Learn

ACTIVE DIRECTORY MANAGEMENT

IN A MONTH OF LUNCHES

SAMPLE CHAPTER

RICHARD SIDDAWAY

MANNING

Learn Active Directory Management in a Month of Lunches

by Richard Siddaway

Chapter 6

brief contents

6 Managing computer accounts

If you were asked "What's in your Active Directory?" your instinctive answer would most likely be users and groups. It's what everyone thinks of because that's where the bulk of the work lies. The other big data set is computers.

Every computer that's part of your Active Directory has an account that has to be managed. This chapter will show you how to manage all the computers in your domain with minimal effort—that's not zero effort, just minimizing the effort.

Why do you need computer accounts in Active Directory? Imagine an organization with thousands of users, each with their own desktop computer. This organization also has 2000 to 3000 servers. Now, you could manage the settings on all of those machines manually—good luck. A better and easier approach is to use Group Policy, which we'll cover in chapters 8 and 9. Using Group Policy requires the machines to be in your Active Directory. Applications like Exchange require that the server is in the AD domain.

You already know a lot about managing computers because you learned how to manage users in chapter 2. Computer accounts in Active Directory are specialized variants of user accounts, which means you can apply a lot of your existing knowledge (re-read chapter 2 to refresh your memory if required). Everything else will be covered in this chapter.

Every object in your Active Directory has a lifecycle. I've mentioned this concept a few times, and make no apologies for the fact that I'm going to keep on mentioning it. Working with the lifecycle of AD objects enables you to keep on top of managing the objects, as well as applying some best practices.

Computer objects have a more limited lifecycle than user objects. The creation and deletion of the object are the obvious endpoints, but you won't perform as many updates to the computer accounts as you do to user accounts. The computer needs to join the domain, which may happen with a preexisting account, or the account can be created as the machine joins the domain.

Each computer has a secure channel to the domain controller it authenticates—yes, computers authenticate as well as users. It happens in the background so you don't see it. This channel may, very occasionally, develop problems and need resetting.

As with any object in your Active Directory, the final act is to delete it. Occasionally you may decide to disable a computer account, but it's not an action that I've used very much; however, you'll learn how to do this for completeness.

The first thing is to create some computer accounts.

There are two main ways to deal with creating computer accounts:

- Create the account and then join the computer to the domain.
- Create the account as you join the computer to the domain.

Which way is best? It depends on how you create your computers. If you use an automated system such as Windows Deployment Services, it expects a computer account to exist. On the other hand, if you build your machines from an image or template, then you'll create the account as you join the machine to the domain.

Another advantage to creating the account in Active Directory, known as prestaging, is that the account is in the correct OU, so the machine will receive the correct GPOs as soon as it joins the domain. Also, you can control who can join the machine to the domain.

Let's see how to create a computer account.

6.1 Creating an AD computer account

Creating a computer account in Active Directory is a much simpler proposition than creating a user account, primarily because you don't need to supply as much information. As usual, you'll start by creating a computer account using GUI tools.

6.1.1 Creating a computer account using ADAC

In the examples in this chapter you're going to create the accounts in the Computers container. This is the default location for computer accounts, but you can create an account in any OU.

> **TIP** It's best practice to keep your computer and user accounts in separate OUs to make the application of GPOs easier.

You can create an account using ADAC by following these steps:

1. Open ADAC.
2. Select the container or OU in which you'll create the computer account (in this case the Computers container).

