Using VBScript to Automate User and Group Administration

Exam Objectives in this Chapter:

- Create and manage groups
- Create and modify groups by using automation
- Create and manage user accounts
- Create and modify user accounts by using automation

In Chapters 3 and 4, you learned how to create and manage user and group accounts using the Active Directory Users and Computers snap-in, directory service command-line commands, CSVDE, and LDIFDE. These skills and concepts will be tremendously valuable on a day-to-day basis. Occasionally, however, you will be faced with a task for which those techniques might be insufficient. For example, you might wish to modify properties of a user object that are not accessible through DSMOD parameters.

The Windows Scripting Host (WSH) enables automation of both simple and sophisticated administrative tasks. It interprets scripts much like the command shell (cmd.exe) interprets batch files. WSH supports scripting in VBScript: a relatively straightforward scripting language that you can use to perform tasks on almost every component of Windows Server 2003, including Active Directory directory service.

Although a thorough understanding of VBScript and its capabilities would be a tremendous benefit to you as a system administrator, that level of detail is beyond the scope of the 70-290 exam and of this book. You do need to be able to recognize scripts that can perform useful, day-to-day user and group management tasks. We will go slightly farther than that in our discussion, as we introduce you to scripting terminology and concepts that will help you understand exactly what the scripts are doing. Do your best to analyze and understand the way the scripts are working, but take comfort that the examination will expect you only to be able to recognize what a script actually does.

VBScript files are typically named with a .vbs extension. If you open a .vbs file using Windows Explorer or from the command prompt using either Cscript.exe or Wscript.exe, WSH interprets the script and performs each command in the script.

Creating a script in Notepad

To create a script, simply open Notepad, type the script, and then save the script with a .vbs extension. Here's a tip: when you choose the Save command in Notepad, surround the filename with quotation marks and include the extension, such as: "Myscript.vbs." Without the quotation marks, Notepad may add a .txt extension, resulting in a filename such as "Myscript.vbs.txt," which will not open in WSH.

Creating Users and Groups with VBScript

Let's start our examination of Active Directory scripts and VBScript by looking at two functioning scripts. The first creates a user account in the Management OU of the contoso.com domain. The second creates a global group account. Notice their similarities.

Create a user account

```
Set objOU = GetObject("LDAP://OU=Management,dc=contoso,dc=com")
Set objUser = objOU.Create("User", "cn=Dan Holme")
objUser.Put "sAMAccountName", "DanHolme"
objUser.SetInfo
```

Create a global group account

```
Set objOU = GetObject("LDAP://OU=Management,dc=contoso,dc=com")
Set objGroup = objOU.Create("Group", "cn=phx-users")
objGroup.Put "sAMAccountName", "phx-users"
objGroup.SetInfo
```

VBScript is referred to as an object-oriented scripting language. Objects are virtual representations of a component of the system. Instead of performing an action directly on the system, you perform the action on the object that represents the system. Instead of setting or querying a property directly from the system, you get or set the property of the object that represents the system.

For example, you'll notice that the first line of each script is the same:

```
Set objOU = GetObject("LDAP://OU=Management,dc=contoso,dc=com")
```

The line is creating what is called an *object reference:* it is creating the virtual representation of a system component as just mentioned. In this case, an object called objOU is being configured (Set) to represent (=) the Management OU in the contoso.com domain. In order to point objOU to the Management OU, a *method* called GetObject queries Active Directory using Lightweight Directory Access Protocol (LDAP) and the distinguished name (DN)of the Management OU. You can safely think of methods as commands or actions. The GetObject method is a native method for WSH and is used regularly to find a specific system component so that an object can be set to refer to that component.

The object name, in this case objOU, is irrelevant—you could call it almost anything you want, such as "Bob," as long as you don't use a name that is also used as a command in VBScript. That's a reasonable limitation that allows you to name things just about any way you want. It is a convention, rather than a requirement, that objects begin with a prefix such as "obj" or "o" to indicate the type of named element.

Each object, once created, supports methods specific to the type of system component that the object represents. As you know, you can create users, groups, or any number of other Active Directory objects within an OU. Therefore, not surprisingly, the reference objOU that represents an OU supports a method to create objects. That method is creatively named Create. Notice the similarity between the two scripts:

```
Set objUser = objOU.Create("User", "cn=Dan Holme")
Set objGroup = objOU.Create("Group", "cn=phx-users")
```

In both scripts a new object reference is being set. Again, the names of the objects (objUser and objGroup) are irrelevant, but the convention is to prefix the name with "obj" and then to use an identifiable name such as "User" or "Group." This time the object references are representing something that hasn't existed yet: we are creating a new object within the Management OU, represented by objOU. As mentioned above, objects referring to OUs support a Create method. We simply have to specify what type of object we are creating and what its common name (CN) will be.

