Windows Server 2003

Enterprise Logon Scripts

Microsoft Corporation

Published: October 2004

Abstract

Logon and Group Policy scripts are powerful, flexible tools that system administrators can use to provide users with a consistent, predictable, and secure computing experience.

Introduction

Technically, a logon script is nothing more than a script that runs whenever a user logs on to a network or, less commonly, whenever a user logs on to a local computer. Logon scripts are not new; in fact, logon scripts have been around almost as long as computer networks have. Many system administrators, for example, got their introduction to logon scripts, and scripting in general, by managing Novell Netware networks. Logon scripts were a staple of Netware networks, performing such tasks as mapping network drives to specified folders on the Netware server. Rather than require users to manually map drives each time they logged on, administrators created logon scripts that carried out this task automatically.

That is why logon scripts are so important: By performing important tasks each time a user logged on, logon scripts not only save administrators and other support personnel from having to manually assist users, but also freed users from having to deal with technical issues. Users can simply start their computers, log on, and assume that when the logon process is complete, they can start working.

Newer and more sophisticated applications have replaced Novell Netware, yet logon scripts remain one of the primary methods that system administrators use to ensure that users have consistent access to resources each time they log on. Providing access to resources is extremely important in itself, but logon scripts simultaneously help automate many other management tasks. For example, logon scripts can inventory computer hardware, perform audits of installed software, ensure that antivirus software is installed and up-to-date, and verify that applications such as Internet Explorer are properly configured. Although you can use alternate methods – such as Group Policy or Microsoft Systems Management Server (SMS) – to do many of these tasks, logon scripts provide a more comprehensive and flexible approach to managing desktop computers. Logon scripts are particularly convenient in organizations that run various versions of Windows operating systems, such as Microsoft® Windows®°98, Windows°NT® Server°4.0, and Windows®°2000). Logon scripts can be configured to run on any of these platforms, which is not necessarily true of other management technologies, most notably, Group Policy.

As computer hardware and software have evolved, so have logon scripts. Originally, the term *logon script* referred solely to a script that automatically runs each time a user logs on to a domain. Today, the term refers to scripts that run in any of the following circumstances:

- A user logs *off* a domain.
- A computer starts.
- A computer shuts down.

Even the traditional logon script (one that automatically runs whenever a user logs on to a domain) has evolved, primarily in the ways in which these scripts are assigned. Depending on your administrative needs (something to be discussed in detail in this whitepaper), you can:

- Directly to an individual user account within a domain.
- Directly to an individual user account on a local computer.
- Indirectly to any number of users or computers by using Group Policy.

Conceptually, logon scripts are very simple — they are scripts that run in response to a specific event, such as a user logging on or a computer shutting down. However, the various types of logon script and ways that they can be assigned is confusing to many administrators. The purpose of this paper is to help clear up that confusion by doing the following:

- Define the different types of logon scripts and explain when one type of logon script might be more appropriate to use than another.
- Explain how to assign the types of logon scripts.
- Discuss useful techniques for writing and debugging logon scripts.
- Provide sample scripts that carry out common logon script tasks.

Types of Logon Scripts

The term *logon script* applies both to specific and generic classes of scripts. The generic class of logon scripts

consists of the following types:

- Group Policy logon scripts
- Group Policy logoff scripts
- Group Policy computer startup scripts
- Group Policy computer shutdown scripts
- Domain user logon scripts
- Local user logon scripts

All logon scripts operate in similar fashion with two main differences between the various types:

- When the script actually executes
- The security context in which it executes.

Aside from those two differences, all logon scripts follow one basic process. When a computer starts up, or a user logs on, the domain controller determines if any startup script has been assigned to either the computer account or the user account. If such a script is found, that script is copied *in memory* from the domain controller to the local computer. The actual script file (for example, logon.vbs) is never copied; logon.vbs will not be found on the local computer. Instead, the script is copied in memory and executed locally on each computer.

Because logon scripts execute locally, you can use scripting technologies, such as the FileSystemObject, that typically cannot be used remotely. For example, if you are working on Computer A, you cannot – without additional configuration – run a script that creates an instance of the FileSystemObject on Computer B. The new Windows Firewall in Windows XP Service Pack 2 is fully scriptable. However, those scripts must run locally .How can you get around the fact that scripts designed to manage Windows Firewall cannot be run remotely? One simple solution is to include those scripts as part of a logon script. For example, your logon script might include code that ensures that the Windows Firewall is running, and that it has been properly configured.

Similarly, computer shutdown scripts and user logoff scripts are copied into memory and run locally each time a computer shuts down or a user logs off.

The only exception to this process is local user logon scripts. Although local user logon scripts execute in similar fashion, they are not copied from a remote server to a local server because they must be stored on the local computer.

Group Policy Logon and Logoff Scripts

As the name implies, Group Policy logon and logoff scripts are assigned to user accounts using Group Policy and apply to all the users in the domain or organizational unit (OU) in which the GPO is assigned. A Group Policy logon script runs only after a user logs on. Because of this, logon scripts often include tasks endemic to a particular user. For example, the script might map a network drive based on the groups that a user belongs to. Logon scripts do not run until a user logs on, and a user cannot log on until the computer has completed startup. Because part of the startup process includes running any Group Policy startup script, logon scripts always run after computer startup scripts have run.

Logoff scripts are configured to run automatically each time a user logs off a domain. Logoff scripts run before computer-shutdown scripts because the logoff process must be completed before the computer can shut down. Like logon scripts, logoff scripts carry out tasks that apply to an individual user. For example, a logoff script might be used to delete files in a user's Temporary Internet Files folder.

You use Group Policy to assign logon and logoff scripts, and the scripts run under the security context of the user logging on or off. The fact that these scripts run under the security context of the individual user has security implications that are discussed later in this paper.

Group Policy Startup and Shutdown Scripts

Group Policy startup and shutdown scripts are similar to logon and logoff scripts, but they apply to the computer account instead of the user account. For example, suppose User A has been assigned a logon script, and User B has not; suppose also that the two users share a computer that has been assigned a startup script. When User A logs on to the computer, two things happen:

- Before User A can log on, the computer startup script runs. User A cannot log on until the startup script has finished.
- After the startup script has completed, User A can log on, and then User A's logon script runs.

When User B logs on to the computer, the startup script runs because startup scripts run whenever anyone starts the computer. However, no logon script runs because no logon script has been assigned to User B. Logon and logoff scripts are tied to user accounts; startup and shutdown scripts are tied to computer accounts. This is

the key difference between the two types of scripts. To perform a task in which the user's identity is critical, such as mapping a drive based on membership in a group or adding a file to a user's My Documents folder, you must use a logon or logoff script. If, instead, you need to perform a task to a computer, such as determine the IP address, verify free disk space on drive C, check for the existence of a specific file, then you can use either a logon/logoff script or a startup/shutdown script.

Startup and shutdown scripts are assigned by using Group Policy and run in the context of the local system account. Because the scripts rely on Group Policy, startup and shutdown scripts cannot be applied to computers running Windows 98 or Windows NT 4.0. The same is true for Group Policy logon and logoff scripts. Like all Group Policy objects, startup and shutdown scripts can be easily applied to all the computers in a site, domain, or OU in which the GPO is assigned.

Domain User Logon Script

Domain user logon scripts are similar to Group Policy logon scripts: they run each time a user logs on to the domain. However, domain user logon scripts are not assigned by using Group Policy; instead, domain user logon scripts are assigned to individual user accounts. This is important because a domain user logon script runs even if the user is logging on from a computer running a version of Windows that Group Policy cannot be applied to, such as Windows 98.

Typically, Group Policy logon scripts are easier to assign and manage than domain user logon scripts. You can assign the same logon script to all the users in your domain by creating one GPO, and then remove that logon script by deleting that same GPO. This alone makes Group Policy logon scripts the preferred method for assigning logon scripts, depending on the versions of Windows you use in your organization. If all your computers are running Windows 2000 or later, and if you use the Active Directory® directory service, you likely have no need for domain user logon scripts. In that case, you should use Group Policy logon scripts.

However, if you are running in a Windows NT 4.0 domain, you need to use domain user logon scripts because Windows NT 4.0 domains do not support Group Policy. Likewise, you might have client computers running Windows 98 or Windows NT 4.0. Because Group Policy cannot be applied to these versions of Windows, you must assign a domain user logon script if you want a script to run each time a user logs on to the domain. Domain user logon scripts run no matter which operating system is installed on a user's computer.

Note that there is no domain user equivalent to the Group Policy logoff script, which automatically runs each time a user logs off the domain. Likewise, there is no equivalent to the Group Policy startup or shutdown scripts. You can assign a user a domain user logon script, but you cannot assign a computer an equivalent domain computer startup or shutdown script. Why not? Domain user logon scripts are provided primarily for backwards compatibility with computers running, such as Windows 98. Because Windows 98 computers cannot have computer accounts in Active Directory, it makes little sense to try and manage those computers using the directory service.

Local User Logon Script

Local user logon scripts run when a user logs on to the local computer instead of a domain. If you log on to a domain, your local user logon script will not run. These scripts run only if you log on locally. That also means that these scripts do not run if you are not connected to the domain but log on using cached credentials. Because logging on using cached credentials is equivalent to logging on to the domain, the local user logon script will not run.

Local user logon scripts are typically of limited use, and for two reasons. First, even when not connected to the network, most users log on to their computers by using cached credentials. This prevents them from having to maintain two sets of passwords — one for the domain, one for local logons — and from having two separate user profiles. Having two separate user profiles is a potential problem if they are not local administrators of the computer.

Second, managing local user logon scripts is difficult. Any such scripts must be individually managed on each individual computer. For example, suppose you have 1,000 computers and you want to run a local user logon script on each computer. At a minimum, you will need to copy the logon script to each of these 1,000 computers, and then assign the logon script to each local user account. This manual management must be repeated any time you change the logon script.

In addition, users typically are not connected to the network when they log on locally. In most cases, users log on locally because they are not connected to the domain. Although you can still take a hardware inventory as part of a local user logon script, the collected information has to be stored locally and is not available for administrative use. Because users have not logged on to the domain, you cannot provide access to resources based on group membership or perform similar tasks that rely on users being authenticated within the domain.