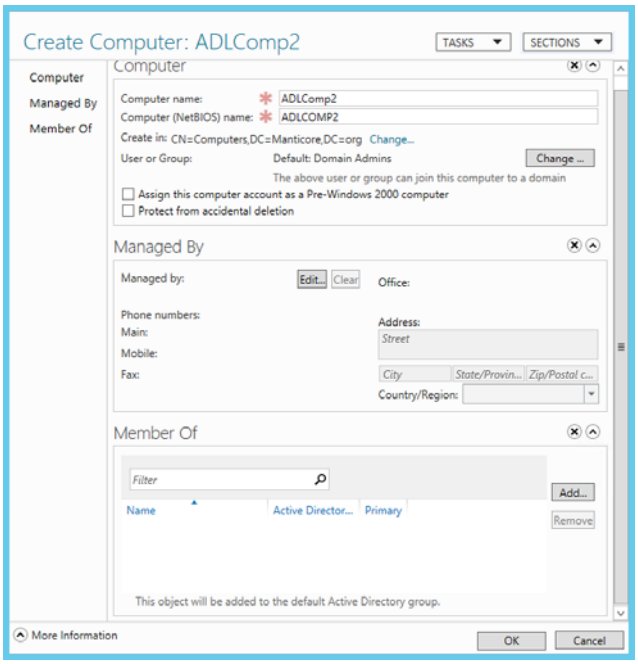

Figure 6.1 Creating a computer account with ADAC. The only mandatory information is the computer name, which auto-populates the NetBIOS field as you enter it. This dialog also provides options for setting Protect from Accidental Deletion, the administrator(s) who will manage the computer, and the groups to which it'll belong.

3. Choose New from the Tasks menu.
4. Choose Computer.
5. The dialog box in figure 6.1 will be displayed.
6. Add the computer name (the NetBIOS field auto-populates as you type).
7. Check Protect from accidental deletion.
8. Click Change to select the users or groups that can join the machine to the domain.
9. Add the computer manager if required.
10. Add the computer to one or more groups if required.
11. Click OK to create the account.

Creating a computer account with ADUC is a similar process.

6.1.2 Creating a computer account using ADUC

ADUC has a slightly different approach to creating computer accounts, and in one important point has an advantage over ADAC in that you can add the computer GUID used by automated deployment systems. You create a computer account in ADUC by following these steps:

1. Open ADUC.
2. Select the container or OU in which you'll create the computer account (in this case the Computers container).

3. Right-click the container name and choose New from the context menu.
4. Choose Computer.
5. The dialog shown in figure 6.2 will be displayed.
6. Add the computer name (the pre-Windows 2000 name will auto-populate).
7. Click Change to modify the user or group that can add the machine to the domain.
8. Click Next.
9. The dialog in figure 6.3 will be displayed.
10. Choose "This is a managed computer" and add the computer GUID as required (only if using automated build systems such as Windows Deployment Services).
11. Click Next.
12. View the summary.
13. Click Finish to complete the creation of the computer object.

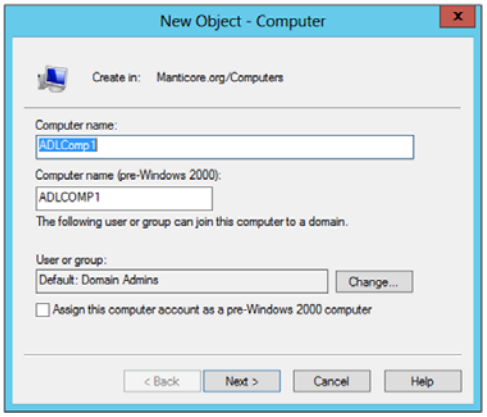

Figure 6.2 Creating a computer account with ADUC. Add the computer name (the pre-Windows 2000 name will auto-populate). Click Next.

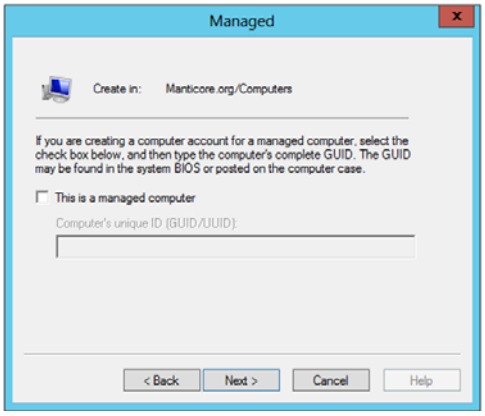

Figure 6.3 Adding a computer's GUID in ADUC. Click Next to continue.