Using the Create method does not write any data to Active Directory. Why? Because object references are virtual representations—everything is happening outside of the actual system component (the OU) so far. And that's a good thing, because at this point in the scripts, the objects would fail to be created anyway because mandatory properties of the objects have not yet been configured.

The CN of an object is a mandatory property and is specified when you use the Create method. Some mandatory properties of users and groups are configured automatically by the system. For example, the security identifier (SID) of the object is created automatically. There is one other mandatory property of user and group objects that must be specified before the object can be created, and that is its SAM Account Name. As you learned in Chapter 3, the SAM Account Name is the downlevel account name for users, groups, and computers, used for compatibility with pre–Windows 2000 operating systems.

The third lines of the scripts configure the SAM Account Names:

```
objUser.Put "sAMAccountName", "DanHolme"
objGroup.Put "sAMAccountName", "phx-users"
```

Now that the object references to the new objects have been created (objUser and objGroup), those objects allow us to set or get properties of the objects, using the Put and Get methods, respectively. The Put method is followed by the LDAP attribute name (sAMAccountName) and the value that we want the attribute to take on (Dan Holme and phx-users).

Again, all of this is happening “virtually” in objects that exist in the computer’s memory only. The final step in a script to create a user or group is to write the information to Active Directory, which is achieved using the user or group object’s SetInfo method:

```
objUser.SetInfo
objGroup.SetInfo
```

Voílà! A user and group are created in Active Directory! You might think that this is a lot of work, compared to right-clicking and choosing New User in the Active Directory Users and Computers snap-in. However, with a little work, you can write a script that pulls information from a database and creates accounts based on that information. As we stated at the beginning of the chapter, with each task you perform as an administrator you should evaluate the time required to do the task manually and accurately against the time it takes to create and test the functionality and accuracy of a script.

Managing Users and Groups with VBScript

The remainder of this chapter will present sample scripts that perform routine administrative tasks for user and group accounts. We will touch on the “behind the scenes” workings of the script, but it is only necessary that you are able to recognize what the scripts achieve.

Add a user to a group

```
Set objGroup = GetObject _
   ("LDAP://cn=phx-users,OU=Management,dc=contoso,dc=com")
objGroup.Add "LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com"
```

This script adds the user we created (Dan Holme) to the group we created (phx-users). To read the script, you must know that the underscore character (“_”) used at the end of a line indicates that the command is continued on the following line. It is a continuation marker. So the first two lines of the script are actually one command that creates an object that refers to the phx-users group. The second command, on the last line, adds the user to the group. That action is achieved by using the Add method supported by objects that refer to groups and pointing to the DN of the user account.

Can you guess how we would remove a user from a group? The exact same script, with the Remove method of the group object, achieves that task.

Scripting Resources

To learn more about scripting, objects, methods, and properties, visit *http://www.microsoft.com/technet/scriptcenter*, or for even more detail, explore *http://msdn.microsoft.com*.

Remove a user from a group

```
Set objGroup = GetObject _
   ("LDAP://cn=phx-users,OU=Management,dc=contoso,dc=com")
objGroup.Remove "LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com"
```

After you've created a user or group object, you will want to be able to configure properties of the object. There are two primary formats for doing this. For most properties the syntax is:

```
objUser.Put "property name", "property value"
```

For example, the following script sets a number of common user properties:

Configure common user properties

```
Set objUser = GetObject _
   ("LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com")
objUser.Put "givenName", "Dan"
objUser.Put "initials", "F."
objUser.Put "sn", "Holme"
objUser.Put "userPrincipalName", "DanHolme@contoso.com"
objUser.Put "displayName", "Dan Holme"
objUser.Put "title", "Marketing Manager"
objUser.Put "profilePath", "\\phxsrv01\Profiles\DanHolme"
objUser.Put "homeDirectory", "\\phxsrv01\HomeFolders\DanHolme"
objUser.Put "homeDrive", "H:"
objUser.Put "scriptPath", "logon.vbs"
objUser.Put "telephoneNumber", "(425) 555-1211"
objUser.Put "mail", "DanHolme@contoso.com"
objUser.Put "wWWHomePage", "http://www.contoso.com"
objUser.Put "physicalDeliveryOfficeName", "Room 4358"
objUser.SetInfo
```

