## Managing Active Directory at the command line

# EX-DIRECTORY

Professionally manage Active Directory users, groups, and other AD objects with Windows PowerShell Version 2.0. **BY ROLF MASUCH**

The introduction of PowerShell [3] sees Microsoft finally implement long overdue changes to Windows scripting. Unfortunately, PowerShell Version 1 was no big help for Active Directory (AD) administrators, but then Microsoft added PowerShell support to its "Common Engineering Criteria" (CEC), making it mandatory for all Redmond products to support it. This means Windows Server 2008 R2 and Windows 7 now have PowerShell Version 2 hard-wired.

## Revamped PowerShell for Active Directory

Microsoft developers have enhanced Active Directory support with a PowerShell provider and 76 additional cmdlets (say "commandlets"). The provider lets you change to an AD like changing a drive by typing *cd ad:*. The cmdlets are an additional, optional instruction set that you can use to manipulate individual Active Directory objects.

Although a script-based approach is possibly overkill for occasional work with individual objects, it becomes essential if you need to handle several dozen or even thousands of objects. The PowerShell is a .Net implementation and thus object oriented throughout; objects can be processed in a structured manner through the use of pipes, removing the need to search text-based output for specific content.

Administrators can manipulate the object properties and methods directly. The introduction of this powerful scripting language is an important milestone for the management of Windows systems, whether clients or servers, and the applications installed on them.

## Requirements

Not every Active Directory will support the use of PowerShell. At least one of the domain controllers (DCs) in the AD has to run Windows Server 2008 R2. For "legacy" systems based on Windows Server 2003, Windows Server 2003 R2, and Windows Server 2008, the Active Directory Management Gateway Service [4] is available as a separate download.

After the installation, the Active Directory Web Services (ADWS) run on the domain controller. To support ADWS, you need to allow TCP port 9389 on the domain controller running the service. If you use a group policy object to configure your firewall, you need to modify your group policies to allow this port for ADWS. The beta versions of the Gateway Service used a different port. Windows PowerShell Version 2 must be installed.

If you have Windows Server 2008 R2 fulfilling the *Active Directory Domain Services* role, the PowerShell module for Active Directory is preinstalled. This module is not installed on any other systems. Instead, the Remote Server Administration Tools (RSAT) module is used [5] [6].

The version for Windows 7 not only installs a full set of tools for GUI-based administration, but also a full set of PowerShell modules. Again, you need PowerShell v2 installed.

The minimum requirement for installing PowerShell is the .NET Framework 2.0. The PowerShell Integrated Scripting Environment (ISE) requires version 3.51 of the frameworks. ISE is the internal editor for PowerShell. All you need now is a user account with the required privileges for the management work you

want to perform, and you are up and running.

## Accessing Active Directory

To manipulate user-type objects with PowerShell, you need a connection to your Active Directory and the *ActiveDirectory* PowerShell. Via the Start menu or ISE, launch a PowerShell console. To load the module with the *Import-Module* cmdlet, use:

```
Import-Module ActiveDirectory
```

Make sure *ActiveDirectory* has no embedded blanks. If your computer is connected to your Active Directory, the system will display a new drive labeled *ad:*. If the connection fails, a warning displays. PowerShell displays warnings in orange and errors in red.

If you want to connect to this drive, or to another domain, see Listing 1. The *Get-Credential* cmdlet retrieves the credentials used to establish the connection and stores it in the *$cred* variable. The *New-PSDrive* cmdlet establishes the connection with the specified parameters. The *PSProvider* switch specifies the provider on which to establish the drive. The *Name* switch defines the name used to reference the drive later. The default here is *AD*, but you can change this to suit your own needs. The next two switches *Root* and *Server* define the server you want to connect to and the mountpoint, but more of that later. Finally, the *Credential* keyword passes in the login information. Without these details, PowerShell uses the data for the currently logged on user.

Because Active Directory is an LDAP directory, PowerShell also supports the functions and requirements of the LDAP protocol. When you establish a connection, it displays all the partitions at the topmost level, root. In the example here, running the *dir* command gives you the output in Listing 2.

