

## Chapter 12

# General Network and Server Management Tasks

### In this chapter:

List FSMO Role Holders.....	224
Create DHCP Scope.....	227
Modify DHCP Options.....	231
Create DNS Host Records.....	234
Create Print Queues.....	238

Network management tasks aren't always the ones you need to perform on multiple computers, but they are often the ones you need to perform frequently or in a hurry (perhaps while troubleshooting a problem), or with great accuracy. Automation can help make information available quickly in time of need, can perform configuration changes with great accuracy, and can make frequently performed tasks less tedious.

Tasks automated in this chapter include:

- Listing the holders of Flexible Single Master Operations (FSMO) roles
- Creating DHCP scopes
- Modifying DHCP scope options
- Creating static DNS records
- Creating print queues

## List FSMO Role Holders



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap12>ListFSMOs>ListFSMOs.wsf`.

Operating System	Supported?	Prerequisites
Microsoft® Windows® 2000 family	No	<ul style="list-style-type: none"> <li>■ 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

Microsoft Active Directory® defines five Flexible Single Master Operations (FSMO) roles, which provide specialized services for various domain and forest functions. Each FSMO role is initially held by the first domain controller in the first forest you create; you can manually reassign these roles to other domain controllers. When troubleshooting domain problems, it can be useful to know which domain controllers hold which FSMO roles. For example, the Schema Master role is required to make changes to the Active Directory schema; if you are attempting to make changes to the schema and having problems, figuring out which domain controller holds the Schema Master role is a good first troubleshooting step. If, for example, that domain controller is unavailable or unreachable, you'll know why you're having problems.

### Performing This Task Manually

The management tools for Active Directory provide a way to discover which domain controllers hold each FSMO role. The various roles' current holders can be viewed by using different tools.

- To view the current Schema Master by using the Active Directory Schema console, with the console open, right-click Active Directory Schema and select Operations Master.
- To view the current Domain Naming Master by using the Active Directory Domains and Trusts console, with the console open, right-click Active Directory Domains And Trusts and select Operations Master.

- To view all other FSMO roles by using the Active Directory Users and Computers console, with the console open, right-click Active Directory Users And Computers, point to All Tasks, and then select Operations Master.

Note that you can also manually change the role holder for each FSMO role using the preceding steps. After selecting Operations Master, you'll see the current role holder and a Change button, which allows you to select a new role holder.

## Example

This script requires no command-line arguments. Simply run it:

```
ListFSMOs.vbs
```

The script will work for only the default domain of the computer on which the script is run, displaying the domain and forestwide roles for that default domain. This script will not work when run from a computer that is not a member of a domain.

## Syntax

This script has no command-line arguments. Set CScript.exe to be your default script processor, as described in Chapter 3, “Working with VBScript.” You can run this script by using the `/?` parameter to display the command's syntax.

## Under the Hood

This script uses Microsoft Active Directory Services Interface (ADSI) to query the FSMO role holders from the domain. One very important limitation of this script is that it can list only the FSMO role holders that are known to and accepted by the domain controller to which the script connects, which is generally the domain controller that authenticated the computer running the script. In a normal, properly functioning domain, all domain controllers will agree on which domain controllers hold which FSMO roles. However, it is possible for a domain to become damaged, in which case disagreement can occur. For example, suppose you take the PDC Emulator role holder offline, and then instruct another domain controller to seize the PDC Emulator role. If you then return the original PDC Emulator role holder to the network, the network will believe that this original role holder is the one and only PDC Emulator; the remaining domain controllers in the domain will believe the *new* holder is the one and only PDC Emulator. In this situation, this script will return different results when querying the original PDC Emulator (which believes it is the official role holder) and any other domain controller (which believes the new role holder is the official one).

An example of how the script works is shown in the following code:

```
Set oRootDSE = GetObject("LDAP://rootDSE")
If Err <> 0 Then
    WScript.Echo "Could not connect to the domain."
    WScript.Quit
End If

' Schema Master
Set oSchema = GetObject("LDAP://" & oRootDSE.Get("schemaNamingContext"))
sSchemaMaster = oSchema.Get("FSMORoleOwner")
Set oNtds = GetObject("LDAP://" & sSchemaMaster)
Set oComputer = GetObject(oNtds.Parent)
WScript.Echo "Forest :          Schema Master: " & oComputer.Name
```

The first line of code connects to the domain; the last five lines of code query the Schema Master role holder name and display that information on the command line. The script repeats this process to obtain the other four FSMO role holder names.