Pre-Windows 2000 computers

In the dialogs for both ADAC and ADUC you'll notice the option to assign this computer account as a pre-Windows 2000 computer.

Ignore it!

This option exists to allow AD to be aware of pre-Windows 2000 systems and interact with the client software that allowed the user (but not the computer) to authenticate against Active Directory. Because those operating systems are all out of support, it's unlikely you'll see them in your environment.

Creating computer accounts with GUI tools is great for the once-in-a-while activity, but if you need to create lots of computer accounts, you need to use PowerShell.

6.1.3 Creating a computer account using PowerShell

One of the great advantages of ADAC is that it's built on top of PowerShell. The version released with Windows Server 2012 shows you the PowerShell commands it used to perform the actions you've initiated in the GUI. This is illustrated in figure 6.4, which shows the PowerShell generated when you created the computer account in figure 6.1.

The code that ADAC generates can be overly verbose. A simplified version would be

```
New-ADComputer -Enabled $true -Name ADLComp3 `
-Path "CN=Computers,DC=Manticore,DC=org" `
-SamAccountName ADLCOMP3
```

You can also set a number of other properties, such as who manages the computer account and the operating system information (completed automatically when the machine joins the domain). PowerShell really comes into its own when you need to create multiple accounts, which you'll see in the lab.

The other way to generate a computer account is to create one as the system joins the domain.

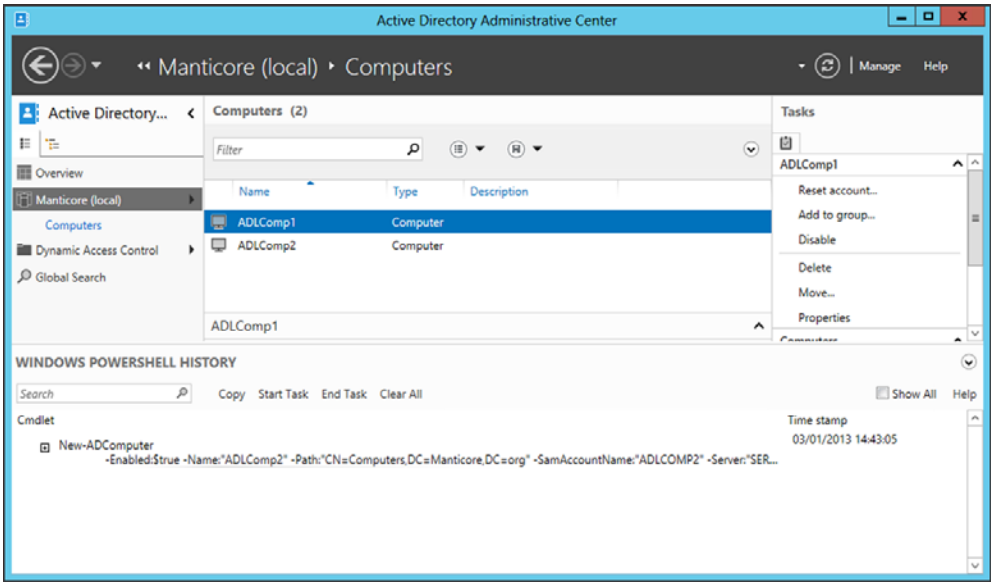

Figure 6.4 PowerShell command to create a computer account. The Windows PowerShell History window is normally minimized. You open it by clicking on the upward-pointing arrow at the bottom-right corner of the ADAC GUI.

6.2 Joining a computer to the domain

By default, any user can join 10 computers to the domain. This is normally thought to be a bad idea, and most organizations use Group Policy to limit who can join a computer to the domain.

A number of prerequisites needs to be in place before you can join the computer to the domain:

- The machine should be configured with the address of at least one DNS server that manages the DNS zone for your Active Directory.
- You need local administrator permissions on the machine.
- You need the account name and password of a user account that has permissions to join the machine to the domain.
- Ideally, the machine should have been renamed to the name you want it to have in Active Directory.

You can join a computer to the domain using a GUI tool or through PowerShell.

