                                 |   |
|---------------------------------|---|
| <code>/ping</code>              | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |
| <code>/log:path</code>          | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument. |
| <code>/verbose</code>           | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/user:username</code>     | Required. Specifies the local user account you want to change.  |
| <code>/password:password</code> | Required. Specifies the new password for the user account.  |

You can run these scripts with the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Active Directory Services Interface (ADSI) to accomplish its work. A common misconception about ADSI is that it works only with Active Directory, but ADSI includes a *WinNT* provider that is designed to work with Microsoft Windows NT® domains as well as local user accounts on standalone and member computers. The main portion of the script code follows:

```
Dim oUser
Set oUser = QueryADSI(sName,"winNT:// " & sName & "/" &
    WScript.Arguments.Named("user") & ",user")
If Not IsObject(oUser) Then
    WScript.Echo " *** Couldn't retrieve user from " & sName
Else
    On Error Resume Next
    oUser.setpassword WScript.Arguments.Named("password")
    If Err <> 0 Then
        WScript.Echo " *** Couldn't change password on " & sname
        WScript.Echo " " & Err.Description
    Else
        verbose " Changed password on " & sName
    End If
End If
```

The script uses this *WinNT* provider to connect to each targeted computer and retrieve the specified user account. (The inclusion of `",user"` ensures that computer, group, or other objects with similar names are not retrieved by accident.) The returned *User* object supplies a *SetPassword* method that is used to change the password.

## Troubleshooting

This script has the potential to create three major types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's users. (Typically, only a member of the Administrators local group has permission to change user passwords.) In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the */ping* argument.

The third potential problem is that one or more of the targeted computers won't have the user account specified. Although this situation wouldn't normally occur for built-in accounts like Administrator (unless you've inconsistently renamed that account on various computers), it could occur for other local accounts that you might have created. In these instances, the script will return an error and continue trying with the next targeted computer.

## To Learn More

- Learn more about the WinNT ADSI provider at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/adsi/adsi/adsi\\_winnt\\_provider.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/adsi/adsi/adsi_winnt_provider.asp).
- Find more local user account management samples in Microsoft Visual Basic® (VBScript) at <http://www.microsoft.com/technet/scriptcenter/scripts/ds/local/users/default.msp>.

## Clear Print Queues



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9\ClearPrintQueue\ClearPrintQueue.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | No         | <ul style="list-style-type: none"> <li>■ WSH 5.6 or later</li> </ul>  |
| Windows XP Professional    | Yes        | <ul style="list-style-type: none"> <li>■ WMI</li> </ul>   |
| Windows Server 2003 family | Yes        | <ul style="list-style-type: none"> <li>■ Administrative permission on targeted computers</li> <li>■ Network connectivity to each remote computer</li> </ul> |

### Description

This script, `ClearPrintQueue.wsf`, will clear documents from one or more print queues on one or more computers. This is typically a troubleshooting or maintenance step performed by an administrator.

### Performing This Task Manually

Windows provides a graphical user interface for managing print queue jobs. Typically, an administrator connects to the printer using the Printers And Faxes folder, and then double-clicks any connected printer to view its queue. Selecting one or more jobs allows these jobs to be paused or cancelled. The `NET PRINT` command-line utility can also be used to delete individual jobs from remote print queues. However, no built-in means exists for clearing multiple queues at once.

### Example

This tool allows you to target one computer or multiple computers. You can clear all print queues on each targeted computer, or you can choose to clear only one named queue. In the latter case, the queue specified must exist with the same name on each targeted computer. To use the script to target a single remote computer named `ServerA` and clear all queues, you would use this code:

```
ClearPrintQueue.wsf /computer:ServerA /all
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named `C:\Computers.txt`, you would use this syntax:

```
ClearPrintQueue.wsf /list:C:\Computers.txt /all
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named West, you would use the following syntax:

```
ClearPrintQueue.wsf /container:west /all
```

Note that the */container* argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain as that of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
ClearPrintQueue.wsf /container:west /queue:MyQueue
```

Note that this example clears a queue named MyQueue (which is the name of the installed printer), which would exist on each targeted computer. Additional arguments provide more functionality to the command; see following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set CScript.exe to be your default script processor, as described in Chapter 3.

|                         |   |
|-------------------------|---|
| <i>/list:path</i>       | One and only one of these is required by the script. Use <i>/list</i> to target a list of computers contained within a text file. Use <i>/computer</i> to target a single computer. Use <i>/container</i> to target an organizational unit within Active Directory. |
| <i>/computer:name</i>   |   |
| <i>/container:name</i>  |   |
| <i>/recurse</i>         | When used with <i>/container</i> , also targets computers contained within nested OUs.  |
| <i>/ping</i>            | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |
| <i>/log:path</i>        | Logs unreachable computer names to the specified file. This file can then be used later, along with the <i>/list</i> argument, to retry these computers. Note that a log is created only when used in conjunction with the <i>/ping</i> argument.                   |
| <i>/verbose</i>         | Causes the script to display more detailed, step-by-step status messages.   |
| <i>/all</i>             | One of these is required. Use <i>/all</i> to clear all queues on each targeted computer, or specify a specific queue by using <i>/queue:queuename</i> .   |
| <i>/queue:queuename</i> |   |

You can run these scripts with the */?* parameter to display the command’s syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The *Win32\_Printer* class is queried (either all instances, representing all queues, or a specific named instance). The following script code performs the work of connecting to the remote queues and clearing them:

```
Dim cQueues, oQueue
If WScript.Arguments.Named.Exists("all") Then
    Set cQueues = QueryWMI(sName, "\root\cimv2", "Select * From
    Win32_Printer", "", "")
Else
    Set cQueues = QueryWMI(sName, "\root\cimv2", "Select * From Win32_Printer Where
    Name = '" & WScript.Arguments.Named("queue") & "'", "", "")
End If
If Not IsObject(cQueues) Then
    WScript.Echo " *** Couldn't retrieve queue(s) from " & sName
Else
    On Error Resume Next
    For Each oQueue In cQueues
        Verbose "Cancelling jobs on " & sName & ": " & oQueue.Name
        oQueue.CancelAllJobs()
        If Err <> 0 Then
            WScript.Echo " *** Error cancelling " & oQueue.Name & " on " & sName & ":"
            WScript.Echo " " & Err.Description
        End If
    Next
End If
```

Each instance of the *Win32\_Printer* class provides a *CancelAllJobs()* method, which performs the work of canceling all outstanding jobs in that print queue.

## Troubleshooting

This script has the potential for several types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's print queues. (Typically, only a member of the Administrators and Power Users or Print Operators groups has permission to cancel all print jobs; ordinary users can typically cancel only their own jobs.) In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the */ping* argument.

The third potential problem is that one or more of the targeted computers won't have the queue specified (if you use the */queue* argument). In these instances, the script will return an error and continue trying with the next targeted computer. It's also possible for the script to encounter an error while trying to cancel one or more jobs—jobs

can become stuck, or the script might be unable to obtain permissions for specified jobs. In these cases, the script will again return an error message and continue trying with the next print queue. However, the script is not designed to retry queues whose jobs have failed to cancel.

## To Learn More

- Find more VBScript samples for managing printers at <http://www.microsoft.com/technet/scriptcenter/scripts/printing/default.aspx>.
- Learn more about the `Win32_Printer` class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_printer.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_printer.asp).

## Print Test Pages



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9\PrintTestPage\PrintTestPage.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | No         | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

### Description

This tool will print test pages, on one or more computers, for all queues or for a named queue. This is a great way to periodically test your queues for proper operation, or a quick way for an administrator or support technician to test a remote print server's queues during troubleshooting.

I've also used this script to quickly identify which physical print devices are connected to a given print server—helpful because some organizations lose track of this information. This script can target all queues on a given print server; print devices producing a test page reveal which devices are connected to a particular server.

### Performing This Task Manually

Windows provides a built-in means for printing test pages on single print queues. To print these pages manually, use this code:

1. Right-click a printer in the Printers And Faxes folder, and select Properties.
2. In the printer's Properties dialog box, click Print Test Page.

A one-page (typically) test is produced that confirms that the printer is working properly and that lists the driver files in use by the printer.

However, Windows provides no built-in means of performing this test on multiple print queues at once.

## Example

This tool allows you to target one computer or multiple computers. You can clear all print queues on each targeted computer, or you can choose to clear only one named queue. In the latter case, the queue specified must exist with the same name on each targeted computer. If you wanted to use the script to target a single remote computer named ServerA and test all queues, you would use this code:

```
PrintTestPage.wsf /computer:ServerA /all
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named C:\Computers.txt, you would use this syntax:

```
PrintTestPage.wsf /list:C:\Computers.txt /all
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named West, you would use the following syntax:

```
PrintTestPage.wsf /container:west /all
```

Note that the */container* argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
PrintTestPage.wsf /container:west /queue:MyQueue
```

Note that this example tests a queue named MyQueue (which is the name of the installed printer), which must exist on each targeted computer. Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set CScript.exe to be your default script processor, as described in Chapter 3.

|                        |   |
|------------------------|---|
| <i>/list:path</i>      | One and only one of these is required by the script. Use <i>/list</i> to target a list of computers contained within a text file. Use <i>/computer</i> to target a single computer. Use <i>/container</i> to target an organizational unit within Active Directory. |
| <i>/computer:name</i>  |   |
| <i>/container:name</i> |   |
| <i>/recurse</i>        | When used with <i>/container</i> , also targets computers contained within nested OUs.  |

|  |   |
|--|---|
| <code>/ping</code>                                 | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |
| <code>/log:path</code>                             | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument. |
| <code>/verbose</code>                              | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/all</code><br><code>/queue:queuename</code> | One of these is required. Use <code>/all</code> to test all queues on each targeted computer, or specify a specific queue by using <code>/queue:queuename</code> .  |

You can run these scripts with the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The `Win32_Printer` class is queried (either all instances representing all queues or a specific named instance). The following script code performs the work of connecting to the remote queues and printing the test page:

```
Dim cQueues, oQueue
If WScript.Arguments.Named.Exists("all") Then
    Set cQueues = QueryWMI(sName, "\root\cimv2", "Select * From Win32_Printer", "", "")
Else
    Set cQueues = QueryWMI(sName, "\root\cimv2", "Select * From Win32_Printer Where
Name = '" & WScript.Arguments.Named("queue") & "'", "", "")
End If
If Not IsObject(cQueues) Then
    WScript.Echo " *** Couldn't retrieve queue(s) from " & sName
Else
    On Error Resume Next
    For Each oQueue In cQueues
        Verbose "Printing test on " & sName & ": " & oQueue.Name
        oQueue.PrintTestPage()
        If Err <> 0 Then
            WScript.Echo " *** Error printing test to " & oQueue.name & " on "
            & sName & ":"
            WScript.Echo " " & Err.Description
        End If
    Next
End If
```

Each instance of the `Win32_Printer` class provides a `PrintTestPage()` method that performs the work of printing a test page to that queue.