Choosing a Logon Script Solution

The wide variety of available logon scripts is sometimes more confusing than helpful to many administrators. In the early days of computing, system administrators who wanted to use a script to manage user workstations

had one option — a traditional logon script, assigned to an individual user account that automatically ran each time that user logged on to the domain. Today, you must choose between logon and logoff scripts, shutdown and startup scripts, domain user logon scripts, and more. With all these possibilities, how do you know which type of logon script is best for you?

Choosing a logon script type (or types) depends on several variables unique to your situation, and there is no right answer to the question "What kind of logons script should I use?" Following are some considerations that can make answering this question easier:

- Your need to support multiple versions of Windows.
- An understanding of the security context in which logon scripts run and the rights and privileges needed to carry out the tasks that logon scripts perform.
- An understanding of the computer startup process, and the times at which the various logon scripts run. Is it better to run a script when a computer starts up, when a user logs on, or when a computer shuts down? Or does it really matter?

These considerations are addressed in the following sections of this paper.

Supporting Multiple Versions of Windows

An important consideration when deciding which type of logon script to use is the version (or versions) of Windows operating system your organization uses. You can use Group Policy logon scripts only with client computers running Windows 2000, Windows XP, or Microsoft® Windows Server™ 2003. You cannot apply Group Policy scripts to computers that run Windows 98 or Windows NT 4.0. Consequently, if you support and manage computers running Windows 98 or Windows NT 4.0, you cannot rely exclusively on Group Policy logon scripts. Similarly, you also cannot apply Group Policy if you are not running Active Directory. If you use a Windows NT 4.0 domain, you must use domain user logon scripts instead. If you do not have Active Directory, or you must support versions of Windows prior to Windows 2000, your ability to rely exclusively on one kind of management system (such as Group Policy logon scripts) is limited.

Versions of Windows operating systems and their compatibility with domain user and Group Policy logon scripts are summarized in Table 1.

Table 1 Logon Script Compatibility by Operating System

Operating System	Compatible with Domain User Logon Scripts	Compatible with Group Policy Scripts
Windows 98	Yes	No
Windows NT 4.0	Yes	No
Windows 2000	Yes	Yes
Windows XP Professional	Yes	Yes
Windows Server 2003	Yes	Yes

The version of Windows installed on your computers also determines the depth and breadth of the management tasks that you can carry out using logon scripts. For more information about using logon scripts with other versions of Windows, see "Working with Previous Versions of Windows" later in this paper.

Logon Scripts and Security

The different operating systems that you must support are the prime determinant of the type of logon script you must use. If you support computers that run Windows 98 and Windows NT 4.0, then your choice is limited to domain user logon scripts. But even if you do not need to support these older versions of Windows, you still have to make some decisions pertaining to the security context in which the various types of logon scripts run.

Logon and logoff scripts (including domain user logon scripts and local user logon scripts) run in the security context of the user who has just logged on or off. This means these scripts can only carry out tasks that the user has the right to perform. If your users are all local administrators, this is not an issue. Local administrators can carry out almost any task on a computer. But if your users are not local administrators, your logon script might attempt to carry out tasks that users do not have the right to perform. For example, suppose you include code for installing a new piece of software on all your computers. If users do not have the right to install software on their computers, this part of the logon script will fail. Depending on the type of error generated and the error handling you do or do not have included in the script, the entire logon script might fail.

> **Important** The fact that logon scripts run under the security context of an individual user means that you should not test logon scripts using only your Domain Administrator account. If you do this, do not assume that the lack of any problems means the scripts will work for everyone. Instead, make sure you

test your scripts by using accounts where the user is a local administrator, where the user is a Power User, and where the user has very limited rights and privileges.

By contrast, computer startup and shutdown scripts run in the local system context. Because the local system has unrestricted access to everything on a computer, the types of tasks that startup and shutdown scripts can perform are greatly expanded. For example, only administrators can change the start mode of a disabled service (even Power Users are denied this right). Suppose you have disabled a service in response to a security bulletin, and now you want to re-start that service. Restarting a disabled service requires you to first change the start mode, and then restart the service. Because this requires administrative privileges, it cannot be included in a logon script unless your users are local administrators. However, it can be included as a startup or shutdown script, because those scripts run under the local system account.

On the other hand, startup scripts run before the user logs on and shutdown scripts run after the user has logged off. With a startup or a shutdown script, you can only carry out tasks that are the same regardless of who is or is not logged on to a computer. For example, you can do a hardware inventory as part of a startup script because a computer has the same amount of installed memory and the same number of USB ports regardless of the current user. However, you cannot map network drives based on group membership because, at the time that the startup script runs, you do not know who the user is.

This means that you must consider the task and the security context required to perform that task when choosing a logon script task. Likewise, you consider whether each task can be performed "generically," or whether the successful completion of the task requires the script can identify the logged-on user.

Logon script types and the security contexts in which they run are summarized in Table 2.

Table 2 Logon Script Compatibility by Operating System

Logon Script Type	Security Context
Group Policy logon	Security credentials of the logged-on user
Group Policy logoff	Security credentials of the logged-on user
Group Policy computer startup	Local system account
Group Policy computer shutdown	Local system account
Domain user logon	Security credentials of the logged-on user
Local user logon	Security credentials of the logged-on user

When Logon Scripts Run

Another factor to consider when choosing a logon script type is the point when the various logon scripts run in the startup-and-shutdown cycle. From startup to shutdown, the lifecycle of logon scripts goes like this:

1. The computer starts.
2. Computer startup scripts assigned by using Group Policy run.
3. After the startup scripts have completed, the Windows Authentication dialog box is displayed, and the user can log on.
4. Any Group Policy logon scripts run.
5. After the Group Policy logon scripts have completed, any domain user logon scripts run.
6. After the domain user logon scripts have completed, the Windows desktop is available, and users can begin working.
7. The user shuts down the computer.
8. While still in the current Windows session, any Group Policy logoff scripts run.
9. The user is logged off.
10. Before the computer actually shuts down, any computer shutdown scripts assigned by using Group Policy run.
11. The computer shuts down.

These timing issues are important for two reasons. First, computer startup and computer shutdown scripts run only when no one is logged on to the computer. This makes startup and shutdown scripts excellent places to do generic tasks such as a hardware inventory or backup. However, it also means that you cannot perform tasks like mapping a network drive or connecting a user to a network printer because, at the time these scripts run, there is no user.

Second, you must consider the perceived impact on the user when running logon scripts. For example, suppose

you have a very extensive inventory script that requires 2 or 3 minutes to complete. If you run this script as a computer startup script, the impact on the user might be less noticeable. Users expect computers to take at least some time to start. However, if you run this as a user logon script, the impact might seem very different. After pressing Ctrl-Alt-Delete and logging on, most users expect to begin working immediately. Waiting 2 or 3 minutes before the desktop is displayed might have a negative impact that exceeds any value gained by taking inventory each time a user logs on.

Lengthy procedures, in other words, are probably best carried out as startup scripts instead of logon scripts. Alternatively, you might consider dividing the script functions. For example, you might do part of the inventory doing computer startup and a second part during logon. That extends both the startup and the logon process, but without making either one seem interminably long.

You can also run long procedures as part of a logoff or shutdown script. The one drawback to this, however, is that a user might assume that the computer has stopped responding and manually turn it off. That not only interferes with the recommended shutdown procedure, but also abruptly terminates your logoff or shutdown scripts.

Comparing Logon Scripts with Alternate Solutions

Today system administrators have a wide variety of management options, ranging from non-Microsoft management systems to technologies, such as Group Policy, WMI, and ADSI, which are included in the Windows operating systems. In many ways, the wealth of administrative options available to you is extremely useful; it provides a management flexibility not found in previous versions of Windows. However, the many available options also pose a big management challenge: Which management option, if any, is right for you? Not only must administrators choose the appropriate type of logon script for their situation, they also must consider whether logon scripts are truly preferred over such things as Group Policy, Microsoft Systems Management Server (SMS), or even remote scripting technologies that do not rely on the startup/shutdown cycle.

Answers to this problem vary by organization. To answer the questions for yourself, you need to compare the advantages and disadvantages of your options in the context of your organization's needs.

Logon Scripts vs. Group Policy

At first glance, there appears to be considerable overlap between logon scripts and Group Policy because scripts can carry out many of the same user activities. On closer look, however, you can see that these technologies complement more than compete. Group Policy is designed largely, to configure user workstations and to lockdown user activities primarily by modifying registry values. However, to configure registry-based values on computers, you must weigh the pros and cons of using Group Policy against the pros and cons of using logon scripts.

Advantages of Using Group Policy

One big advantage of using Group Policy is that much of the hard work is already done for you. Suppose you want to ensure that screensaver password protection is enabled on all your computers. With Group Policy, you simply locate and enable the password-protection policy within the Group Policy Object Editor. By contrast, with a logon script you need to know which registry value to modify, the data type that the registry uses, and the correct configuration value to use. Then you need to write a script to make that change. Often it is easier to configure a GPO than to write a script.

Typically, Group Policy has the following advantages over logon scripts:

- **Typically, it is easier to create a GPO that configures a setting than to write script code.** The Group Policy Object Editor makes applying registry-based settings to a computer easier. For example, to password-protect a screensaver, you only need to locate the **Password protect the screen saver** policy within the Group Policy Object Editor, and then click **Enable**. By contrast, a script requires you to type and then test the following code:

```
HKEY_CURRENT_USER = &H80000001
strComputer = "."
Set objReg = GetObject("winmgmts:\\" & strComputer & _
    "\root\default:StdRegProv")
strKeyPath = "Control Panel\Desktop"
objReg.CreateKey HKEY_CURRENT_USER, strKeyPath
ValueName = "ScreenSaverIsSecure"
strValue = "1"
objReg.SetStringValue HKEY_CURRENT_USER, strKeyPath, ValueName, strValue
```

  Typing and testing code such as the preceding example is not particularly difficult. Still it requires more effort, and that method can be more prone to error than making a selection within the Group Policy Object Editor. You can create a GPO to configure several settings in a few minutes. That is not always the case

with writing a script. If you need to configure registry settings and you are already using Group Policy, creating or modifying a GPO is typically faster and easier than writing a script to do the same task.