6.2.1 Using GUI tools to join a machine to the domain

Using GUI tools is the traditional method of joining a machine to the domain. The procedure involves the following steps:

1. Open Control Panel.
2. Choose System.
3. Choose Advanced System Settings.
4. Choose Computer Name.
5. Choose Change.
6. The dialog shown in figure 6.5 will be displayed.
7. Choose the Domain button and supply a domain name.
8. Click OK.

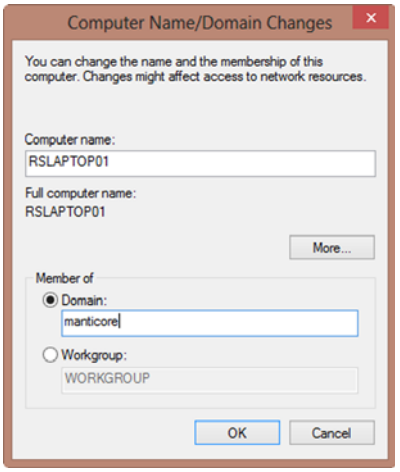

Figure 6.5 Dialog to add a computer to the domain

You'll be prompted for the name and password of an account with permissions to add the computer to the domain. Supply the data and click OK. You'll see a message welcoming you to the domain and prompting you to reboot. The reboot is required. The machine will not function as part of the domain until the reboot has occurred. The computer account will be created in the default location, usually the Computers container.

TIP If you're running a system with PowerShell v3, you can display the Control Panel System applet using this PowerShell command:

```
Show-ControlPanelItem -Name System
```

The alternative approach is to use a PowerShell cmdlet.

6.2.2 Using PowerShell to join a machine to the domain

The following code can be used to join a machine to the domain:

```
$cred = Get-Credential
Add-Computer -Credential $cred -DomainName Manticore `
-OUPath "OU=Security Servers,OU=Servers,DC=Manticore,DC=org" -Force
Restart-Computer
```

The `Get-Credential` cmdlet displays a dialog box for you to enter a user ID and password. Make sure that you enter the user ID in the domain/user format. The password will be masked as you enter it. The `Add-Computer` cmdlet needs the name of the domain and the credential. The `OUPath` is optional but useful, because it controls where the account is created and saves the work of moving it later. The `-Force` parameter suppresses the conformation prompt, which is the best way to proceed in a script. The final command, `Restart-Computer`, performs the reboot.

Once your machine has been added to the domain, it communicates across a secure channel with the domain controller. That channel may need to be tested if you're having issues with the machine.

6.3 Managing the secure channel

All computers (Windows 2000 and later) in an AD domain communicate with a domain controller. They authenticate over that channel every time the machine is restarted.

A password is used to authenticate the machine. This password is automatically reset every 30 days—you don't have to do anything, it just happens. The secure channel is a robust mechanism and usually doesn't go wrong.

This doesn't mean that it can't go wrong. Sometimes you'll see problems that manifest when you try to log on:

- You receive a message that states no logon servers are available.
- You receive a message that Windows cannot connect to the domain because a domain controller isn't available.
- You receive a message that your computer account wasn't found.

The exact wording will depend on your version of Windows. All of these messages mean that the secure channel between the computer and the domain has become corrupted. This can happen in a number of scenarios:

- A desktop machine has been switched off for more than 30 days, which is common in classroom scenarios.
- A user has had extended leave and their machine hasn't been switched on.
- A spare laptop has been kept in a cupboard and not used.
- A preconfigured server was started for the first time more than 30 days after configuration.
- A virtual machine hasn't been started for an extended period, which is very common in lab environments.

These scenarios all have one thing in common: the machine hasn't been connected to the domain for more than 30 days, so the password on its secure channel has expired. The domain controller doesn't recognize the password and therefore won't authenticate the machine.

There are a few ways to fix this problem:

- Remove the machine from the domain and rejoin it to the domain.
- Choose the computer object in ADAC. Choose Reset Account from the Task menu. A confirmation request will be displayed and the reset will occur.
- Right-click the computer object in ADUC and choose Reset Account. A confirmation request will be displayed and the reset will occur.