## Troubleshooting

The most common problem with this script is its inability to connect to the domain, which usually occurs only on a computer that is not a member of a domain at all. On a domain member computer, this script rarely has a problem; even if it is unable to connect to the computer's preferred domain controller, it will automatically attempt to contact another domain controller to obtain the FSMO information.

## To Learn More

- Learn more about the FSMO roles and how to manually view their holders and transfer holders at <http://support.microsoft.com/default.aspx?kbid=324801&product=winsvr2003>.
- Read more about the FSMO roles and their function within Active Directory by referring to the Windows Server 2003 Help and Support Center.

## Create DHCP Scope



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap12\CreateDHCPScopes\CreateDHCPScopes.wsf`.

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

### Description

The Dynamic Host Configuration Protocol (DHCP) server included with Microsoft Windows Server 2003 is designed to support one or more *scopes*, which are ranges of IP addresses, from which addresses are issued to DHCP clients. Creating new scopes is typically performed through the DHCP Server console, a graphical administrative tool. However, using the console to configure a new server with a large number of scopes can be time-consuming and error-prone. Windows Server 2003 provides a scriptable command-line tool, Netsh, that can be used to create new scopes from the command line. However, this tool can be complex and difficult to use manually. The `CreateDHCPScopes.wsf` script automates the Netsh tool, enabling easier bulk creation of multiple DHCP scopes.

### Performing This Task Manually

Using the DHCP Server console to create a new scope is a straightforward task. Typically, a new scope includes a range of IP addresses that define the scope as well as one or more scope options. Most environments define only a router option for the scope and define global rather than scope options for name resolution servers.

You can also use the Netsh command to manually configure a new scope. For example, the following will create a new scope named `MyScope` for the `192.168.1.0/24` subnet, on a DHCP server at `192.168.12.2`:

```
Netsh dhcp server 192.168.12.2 add scope 192.168.1.0 255.255.255.0 MyScope
ScopeComment
```

Additional steps are required to add an IP address range and any scope options to the newly created scope.

## Example

This script can be executed as follows:

```
CreatedHCPScopes.wsf /server:192.168.12.2 /scope:192.168.1 /name:MyScope
```

The script assumes a 24-bit subnet mask and assumes that the router for this subnet is at the .1 host address. It assigns an IP address range from .10 through .254 for use by DHCP clients. The new scope has the following properties:

- A subnet mask of 255.255.255.0
- A name of MyScope
- A router option (DHCP option value 003) of 192.168.1.1
- An IP address range from 192.168.1.10 through 192.168.1.254

The scope is created on a DHCP server at address 192.168.12.2. After creating the scope, adding the IP address range, and specifying the router scope option, the script activates the scope.

## Syntax

This script has several command-line arguments, which are described in the following table. Set CScript.exe to be your default script processor, as explained in Chapter 3.

<code>/server:address</code>	Specifies the name or IP address of the DHCP server. This must be a Windows Server 2003 server running the Windows DHCP Service.
<code>/scope:address</code>	Specifies the subnet address for the new scope. This must be only three octets. For example, to create a scope for 192.168.5.0/24, specify <code>/scope:192.168.5</code> .
<code>/name:string</code>	Specifies the name of the new scope.

You can run this script by using the `/?` parameter to display the script's syntax.

## Under the Hood

This script uses the Windows Script Host's *WshShell* object (specifically, the object's *Run()* method) to execute the Netsh command. The following script code does most of the work:

```
Dim sCmd
sCmd = "netsh dhcp server " & WScript.Arguments.Named("server") & _
      " add scope " & WScript.Arguments.Named("scope") & ".0 255.255.255.0 " & _
      WScript.Arguments.Named("name") & " scriptedScope"
GoRun sCmd
```

```
'create scope range
sCmd = "netsh dhcp server " & WScript.Arguments.Named("server") & _
      " scope " & WScript.Arguments.Named("name") & _
      " add iprange " & WScript.Arguments.Named("scope") & ".10 " & _
      WScript.Arguments.Named("scope") & ".254"
GoRun sCmd

'create router Option
sCmd = "netsh dhcp server " & WScript.Arguments.Named("server") & _
      " scope " & WScript.Arguments.Named("name") & _
      " set optionvalue 003 IPADDRESS " & WScript.Arguments.Named("scope") & ".1"
GoRun sCmd

'activate scope
sCmd = "netsh dhcp server " & WScript.Arguments.Named("server") & _
      " scope " & WScript.Arguments.Named("name") & _
      " set state active"
GoRun sCmd

WScript.Echo "Commands completed"
```