## Troubleshooting

This script has the potential to experience several types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's print queues. A permissions issue is rare, however, because you're merely printing a document, which all users generally have permissions to do. Any permissions issue you encounter will more likely be a result of connecting to WMI itself, a task that often only Administrators have permission to do. In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the `/ping` argument.

The third potential problem is that one or more of the targeted computers won't have the queue specified (if you use the `/queue` argument). In these instances, the script will return an error and continue trying with the next targeted computer. It's also possible for the script to encounter an error while trying to print the test page. This is also rare—Windows typically sends the test page job to the printer much like it would for any other print job. Problems will show up in the printer's queue monitoring window, but problems won't usually result in an error being returned to the script.

## To Learn More

- Find more VBScript samples for managing printers at <http://www.microsoft.com/technet/scriptcenter/scripts/printing/default.mspx>.
- Learn more about the `Win32_Printer` class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_printer.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_printer.asp).

## List Running Processes



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9>ListProcesses>ListProcesses.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | Yes        | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

### Description

This tool will list all processes running on one or more computers. Optionally, the script's output can be directed to a comma-separated values (CSV) file, providing you with a report of running processes. This file is a useful inventory tool because it allows you to more easily audit the processes running on your computers and look for unauthorized applications, outdated software, and so forth.

Some viruses, spyware, and malware install themselves as innocent-sounding services; I've used this script in the past to quickly inventory multiple computers to see whether any of them were running a process known to be associated with malware or a particular attack.

### Performing This Task Manually

Windows Task Manager provides a list of all processes running on the local computer; command-line tools such as `PLIST`, `RLIST`, and `TLIST` (which have been available in various versions of the Microsoft Windows Resource Kits) also have the capability of listing the processes on a single remote computer. However, there is no built-in means for listing the processes on multiple remote computers.

### Example

This tool allows you to target one computer or multiple computers. To use the script to target a single remote computer named `ServerA` and list the running processes to the console, you would use the following code:

```
ListProcesses.wsf /computer:ServerA
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named `C:\Computers.txt`, you would use this syntax:

```
ListProcesses.wsf /list:c:\Computers.txt /output:c:\procllist.csv
```

Note that this syntax outputs the process list to a CSV file named `C:\Procllist.csv`. This file is especially useful when targeting multiple computers, because the amount of information generated can be awkward to review in a command-line window.

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named `West`, you would use the following syntax:

```
ListProcesses.wsf /container:west /output:c:\procllist.csv
```

Note that the `/container` argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
ListProcesses.wsf /container:west /output:c:\procllist.csv
```

Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set `CScript.exe` to be your default script processor, as described in Chapter 3.

|                              |   |
|------------------------------|---|
| <code>/list:path</code>      | One and only one of these is required by the script. Use <code>/list</code> to target a list of computers contained within a text file. Use <code>/computer</code> to target a single computer. Use <code>/container</code> to target an organizational unit within Active Directory. |
| <code>/computer:name</code>  |   |
| <code>/container:name</code> |   |
| <code>/recurse</code>        | When used with <code>/container</code> , also targets computers contained within nested OUs.  |
| <code>/ping</code>           | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |
| <code>/log:path</code>       | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument.                         |
| <code>/verbose</code>        | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/output:path</code>    | Optionally logs the script's output to a CSV file.  |

You can run these scripts with the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The *Win32\_Process* class is queried (returning all instances, with each instance representing one running process). The following script code performs the work:

```
Dim cProcesses, oProcess, sOutput
Verbose " Connecting to WMI on " & sName
Set cProcesses = QueryWMI(sName,"root\cimv2","select * From Win32_Process","","")
If Not IsObject(cProcesses) Then
    WScript.Echo " *** Couldn't connect to WMI on " & sName
Else
    For Each oProcess In cProcesses
        sOutput = sName & ", " & oProcess.Name & ", " & oProcess.CommandLine & ", "
    & oProcess.ProcessID
        If WScript.Arguments.Named.Exists("output") Then
            LogFile WScript.Arguments.Named("output"),sOutput,False
        Else
            WScript.Echo sOutput
        End If
    Next
End If
```

The variable *sOutput* is used to contain information about each process, including its name, process ID, and the command-line used to execute the process. The *sOutput* variable is then printed in the command-line window or, if you use the */output* argument, to the specified CSV file.

## Troubleshooting

This script has the potential to result in two types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's processes. (Typically, only a member of the Administrators group has permission to do so.) In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the */ping* argument.

## To Learn More

- Find more VBScript samples for managing and monitoring processes at <http://www.microsoft.com/technet/scriptcenter/scripts/os/process/default.mspx>.
- Learn more about the *Win32\_Process* class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_process.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_process.asp).

## Start a Process



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9\StartProcess\StartProcess.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | Yes        | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

### Description

This tool will start a new process, using the specified command line, on one or more computers. The results of this tool will differ somewhat depending on the precise circumstances in which you run the script. For example, when a user starts a new process (such as an application) using Microsoft Windows Explorer, the process runs under the user's normal security context and is visible on the user's desktop. Processes started by other users, however, are generally prevented from interacting with the logged-on user's desktop. Because this script uses *your* credentials to connect to remote computers, the processes you start will usually be running under your security context on the remote computer and might not be visible to the user logged on to that computer. So you might not be able to use this script to, for example, open a new instance of Microsoft Notepad, which will be visible to the logged-on user. Consider this tool to be more appropriate for starting new background maintenance tasks (such as disk defragmenters), which require no user interaction.