Using ADAC or ADUC to reset the account results in you having to remove the machine from the domain and rejoin it back to the domain. This requires a number of reboots and is very time-consuming.

None of these options provide a way to test that the secure channel is the cause of the problem. A simpler approach is to use the PowerShell `Test-ComputerSecureChannel` cmdlet, as shown in figure 6.6.

The command `Test-ComputerSecureChannel` performs the test. Use the `-Verbose` parameter for more output. A value of `False` is returned if the secure channel is corrupted (a return value of `True` indicates the channel is good). The cmdlet can be rerun with the `-Repair` switch to reset the password and repair the secure channel. These options are shown in figure 6.6.

Alternatively, you can test and fix in one go:

```
if (-not(Test-ComputerSecureChannel)){
 Test-ComputerSecureChannel -Repair -Verbose
}
```

The only drawback here is that you don't get any feedback if the channel is good.

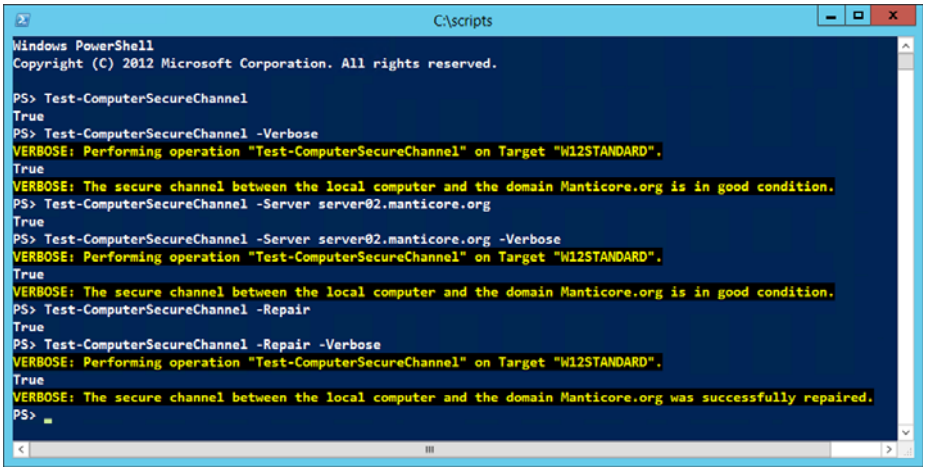

Figure 6.6 Using `Test-ComputerSecureChannel`

Testing and repairing the secure channel with PowerShell has one great advantage: you don't have to remove the machine from the domain and then rejoin it, which is a huge timesaver.

There isn't much more you can do for a computer account. It's worth setting a description on your servers—use the techniques you leaned in chapter 3, but use `Set-ADComputer`'s `-Description` parameter. Computer objects in Active Directory are low-maintenance, but they do have a finite lifecycle and eventually need to be deleted.

6.4 Deleting a computer account

Deleting a computer account is the last step in the object's lifecycle. There are a few reasons for deleting an account:

- The computer is no longer required.
- The computer has been rebuilt with a new name.
- The computer has been removed from the domain and the account wasn't deleted automatically.

Discovering computer accounts to delete usually happens in one of two ways. Either you know you need (or are told that you need) to delete the account because of other activities taking place, or you run a regular test to discover computer accounts that may need deleting.

You can run the following PowerShell snippet to discover inactive (haven't logged on to the domain) computer accounts:

```
Search-ADAccount -AccountInactive  -ComputersOnly |
Format-Table Name, DistinguishedName, LastLogonDate, ObjectClass  -Autosize
```

There are some drawbacks to this:

- The time period after which the account is deemed to be inactive isn't defined.
- It takes no account of which domain controller is contacted.
- In my testing it also returns managed service accounts.

The last point needs a bit more clarification. Using `Search-ADAccount -AccountInactive` returns user and computer accounts. Adding the `-UsersOnly` switch returns just users, excluding managed service accounts. I suspect that the `-ComputersOnly` switch tells the cmdlet to return all accounts that aren't users; unfortunately, this also includes managed service accounts.