You can make changes to some of the script's code so that you can change some of the script's basic operations. At the beginning of the script, the following code specifies how the new scope will be created:

```
sStart = "10"
sFinish = "254"
sRouter = "1"
```

Change *sStart* from 10 and *sFinish* from 254 to other values to change the IP address range included in the scope. For example, to have the script always create scopes with IP address ranges from .12 through .100, change *sStart* to 12 and *sFinish* to 100. If your routers are typically configured to use a .2 host address, change *sRouter* to 2, and each new scope's router option will use the .2 host address.

This script is most useful in environments in which each subnet is configured in a standardized fashion—that is, client addresses issued from range *x* to *y*, router at host address *z*, and all subnets configured to use Class C (24-bit) subnet masks.

## Troubleshooting

Because the Netsh command is not a scriptable COM object, it can't provide detailed feedback when a command fails. For this reason, troubleshooting this script can be difficult. Thus, you need to ensure that you have sufficient permissions to create scopes on the targeted DHCP server and that you provide acceptable values for the script's arguments. The script runs the Netsh command four times to create the scope, add the IP address range, create the router option, and activate the scope; each time, the Netsh command output will appear in a command-line window, where you might also be prompted for user credentials if necessary.

## To Learn More

- Learn more about using the Netsh command to create DHCP scopes at [http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/netsh\\_dhcp\\_example.asp](http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/netsh_dhcp_example.asp).
- Read more about Windows Server 2003 DHCP Server at <http://www.microsoft.com/technet/prodtechnol/windowsserver2003/serverroles/dhcpserver/default.mspx>.

## Modify DHCP Options

This sample script is built into Windows Server 2003.

Operating System	Supported?	Prerequisites
Windows 2000 family	No	■ WSH 5.6 or later
Windows XP Professional	No	■ WMI
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

Windows DHCP servers support a number of server and scope options. These options are used to pass IP configuration information to DHCP clients. Specifically, *server options* are passed to all DHCP clients obtaining an address from the server, whereas *scope options* are passed only to clients obtaining an address for the associated scope. Thus, the options passed to a client are all options for that client's issuing scope and all server options.

Automating the maintenance of DHCP options can make network configuration changes easier. For example, if your company has multiple DHCP servers (not uncommon in large environments), they might all be configured to issue the same DNS server addresses to clients (also a common scenario). Should those DNS server addresses change, automation can help ensure that the change is consistently and quickly applied to all DHCP servers.

### Performing This Task Manually

Most administrators manage server and scope options from within the DHCP Server console, a graphical user interface. The interface makes changing multiple options for a single server or scope easy, but making the same changes in multiple scopes or on multiple servers can be time-consuming and error-prone.

### Example

The Netsh command-line tool provides a means to modify DHCP server and scope options from the command line. By using the tool in a batch file, you can automate the process of changing multiple scopes or servers simultaneously.

To change a server option, run:

```
netsh dhcp server a.a.a.a set optionvalue xxx datatype data
```

Where:

- *A.a.a.a* is the IP address of the DHCP server; alternately, you might specify `\\servername` to use the server's name.
- *Xxx* is the number option code you want to change.
- *Datatype* is the type of data contained within the option. This might be BYTE, WORD, DWORD, STRING, or IPADDRESS. IPADDRESS is perhaps the most common.
- *Data* is the actual data to be assigned to the option, such as an IP address.

Changing a scope option is similar to changing a server option—you simply identify the scope in addition to the server:

```
Netsh dhcp server a.a.a.a scope scope set optionvalue xxx datatype data
```

For example, to set the Router option (whose option code is 003) to 192.168.1.1 for a scope 192.168.1.0 on the DHCP server at 192.168.12.2, you would run this code:

```
Netsh dhcp server 192.168.12.2 scope 192.168.1.0 set optionvalue 003
IPADDRESS 192.168.1.1
```

Common option codes and their data types include:

- **002–Time Offset** DWORD. Specify the offset in seconds.
- **003–Router** IPADDRESS. Can provide multiple addresses separated by spaces.
- **004–Time Server** IPADDRESS
- **006–DNS Servers** IPADDRESS
- **015–DNS Domain Name** STRING
- **044–WINS/NBNS Servers** IPADDRESS
- **046–WINS/NBT Node Type** DWORD

Note that not all DHCP clients support all the preceding options, but they can all be assigned to the DHCP server for those clients that do support them.