- **Group Policy has no dependency on the scripting host**. If all your workstations run Windows 2000 or later, Group Policy simply runs. You do not need to install, update, or maintain additional software on the client computers. This is not necessarily the case with scripting. For example, Windows 2000 includes Windows Script Host (WSH) version 2.0. Windows XP includes WSH version 5.6. If your scripts take advantage of the new capabilities included with WSH 5.6 (such as the Exec method), your scripts will fail on your Windows 2000 computers unless (and until) you upgrade to WSH 5.6, and then you must do one of the following tasks:
  - Upgrade Windows Script Host on all your computers that run Windows 2000.
  - Include code that checks each computer for the installed the version of WSH, and then take a different course of action based on the results.

Group Policy does not require that effort, even though some Group Policy settings for Windows XP are not valid on Windows 2000. That is because the built-in Group Policy templates make the appropriate correction when a setting is supported on Windows XP but not on Windows 2000.

Disadvantages of Using Group Policy

Group Policy is designed to make the same registry changes on a number of computers simultaneously. That is a definite advantage if your management tasks are based on registry settings. However, what if you need to perform tasks that have nothing to do with registry settings, such as a hardware inventory or network-drive mapping? Logon scripts can map network drives and other tasks that Group Policy cannot do.

- **Group Policy is not supported on Windows 98 or Windows NT 4.0**. Group Policy cannot be applied to computers running Windows 98 or Windows NT 4.0. If you have computers running these operating systems, Group Policy is not an exclusive option even though you can develop a Group Policy solution for computers running Windows 2000 or later, and then develop an alternate solution for computers running Windows 98 and Windows NT 4.0. Note that Windows 98 and Windows NT 4.0 have only limited support for scripted solutions as well. However, by installing Windows Script Host, ADSI, and WMI, you can greatly increase your ability to manage these computers by using scripts. Short of upgrading to a newer version of Windows, there is no way to "Group Policy-enable" Windows 98 and Windows NT 4.0 computers.
- **Group Policy cannot carry out all the tasks commonly associated with logon scripts**. Group Policy is an excellent way to configure the user interface on client workstations, and it also provides the ability to lock down these computers by enabling settings that prevent users from actions such as running unauthorized software or changing their display resolution. However, locking down and configuring workstations is typically only part of the work carried out by a logon script. You can use logon scripts to collect hardware and software inventories, to map network drives and printers based on group membership, to backup and clear event logs, and to verify that computers have adequate disk space — tasks that Group Policy cannot do.
- **Group Policy has limited reporting features**. In Windows XP and Windows Server 2003, you can use the Resultant Set of Policy (RSoP) snap-in to identify the settings that changed when Group Policy was applied. However, it is very difficult to use RSoP to determine if any settings were not applied and, if not, why not. By contrast, you can include error-handling code within your script so that you can minutely track and report every operation that the script completes. For example, you might include code in a logon script that attempts to stop a service. If that service cannot be stopped because the user does not have the right to stop a service, the error code can be captured and reported back to you. You can then stop the service by using other means.

Using Group Policy for a certain set of tasks does not mean you cannot use a logon script for another set of tasks. Many organizations use Group Policy to configure generic settings on a computer, such as requiring all screensavers be password protected, and then use a logon script for other tasks, such as mapping network drives based on group membership.

Conflicts Between Group Policy and Logon Scripts

If you use both Group Policy and logon scripts, you might have a situation where a setting in your GPO conflicts with a setting in a logon script. Suppose you use Group Policy to set the screensaver on a computer. At the same time, you deploy a logon script that configures a different screensaver name. When a GPO and a logon script conflict, which setting is enforced?

The answer depends on which portion of the registry your script configures. For example, when a user manually selects a screensaver, the screensaver name is recorded in this registry location:

```
HKEY_CURRENT_USER\Control Panel\Desktop\Scrnsave.exe
```

Suppose your script configures the screensaver by writing to this same registry location. In that case, the script setting is overwritten by any Group Policy setting that also configures the screensaver name because

Group Policy settings are written here:

```
HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Control Panel\Desktop\Scrnsave.exe
```

By default, the operating system checks the Software\Policies key and applies any settings found there. At the same time, the operating system bypasses any similar settings configured elsewhere in the registry. Settings configured in Software\Policies always overwrite similar settings elsewhere in the registry. As long as you avoid writing scripts that modify the Software\Policies key, a script will not overwrite a Group Policy setting.

If you do write a script that modifies Software\Policies, the setting configured by the script is enforced. That is because logon scripts run after Group Policy is configured. In that case, Group Policy sets the screensaver name, and then moments later your script overwrites that name. In most organizations, however, this is not a problem because Software\Policies is a protected area of the registry. Unless your users are the local administrators of their computers, they cannot modify this part of the registry. Because logon scripts run within the context of the logged-on user, scripts attempting to modify Software\Policies fail for most users.

Logon Scripts vs. SMS

Logon scripts might also appear to conflict with dedicated management applications such as Microsoft Systems Management Server (SMS). Again, both technologies can do many of the same tasks. You can take inventory using scripts or by using SMS. You can install software using scripts or by using SMS. So which technology should you use?

Sometimes the choice is easy. Certain tasks can be carried out by using logon scripts that cannot be carried out by using SMS, and vice-versa. Typically, SMS is the preferred solution when either technology can perform a task, such as taking hardware inventory. . That is due to several factors:

- SMS does not require you to write and test a script.
- SMS has rich reporting capabilities.
- SMS is more robust and error-proof than a script you write yourself.

On the other hand, scripting is free, and SMS is not. This is perhaps the most important consideration. If you want to use SMS as a complete management tool, taking advantage of all its capabilities, then it is well worth the investment and is likely the better choice. However, if you want to use SMS to for only one task, such as to inventory computer hardware, then you should consider writing a script instead.

> **Note** Interestingly, SMS uses WMI to perform hardware inventories. In that event, there is little difference between using SMS and using a script to conduct a hardware inventory.

At other times you might find it advantageous to use both a logon script and SMS. For example, the SMS client runs as a service on each computer, meaning that users who have the requisite privileges can disable that service. If they do disable the service, the next time the computer starts and the user logs on, SMS cannot perform its management tasks. A solution for that is to use a logon script that checks the status of the SMS client and, if necessary, restarts it each time a user logs on. Then, SMS can then take over.

Logon Scripts vs. Other Scripting Solutions

One reason logon scripts initially became an important management tool is because logon scripts run on the local computer. That was important in the earlier days of computing because system administrators had few options for running scripts or executable files remotely. Today, you can easily write a script, start it on your workstation, and have that script remotely gather information from every other computer in your domain. The obvious question then is if scripts are not capable of acting against remote computers, is there any reason to still use logon scripts?

In some cases, the answer might be no. If your logon script only does inventories hardware, and then backs up and clears event logs, you might not be a need a logon script. Computer hardware — particularly on client workstations — rarely changes from day to day. Because of that, using a standalone script to do an inventory once a month or every few months might be sufficient. Similarly, event logs rarely fill to overflowing in a day. Running a separate script periodically to back up and clear event logs might be all you need.

However, your ability to run standalone system-administration scripts also depends partly on your organization policies. For example, as an energy savings measure, many organizations require users to turn their computers off when they leave for the day. Because those computers are not running, they are not available to standalone system-administration scripts. Other organizations, particularly those where users share computers, require users to log off when leaving for the day. In those situations, scripts can return generic information about a computer, such as installed hardware, but cannot carry out any tasks specific to the logged-on user because there is no logged-on user.

Typically, you are likely find that dividing your administrative tasks between logon scripts and standalone scripts is the best approach. Standalone scripts are best for tasks that do not have to be performed on a daily basis, and that are do not have to be customized based on who is logged on the computer. Conversely, logon scripts are a better approach for tasks that should be carried out each time a user logs on — such as checking

the status of the Windows Firewall, verifying that antivirus software has been installed, or making sure the Messenger service is not running. Logon scripts are also better for tasks that rely on knowing who is actually logged-on to a computer, such as mapping a network drive based on group membership.

Understanding When Logon Scripts Do Not Run

Logon scripts run whenever a user directly logs on to the domain — the operative phrase being "directly logs on." When a user does not directly log on to a domain, logon scripts do not run. As a system administrator, it is important for you to be aware of when that occurs.

Here are three scenarios in which a logon script will not run:

- **A user unplugs the computer's Ethernet cable, logs on using cached credentials, and then plugs the Ethernet cable back in after logon**. Cached credentials enable a user to logon even when a domain controller server is not available. After the user logs on and reconnects the Ethernet cable, the computer periodically attempts to connect to a domain controller and receive an updated authentication token. However, no attempt will be made to run a logon script because the user is already logged on.

  The reason is that a user can purposely or inadvertently bypass your logon scripts without much difficulty. If the network is down or the domain controller is unavailable, a user who logs on unavoidably bypasses the logon scripts.

- **A user connects to a domain over a VPN connection**. Because the user is already logged on, the logon script does not run. Also, when users connect over dial-up connections, logon scripts typically do not run.

- **A user never logs on or logs off**. This is perhaps the most common way to defeat a logon script. As hardware and software become more and more reliable, the need to restart a computer decreases. In turn, many users see no reason to shut down or log off a computer when they leave for the day. Instead, they lock the workstation, which is a convenience for them. The next day, they simply unlock their workstations and continue where they left off. However, this can cause management problems for system administrators; logon scripts never run because the user never logoffs.

Note Logon scripts are not the only management technology that can be defeated. Group Policy, for example, is periodically refreshed whether or not a user logs off. As a result, some Group Policy settings can be modified even if a user never logs off. However, other Group Policy settings, such as software installation and maintenance, are not changed during a Group Policy refresh. If a user never logs off, GPOs for installing, deleting, or updating software are never applied.

Obviously this can pose real problems. If you rely on logon scripts as a way to configure desktop settings or to verify that antivirus software is up-to-date, those settings might never be applied if a user always logs on over a VPN connection or by using cached credentials. How can? (For that matter, how can you?) Two management techniques that can tell you if the problem exists and help you deal with it follow:

- Monitoring system uptime
- Tracking logon script activity

Alternatively, consider whether critical tasks should be left to logon scripts. If logon scripts perform tasks such as mapping network drives or connecting users to printers, then most likely the network is not placed in jeopardy if a user bypasses the logon script. If a user fails to update the antivirus software, however, the consequence can extend far beyond an inconvenience to that one user.