The lack of user interaction can also pose problems. If you start a process that requires some user interaction, but the process is not visible to the logged-on user, the process might “hang” while it waits for a response to a dialog box or some other interactive element. Try to avoid remotely starting processes that might require such interaction.

### Performing This Task Manually

Windows includes a command-line tool named Rexec, which can start a process on a remote computer. The following, for example, will run a process named `MyProcess.exe` on a computer named `Client1`:

```
Rexec Client1 MyProcess.exe
```

However, Rexec is unable to target multiple computers simultaneously.

## Example

This tool allows you to target one computer or multiple computers. If you wanted to use the script to target a single remote computer named ServerA and start a process named MyProcess.exe, you would use the following code:

```
StartProcess.wsf /computer:ServerA /cmd:MyProcess.exe
```

Note that the */cmd* argument requires a complete path to the process you want started unless the process specified is in the system search path (which generally includes the Windows and Windows\System32 directories).

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named C:\Computers.txt, you would use this syntax:

```
StartProcess.wsf /list:C:\Computers.txt /cmd:MyProcess.exe
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named West, you would use the following syntax:

```
StartProcess.wsf /container:west /cmd:MyProcess.exe
```

Note that the */container* argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
StartProcess.wsf /container:west /cmd:MyProcess.exe
```

Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set CScript.exe to be your default script processor, as described in Chapter 3.

|                        |   |
|------------------------|---|
| <i>/list:path</i>      | One and only one of these is required by the script. Use <i>/list</i> to target a list of computers contained within a text file. Use |
| <i>/computer:name</i>  | <i>/computer</i> to target a single computer. Use <i>/container</i> to target   |
| <i>/container:name</i> | an organizational unit within Active Directory.   |
| <i>/recurse</i>        | When used with <i>/container</i> , also targets computers contained within nested OUs.  |

|                        |   |
|------------------------|---|
| <code>/ping</code>     | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |
| <code>/log:path</code> | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument. |
| <code>/verbose</code>  | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/cmd:path</code> | Required. Complete path and file name of the executable to start. This path can be a UNC, and each targeted computer will load the executable from that UNC and execute it locally.   |

You can run these scripts with the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The entire `Win32_Process` class is retrieved rather than a specific instance of that class; the class itself provides a `Create()` method that creates a new instance of the class with the specified parameters. One return parameter is the new process's process ID, which the script will output to the command-line window if you specify the `/verbose` argument.

```
Dim oProcesses, ErrResult, iID
Verbose " Connecting to WMI on " & sName
On Error Resume Next
Set oProcess = GetObject("winmgmts:\\\" & sName & "\root\cimv2:win32_Process")
If Err <> 0 Then
    WScript.Echo " *** Couldn't connect to WMI on " & sName
    LogBadConnect(sName)
Else
    ErrResult = oProcess.Create(WScript.Arguments.Named("cmd"),Null,Null,iID)
    If ErrResult = 0 Then
        Verbose " Created ID " & iID & " on " & sName
    Else
        WScript.Echo " *** Couldn't start on " & sName & ": " & ErrResult
    End If
End if
```

## Troubleshooting

This script has the potential to result in several types of errors. First, a remote computer won't be available. Second, you won't have permission to start remote processes. Generally, local Administrator permissions are required to start remote

processes via WMI. In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the */ping* argument.

The third potential problem is that the targeted computers will be unable to access the executable you specify. This will result in the script displaying an error, which will likely occur for all targeted computers. In most environments, WMI will attempt to access the specified executable using *your* credentials, so if you must, specify a UNC or local file path to which your user account has at least Read and Execute privileges.

## To Learn More

- Find more VBScript samples for managing and monitoring processes at <http://www.microsoft.com/technet/scriptcenter/scripts/os/process/default.aspx>.
- Learn more about the *Win32\_Process* class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_process.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_process.asp).

## End a Process



**On the CD** The sample script and be found on the CD that accompanies this book at `\Chap9\EndProcess\EndProcess.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | Yes        | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

### Description

This tool will end a named process on one or more computers. This is an excellent way to stop undesired processes that might be running. Note that you must provide the name of each process, which is sometimes slightly different from the name of its command-line executable. The name sometimes includes a file name extension, such as `.exe`, but not always; I recommend using the *ListProcesses* script, described in this chapter, to list the processes on a representative computer. That script lists, among other details, each process's name.

### Performing This Task Manually

Windows Task Manager can be used to end individual tasks (processes) on the local computer. A KILL command-line utility has also been included with various versions of the Windows Resource Kits; this tool can be used to end processes either by name (in some versions) or by process ID number (often requiring a second tool like PLIST or TLIST to discover the process number). Windows doesn't provide a built-in means for terminating processes on multiple computers at once.