Turning the Netsh command into an automation tool is simple—just create a new text file with a .bat file name extension. If, for example, you need to change the DNS server option to 192.168.7.54 on three servers, the .bat file would contain the following:

```
Netsh server \\server1 set optionvalue 006 IPADDRESS 192.168.7.54
Netsh server \\server2 set optionvalue 006 IPADDRESS 192.168.7.54
Netsh server \\server2 set optionvalue 006 IPADDRESS 192.168.7.54
```



**Tip** Using the Copy and Paste features in Microsoft Notepad makes it easier to duplicate this line three times and reduces the chance you will introduce a typo when manually creating the second and third lines.

Saving and double-clicking the .bat file will make your change on all three servers.

## Syntax

server address	Specifies the name or IP address of the DHCP server. This must be a Windows Server 2003 server running the Windows DHCP Service. When specifying a name, use the format \\servername.
scope scope	Specifies the scope when changing a scope option.
optionvalue code type value	Specifies the option code for the option you want to change or set, the type (DWORD, WORD, STRING, IPADDRESS, or BYTE), and the value to change (or set) the option to.

## Under the Hood

The Netsh command provides a great deal of command-line network configuration functionality. It primarily relies on remote procedure calls (RPCs) to connect to remote servers and perform its work; firewalled servers might block these ports, which is a consideration when using this tool.

## Troubleshooting

Netsh provides excellent feedback when errors occur, and the most common errors are syntax-related. Before creating a .bat file containing multiple Netsh commands, take a moment to test a sample command against a nonproduction DHCP server to verify the command's accuracy and operation; then create your .bat file based on that tested command.

## To Learn More

- Learn more about Netsh scripting at [http://www.brienposey.com/kb/netsh\\_scripting.asp](http://www.brienposey.com/kb/netsh_scripting.asp).
- Read more about DHCP-specific uses of Netsh at [http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/netsh\\_dhcp.mspx](http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/netsh_dhcp.mspx).
- Read the complete list of industry-standard DHCP options at <http://www.iana.org/assignments/bootp-dhcp-parameters>.

## Create DNS Host Records



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap12\CreateARecords\CreateARecords.wsf`.

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

### Description

Although most Windows-based environments utilize Dynamic DNS (DDNS) to register host records, many environments still require the use of static host records in DNS for computers and devices that are not DDNS-capable (such as legacy devices and many non-Microsoft operating systems). This script, `CreateARecords.wsf`, automates the process of creating host records for both IP and IPv6 addresses (stored in DNS A and AAAA records, respectively).

Manual record creation is also necessary on systems where DDNS has been disabled. DDNS is often disabled on DNS servers that connect to the Internet, because having DDNS enabled presents a potential security risk. This script is also useful for these situations and does not require DDNS to be enabled.



**Note** This script functions only with the Windows Server 2003 DNS Service. It can also work with the Microsoft Windows 2000 Server DNS Service, provided the DNS server has the Windows Management Instrumentation (WMI) provider installed. See the “Learn More” links at the end of this section for more information about this provider and its installation procedures.

### Performing This Task Manually

Administrators typically use the DNS Server console when they need to manually create records. The procedure for creating A (IP address) and AAAA (IPv6 address) records is similar. To create an A record, right-click the zone in which the records should be created and select **New Host (A)**. In the **New Host** dialog box, type the host name and IP address, and click **Add Host** to save the information. The graphical user interface is not difficult to use, but creating records in bulk can be time-consuming.

## Example

This script is designed to read host names and addresses from a text file in which the host name, IP address, and IPv6 address (optional) are separated by commas. A sample text file might look like this:

```
Server1,192.168.0.12,3ffe:ffff:0100:f101:0210:a4ff:fee3:9566
Server2,192.168.7.43,3ffe:ffff:100:f101:210:a4ff:fee3:9566
Server3,192.168.7.216,3ffe:ffff:100:f101::1
```

As shown in the preceding text file example, both full and abbreviated IPv6 addresses are acceptable. The file does not contain a header row. If you don't want to specify IPv6 addresses, omit that information.



**Note** When specifying IPv6 addresses, you typically provide the address that any client can use to reach the server. Link-local addresses are not usually registered with DNS, for example, because they are valid only on the computer's local subnet. Private addresses (typically routable within an intranet) and global addresses (which are routable on the Internet) are suitable for DNS registration.