For that reason, you might want to use a separate script to periodically validate such things as antivirus software. In addition to checking antivirus software by using a logon script, you can create a separate script to run at scheduled intervals. That methodically connects to each of your computers and reports the status of the antivirus software. Such a script does not replace your logon scripts; it is used in conjunction with your logon scripts.

Determining System Uptime

System uptime represents the amount of time a computer has been running since it last booted up. If you are using computer startup and shutdown scripts, the system uptime can give you a good indication of whether or not those scripts are being applied. Suppose the system uptime on a given computer is 168 hours (7 days). If you released a new startup script 24 hours ago, you know that particular script has not been applied on this computer because startup scripts run each time a computer starts, and it has been 7 days since this computer was started.

You can use a WMI script to determine how long a computer has been running. A simple script for determining system uptime is shown in Listing 1. This script can easily be modified to perform some action based on that uptime. For example, if a computer has been running continuously for 72 hours, you can log an event in the event log, send an alert to an administrator, or notify the computer owner that the computer needs to be periodically restarted to remain in compliance with company policies.

Listing 1 Determining System Uptime

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set colOperatingSystems = objWMIService.ExecQuery _
    ("Select * from Win32_OperatingSystem")

For Each objOS in colOperatingSystems
    dtmBootup = objOS.LastBootUpTime
    dtmLastBootupTime = WMIDateStringToDate(dtmBootup)
    dtmSystemUptime = DateDiff("h", dtmLastBootUpTime, Now)
    Wscript.Echo dtmSystemUptime
Next

Function WMIDateStringToDate(dtmBootup)
    WMIDateStringToDate = CDate(Mid(dtmBootup, 5, 2) & "/" & _
              Mid(dtmBootup, 7, 2) & "/" & Left(dtmBootup, 4) _
                  & " " & Mid (dtmBootup, 9, 2) & ":" & _
                      Mid(dtmBootup, 11, 2) & ":" & Mid(dtmBootup, _
                          13, 2))
End Function
```

You can schedule a script like the preceding example to run every night and check the system uptime on several computers. Depending on the policies of your organization, you can have this monitoring script restart any computer that has been running longer than the maximum allowed hours.

Tracking Logon Script Activity

System uptime gives you an approximate indication of whether startup and shutdown scripts are being run on a computer. You can safely assume that no startup or shutdown scripts have run since the computer last restarted. However, system uptime cannot say for sure that a startup script did run because the user might have been disconnected from the network when restarting the computer. System uptime shows that the computer was recently restarted, but it does not indicate that the computer was not connected to the network at the time and that the startup script did not run. Likewise, system uptime does not say whether logon or logoff scripts recently ran on the computer.

Therefore, you might find it useful to explicitly log the last time logon scripts of all types were run on a computer. A simple way to track the running of logon scripts is to create a registry value to store this information. For example, the script in Listing 2 creates a registry value named LogonScriptLastRun, and then populates that value with the current date and time. If you include this code in your logon script, then each time the logon script runs, it updates the value found in LogonScriptLastRun. You can then create similar registry values to track any logon scripts you use, such as a computer startup script.

Listing 2 Recording When a Logon Script Ran

```
Const HKEY_CURRENT_USER = &H80000001
strComputer = "."
Set objReg=GetObject("winmgmts:\\" &_
    strComputer & "\root\default:StdRegProv")
strKeyPath = "Logon Scripts"
strValueName = "LogonScriptLastRun"
objReg.CreateKey HKEY_CURRENT_USER, strKeyPath
strDate = CStr(Date)
strTime = CStr(Time)
strValue = strDate & " " & strTime
objReg.SetStringValue _
    HKEY_CURRENT_USER,strKeyPath,strValueName, strValue
```

At that point, you can then use a script – such as the one shown in Listing 3 – to retrieve this information, and thus determine the last time a logon script was run for a specific user.

Listing 3 Determining When a Logon Script Ran

```
Const HKEY_CURRENT_USER = &H80000001
strComputer = "."
Set objReg=GetObject("winmgmts:\\" &_
    strComputer & "\root\default:StdRegProv")
```

```
strKeyPath = "Logon Scripts"
strValueName = "LogonScriptLastRun"
objReg.GetStringValue _
    HKEY_CURRENT_USER,strKeyPath,strValueName, strValue
Wscript.Echo strValue
```

Even this method is not completely fail-safe because a user can use Regedit to modify the registry value. However, most users neither know how nor want to manually make such changes. If you want a more robust method of tracking logon script activity, you should have logon scripts either write an event to the event log or update a database each time they run.

Configuring Client Computers

When designing a logon-script strategy for your organization, much of your time and effort is concentrated on the server side. That makes sense because the server is where your logon scripts will physically reside, and the server is where logon scripts are configured and assigned. Nevertheless, it is important to remember that the server represents only half of the logon script equation. You must also pay some attention to your client computers and how they are configured.

If all the computers in your organization are running Windows 2000, Windows XP, or Windows Server 2003, then you can run logon scripts against those computers without installing additional software. If you are running Windows 2000, it is recommended that you upgrade to the latest version of Windows Script Host, version 5.6. However, you do not need to upgrade ADSI or WMI because there are no upgrades for either of these technologies.

The versions of WMI in Windows XP and Windows Server 2003 are greatly enhanced compared to the version in Windows 2000, and to the version available for Windows 98 and Windows NT 4.0. However, there is no way to upgrade the Windows 2000 version of WMI to make it 100% compatible with the version in Windows XP or Windows Server 2003. Consequently, there are many tasks you can script against Windows XP computers that cannot be scripted against Windows 2000 computers.

Incidentally, this is not the case with ADSI. With one exception, the ADSystemInfo class, ADSI is equivalent on all supported Windows platforms. The only additional differences are enhancements, such as support for setting Terminal Services properties, which are more for server management than for logon scripts.

For computers running Windows 98 or Windows NT 4.0, it is a very different story. Neither WMI nor ADSI is installed with these platforms. As a result, this severely limits your ability to carry out useful tasks using any scripts, including logon scripts. You can still run batch files and use command-line tools to carry out certain tasks, such as mapping network drives; however, you will find it difficult to do tasks such as a software or hardware inventory. In theory, you can gather a variety of nonMicrosoft command-line tools to inventory all the hardware on a computer. In practice, however, it is difficult to locate tools that can replicate the features found in WMI and probably impossible to find tools that can save data in the same, consistent format even to take a hardware inventory. It is unlikely that any of those nonMicrosoft tools can help you take a software inventory. For that, you need an additional set of nonMicrosoft utilities.

Thus, it is highly recommended that you install WSH, WMI, and ADSI on your Windows NT 4.0 and Windows 98 computers. Although not officially supported, the Windows 98 versions of these scripting technologies appear to work on computers that run Windows 95. However, the corresponding Windows NT 4.0 versions will not run on computers that run Windows NT versions 3.5 or 3.1. Installing these technologies takes time and effort. On the other hand, it takes at least that much time to install the required number nonMicrosoft utilities. Ultimately, the ability to manage so many previously unmanageable items makes installing WSH, WMI, and ADSI worth your time and energy.

For more information about how to obtain and install WSH, WMI, and ADSI for Windows 98 and Windows 2000, see the TechNet Script Center Scripting FAQ.

Working with Previous Versions of Windows

Installing WSH, WMI, and ADSI on your computers that run Windows NT 4.0 and Windows 98 gives you the potential to apply domain user logon scripts to any user who logs on to the domain using one of those computers. However, installing the scripting technologies is necessary but not sufficient when it comes to logon scripts. Even with WSH, WMI, and ADSI installed, Windows NT 4.0 and Windows 98 computers do not recognize VBScript or Jscript files as legitimate logon-script file types. If you assign a VBScript file (such as logon.vbs) to a user who logs on from a Windows 98 computer, that script will not run when the user logs on.

Instead, you can only assign a command shell script (also called batch file) to users logging on to computers running Windows 98 or Windows NT 4.0. Only after the initial command-shell script start can you call a VBScript or a Jscript script.

This complicates system administration because you must do one of two things:

- Create and maintain separate scripts for your Windows 98 and Windows NT 4.0 users.
- Create a generic script that can work on any version of Windows.

Typically, the second option – creating a generic script that can run on any version of Windows – is the preferred option. Although it is not difficult to create separate scripts for each version of Windows, assigning those scripts can be a challenge. If User A logs on from a Windows 98 computer, you can assign that user Win98_logon.bat. But suppose that user upgrades to Windows XP or logs on from a computer running Windows 2000. In such case, that user does not receive the appropriate logon script, and problems are likely to occur. For example, a user who relies on drive X being mapped to a specific network share might discover that drive X has not been mapped at all.

A better solution is to create a generic batch file that identifies the operating system being used on a computer and then calls the appropriate logon script based on that operating system. For example, the sample batch file shown in Listing 4 echoes the operating system in use on a computer. To run this script, copy it, paste it into Notepad, and then save it as a .bat file.

Listing 4 Detecting the Type of Windows Client Operating System

```
@echo off
If "%OS%"=="" Goto WIN9X
For /f "delims=" %%i In ('ver') Do Set OSVersion=%%i
For /f "delims=[]. tokens=4" %%i In ("%OSVersion%") Do Set Build=%%i
If "%Build%" LEQ "1381" Set OSType=WinNT
If "%Build%" EQU "2195" Set OSType=Win2K
If "%Build%" EQU "2600" Set OSType=WinXP
If "%Build%" EQU "3790" Set OSType=Win2K3
Goto END

:WIN9X
   Set OSType=Win9X
   Goto END

:END
   If "%OSType%"=="" Set OSType=Unknown
   Echo Client OS Type: %OSType%
```

A more practical version of this script not only identifies the current operating system, but also starts the domain logon script, passing the operating system name as a startup parameter. For example, the line of code shown in Listing 5 starts the script Logon.vbs in a minimized window, passing the value WinNT as the startup parameter.