### Example

This tool allows you to target one computer or multiple computers. If you wanted to use the script to target a single remote computer named `ServerA` and terminates a process named `MyProcess.exe`, you would use this code:

```
EndProcess.wsf /computer:ServerA /name:MyProcess.exe
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named `C:\Computers.txt`, you would use this syntax:

```
EndProcess.wsf /list:C:\Computers.txt /name:MyProcess.exe
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named `West`, you would use the following syntax:

```
EndProcess.wsf /container:west /name:MyProcess.exe
```

Note that the `/container` argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain as that of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
EndProcess.wsf /container:west /name:MyProcess.exe
```

Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set `CScript.exe` to be your default script processor, as described in Chapter 3.

|                                |   |
|--------------------------------|---|
| <code>/list:path</code>        | One and only one of these is required by the script. Use <code>/list</code> to target a list of computers contained within a text file. Use <code>/computer</code> to target a single computer. Use <code>/container</code> to target an organizational unit within Active Directory. |
| <code>/computer:name</code>    |   |
| <code>/container:name</code>   |   |
| <code>/recurse</code>          | When used with <code>/container</code> , also targets computers contained within nested OUs.  |
| <code>/ping</code>             | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |
| <code>/log:path</code>         | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument.                         |
| <code>/verbose</code>          | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/name:processname</code> | Required. Identifies the name of the process to terminate. This must be the actual process name, which might include a file name extension for some processes.  |

You can run these scripts with the `/?` parameter to display the command’s syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The *Win32\_Process* class is queried for the specified instance. The following script code performs the work of connecting to the remote computer and terminating the selected process:

```
Dim cProcesses, oProcess
Set cProcesses = QueryWMI(sName,"root\cimv2","Select * from Win32_Process Where Name
= '" & wscript.arguments.named("name") & "',"", "", "")
If Not IsObject(cProcesses) Then
    WScript.Echo " *** Could not retrieve process (or process does not exist) on "
& sName
Else
    For Each oProcess In cProcesses
        On Error Resume Next
        oProcess.Terminate()
        If Err <> 0 Then
            WScript.Echo " *** Error terminating on " & sName
            WScript.Echo " " & Err.Description
        Else
            Verbose " Terminated on " & sName
        End If
    Next
End If
```

Each instance of the *Win32\_Process* class provides a *Terminate()* method that attempts to end the process.

## Troubleshooting

This script has the potential to result in several types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's processes (typically, only a member of the Administrators group will have permission to terminate a process). In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the */ping* argument.

The third potential problem is that one or more of the targeted computers won't have the process. In these instances, the script will return an error and continue trying with the next targeted computer. It's also possible for the script to encounter an error while trying to terminate the process; some processes can be configured, for security

reasons, to resist termination by administrators, whereas other processes might encounter an error while terminating and fail to terminate properly. In these cases, the script will display an error code and continue trying the next targeted computer.

Note that terminating a process is not the same as ending it cleanly. Terminating a process will often result in the loss of whatever data the process was working with; you should terminate processes only when you're certain that doing so will not destabilize Windows, other processes or services, or result in a loss of important data.

## To Learn More

- Find more VBScript samples for managing and monitoring processes at <http://www.microsoft.com/technet/scriptcenter/scripts/os/process/default.aspx>.
- Learn more about the `Win32_Process` class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_process.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_process.asp).

## List Page Files



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9>ListPageFile>ListPageFile.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | Yes        | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

### Description

This tool will list the page file properties for all page files on one or more computers. For each page file, you are shown its host drive, initial size, and maximum size. Optionally, the tool can save this information to a CSV file for long-term archiving or reporting. This is a useful way to quickly check a number of computers to ensure that their page files are optimally configured or configured to meet your organization's policies.

### Performing This Task Manually

In the Windows operating system, your primary means of managing the page file is by using the Properties dialog box of My Computer (specifically, the Advanced tab). Follow these steps to view page file settings:

1. Right-click My Computer.
2. Select Properties to open the System Properties dialog box.
3. Select the Advanced tab.
4. Under the Performance section, click Settings to display the Performance Options dialog box.
5. Select the Advanced tab.
6. Under the Virtual Memory section, click Change to display the Virtual Memory dialog box.

Using this procedure to review the page file settings of multiple computers, however, can be tedious.

## Example

This tool allows you to target one computer or multiple computers. If you wanted to use the script to target a single remote computer named ServerA and display its page file settings in the command-line window, you would use this code:

```
ListPageFile.wsf /computer:ServerA
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named C:\Computers.txt, you would use this syntax:

```
ListPageFile.wsf /list:C:\Computers.txt /output:C:\MyFile.csv
```

Note that this saves the output to a CSV file, which can be easier to review than the command-line window when targeting multiple computers. Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named West, you would use the following syntax:

```
ListPageFile.wsf /container:west /output:C:\MyFile.csv
```

Note that the */container* argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain as that of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
ListPageFile.wsf /container:west /output:C:\MyFile.csv
```

Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set CScript.exe to be your default script processor, as described in Chapter 3.

|                              |   |
|------------------------------|---|
| <code>/list:path</code>      | One and only one of these is required by the script. Use <i>/list</i> to target a list of computers contained within a text file. Use <i>/computer</i> to target a single computer. Use <i>/container</i> to target an organizational unit within Active Directory. |
| <code>/computer:name</code>  |   |
| <code>/container:name</code> |   |
| <code>/recurse</code>        | When used with <i>/container</i> , also targets computers contained within nested OUs.  |
| <code>/ping</code>           | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |

|                           |   |
|---------------------------|---|
| <code>/log:path</code>    | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument. |
| <code>/verbose</code>     | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/output:path</code> | Saves the script's output to a CSV file specified in <i>path</i> .  |

You can run these scripts with the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. All instances of the `Win32_PageFile` class are queried, and specific properties of each instance are used in the script's output. The main work of the script is performed by this code:

```
Dim cFiles, oFile, sOutput
Verbose " Connecting to WMI on " & sName
Set cFiles = QueryWMI(sName,"root\cimv2","Select * From win32_PageFile","", "")
If Not IsObject(cFiles) Then
    WScript.Echo " *** Couldn't connect to WMI on " & sName
Else
    For Each oFile In cFiles
        sOutput = sName & "," & oFile.Drive & "," & oFile.InitialSize & ","
    & oFile.MaximumSize
        If WScript.Arguments.Named.Exists("output") Then
            LogFile WScript.Arguments.Named("output"),sOutput,False
        Else
            WScript.Echo sOutput
        End If
    Next
End If
```

## Troubleshooting

This script has the potential to result in two types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's page files. In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the `/ping` argument.

## To Learn More

- Find more VBScript samples for managing page files at <http://www.microsoft.com/technet/scriptcenter/scripts/os/pagefile/default.msp>.
- Learn more about the `Win32_PageFile` class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_pagefile.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_pagefile.asp).

## Modify Page File Settings



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9\ModifyPageFile\ModifyPageFile.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | Yes        | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

### Description

This tool will modify the initial and maximum sizes for all page files on one or more computers. This is a useful, quick way to reconfigure multiple computers to have consistent page file settings.

### Performing This Task Manually

In Windows, your primary means of managing the page file is by using the Properties dialog box of My Computer (specifically, the Advanced tab). Follow these steps to view page file settings:

1. Right-click My Computer.
2. Select Properties to open the System Properties dialog box.
3. Select the Advanced tab.
4. Under the Performance section, click Settings to display the Performance Options dialog box.
5. Select the Advanced tab.
6. Under the Virtual Memory section, click Change to display the Virtual Memory dialog box.

Using this procedure to modify the page file settings of multiple computers, however, can be tedious.

### Example

This tool allows you to target one computer or multiple computers. You must specify the `/initsize` and `/maxsize` parameters, which both accept a size in megabytes (MB). If you wanted to use the script to target a single remote computer named ServerA and set

its page file to have an initial size of 128 MB and a maximum size of 384 MB, you would use this code:

```
ModifyPageFile.wsf /computer:ServerA /initsize:128 /maxsize:384
```

Note that for computers with multiple page files (many administrators, for example, create page files on multiple physical drives to help spread the page file workload), each page file will be configured to have the initial and maximum size you specify.

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named C:\Computers.txt, you would use this syntax:

```
ModifyPageFile.wsf /list:C:\Computers.txt /initsize:128 /maxsize:384
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named West, you would use the following syntax:

```
ModifyPageFile.wsf /container:west /initsize:128 /maxsize:384
```

Note that the */container* argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain as that of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
ModifyPageFile.wsf /container:west /initsize:128 /maxsize:384
```

Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set CScript.exe to be your default script processor, as described in Chapter 3.

|                              |   |
|------------------------------|---|
| <code>/list:path</code>      | One and only one of these is required by the script. Use <i>/list</i> to target a list of computers contained within a text file. Use <i>/computer</i> to target a single computer. Use <i>/container</i> to target an organizational unit within Active Directory. |
| <code>/computer:name</code>  |   |
| <code>/container:name</code> |   |
| <code>/recurse</code>        | When used with <i>/container</i> , also targets computers contained within nested OUs.  |
| <code>/ping</code>           | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |

|                           |   |
|---------------------------|---|
| <code>/log:path</code>    | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument. |
| <code>/verbose</code>     | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/initsize:MB</code> | Required. Specifies the initial page file size in megabytes.  |
| <code>/maxsize:MB</code>  | Required. Specifies the maximum page file size in megabytes.  |

You can run these scripts with the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The following code performs the main work of this script:

```
Dim cFiles, oFile
Verbose " Connecting to WMI on " & sName
Set cFiles = QueryWMI(sName,"root\cimv2","select * From Win32_PageFileSetting","", "")
If Not IsObject(cFiles) Then
    WScript.Echo " *** Couldn't connect to WMI on " & sName
Else
    For Each oFile In cFiles
        Verbose "Changing on " & sName
        On Error Resume Next
        oFile.InitialSize = WScript.Arguments.Named("initsize")
        If Err <> 0 Then
            WScript.Echo " *** Error setting initsize on " & sName
            WScript.Echo " " & Err.Description
        Else
            Verbose " Set initsize on " sName
        End If
        oFile.MaximumSize = WScript.Arguments.Named("maxsize")
        If Err <> 0 Then
            WScript.Echo " *** Error setting maxsize on " & sName
            WScript.Echo " " & Err.Description
        Else
            Verbose " Set maxsize on " sName
        End If
        oFile.Put_
        If Err <> 0 Then
            WScript.Echo " *** Error setting sizes on " & sName
            WScript.Echo " " & Err.Description
        Else
            Verbose " Set sizes on " sName
        End If
    Next
End If
```

Notice that the *Win32\_PageFile* class is retrieved from each computer, and the class's *InitialSize* and *MaximumSize* properties are modified. These are some of the very few WMI properties that can be directly read and written by a script; most WMI properties are read-only, and you make changes by using methods of the class. In the example in the preceding code, you must use the class's *Put\_()* method to save the changes made to the properties.

## Troubleshooting

This script has the potential to result in several types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's page files. (Typically, only a member of the Administrators group has permission to do so.) In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the */ping* argument.

The third potential problem is that one or more of the targeted computers won't have the disk space needed to support the page file sizes you specify. Be very careful to not specify a page file that would exceed the available disk space; Chapter 10, "File, Disk, and Volume Management," includes tools to help you automate the process of inventorying available drive space on multiple computers.

## To Learn More

- Find more VBScript samples for managing page files at <http://www.microsoft.com/technet/scriptcenter/scripts/os/pagefile/default.msp>.
- Learn more about the *Win32\_PageFile* class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_pagefile.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_pagefile.asp).

# Shut Down or Restart Computers



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9\ShutdownRestart\ShutdownRestart.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | Yes        | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

## Description

This tool will shut down, restart, log off, or power off multiple computers. The power off functionality must be supported in the computers' hardware and BIOS; if it is not, the power off command will be interpreted as a simple shutdown command. This tool will, by default, issue *normal* commands for shut down, restart, and so forth, meaning individual applications could intercept and abort the command. This tool can optionally *force* the specified operation, which will give applications a fixed amount of time in which to shut down properly, and then it can terminate any applications that fail to do so.

## Performing This Task Manually

Windows provides well-known means for performing shutdown, restart, and logoff operations for an individual computer from the graphical user interface. The Windows Support Tools (and some versions of the Windows Resource Kits) also include a `Shutdown.exe` command-line utility, which can be used to shut down or restart a single remote computer. There are no built-in means to target multiple computers at once, however.

## Example

This tool allows you to target one computer or multiple computers. If you wanted to use the script to target a single remote computer named `ServerA` and shut it down, you would use this code:

```
ShutdownRestart.wsf /computer:ServerA /action:shutdown
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named `C:\Computers.txt`, you would use this syntax:

```
shutdownRestart.wsf /list:C:\Computers.txt /action:shutdown
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named `West`, you would use the following syntax:

```
shutdownRestart.wsf /container:west /action:shutdown
```

Note that the `/container` argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain as that of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
shutdownRestart.wsf /container:west /action:shutdown
```

Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set `CScript.exe` to be your default script processor, as described in Chapter 3.

|                              |   |
|------------------------------|---|
| <code>/list:path</code>      | One and only one of these is required by the script. Use <code>/list</code> to target a list of computers contained within a text file. Use <code>/computer</code> to target a single computer. Use <code>/container</code> to target an organizational unit within Active Directory. |
| <code>/computer:name</code>  |   |
| <code>/container:name</code> |   |
| <code>/recurse</code>        | When used with <code>/container</code> , also targets computers contained within nested OUs.  |
| <code>/ping</code>           | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |
| <code>/log:path</code>       | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument.                         |
| <code>/verbose</code>        | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/action:command</code> | Required. <i>Command</i> must be one of these: <i>restart</i> , <i>logoff</i> , <i>shutdown</i> , or <i>poweroff</i> .  |
| <code>/force</code>          | Forces the command specified in <code>/action</code> , preventing an application or process from aborting the specified command.  |

You can run these scripts with the `/?` parameter to display the command’s syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The *Win32\_OperatingSystem* class is queried because that class provides methods for performing a shutdown. The following script code performs the work of connecting to the remote computers and performing the specified command:

```
Dim cSystems, oSystem, iAction
Select Case lcase(WScript.Arguments.Named("action"))
    Case "shutdown"
        iAction = 1
    Case "restart"
        iAction = 2
    Case "logoff"
        iAction = 0
    Case "poweroff"
        iAction = 8
    Case Else
        WScript.Echo "*** Unknown action " & WScript.Arguments.Named("action")
End Select
If WScript.Arguments.Named.Exists("force") Then
    iAction = iAction + 4
End If
Verbose " Connecting to WMI on " & sName
Set cSystems = QueryWMI(sName,"root\cimv2","select * From Win32_OperatingSystem",
"", "")
If Not IsObject(cSystems) Then
    WScript.Echo " *** Couldn't connect to WMI on " & sName
Else
    For Each oSystem In cSystems
        On Error Resume Next
        oSystem.Win32Shutdown(iAction)
        If Err <> 0 Then
            WScript.Echo " *** Couldn't perform action on " & sName
            WScript.Echo " " & Err.Description
        Else
            Verbose "Successful on " & sName
        End If
    Next
End If
```

The *Win32\_OperatingSystem* class also has a *Shutdown()* method, but this script uses the more flexible *Win32Shutdown()* method, which supports additional commands such as power off as well as the ability to force the specified command.

## Troubleshooting

In this script, other than connectivity issues (which can be avoided by specifying the */ping* argument to test for connectivity), only a lack of permissions will result in an error. Typically, members of a computer's local Administrators group have permissions to initiate remote shutdown, restart, poweroff, or logoff.

## To Learn More

- Learn more about the *Win32\_OperatingSystem* class and its *Win32Shutdown()* method at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32shutdown\\_method\\_in\\_class\\_win32\\_operatingsystem.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32shutdown_method_in_class_win32_operatingsystem.asp).
- Read more about the Shutdown.exe command-line tool.
  - Windows 2000: <http://support.microsoft.com/default.aspx?scid=kb;en-us;317371>.
  - Windows Server 2003: <http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/shutdown.asp>.