Running the script requires you to specify the text file path and name, as well as the DNS server name (or address) and whether you want IPv6 records created:

```
createARecords.wsf /dns:MyDNSServer /list:c:\hosts.csv /ipv6 /domain:company.com
```

You must also specify the domain in which the records will be created, and the DNS server specified must host a writable (primary) copy of that domain's zone. Note that specifying the `/ipv6` argument will cause errors if the input file does not contain IP and IPv6 addresses for each host.

## Syntax

This script has several command-line arguments, which are listed in the following table. Set `CScript.exe` to be your default script processor, as described in Chapter 3.

<code>/dns:name</code>	Specifies the name or IP address of the Windows DNS server.
<code>/list:file</code>	Specifies the path and file name of the text file containing the hosts and IP addresses to be added.
<code>/domain:name</code>	Specifies the name of the domain in which the records should be created.
<code>/ipv6</code>	Specifies that both IP and IPv6 records should be added to DNS for each host.

You can run this script with the `/?` parameter to display the script's syntax.

## Under the Hood

This script uses WMI to connect to a DNS server and create new records. The following code performs the bulk of the work:

```

Do Until oTS.AtEndOfStream
    sData = Split(oTS.ReadLine,",")
    sName = sData(0)
    sAddress = sData(1)

    'Create AAAA records
    if wscript.Arguments.Named.Exists("ipv6") Then
        sIPv6Address = sData(2)
        Set oItem = oWMIService.Get("MicrosoftDNS_AAAAType")
        WScript.Echo " Creating AAAA record for " & sName
        errResult = oItem.CreateInstanceFromPropertyData (sDNSServer, sDomain, sName,
1, 600, sIPv6Address)
        If errResult <> 0 Then
            WScript.Echo " ** Error: " & Err.Description
        End If
    End If

    'Create A records
    Set oItem = oWMIService.Get("MicrosoftDNS_AType")
    WScript.Echo " Creating A record for " & sName
    errResult = oItem.CreateInstanceFromPropertyData (sDNSServer, sDomain, sName, 1,
600, sAddress)
    If errResult <> 0 Then
        WScript.Echo " ** Error: " & Err.Description
    End If
End If
Loop

```

The variable *oWMIService* is set to an instance of the DNS server's WMI service, which provides the interface for querying and creating DNS records. The WMI namespace used, *root\MicrosoftDNS*, is available only when the DNS WMI provider is installed (which is always the case on computers running Windows Server 2003 and the Microsoft DNS service).

## Troubleshooting

Several things can go wrong with this script, although the most common problem concerns permissions. The user account running the script must have permission to create new DNS records on the DNS server specified.

Other problems generally come from the DNS server itself. For example, the DNS server might not allow a record to be created if an existing record with the same name exists; the outcome depends on how the permissions on that record are configured. The DNS server might also be configured to disallow certain record types (unlikely with A records but more likely with AAAA records, which are less commonly used). In all cases, the script will display an error message if the DNS server returns one.

Other errors usually result from improperly formatted input files. Always check to make sure your input file's format matches the example shown earlier and that the first line of the file contains a host name and an IP address, and not a header row.

## To Learn More

- Learn more about the DNS WMI provider at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dns/dns/dns\\_wmi\\_provider\\_overview.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dns/dns/dns_wmi_provider_overview.asp).
- Get DNS WMI scripting examples from [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dns/dns/dns\\_wmi\\_provider\\_scripting\\_examples.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dns/dns/dns_wmi_provider_scripting_examples.asp).
- Download and install the DNS WMI provider from [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dns/dns/installing\\_the\\_provider.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dns/dns/installing_the_provider.asp) (for Windows 2000 Server only; the provider is preinstalled on Windows Server 2003).
- Learn more about IPv6 at <http://www.ipv6.org/>.

## Create Print Queues



**On the CD** The sample script can be found on the CD that accompanies this book at `\Chap12\CreatePrintQueues\CreatePrintQueues.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 ■ Network connectivity to each remote computer

### Description

This script, `CreatePrintQueues.wsf`, is used to create multiple print queues. (A print queue is referred to as a *printer* in Microsoft Windows terminology. The physical hardware is referred to as a *print device*.) Because an administrator rarely needs to create the same kind of print queue on multiple computers, this script is designed to create multiple printers on a single computer. This can be useful for restoring a print server that contains multiple queues, or for migrating a print server's queues to a new server.