You can fix all of the issues by defining an explicit timespan that accounts for AD replication and also filtering out managed service accounts:

```
Search-ADAccount -AccountInactive -ComputersOnly -TimeSpan "90:00:00"|
where ObjectClass -eq "Computer" |
Format-Table Name, DistinguishedName, LastLogonDate, ObjectClass -Autosize
```

This will search for computer accounts that have been inactive for 90 days—the timespan is read as days:hours:minutes. A filter only allows computer objects through:

```
where ObjectClass -eq "Computer"
```

The code completes by displaying the selected properties in a table. Now that you know which computers have inactive accounts, how do you delete them?

> **TIP** If for some reason a domain controller appears on your list, investigate carefully before deleting it. GUI tools in Windows Server 2008 R2 and Windows Server 2012 remove the other references in Active Directory to the domain controller if you delete the computer object, but in earlier versions of Windows you need to clean up the domain controller references (see chapter 11).

If you only have one or two computer accounts to delete, you can find them in Active Directory (using either ADAC or ADUC); right-click the object and choose Delete. You'll get a message box asking you to confirm the deletion. Click Yes to delete and No to cancel the action.

In a situation where you have many objects to delete, this can be achieved using a PowerShell script:

```
Import-Csv -Path C:\Scripts\ADLunches\computersTodelete.txt |
foreach {
 Remove-ADComputer -Identity $_.Name -Confirm:$false
}
```

The computer names are in a file. The names are passed to `Remove-ADComputer`, which performs the deletion. This technique is covered further in the lab section. `Remove-ADComputer` can also be used to delete individual accounts.

This concludes your lesson on computer accounts—time for some lab work.

6.5 Lab

This lab contains exercises based on the computer object lifecycle. You'll practice creating, managing, and deleting computer accounts.

6.5.1 Create computer accounts

Try creating the computer accounts shown in the figures and code snippets in section 6.1, using the instructions provided. The accounts are all created in the Computers container. The names are

- ADLComp1
- ADLComp2
- ADLComp3

The computer name and pre-Windows 2000 (samAccountName) should be identical. It's possible for them to be different, but there's no good reason to do so. The accounts should be created in an enabled state. Do you have to do anything in the GUI to achieve this?

6.5.2 Create computer accounts in bulk

Creating accounts in bulk is required if you have a large number of servers coming online or if you're performing a desktop refresh. This task is easiest performed using

PowerShell. Create a CSV file with the computer name and the description. An example would be

```
Name,Description
ADLComp10, "Test Machine for AD Lunches"
ADLComp11, "Test Machine for AD Lunches"
ADLComp12, "Test Machine for AD Lunches"
```

The file can have a CSV or TXT extension. You can use code like this to create the computer accounts:

```
Import-Csv -Path C:\Scripts\ADLunches\computers.txt |
foreach {
 New-ADComputer -Enabled $true -Name $_.Name `
-Path "CN=Computers,DC=Manticore,DC=org" `
-SamAccountName $_.Name   `
-Description $_.Description -PassThru
}
```

Try creating a file and running the code. How could you alter the code so that accounts were created in different OUs?

6.5.3 Searching for computer accounts

There's some useful information stored in the computer account object. You can search on this information. Try these searches and view the results. *Hint:* you'll need to modify the computer names for your environment.

FIND ALL COMPUTERS

This will display all computers in your Active Directory:

```
Get-ADComputer -Filter * | select Name, DistinguishedName
```

FIND A SPECIFIC COMPUTER

Use this snippet to view all the properties of a specific computer:

```
Get-ADComputer -Identity dc02 -Properties *
```

FIND A SPECIFIC OPERATING SYSTEM

This displays all computers with a Windows Server 2012 operating system:

```
Get-ADComputer -Properties *  `
-Filter "OperatingSystem -like '*Server 2012*'" |
select Name, DistinguishedName
```

How could you modify this for other operating systems? *Hint:* try using this to see the OS versions present in your AD.

```
Get-ADComputer -Properties * -Filter * | select OperatingSystem –Unique
```

Examine the full list of properties for a computer object. How can you filter on the service pack as well as on the operating system?