To work with the typical tree structure, you can use the *cd* command to change to the domain partition. Alternatively, you can specify the required partition directly with the *Root* parameter. This then is your mountpoint. Before doing this, you might want to change the drive. When you specify the partition name, you must use X.500 notation. The use of canonical names, such as *woodgrovebank.com*, is not currently supported. The command

```
cd ad:
cd "DC=WoodgroveBank,DC=com"
```

changes to the required partition.

## Working with User Accounts

The *ActiveDirectory* module includes 76 cmdlets for Active Directory. To discover the cmdlet useful for working with user accounts, ask PowerShell for help. The *Get-Command* cmdlet supports a wildcard search and the *Type* parameter to provide a filter function:

```
get-command *user* -type cmdlet | ➥
            format-table -autosize
```

This query results in five cmdlets (Listing 3).

With the exception of the *Get-ADUserResultantPasswordPolicy* cmdlet, which performs a resulting set of policy (RSOP) query against the password policies for a user, you can use all of these cmdlets for manipulating user objects. For example, the *New-ADUser* cmdlet adds a new user. To change user properties, you would choose *Set-ADUser*. The *New-ADUser* command creates a new user account for Mark Eting in the Marketing organizational unit (Listing 4). The *Name* option passes in the value for a user's *CommonName* (CN). The display name is specified by the *DisplayName* switch. To manipulate the user object, you need to run the *Get-ADUser* cmdlet and store the object in a variable:

```
$User = Get-ADUser MarkEting ➥
        -Properties *
```

Then you can use the object's individual properties to access the values. For example, *$User.PasswordExpired* checks to see whether a user's password has expired. User accounts can be deleted either by passing in their unique names or by piping. The *Confirm* option sup-

### Listing 1: Access Active Directory

```
01 $cred = Get-Credential
02 New-PSDrive -PSProvider ActiveDirectory -Name AD -Root ""
               -Server "nyc-dc1.woodgrovebank.com" -Credential $cred
```

### Listing 2: AD LDAP Information

```
01 Name            ObjectClass    DistinguishedName
02 ----            -----------    -----------------
03 WoodgroveBank   domainDNS      DC=WoodgroveBank,DC=com
04 Configuration   configuration  CN=Configuration,DC=WoodgroveBank,...
05 Schema          dMD            CN=Schema,CN=Configuration,DC=Wood...
06 DomainDnsZones  domainDNS      DC=DomainDnsZones,DC=WoodgroveBank...
07 ForestDnsZones  domainDNS      DC=ForestDnsZones,DC=WoodgroveBank...
```

### Listing 3: Filtering Commands for *user*

```
01 CommandType     Name                          Definition
02 -----------     ----                          ----------
03 Cmdlet          Get-ADUser                    Get-ADUser -Filter <String> ...
04 Cmdlet          Get-ADUserResultantPasswordP... Get-ADUserResultantPasswordP...
05 Cmdlet          New-ADUser                    New-ADUser [-Name] <String> ...
06 Cmdlet          Remove-ADUser                 Remove-ADUser [-Identity] <A...
07 Cmdlet          Set-ADUser                    Set-ADUser [-Identity] <ADUs...
```

### Listing 4: Create a New User

```
01 New-ADUser -Name "Mark Eting" -SamAccountName "MarkEting" -GivenName "Mark"
              -Surname "Eting" -DisplayName "Mark Eting"
              -Path 'OU=Marketing,OU=NYC,DC=WoodgroveBank,DC=com'
              -OtherAttributes @{'msDS-PhoneticDisplayName'="EtingMark"}
```