Some user properties are stored in arrays. An array is a structure that holds more than a single value. For example, a user can have several telephone numbers, so the otherTelephone attribute is declared to be multivalued, or an array. To store or modify values in an array, you use the PutEx, or "put extended" method of the user object. PutEx can use a total of four operations. These operations are clear, update, append, or delete, and each are declared by using either their individually assigned value or constant—a name for a value. For example, to replace the value of a specified attribute, PutEx is followed by the assigned number 2 or by the constant ADS_PROPERTY_UPDATE, which indicates an update, followed by the property name and the value(s) passed in an array. The following script sets the PutEx update constant called ADS_PROPERTY_UPDATE . The PutEx constant is used in the script instead of the actual number 2. This is purely convention. The theory is that it is easier to understand what the script is doing when you read a name like "ADS_PROPERTY_UPDATE" than it is to see the number 2 and be unsure what that number indicates. (For an explanation of PutEx operations, refer to "Reading and Writing User Account Attributes" in the online "Windows 2000 Scripting Guide" at *http://www.microsoft.com/technet/script–center/guide/sas_usr_nehi.mspx*).

Configure common user properties

```
Const ADS_PROPERTY_UPDATE = 2
Set objUser = GetObject _
   ("LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com")
objUser.PutEx ADS_PROPERTY_UPDATE, _
"description", Array("Management staff")
objUser.PutEx ADS_PROPERTY_UPDATE, _
"otherTelephone", Array("(800) 555-1212", "(425) 555-1213")
objUser.PutEx ADS_PROPERTY_UPDATE, _
"url", Array("http://www.contoso.com/management")
objUser.SetInfo
```

To set a user's password, use the SetPassword method of the user object. Note that unlike setting other properties of a user, you do not use SetInfo to write the password to Active Directory: password changes are written directly to the directory by the SetInfo method.

Set a user's password

```
Set objUser = GetObject _
   ("LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com")
objUser.SetPassword "i5A2sj*!"
```

If you create a user account with a script, the user account will be disabled. After configuring appropriate logon credentials (the sAMAccountName, userPrincipalName, and password), you need to enable the account using the following script. Note that the object must exist in the directory—it must have been written to Active Directory with its first SetInfo method—before it can be enabled or disabled.

Enable a user account

```
Set objUser = GetObject _
   ("LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com")
objUser.AccountDisabled = FALSE
objUser.SetInfo
```

The final scripts we'll present involve listing the members of a group and listing the groups to which a user belongs. These scripts involve more complex structures, including loops that repeat multiple times using a syntax that begins with a For statement and ends with a Next statement. They also involve displaying information using the WScript.Echo statement. The details of the scripts are beyond the requirements of the exam and the scope of this book, but you should look at the scripts and get a general feeling for what they accomplish. Each script is documented with comments that begin with an apostrophe (" ' ").

List the members of a group

```
' This script lists the members of a group
' Note that it does not look for multiple levels of nesting
' It is only listing the direct membership
'
```

```
' Create an object representing the desired group
Set objGroup = GetObject _
   ("LDAP://cn=phx-users,OU=Management,dc=contoso,dc=com")
' Loop through each of the group's Members attribute
For each objMember in objGroup.Members
   ' Display the member
   Wscript.Echo objMember.Name
Next
```

List the local groups to which a user belongs

```
' This script looks through the local groups on a computer
' and displays the names of groups to which a specified user
' belongs
'
' The following line sets a string variable to represent the
' name of the computer on which the script will run.
' Substitute a period "." for the computer name and the script
' will act on the local computer on which the script is executed
strComputer = "phxdesktop01"
' The following line sets a string variable to represent the
' name of the user we are looking for in local groups
strUserName = "danholme"
' The next two lines query the computer's *local* directory (SAM)
' and filter out all objects except groups
Set colGroups = GetObject("WinNT://" & strComputer & "")
colGroups.Filter = Array("group")
' The following line loops through each group that was found
' on the system
For Each objGroup In colGroups
    ' The following loops through each user belonging
    ' to the group and checks to see if the user name is
    ' the one we're looking for
    For Each objUser in objGroup.Members
        If objUser.name = strUserName Then
            ' If we've found the user in the group, display
            ' the group name
            Wscript.Echo objGroup.Name
        End If
    ' Loop to the next user in the group
    Next
' Loop to the next group on the computer
Next
```