```
Start /MIN cscript.exe %~dp0\Logon.vbs WinNT
```

Within Logon.vbs, you will also need to include code that reads the startup parameter, and then branches accordingly.

Listing 5 Using the Client Type Information to Start the Main Logon Script

```
@echo off
If "%OS%"=="" Goto WIN9X
For /f "delims=" %%i In ('ver') Do Set OSVersion=%%i
For /f "delims=[]. tokens=4" %%i In ("%OSVersion%") Do Set Build=%%i
If "%Build%" LEQ "1381" Goto WINNT
If "%Build%" EQU "2195" Goto WIN2K
If "%Build%" EQU "2600" Goto WINXP
If "%Build%" EQU "3790" Goto WIN2K3
Goto ERROR

:WIN9X
   If Not Exist %WINDIR%\Command\cscript.exe Goto LEGACY
   Start /MIN cscript.exe %0\..\Logon.vbs Win9X
   Goto END

:WINNT
   If Not Exist %WINDIR%\System32\cscript.exe Goto LEGACY
   Start /MIN cscript.exe %~dp0\Logon.vbs WinNT
   Goto END

:WIN2K
   Start /MIN cscript.exe %~dp0\Logon.vbs Win2K
   Goto END

:WINXP
```

```
   Start /MIN cscript.exe %~dp0\Logon.vbs WinXP
   Goto END

:WIN2K3
   Start /MIN cscript.exe %~dp0\Logon.vbs Win2K3
   Goto END

:ERROR
   Echo Unknown client operating system.
   Goto END

:LEGACY
   Rem Add tasks unique to Windows 9x and/or Windows NT
   Rem legacy clients that do not have WSH installed.
   Goto END

:END
```

Assigning Logon Scripts

Much of the confusion surrounding the use of logon scripts revolves around the logistics of placing logon scripts on a domain controller and then assigning those logon scripts to users or computers. This confusion arises because administrators mistakenly believe that:

- Logon scripts can be placed in any of a number of locations. Generally speaking, this is not true. Instead, logon scripts should always be placed in the Netlogon shared folder. This ensures that the scripts will be replicated to all other domain controllers in the domain and that the scripts will be accessible to Windows 98 and Windows NT 4.0 clients.
- Logon scripts can be assigned in many ways. It is true that that there are different ways to assign script types. However, within a given script category, such as Group Policy logon scripts, there are very few specific ways to assign a logon script. For example, Group Policy logon scripts can only be assigned by using Group Policy, and domain user logon scripts can only be assigned by modifying an individual user account. This can be done in one of two ways, either by using Active Directory Users and Computers, or by writing a script.

This paper provides the explanation and procedures you require to assign each logon-script type.

Assigning Group Policy Logon Scripts

Assigning logon scripts by using GPOs is remarkably easy. After you create the logon script, you simply do the following:

1. Copy the script to the Netlogon folder on a domain controller (typically, %windir%\Sysvol\Sysvol\Scripts).
2. Use Active Directory Users and Computers to create a GPO, and assign the logon script using that GPO.

Note There is no requirement that scripts be copied to a particular Netlogon folder. If the scripts are in the Sysvol folder, they can be used as a Group Policy logon script and will be properly replicated to all the domain controllers. However, placing the scripts in Netlogon makes it easy to locate all the logon scripts used in the domain, and ensures that scripts are accessible to users running Windows NT 4.0 or Windows 98.

Assign a logon script using Active Directory Users and Computers

1. In Active Directory Users and Computers, right-click the container where you want to add the logon script policy, and then click Properties.
2. In the container Properties dialog box, on the Group Policy tab, do one of the following:
   - To add the logon script information to an existing policy, select the name of the policy and then click Edit.

   -Or-

   - To add the logon script information to a new policy, click New, type a name for the new policy, press ENTER, and then click Edit.
3. If you are adding a startup or shutdown script, in Group Policy, expand **Machine Configuration**:

   -Or-

   - If you are adding a logon or logoff script, expand **User Configuration**.
4. Click Windows Settings.
5. Double-click the appropriate script type as follows:

- If you are adding logon script information, double-click **Logon**.

-Or-

- If you are adding logoff script information, double-click **Logoff**.

6. In the **Properties** dialog box, click **Add**.
7. In the **Add a Script** dialog box, click **Browse**, and then locate the script being added. Select the script in the list of files, and then click **Open**.
8. In the **Add a Script** dialog box, click **OK**.
9. In the **Properties** dialog box **OK**.
10. Close Group Policy, and then click **OK** in the **Properties** dialog box.

Although the process of assigning as logon script is simple enough, there are several factors to bear in mind when you make such an assignment. These factors include understanding what happens if you:

- Assign multiple logon scripts within a single GPO
- Assign multiple GPOs (each of which assigns a logon script) to a specific Active Directory container
- Assign logon scripts to multiple Active Directory containers

Assigning Multiple Logon Scripts Within a Single GPO

Using Group Policy Object Editor you can assign multiple logon scripts within a single GPO.

Any time multiple logon scripts are assigned in a single GPO, the scripts are processed beginning with the first script in the list, and ending with the last script in the list. In the preceding example, A_logon.vbs runs, and then B_logon.vbs runs. That is because A_logon.vbs is the first script in the list.

In many cases it might not matter to you which script runs first. However, there might be time when the order in which scripts are processed is important. For example, you might need B_logon.vbs to run first because it configures a setting that your other logon scripts require in order to complete successfully. In that case, you must move B_logon.vbs to the top of the list.

> **Note** Processing logon scripts in order assumes you have configured Group Policy to ensure that logon scripts are processed synchronously. For more information, see the section of this document titled Group Policy Settings for Logon Scripts. If you are not running logon scripts synchronously, then the order of processing cannot be guaranteed. Most likely, the scripts will attempt to run at the same time.

Assigning Multiple GPOs to a Container

Group Policy enables you to assign multiple GPOs to the same container. Each GPO in turn assigns a logon script.

Again, GPOs (and thus logon scripts) are processed from the top of the list down. In the preceding example, that means that the logon script assigned in LogonScriptPolicy1 is executed first, followed by the logon script assigned in LogonScriptPolicy2. If the script in LogonScriptPolicy2 must run before any other logon script, then you must move LogonScriptPolicy2 to the top of the GPO list.

Assigning Logon Scripts to Multiple Containers

You can assign GPOs (and thus Group Policy logon scripts) at multiple spots in Active Directory. For example, you can assign a logon script at the domain level, and then assign a second logon script at the OU level. The logon script farthest from the actual user or computer account always runs first; in this case, that means the domain-level logon script runs, followed by the OU-level logon script. Domain user logon scripts — logon scripts assigned to the user profile — always run last because all the Group Policy logon scripts have completed.

For example, suppose a user has an account in the Managers OU, which is in the Human Resources OU, which is in the fabrikam.com domain. In addition, suppose a separate GPO is applied to each of those containers, and each GPOI assigns a logon script to that same user. When that user logs on, the logon scripts run in the following order:

1. The logon script assigned at the domain level (fabrikam.com) runs first.
2. The logon script assigned in the Human Resources OU runs second.
3. The logon script assigned in the Managers OU runs last.

Why is this important? For one thing, this means a logon script assigned at the OU level trumps a logon script assigned at the domain level. Suppose you assign a logon script at the domain level, and one of tasks the script performs is to map drive X to the shared folder \\atl-fs-01\public. This ensures that all your users have drive X mapped to the same network location: \\atl-fs-01\public.

However, suppose another administrator assigns a logon script at the OU level; one thing that *this* logon script does is map drive X to \\atl-fs-02\accounting. If that happens, the logon script assigned at the OU level takes precedence over the logon script assigned at the domain level. As a result, users in this OU will not have drive X mapped to the public folder. Instead, these users will have drive X mapped to \\atl-fs-02\accounting. Needless to say, this can create problems for users running applications or scripts that expect to have drive X mapped to \\atl-fs-01\public. It also causes problems for users who have been given generic instructions such as, "After you finish your report, save it to drive X."

If multiple administrators assign logon scripts, it is important to keep track of what these administrators are doing, and where they are doing it.

Group Policy Settings for Logon Scripts

Group Policy enables you to determine which logon scripts are run in your organization; at the same time, it also enables you to determine how logon scripts are run. Group Policy settings for managing logon script control and execution are summarized in Table 3.

Table 3 Group Policy Settings for Managing Logon-Script Control and Execution

Group Policy Setting	Description and Explanation
Run logon scripts synchronously	By default, logon scripts run asynchronously. This means two things. First, if you have multiple logon scripts, you cannot guarantee the order in which those scripts run. Second, when logon scripts are processed asynchronously, Windows Explorer can start before the logon scripts finish processing. This means that users have access to Task Manager and can terminate any logon scripts that are still running. To prevent multiple logon scripts from running at the same time and competing for resources, run logon scripts synchronously. This ensures that Logon Script A starts *and finishes* before Logon Script B begins. In addition, when run synchronously, all logon scripts finish before Windows Explorer starts, reducing the chance of a user interrupting a logon script. One potential problem in running scripts synchronously is that it might take longer for the logon process to complete; this is because users do not have access to Windows Explorer until all the scripts finish processing. For example, suppose Script A takes 2 minutes to complete and Script B takes another 2 minutes to complete. Run asynchronously, the scripts can theoretically complete in 2 minutes; run synchronously, they take a full 4 minutes to finish. Needless to say, 4 minutes represents a long wait for a user logging on to a computer, ,which might be an argument for asynchronous processing, if the order in which the scripts are run does not matter. Alternatively, you might consider moving some of the logon script functionality out of the logon script. For example, Logon Script B does nothing but take a hardware inventory. Instead of take inventory each time a user logs on, you can either include this as part of a computer startup or shutdown script, or run it periodically as a standalone script.
Run legacy logon scripts hidden	By default, logon scripts written as either .bat or .cmd files (so-called "legacy" logon scripts) run in a visible command window; when executed, a command window open up on the screen. To prevent a user from closing the command window (and thus terminating the script), you can the **Run legacy logon scripts hidden** enable policy. This ensures that all legacy logon scripts run in a hidden window.
Run logon scripts visible	Logon scripts run using the default script host (either CScript or WScript) configured on each computer. If the script host is set to Cscript, scripts run, by default, in "hidden" mode, meaning no command window is displayed. (If the script host is set to Wscript, scripts run hidden whether this policy is enabled or disabled.) Typically, running logon scripts in a hidden window is the preferred setting. If the command window appears on screen, a user can

	conceivably terminate the window and thus terminate the logon script. With scripts running in hidden mode, scripts can be terminated only by opening Task Manager and terminating the script process. Because Task Manager is typically not available until the scripts have finished executing, it is difficult for users to prematurely terminate their logon scripts. There are times, however, when it is useful to enable this policy and run logon scripts in a visible command window. For example, when testing a logon script, you might want to run the script in a visible command window so that you can better track the script progress and interrupt it if necessary. In that case, you might want to enable the **Run logon scripts visible** policy.
Run logoff scripts visible	This policy is similar to **Run logon scripts visible**, except that it affects logoff scripts instead of logon scripts.

Similar settings are available for computer startup and shutdown scripts.

Running Logon Scripts Synchronously

It is possible to assign a user more than one logon script. While there are occasional advantages to assigning multiple logon scripts to a single user, this opens the possibility of scripts running g simultaneously, and possibly "colliding." Because of that, it is useful to understand how Windows processes multiple logon scripts, and how you can configure Group Policy to avoid logon script collisions.

For example, suppose you assign a Group Policy logon named script GP_logon.vbs at the domain level; by default, this script then applies to all users in the domain. At the same time, you assign a domain user logon script (User_logon.vbs) to an individual user account. When this user logs on, both logon scripts will run, most likely at the same time. That can create competition for system resources; it might even result in a script failing. For example, suppose User_logon.vbs requires GP_logon.vbs to finish before it can carry out some action. If the two scripts run concurrently, User_logon might end up failing simply because GP_logon has not finished running. To avoid such conflicts, use Group Policy to configure logon scripts to run synchronously; that ensures that GP_logon.vbs runs and finishes before User_logon.vbs even begins.

Likewise, you can use Group Policy to assign multiple scripts as part of a single logon script configuration. When you do this, and when you configure Group Policy to run logon scripts synchronously, the logon script listed first in the Logon Properties dialog box will run first; when finished, the logon script listed second will run. If you do not configure Group Policy to run logon scripts synchronously, then the order and timing of logon script cannot be guaranteed.

Typically, it is better to use a single logon script instead of a series of logon scripts to avoid competition for resources. In addition, a single logon script is much easier to manage than multiple scripts.

However, there might be times when it is advantageous to assign more than one logon script. For example, suppose there is a new virus outbreak that, among other things, modifies the HKEY_CURRENT_USER portion of the registry. You might create a temporary logon script that correctly configures the registry when the user logs on (because the problems are in HKEY_CURRENT_USER, changes must be made on a per-user, per-computer basis). After a patch has been released and applied to all your computers, you can then remove the now-unnecessary temporary logon script. In a case such as this, you might find it easier to create and test your temporary solution as a separate script rather than incorporating new code into your main logon script. This is especially true because you intend to remove the code as soon as the patch has been applied.

Assigning Domain User Logon Scripts

Group Policy logon scripts make it easy to assign the same logon script to any number of users: you simply assign a GPO to a given container (typically a domain or OU), and then any logon scripts in that GPO are automatically assigned to all the user or computer accounts within that container. This makes Group Policy logon scripts an excellent tool for generic administration of user and computer accounts.

Domain user logon scripts, by contrast, are assigned to individual user accounts (although the same script can be individually assigned to any number of user accounts). Because domain user logon scripts are tied to individual user accounts instead of to Active Directory containers, they can be more difficult to manage. For example, suppose you want Domain User Logon Script A assigned to all the users in a specific OU. To do this, you must assign that script to each individual user account in that OU. Furthermore, if you create a new user account, the account will not inherit that logon script; instead, you must assign the script to the new account. Group Policy logon scripts provide much more automated system administration. If you assign a logon script to an OU, by default that script applies to all users in the OU, as well as to any new users who join the OU. This is not true of domain user logon scripts.

Because assigning domain user logon scripts is more complex than assigning logon scripts by using Group

Policy, you might wonder why use domain user logon scripts. Domain user logon scripts might be appropriate in the following situations:

1. You are not running Active Directory. Group Policy logon scripts are only available in Active Directory; they are not available in workgroups or Windows NT 4.0 domain.
2. You have client computers running Windows NT 4.0 or Windows 98. Group Policy – including Group Policy logon scripts – cannot be applied to these computers.
3. Your Active Directory structure does not allow you to deploy a logon script to a subset of users by assigning a GPO to a specific OU. For example, suppose you want to assign a specific logon script to all the users in Human Resources and not to users in any other department. If the Human Resources accounts are all located in the same OU, you can assign those users the same logon script by applying a GPO to that organizational unit. But what if Human Resources accounts are in an OU that includes users from other departments? In that case, you will find it much more difficult to use Group Policy to assign a specific logon script just to Human Resources personnel. (By default GPOs apply to all the accounts in an OU, regardless of properties such as department membership.)

There are two important things to note about domain user logon scripts:

- The term *logon script* is used in its most literal sense. You can only assign a single script to run whenever a user logs on, and that single script can call other scripts. There is no such thing as a domain user logoff script, or a domain computer startup or shutdown script.
- Assuming you run logon scripts synchronously, domain user logon scripts always run after Group Policy is applied —thus after any Group Policy logon scripts run. If you use both Group Policy logon scripts and domain user logon scripts, you must make sure that your Group Policy scripts have no dependencies on the domain user scripts. That is because the he Group Policy scripts will complete before the domain user scripts even begin.

Assigning Domain User Logon Scripts in Active Directory Domains

If you are using Active Directory, you can assign domain user logon scripts either by using Active Directory Users and Computers to modify the user account profile, or by using a script to configure the scriptPath attribute. To assign a domain user logon script in Active Directory, create the script, and then copy it to the Netlogon folder on the domain controller (typically %windir%\sysvol\sysvol\scripts). Placing the script in Netlogon ensures that it can be found by Windows NT and Windows 98 clients, and that it will be properly replicated to all the domain controllers in the domain.

After the script has been copied to the Netlogon folder, assign the script using one of the following procedures:

To assign a logon script using Active Directory Users and Computers

1. In Active Directory Users and Computers, locate the user account, right-click the account name and then click **Properties**
2. In the **Properties** dialog box, on the **Profile** tab, enter the name of the logon script in the **Logon script** box. If the script is stored in the Netlogon folder, type the script name (for example, Logon_script.vbs).

   -Or-

   If the script is stored in a subfolder within Netlogon, include the folder name as part of the script name. For example, if the script is stored in the FinanceUsers subfolder, enter FinanceUsers\Logon_script.vbs in the **Logon script** box.
3. Click **OK**.

Alternatively, you can use an ADSI script such as the one shown in Listing 6 to assign logon scripts to individual user accounts or to a collection of user accounts, such as all the user accounts in an OU (see Listing 7).

To assign a domain user logon script by using scripts

1. Use the GetObject method to bind to the user account in Active Directory.
2. Use the Put method to set the value of the scriptPath attribute to "Logon.vbs" (see Listing 6).
3. Use the SetInfo method to save the changes and write the new scriptPath to Active Directory.

Listing 6 Assigning a Domain User Logon Script to a Single User