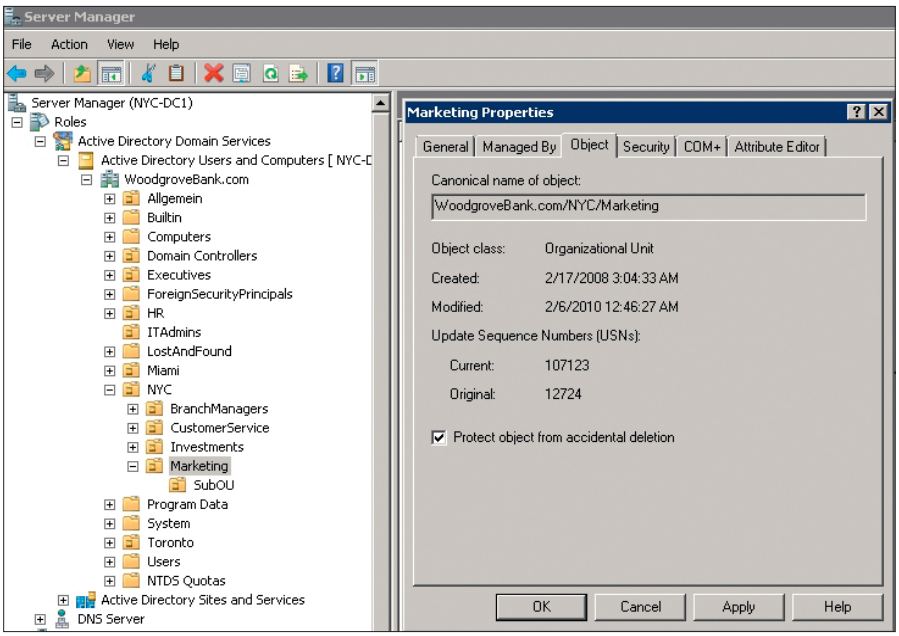


**Figure 1: In the MMC user interface, you can disable the delete request.**

presses the confirm dialog for the deletion.

```
Remove-ADUser MarkEting ➥
   -Confirm:$false
Get-ADUser MarkEting | Remove-ADUser ➥
   -Confirm:$false
```

## Working with Groups

Just like the cmdlets for manipulating user accounts, it makes sense to discover the cmdlets for working with Active Directory groups. Again the *Get-Command* is a useful aid, but this time, use the *Module* option to restrict the search to cmdlets from this module. The results are more extensive than for user accounts (Listing 5).

The following command lists the members of the *NYC_MarketingGG* group by their *CommonName*:

```
Get-ADGroupMember -Identity NYC_➥
    MarketingGG | %{$_.Name}
```

The *New-ADGroup* cmdlet is used to create a new group. Don't forget to specify the *-GroupType*, or you will be confronted with a dialog without options. The supported values for this option are *Global*, *Universal*, and *DomainLocal*.

```
New-ADGroup -GroupScope DomainLocal ➥
            -Name NYC_all
```

Adding new members to existing groups is also really easy. Just grab the required user account with the *Get-ADUser* cmdlet and then pipe it to the *Add-ADGroupMember* cmdlet.

```
Get-ADUser MarkEting | ➥
   %{Add-ADGroupMember ➥
   NYC_MarketingGG $_}
```

It is simpler to specify the username directly in the call to the cmdlet:

```
Add-ADGroupMember NYC_MarketingGG ➥
                  MarkEting
```

The command for removing users from groups is just as simple:

```
Remove-ADGroupMember NYC_MarketingGG ➥
   MarkEting -confirm:$false
```

As with deleting a user, *Confirm* suppresses the confirmation dialog.

## Managing Other Objects

Active Directory gives you many other object types besides users and groups; for example Computers, Organizational Units (OUs), Group Policies, and so on. To find the right cmdlets for these ob-