FIND EXPIRED ACCOUNTS

```
Search-ADAccount -AccountInactive -ComputersOnly -TimeSpan "90:00:00"|
where ObjectClass -eq "Computer" |
Format-Table Name, DistinguishedName, LastLogonDate, ObjectClass -Autosize
```

Modify the timespan to determine the optimum for your environment. Could you change this to find disabled computer accounts?

6.5.4 Managing the secure channel

Pick a computer (noncritical) from your environment and log on to it. Use the PowerShell commands from section 6.3 to test and reset the secure channel.

Reset the account from GUI tools. What happens to the computer account? Rejoin the computer to the domain.

6.5.5 Deleting computer accounts

You created three computer accounts in section 6.5.1. Delete those three accounts:

- ADLComp1—delete with ADAC
- ADLComp2—delete with ADUC
- ADLComp3—delete with PowerShell

Take some of the computer names you used for bulk creation in section 6.5.2 and create a CSV file like this:

```
Name
ADLComp21
ADLComp22
ADLComp23
ADLComp24
```

Use the bulk deletion code to remove those accounts:

```
Import-Csv -Path C:\Scripts\ADLunches\computersTodelete.txt |
foreach {
 Remove-ADComputer -Identity $_.Name -Confirm:$false
}
```

Is there a way to make this any safer? How would you change the code so you confirmed each deletion?

6.6 Ideas for on your own

Managing computer accounts is not as big a task as managing user accounts, but you can make your life more efficient by adopting some automation. Here are some ideas to build into your administration work cycles:

- Use the scripts in this chapter to create a mechanism for removing expired accounts. *Hint:* make it a two-stage process to find the expired accounts and examine them to determine which to delete. Then perform the deletion.
- Create a process for the bulk creation of computer accounts.
- Create a script that tests the service-pack level of your computers. You can then proactively start reporting which machines need to be service packed.

This concludes our examination of computer accounts. Next, we'll look at organizational units.

WINDOWS SERVER

Learn ACTIVE DIRECTORY MANAGEMENT IN A MONTH OF LUNCHES

RICHARD SIDDAWAY

At the heart of your Windows network is Active Directory, the control center for administration, security, and other core management functions. If you're new to Active Directory administration—or if you find yourself unexpectedly thrust into that role—you'll need to get up to speed fast.

Learn Active Directory Management in a Month of Lunches is a hands-on tutorial designed for IT pros new to Active Directory. Without assuming previous administration experience, the book starts by walking you through the most important day-to-day system management tasks. You'll learn how to administer AD both from the GUI tools built into Windows and by using PowerShell at the command line. Along the way, you'll touch on best practices for managing user access, setting group policies, automating backups, and more.

What's Inside

- ADM tasks you'll need every day
- GUI and command line techniques
- Content tested by new administrators
- Well-illustrated, clearly explained examples

This book assumes no prior experience with Active Directory or Windows administration. Examples are based in Windows Server 2012.

Richard Siddaway is an experienced all-around Windows administrator with two decades of experience. He's the author of *PowerShell in Practice* and *PowerShell and WMI*, and coauthor of *PowerShell in Depth*.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/LearnActiveDirectoryManagementinaMonthofLunches

"A great way to learn Active Directory or strengthen your expertise."

—Joseph Moody, MVP
DeployHappiness.com

"An efficient how-to book."

—Ken A Baker, CH Mack

"Perfect resource for those new to the world of ADM."

—James Berkenbile
Berkenbile Consulting

"Just what you expect from the *Lunches* series—direct, to the point, and immediately effective in the real world."

—Jason Helmick
Author of *Learn Windows IIS in a Month of Lunches*

ISBN-13: 978-1617291197
ISBN-10: 1617291196
54499
9 781617 291197

MANNING US $44.99/CAN $47.99 [INCLUDING EBOOK]