List the groups to which a user belongs

```
' This script shows the groups to which a user belongs
' Note that it does not check for multiple levels of nesting
' It is only looking for groups to which the user *directly*
' belongs. It also does not show the user's Primary Group.
' See the next script for an example of how to show the
' primary group.
'
' Create an object referencing the specified user
Set objUser = GetObject _
   ("LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com")
```

```
' Create an object representing the MemberOf attribute of the user
objMemberOf = objUser.GetEx("MemberOf")
' Display an introductory message
WScript.Echo VbCrLf & "The user is a member of:"
' Loop through each group that is listed in the MemberOf attribute
For Each Group in objMemberOf
   ' Display that group
   Wscript.Echo Group
Next
```

List the groups to which a user belongs and the user's primary group

```
' This script is more complex.
' It lists all of the groups in Active Directory to which a user
' belongs directly (including the primary group) but does not
' show groups to which the user belongs indirectly through
' group nesting
'
' Indicate that we want to skip any errors
On Error Resume Next
' Create a constant representing a common error value
Const E_ADS_PROPERTY_NOT_FOUND = &h8000500D
' Create an object referencing the desired user
Set objUser = GetObject _
    ("LDAP://cn=Dan Holme,OU=Management,dc=contoso,dc=com")
' Retrieve the list of groups to which the user directly belongs
arrMemberOf = objUser.GetEx("memberOf")
' In case an error was returned,
' we evaluate whether the error returned
' indicates that the user belongs to no groups
If Err.Number = E_ADS_PROPERTY_NOT_FOUND Then
    WScript.Echo "The memberOf attribute is not set."
' otherwise, we list the groups that the user belongs to
Else
    WScript.Echo "Member of: "
    ' Loop through each group listed in the MemberOf attribute
    ' and display the group name
    For each Group in arrMemberOf
        WScript.Echo Group
    Next
    Err.Clear
End If

' The user's primary group (usually Domain Users)
' is not included in the MemberOf attribute
' These lines find the primary group, which is indicated by
' an integer in the primaryGroupID attribute of the user,
' then take that integer and perform an Active Directory
' database query to locate the name that corresponds to that
' primary group ID

' Retrieve the user's Primary Group attribute
intPrimaryGroupID = objUser.Get("primaryGroupID")
Set objConnection = CreateObject("ADODB.Connection")
objConnection.Open "Provider=ADsDSOObject;"
Set objCommand = CreateObject("ADODB.Command")
```

```
objCommand.ActiveConnection = objConnection
objCommand.CommandText = _
  "<LDAP://dc=contoso,dc=com>;(objectCategory=Group);" & _
  "distinguishedName,primaryGroupToken;subtree"
Set objRecordSet = objCommand.Execute

While Not objRecordset.EOF
    If objRecordset.Fields("primaryGroupToken") = intPrimaryGroupID Then
        WScript.Echo "Primary group:"
        WScript.Echo objRecordset.Fields("distinguishedName") & _
            " (primaryGroupID: " & intPrimaryGroupID & ")"
    End If
    objRecordset.MoveNext
Wend
objConnection.Close
```

As you can see from the last two examples, there is often more than one way to script a particular task. Scripting simple tasks is straightforward, and you should make every effort to use scripts and increase your comfort level and familiarity with scripting. You'll find that soon even more complex scripts are understandable. The best news of all is that Microsoft TechNet and MSDN provide numerous examples of scripts, and in most cases you need only to modify a few pieces of a sample script to make it work in your environment.

Exam Highlights

Key Points

Before taking the exam, review the key points and terms that are presented below to help you identify topics you need to review. Return to the lessons for additional practice and for pointers to more information about topics covered by the exam objectives.

- Windows Script Host (WSH) supports VBScript files. VBScript is a relatively straightforward scripting language that enables you to script just about any task on a Windows Server 2003 system.

Key Terms

method An action or command that is performed on an object. The methods that are available for an object depend on the system component to which the object refers. For example, an object referring to an OU exposes the Create method to create new objects in the OU. An object referring to a group exposes the Add method to add members.

object reference A virtual representation of a particular system component. In scripting, object references are created and then the object can be used to expose methods and properties.