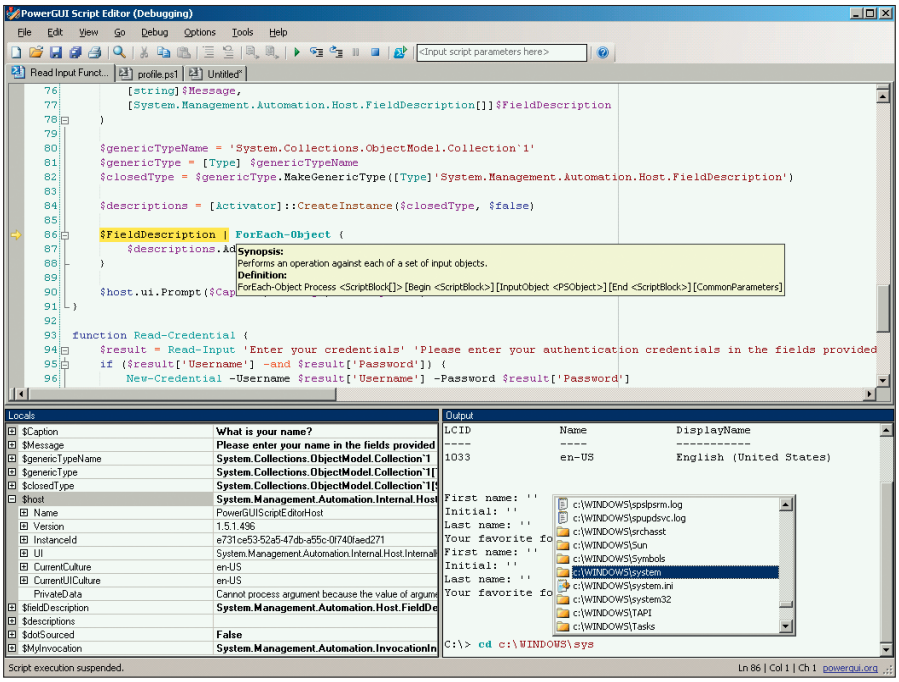


**Figure 2: PowerGUI is one of the free editors with built-in PowerShell support.**

**Listing 5: Commands for *group***

```
01 Get-Command *group* -Module
   ActiveDirectory
02
03 Name
04 ----
05 Add-ADGroupMember
06 Add-ADPrincipalGroupMembership
07 Get-ADAccountAuthorizationGroup
08 Get-ADGroup
09 Get-ADGroupMember
10 Get-ADPrincipalGroupMembership
11 New-ADGroup
12 Remove-ADGroup
13 Remove-ADGroupMember
14 Remove-ADPrincipalGroupMembership
15 Set-ADGroup
```

**THE AUTHOR**

Rolf Masuch works as a product manager with ABSC GmbH. He has been a Microsoft Certified Systems Engineer for Windows Server 2003, Messaging Specialist, and Microsoft Certified Trainer since 1996. He is also a Microsoft Certified IT Professional for Enterprise Desktop Administrator Windows 7 and Enterprise Administrator Windows Server 2008. As the founder of the German-language PowerShell user group D/A/CH, Rolf is in a perfect position to plumb the depths of the command-line tool.

jects, you again need the *Get-Command* cmdlet with the appropriate wildcards. The search keys here are *Computer*, *Org*, *Policy*, or even *Server*. If you don't find a cmdlet for your task, you can use the generic cmdlets for objects. Note that these cmdlets might not have such a rich selection of options as the cmdlets optimized for special object types. The search key for the generic cmdlets is *Object*, of course.

This search also returns the *Restore-ADObject* cmdlet, which restores accidentally deleted objects. With the *Get-ADObject* cmdlet and its *IncludeDeletedObjects* option, you can identify deleted objects and pipe them to *Restore-ADObject*, which restores the objects.

Organizational units have four cmdlets. To create an OU, use the *New-ADOrganizationalUnit* cmdlet with the name of the new OU. *Remove-ADOrganizationalUnit* lets you delete an OU. Watch out for the Active Directory version. If you use Windows Server 2008, the new OU will automatically be flagged *Protect object from accidental deletion*. To enable this feature, you might need to go to the MMC GUI in *Active Directory Users and Computers* in the View menu's *Advanced Features* item. After doing so, you can click the *Object* tab and remove the flag (Figure 1). Until you have entered the following, you will not be able to delete:

```
Remove-ADOrganizationalUnit ➥
   -Identity 'OU=Marketing,OU=NYC,➥
             DC=WoodgroveBank,DC=com'
```

Unfortunately, you can't use the *Confirm* switch here.