## Modify Boot.ini Files



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap9\ModifyBootIni\ModifyBootIni.wsf`.

| Operating System           | Supported? | Prerequisites   |
|----------------------------|------------|---|
| Windows 2000 family        | Yes        | ■ WSH 5.6 or later  |
| Windows XP Professional    | Yes        | ■ WMI   |
| Windows Server 2003 family | Yes        | ■ Administrative permission on targeted computers<br>■ Network connectivity to each remote computer |

### Description

This tool will modify the Boot.ini file on one more computers so that it has a new startup delay time. This delay time specifies how long the Windows operating system selection screen is displayed before Windows starts the default operating system.

Many Windows computers will not display an operating system selection screen. When only one operating system and the Recovery Console is not installed, Boot.ini contains only one entry, and Windows always starts that entry without displaying the selection screen. However, it is a best practice (at least on servers) to install the Recovery Console, which will cause the selection screen to be displayed when the server starts. Consistently configuring the delay time on this screen will make your servers more predictable and easier to manage in the event the Recovery Console is needed.

### Performing This Task Manually

To manually modify the startup delay:

1. Right-click My Computer.
2. Select Properties to display the System Properties dialog box.
3. Select the Advanced tab.
4. Under the Startup And Recovery section, click Settings to display the Startup And Recovery dialog box.
5. Modify the Time to display a list of operating systems values as desired. (The default is 30 seconds.)

## Example

This tool allows you to target one computer or multiple computers. If you wanted to use the script simply to target a single remote computer named ServerA and set its startup delay to 45 seconds, you would use this code:

```
ModifyBootIni.wsf /computer:ServerA /delay:45
```

You can also target a list of computers from a text file. The text file is expected to contain one computer name per line and no other information. Assuming the file is named C:\Computers.txt, you would use this syntax:

```
ModifyBootIni.wsf /list:C:\Computers.txt /delay:45
```

Finally, you can target an entire organizational unit of computer accounts. If your domain contains an OU named West, you would use the following syntax:

```
ModifyBootIni.wsf /container:west /delay:45
```

Note that the */container* argument will work against only the default domain of the computer running the script. In other words, the OU specified must exist within the same domain as that of the computer running the script. If the specified OU has nested OUs, you can include their computer accounts as well by specifying one additional argument:

```
ModifyBootIni.wsf /container:west /delay:45
```

Additional arguments provide more functionality to the command; see the following section titled “Syntax.”

## Syntax

These scripts can be executed as command-line utilities. Set CScript.exe to be your default script processor, as described in Chapter 3.

|                              |   |
|------------------------------|---|
| <code>/list:path</code>      | One and only one of these is required by the script. Use <i>/list</i> to target a list of computers contained within a text file. Use <i>/computer</i> to target a single computer. Use <i>/container</i> to target an organizational unit within Active Directory. |
| <code>/computer:name</code>  |   |
| <code>/container:name</code> |   |
| <code>/recurse</code>        | When used with <i>/container</i> , also targets computers contained within nested OUs.  |
| <code>/ping</code>           | Verifies the connectivity to all targeted computers prior to attempting a connection. Using this argument will reduce the timeout wait when one or more computers cannot be reached on the network.   |

|                             |   |
|-----------------------------|---|
| <code>/log:path</code>      | Logs unreachable computer names to the specified file. This file can then be used later, along with the <code>/list</code> argument, to retry these computers. Note that a log is created only when used in conjunction with the <code>/ping</code> argument. |
| <code>/verbose</code>       | Causes the script to display more detailed, step-by-step status messages.   |
| <code>/delay:seconds</code> | Required. Specifies the number of seconds to display the operating system selection screen before starting the default entry.   |

You can run these scripts with the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Windows Management Instrumentation (WMI) to connect to each targeted computer. The `Win32_ComputerSystem` class provides access to major properties of the `Boot.ini` file, as shown in this portion of the script:

```
Dim cSystems, oSystem, sOutput
Verbose " Connecting to WMI on " & sName
Set cSystems = QueryWMI(sName,"root\cimv2","select * From Win32_ComputerSystem","","")
If Not IsObject(cSystems) Then
    WScript.Echo " *** Couldn't connect to WMI on " & sName
Else
    For Each oSystem In cSystems
        On Error Resume Next
        oSystem.SystemStartupDelay = WScript.Arguments.Named("delay")
        oSystem.Put_
        If Err <> 0 Then
            WScript.Echo " *** Error changing " & sName
            WScript.Echo " " & Err.Description
        Else
            Verbose " Changed " & sName
        End If
    Next
End If
```

The `SystemStartupDelay` property—one of the few writable WMI properties—configures the startup delay; the `Put_()` method is required to save the changes to this property.

## Troubleshooting

This script has the potential to result in two types of errors. First, a remote computer won't be available. Second, you won't have permission to work with the remote computer's `Boot.ini` properties. (Typically, only a member of the Administrators group will

have permission to do so.) In either case, you'll see an error message that alerts you to the problem for that computer. Make sure you have the appropriate permissions and that the computer is reachable via the network from the computer running this script. Unreachable computers can optionally be logged to a text file for a later attempt, and you can speed up the script by specifying the */ping* argument.

## To Learn More

- Find more VBScript samples for managing startup and shutdown properties at <http://www.microsoft.com/technet/scriptcenter/scripts/desktop/state/default.aspx>.
- Learn more about remote access to the *Win32\_ComputerSystem* class at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/accessing\\_a\\_remote\\_wmi\\_win32\\_computersystem\\_object.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/accessing_a_remote_wmi_win32_computersystem_object.asp).