```
1  Set objUser = _
2      GetObject("LDAP://cn=myerken,ou=management,dc=fabrikam,dc=com")
3
4  objUser.Put "scriptPath", "Logon.vbs"
5  objUser.SetInfo
```

To assign the same logon script to all users in a domain

1. Use the GetObject method to bind to the Management OU in Active Directory.
2. Set a filter on the returned data to ensure that only User objects are included in the collection. Without this filter, your script tries to assign logon scripts to computers, printers, contacts, and any other type of object found in the OU.
3. For each user account in the OU, use the Put method to set the value of the scriptPath attribute to "Logon.vbs" (see Listing 7).
4. Use the SetInfo method to save the changes and write the new scriptPath to Active Directory.

Listing 7 Assigning a Domain User Logon Script to Multiple Users

```
Set objOU = GetObject("LDAP://ou=management,dc=fabrikam,dc=com")
objOU.Filter = Array("User")

For Each objUser In objOU
    objUser.Put "scriptPath", "Logon.vbs"
    objUser.SetInfo
Next
```

To assign the same logon script to *all* users in a domain, you can write a script that searches Active Directory and returns a list of all the user accounts. The script can then connect to individual accounts and make the assignment. Searching also makes it easy to assign specific user accounts to a subset of users. For example, you can assign a special logon scripts to all the members of the Human Resources Department regardless of the OU in which the user accounts resides.

For an example of assigning a domain user logon script to user accounts, see the script shown in Listing 8. This script retrieves a list of all the user accounts in the fabrikam.com domain, and then assigns each of those accounts the domain user logon script, logon.vbs:

Listing 8 Assigning a Domain User Logon Script to All Your User Accounts

```
On Error Resume Next

Const ADS_SCOPE_SUBTREE = 2

Set objConnection = CreateObject("ADODB.Connection")
Set objCommand =   CreateObject("ADODB.Command")
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection

objCommand.Properties("Page Size") = 1000
objCommand.Properties("Searchscope") = ADS_SCOPE_SUBTREE

objCommand.CommandText = _
    "SELECT ADsPath FROM 'LDAP://dc=fabrikam,dc=com' WHERE " _
        & "objectCategory='user'"
Set objRecordSet = objCommand.Execute

objRecordSet.MoveFirst
Do Until objRecordSet.EOF
    strPath = objRecordSet.Fields("AdsPath").Value
    Set objUser = GetObject(strPath)
    objUser.scriptPath = "logon.vbs"
    objUser.SetInfo
objRecordSet.MoveNext
Loop
```

For more information about using scripts to search Active Directory, see the TechNet Webcast Poking Your Nose into Active Directory.

Assigning Domain User Logon Scripts in a Windows NT 4.0 Domain

Domain user logon scripts can also be assigned within a Windows NT 4.0 domain; allowing for differences in the graphical user interface, the process is very similar to assigning domain user logon scripts in Active Directory. There is one very important difference, however: you cannot assign a .vbs or .js file as a logon script in a Windows NT 4.0 domain.

Computers running Windows NT 4.0 and Windows 98 are both capable of running WSH scripts (although you will first have to install WSH on the Windows 98 computers). However, neither platform is designed to run a WSH script *as a logon script*. If you assign a WSH script (such as logon.vbs) to a user logging on from a Windows NT 4.0 computer, the script will not run. Instead, you must assign a .bat or a .cmd file (for example, Logon.cmd) as the logon script. In fact, User Manager for Domains, the primary administration tool for Windows NT 4.0 domains, is designed to accept only .cmd or .bat files as logon scripts; it is not designed to accept .vbs or .js files.

This does not mean you are limited to using batch files as logon scripts; it only means that the initial logon script must be a batch file. After the initial logon script has started, it can call other utilities, including .vbs or .js files. Consequently, your logon.cmd file might consist of nothing more than a single line of code, one that that calls a VBScript file:

```
start cscript.exe real_logon.vbs
```

You can assign logon scripts to NT 4.0 domain users either by using User Manager for Domains, or by writing an ADSI script.

To assign a logon script by using User Manager for Domains

1. In User Manager for Domains, double-click the account or accounts you want to assign a logon script to. (To select multiple accounts, click the first account, then hold down the Ctrl key and click additional accounts. To open the joint **Properties** dialog box, double-click any of the selected accounts.)
2. In the **User Properties** dialog box, click **Profile**.
3. In the **User Environment Profile** dialog box, in the **Logon Script Name** box, type the name of the logon script. If the script is stored in the %windir%\System32\Repl\Import\Scripts folder, you need only type the name of the script (for example, logon.cmd). Otherwise, type the relative path to the file (for example, \FinanceUsers\logon.cmd). Click **OK**.
4. In the **User Properties** dialog box, click **OK**.

You can also use a script to assign domain user accounts in a Windows NT 4.0 domain. Note that you must have ADSI installed on your domain controller. For more information about installing ADSI and other scripting technologies on Windows NT 4.0 computers, see the section "Working with Previous Versions of Windows" earlier in this paper, Working with Previous Versions of Windows.

To assign NT 4.0 domain user logon scripts by using scripts

Listing 9 contains a script that assigns a logon script (logon.bat) to the KenMyer user account in the Fabrikam domain. To carry out this task, the script must perform the following steps:

1. Use GetObject to bind to the domain user account (KenMyer).
2. Set the value of the LoginScript attribute to the name of the script that will run each time the user logs on. Note that this must be either a .bat or a .cmd file.
3. Use the SetInfo method to write the changes to the user account.

Listing 9 Assigning a Logon Script in a Windows NT 4.0 Domain

```
1    Set objUser = GetObject("WinNT://FABRIKAM/KenMyer")
2    objUser.LoginScript = "Logon.bat"
3    objUser.SetInfo
```

Using scripts, you can also quickly assign the same logon script to all the users in the domain. Listing 10 contains a script that assigns the same logon script (Logon.bat) to all the users in the Fabrikam domain. To complete this task, the script must perform the following tasks:

1. Use GetObject to connect to the Fabrikam domain.
2. Create a filter to ensure that only user accounts appear in the resulting collection. Without this filter, the script attempts to assign a logon script to printers, services, groups, and other objects stored in the domain.
3. For each user in the collection, set the value of the LoginScript attribute to "Logon.bat."
4. Use the SetInfo method to write the changes to the user account.

Listing 10 Assigning a Logon Script to all the Users in a Windows NT 4.0 Domain

```
Set objDomain = GetObject("WinNT://FABRIKAM")
objDomain.Filter = Array("User")
For Each objUser in objDomain
    objUser.LoginScript = "Logon.bat"
    objUser.SetInfo
Next
```

Note you cannot use ADSI to search for user accounts within a Windows NT 4.0 domain. This can be a problem if you want to apply the logon script only to a subset of users. Instead of being able to search for users meeting the specified criteria, you will need to retrieve the entire list of domain users, and then individually check each account to see if it fits the specified criteria.

Assigning Local User Logon Scripts

As noted previously, local user logon login scripts are typically of limited use, especially when users not connected to the network log on using cached credentials rather than directly logging on to the local computer. If you have a need for local user logon scripts, however, you can assign these scripts using either the Computer Management snap-in or an ADSI script.

Before you can assign a local user logon script, you must create a shared folder named Netlogon on the local computer. Any local user logon scripts you want to assign must then be placed within that folder.

To assign a logon script using the Computer Management utility

1. On the desktop, right-click the My Computer icon and then click **Manage**.
2. In the Computer Management snap-in, expand **Local Users and Groups**, and then click **Users**.
3. Right-click the account name and then click **Properties**.
4. On the **Profile** tab, type the path to the logon script in the **Logon script** box. If the script is stored in the Netlogon folder on the local computer, you need only type the script name (for example. local_logon.vbs).
5. Click **OK**.

As is the case with domain user logon scripts, you can use ADSI to assign a local user logon script to a local user account.

Using Scripts to Assign Local User Logon Scripts

Listing 11 contains a script that assigns a logon script (Logon.vbs) to the KenMyer account on a local workstation. To carry out this task, the script must perform the following steps:

1. Use GetObject to bind to the user account in question (in this example, the KenMyer account on atl-ws-01).
2. Set the value of the LoginScript attribute to the name of the logon script. This script must be stored on the local computer in a folder with the share name Netlogon.
3. Use the SetInfo method to write the changes to the local user account.

Listing 11 Assigning a Local User Logon Script

```
Set objUser = GetObject("WinNT://atl-ws-01/KenMyer")
objUser.LoginScript = "Logon.vbs"
objUser.SetInfo
```

You can also assign the same logon script to all local users by following a process similar to that shown in Listing 12. The only difference is that you use the WinNT provider to connect to the local computer instead of to the domain.

Listing 12 Assigning a Logon Script to all Local Users

```
Set objDomain = GetObject("WinNT://atl-ws-01")
objDomain.Filter = Array("User")
For Each objUser in objDomain
    objUser.LoginScript = "Logon.vbs"
    objUser.SetInfo
Next
```

Writing Logon Scripts

Logon scripts are often treated as if they were different from other types of scripts. For example, scripting books often have an entire chapter devoted to writing logon scripts. This makes it appear as though the code used in logon scripts is not the same as that used in other scripts. In truth, writing a logon script is no different than writing any other kind of script; there is nothing special about the script code itself.

In fact, it can be argued that logon scripts are easier to write than other system administration scripts. This is because logon scripts always run locally on the computer in question. As a result, you can use Windows Script Host methods such as MapNetworkDrive that typically run only on the local computer. Without additional configuration and without using the WSHController object, you cannot use MapNetworkDrive to map a network drive on a remote computer. Likewise, you can make full use of other scripting objects, like the FileSystemObject and the Shell object, that are also designed to run locally instead of remotely.

> **Note** Because logon scripts are designed to be run locally, they are very easy to test while they are being developed. They do not have to be deployed as a logon script in order to run Instead, you can write some code and then simply run the script from Windows Explorer as you do any other scrip. During initial testing, there is no need to configure the script as an actual logon script.

To write a logon script you only need to open Notepad and begin writing. Write the script as though it will be run on the local computer, which is where it will be run. When you are finished, do one of the following:

- If the logon script is to be applied using Group Policy, save the script to the Netlogon folder on any domain controller. Active Directory replication will ensure that the script is properly copied to all the other domain controllers.
- If the logon script is to be applied as a domain user logon script, save the script to the Netlogon shared folder on any domain controller.
- If the logon script is being assigned in a Windows NT 4.0 domain, save the script to the %systemroot% \Repl\Import\Scripts folder.

At that point, assign the logon script using one of the procedures discussed previously, and you are ready to go.

As to what sorts of things should actually be included in a logon script, that depends on your individual needs. Sample code for tasks commonly included in logon scripts can be found in Appendix A; however, these tasks should not be considered exhaustive, nor should they be considered tasks that must be included in every logon script. If you want to take an inventory of installed software as part of the logon script process, then include that sample script (or some variation of it) in your actual logon script. But if you do not want to take an inventory of installed software, then do not feel compelled to include that code.

Logon scripts are designed to retrieve information or make configuration changes each time a user logs on or off, or each time a computer starts up or shuts down. Is there information – such as a list of currently-installed hot fixes – that you want to see each time a user logs on? If so, one easy way to retrieve that information is through a logon script. Is there information – like a hardware inventory – that you want to see every three or four months? If so, that information probably should not be retrieved by using a logon script. That only slows the logon process and provides you with extraneous data you that you do not need.

The following sections of the document will discuss generic techniques used in writing logon scripts, including:

- Testing logon scripts.
- Encoding logon scripts.
- Recording logon script data.

As noted previously, specific examples of logon script tasks are found in Appendix A.

Testing Logon Scripts

Although all scripts should be thoroughly tested before being deployed in your production environment, this is especially true of logon scripts. Suppose a mistake in syntax causes the script to hang partway through without completing. If you deploy this script in your production environment, every user who tries to logon will encounter problems: logon will be exceedingly slow (assuming it even completes), and your users might not receive the expected settings and configuration. In turn, this can keep them from accessing needed resources, and keep you from performing such key tasks as asset management and security auditing.

The following sections provide guidelines and suggested procedures for testing logon scripts. More detailed information about testing and debugging the code included in a logon script can be found in Appendix C of this document, Using the Microsoft Script Debugger.

Having a Backup Script That Works

Having backups of everything you do is always a good idea; it is especially important, however, to have backups of all the logon scripts deployed in your organization. For example, suppose you have a logon script

(logon.vbs) that has been successfully deployed in your organization. Now you want to add a new feature to the script. Probably the *worst* thing you can do is open logon.vbs in Notepad, change the code, and then saved the revised script.

What is wrong with this seemingly simple and easy approach? Suppose you make a mistake, and the script does not work as expected. You can just remove the script from the Sysvol or Netlogon folder. If the script cannot be found, it cannot be run. Removing the script altogether protects currently users logging on.

This situation also introduces a new dilemma: after you remove your only logon script, no logon script runs when users log on. If you rely on your logon scripts to lockdown user desktops or to provide users easy access to resources, those capabilities are also lost.

Instead of modifying your existing logon script, you should follow a process similar to this:

1. Make a copy of your existing logon script (copy_of_logon.vbs).
2. Make your changes to copy_of_logon.vbs.
3. Leave logon.vbs in place, and allow it to continue serving users who log on while you are revising the script.
4. After the revisions are complete and after the new script has been tested, rename the current logon script. For example, you can rename logon.vbs to last_known_good_logon.vbs.
5. Rename copy_of_logon.vbs to logon.vbs.

As soon as the revised script is renamed, it becomes the logon script for any users logging on to the domain. If you later discover that the revised script is not working properly, you have a quick fix: remove logon.vbs (the revised script) and rename last_known_good_logon.vbs to logon.vbs. That fix restores the previous functionality.

Providing a Way to Rollback Changes When Possible

Sometimes logon scripts do nothing more than gather information: taking a hardware and software inventory, checking for free disk space, ensuring that service packs and hot fixes have been installed, etc.. At other times, however, logon scripts might make configuration changes on a computer: disabling services, changing registry settings, configuring NTFS permissions. If your script makes configuration changes, it is a good idea to have scripts available that can undo those changes. For example, suppose your logon script changes NTFS permissions. No sooner does it do so than your help desk is deluged with calls from users no longer able to access a particular resource. Support personnel should have a script that can quickly and easily reset those permissions.

Admittedly, this requires you to, in effect, create two logon scripts: one to make changes, the other to reverse those changes. However, the extra work required to create this undo script is invaluable insurance should you ever need to reset a change made by your logon script.

Testing Logon Scripts on All Relevant Platforms

In at least one respect logon scripts are more "dangerous" than many other system administration scripts. That's because the same logon script is typically run by a large number of users; in fact, it is not unusual for all the users in a domain to run the same logon script.

There's nothing wrong with that; in fact, this make it easy for you to enforce organization policies across the entire enterprise. The one potential problem is the fact that any mistake you make will be multiplied by the number of users who run the script; in theory, a poorly-written logon script could result in a scenario where no one in the entire domain can log on to the network.

This emphasizes the importance of thoroughly testing logon scripts before they are deployed in your actual production environment. It is always a good idea to test *any* script before deploying it; it is a *really* good idea to test a script that might impact hundreds or even thousands of users.

It is equally important to test the script on all your client platforms. As noted previously, the versions of WSH, WMI, and ADSI are not identical on all Windows platforms. Because of this, you cannot assume that a script that runs without problems on a Windows XP computer will also run without problems on a Windows 2000 or Windows 98 computer. Test your logon scripts on all platforms before deploying them.

Deploying New Logon Scripts in Phases

Testing, no matter how thorough, can never uncover all the potential bugs in a script; there are too many extraneous variables to reproduce in the lab. Therefore, even after your logon script has been successfully deployed in a lab setting, it is a good idea not to immediately implement it throughout the organization. Instead, you might set aside an OU of more advanced users willing to serve as beta testers for new logon scripts. (Why more advanced users? Because they are less likely to be frustrated by any problem that might arise and more apt to work around any problem.) Assign the new logon script to that OU, and give the beta

testers exclusive use of the script for a few days. Assuming they report no problems, you might then apply the script to another OU, perhaps one with relatively few users. Again, wait a day or two to ensure that no problems or complaints arise, and *then* apply the script on a widespread basis.

Encoding Logon Scripts

One of the golden rules of scripting is that you should never include confidential information (such as the Administrator password) in a script. This is especially true in a logon script. A user cannot run a logon script without having Read and Read & Execute access to the folder where the script is stored. By definition, users can open and read (though not modify) any logon script assigned to them. Placing confidential information inside a logon script is simply asking for trouble.

If there is no confidential information included in a logon script, you might wonder why you should encode these scripts. As a child, you might have written secret messages that used a code such as this:

A = 1

C = 2

T = 20

Thus the word "cat" is written like this:

1 2 20

When you encode a script you use a similar and more sophisticated approach, replacing the characters in the script code with other characters, much like replacing the word "cat" with the seemingly-meaningless *1 2 20*, yet it can be easily translated by anyone who knows the code, including the script engine..

So does it really matter if users know that you are mapping network drives for them? Probably not. However, some users resent having their computers (and, in a sense, their activities) managed; this is especially true in organizations where users are local administrators and thus expect to be given free rein to manage their systems as they see fit. If you have users who balk at being managed, you might consider encoding your scripts. If users know the settings you are configuring, they can figure out how to unconfigure those same settings. As a result, they might undo any changes made by the logon script.

> **Note** If your users are local administrators they can undo those configuration changes. However, if they do not know what changes were made, it is less likely they can undo those changes. You might configure a particular setting that users want to undo, but they do not know how to do this. If they see code in your logon script that sets a specific registry value to 1, however, they have a clue that setting that same value to 0 might defeat your management efforts.

One easy way to disguise your code is to run your finished script through the Microsoft Script Encoder. The Script Encoder takes a standard .vbs file and creates a special .vbe file that, although it still carries out the functions of the original script, contains no readily-recognizable code. You can download the Microsoft Script Encoder.

For example, here is a simple script that sets the value of two variables, adds the two variables, and then displays the resulting sum:

```
1  intFirstNumber = 2
2  intSecondNumber = 3
3  intSum = intFirstNumber + intSecondNumber
4  Wscript.Echo intSum
```

Here is the same script after being run through the Script Encoder:

```
1  #@~^cQAAAA==r   YobDkY1!:(nD,'~
2  @#@&rUD?+1Gx91Es8nD,',&@#@&rUD?;:,xPbxOobD/OH!:8+M~3Pr
3     YU+mKx9HEs4nD@#@&   d1DkaORAmtK~rxD?!:@#@&9CMAAA==^#~@
```

Although the Script Encoder is effective, it is important to note that the script is *encoded* rather than encrypted. The Script Encoder uses a relatively simple algorithm that a determined attacker can defeat; in fact, there are utilities available on the Internet that can decode a .vbe file. The Script Encoder keeps most users from being able to read script code, but it cannot keep *all* users from being able to read the script code. Encoding does not equal security; running the Script Encoder does not suddenly give you license to do things like include the administrator password in your scripts.

Using the Microsoft Script Encoder

To encode a script using the Microsoft Script Encoder, run the Script Encoder executable (Screnc.exe) from the

command prompt, supplying two parameters:

- The name of the script being encoded.
- The name you want to give the encoded script.

Suppose you have a script named Logon.vbs, and you want to create an encoded version named Encoded_logon.vbs. In that case, you type this at the command prompt:

```
screnc logon.vbs encoded_logon.vbe
```

There are two things to keep in mind here. First, you must give the encoded script a .vbe file name extension, otherwise the script will not execute. Second, after running the preceding command, you will be left with two files: the original script (Logon.vbs) and the encoded script (Encoded_logon.vbe).

> **Important** *Do not throw away the original script*. You cannot make any changes to an encoded script; changing even a single character results in a script that will not run. Always keep a copy of the original script, and make any needed revisions there. After making those changes, you can then re-run the Script Encoder and create a new version of Encoded_logon.vbe.

Recording Logon Script Data

One very important aspect of logon scripts is recording any data collected by that script because there is little reason to take a hardware inventory if you do not save that information for future use. Unfortunately, techniques used in recording the large amounts of data typically gathered by a logon script lie outside the scope of this paper. The recommended technique (saving data in XML format) requires a separate whitepaper of its own. Therefore, this section of this paper only discusses the pros and cons of three different techniques for recording data. For sample code for saving data as an XML file, see Appendix B in this paper.

Having the Data Recorded in a Database

If you have several computers, you will want the data collected by your logon scripts to be stored in a database. That is the only way you can analyze and make use of that data. That means you can have your logon scripts do one of two things: Record data directly to the database. Or store data in an intermediate format, such as a text file or XML file, and then you can use an automated process to pull that data into the database.

Having the logon script write data directly to the database eliminates the intermediate step. It might also be the best approach depending on how many users you have and when those users typically log on. Databases are designed to handle enormous amounts of data, but not necessarily hundreds of simultaneous connections. This can be a problem in organizations with hundreds of users who start work at 8:00 AM and promptly log on to the network. If users have more flexible hours, thus dispersing logons more equally throughout the day, you might be able to have your logon scripts write directly to a database. If not, a better approach might be to save the data to an XML, then later use a script or other automated procedure to collect that data and add it to the database.

For more information about using scripts to read from and write to a database, see the TechNet Webcast "Database Scripting for System Administrators".

Having the Data Recorded in a Text File

Text files represent a quick and easy way to record data, provided that you are keeping track of a limited amount of data. For example, suppose your logon script only needs to record two things: the name of the computer and the current state of the Messenger service. If that is the case, a simple comma-separated-values file such as this will suffice:

```
atl-ws-01,Running
```

With very small amounts of data, text files are both easy to write to and easy to read from.

However, suppose you want record the state of all the services on a computer. On a typical Windows XP computer, you are likely to have 80 or more services running. Furthermore, the number of services varies from computer to computer depending on the software installed. Consequently, a text file such as this is meaningless:

```
atl-ws-01,Running,Running,Running,Stopped,Running
```

Instead, at a minimum you need a text file similar to this:

```
atl-ws-01,Messenger,Running,Alerter,Running,Browser,Running,
```

Although data it is not difficult to record in this fashion, parsing the data can be challenging, especially if you want to record more information than just service states. For that reason, text files are not recommended unless you keep a bare minimum of data. If you only need to record the computer name, the user name, and the date and time that the logon script ran, then a text file will suffice. Anything beyond that bare minimum, however, requires you to either write data directly to a database or to save that data to an XML file.

For more information about using scripts to read from and write to text files, see "Reading and Writing Text Files".

Having the Data Recorded in an XML File

XML (eXtensible Markup Language) is a plain-text format designed specifically to store data. XML is supported by database products such as Microsoft SQL Server. As a result, it represents an excellent intermediate storage place for data generated by a logon script. Data can be saved in an XML file, and then easily imported into and parsed by a database application.

Because XML is a plain-text format, it is easy to save data as an XML document as it is to save that data as a comma-separated-values file. For example, here is a very simple XML document that shows information about the Automatic Updates service on a computer:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<OPERATING_SYSTEM>
  <SERVICE>
    <NAME>wuauserv</NAME>
    <DISPLAY_NAME>Automatic Updates</DISPLAY_NAME>
    <STATE>Running</STATE>
  </SERVICE>
</OPERATING_SYSTEM>
```

XML documents such as this can easily be extended not only to store data for all the services on a computer, but also to store data about other items. A single XML document can contain data about services, such as hardware, software, and anything else you want to keep track of. This makes XML the preferred format for temporary data storage: a logon script can gather information and store that data in a temporary file. You can then write a script or use another type of automated procedure to read these temporary files and add the data to the appropriate table in a database.

For more information about using scripts to read from and write to an XML file, see "Understanding XML".

Continue to Appendices