### Performing This Task Manually

Creating a printer involves using the Printers And Faxes folder, which is accessible from Control Panel. In the Printers And Faxes folder, an Add A Printer link is provided, which launches a wizard that collects the necessary information to set up the new printer. This process can be cumbersome and time-consuming when you need to set up multiple printers, so the script streamlines it. The script can also automatically enable the sharing of printers so that they are available to network users. However, the script uses the default share permissions for newly shared printers and does not provide a facility for specifying custom shared printer permissions.

### Example

This script provides two basic modes of operation. The first mode reads the existing printers on a server and writes their information into a text file. The second mode reads a text file—such as the kind created by the first mode—and creates printers based on the data contained within the file.

The first mode is executed like this:

```
createPrintQueues.wsf /server:Server1 /output:C:\Printers.csv
```

This code will connect to a server named Server1 and write its printer information to a file named Printers.csv. This file is a comma-separated values (CSV) file formatted as follows:

```
DriverName,PortName,DeviceID,Location,Network,Shared,ShareName
```

The second mode of the script reads a CSV file formatted in that fashion and creates printers:

```
createPrintQueues.wsf /server:Server2 /input:C:\Printers.csv
```

Keep in mind that this script does not install printer drivers; it simply creates queues that reference those drivers. For example, the script is capable of creating a queue for a Brand X print device, but it cannot install the Brand X device drivers. All device drivers needed to create the queues listed in the input file must be installed before running the script.

## Syntax

This script has several command-line arguments, which are listed in the following table. Set CScript.exe to be your default script processor, as described in Chapter 3.

<code>/server:name</code>	Specifies the name or IP address of the server to target.
<code>/input:file</code>	Specifies the path and file name of the CSV text file containing the printer information; cannot be used in conjunction with <code>/output</code> .
<code>/output:file</code>	Specifies the path and file name of the CSV text file that will contain the printer information; cannot be used in conjunction with <code>/input</code> .

You can run this script with the `/?` parameter to display the script's syntax.

## Under the Hood

The following code snippet is used to inventory existing printers and write their information to a file:

```
Set oTS = oFSO.CreateTextFile(WScript.Arguments.Named("output"),True)
If Err <> 0 Then
    WScript.Echo "*** Could not open output file"
    WScript.Echo Err.Description
    WScript.Quit
End If

Set cPrinters = oWMI.ExecQuery("Select * From Win32_Printer")
For Each oPrinter In cPrinters
    sData = oPrinter.DriverName
    sData = sData & "," & oPrinter.PortName
    sData = sData & "," & oPrinter.DeviceID
    sData = sData & "," & oPrinter.Location
    sData = sData & "," & oPrinter.Network
```

```

sData = sData & "," & oPrinter.Shared
sData = sData & "," & oPrinter.ShareName
WScript.Echo sData
oTS.WriteLine sData
Next

```

This is a simple WMI query for all available instances of the *Win32\_Printer* class. Using the script's */input* argument runs the following code, which reads information from a file and creates new printers by spawning instances of the *Win32\_Printer* class:

```

sData = ots.ReadLine
sData=Split(sData,",")
Set oPrinter = oWMI.Get("Win32_Printer").SpawnInstance_
oPrinter.DriverName = sData(0)
oPrinter.PortName = sData(1)
oPrinter.DeviceID = sData(2)
oPrinter.Location = sData(3)
oPrinter.Network = sData(4)
oPrinter.Shared = sData(5)
oPrinter.ShareName = sData(6)
oPrinter.Put_
If Err <> 0 Then
    WScript.Echo "*** Error creating " & sData(0)
    WScript.Echo Err.Description
End If

```

The script expects the data in the input file to be within acceptable ranges and properly formatted. Use the */output* switch to generate a sample file if you plan to manually create your own input files.

This script works best with networked printers that are accessed via TCP/IP printing (also called LPD/LPR or IP printing). Other types of printers might require local printer port configurations, which the script cannot automatically create.

## Troubleshooting

To run this script, you must have permission to create new printers on the targeted server. The script will not be able to create all printer types. Those using specialized ports, those using drivers that are not installed, and some printers utilizing USB connectivity will often cause errors. You can edit the file produced with the */output* argument to remove any printers that you do not want created or that are causing problems. If a printer cannot be created, the script will try to obtain and display error information. Removing the printer from the input file will prevent the error from recurring.

## To Learn More

- Find more Microsoft Visual Basic® Script (VBScript) samples that deal with printers and print queues at <http://www.microsoft.com/technet/scriptcenter/scripts/printing/servers/default.aspx>.