## Summary

PowerShell Version 2 and the *ActiveDirectory* module finally bring Active Directory management to the command line. PowerShell's strict object orientation makes it possible to manipulate elements in the Active Directory directly and to use all the methods and properties of the objects in question. Whereas other languages such as VBS previously had to address each individual object separately before manipulating it, a simple *dir* for a user OU in PowerShell will output all the user objects in the OU so that you can pipe them for ongoing processing. The integrated ISE editor is a useful tool to display scripts, your input, and the resulting output.

The RSAT tool lets you perform all of these tasks from a Windows-based client system, in which you can set up your personal scripting environment. This could include more powerful editors such as the free PowerGUI by Quest (Figure 2) [7] or the commercial PowerShell Plus variant by Idera [8] or PrimalScript by Sapien (Figure 3) [9]. At the end of the day, PowerShell helps administrators, especially in larger Active Directory environments with large numbers of objects, manage their Active Directory with less clicking, and the results are documented up front and more easily reproducible thanks to scripting. ■

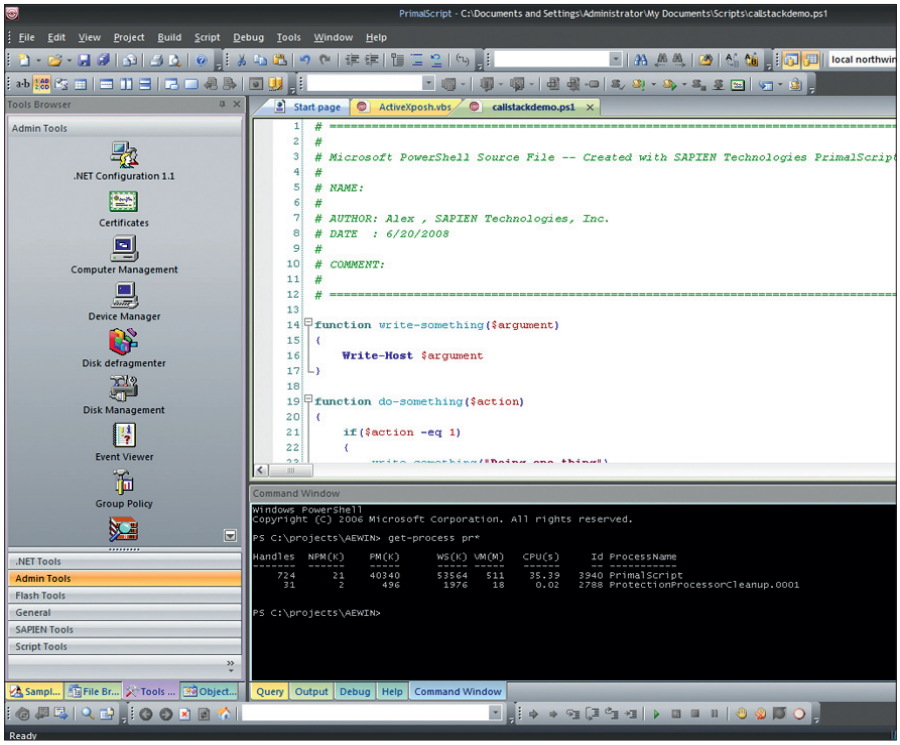


**Figure 3: PrimalScript is a commercial editor with built-in PowerShell support.**

## INFO

[1] PowerShell community: *http://powershellcommunity.org/*

[2] PowerShell user group: *http://www.powershell-ag.de/ps*

[3] Download PowerShell version 2.0: *http://support.microsoft.com/kb/968930*

[4] Active Directory Management Gateway Service (Active Directory Web Service for Windows Server 2003 and Windows Server 2008): *http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=008940c6-0296-4597-be3e-1d24c1cf0dda*

[5] Remote Server Administration Tools for Windows 7: *http://www.microsoft.com/downloads/details.aspx?familyid=7D2F6AD7-656B-4313-A005-4E344E43997D&displaylang=en*

[6] Description of Remote Server Administration Tools for Windows 7: *http://support.microsoft.com/kb/958830/en-us*

[7] PowerGUI: *http://powergui.org/downloads.jspa*

[8] PowerShell Plus: *http://powershellplus.com/*

[9] PrimalScript: *http://www.primaltools.com/products/info.asp?p=PrimalScript*