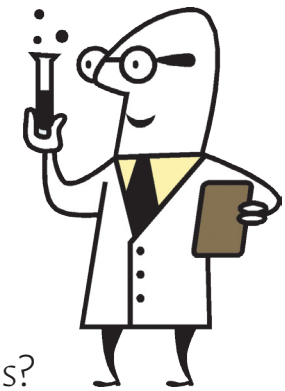
Naively taking on local users and groups



THE MICROSOFT SCRIPTING GUYS ANSWER

How can I manage security descriptors on files and folders?

BELIEVE IT OR NOT, there was a time – long ago, of course – when the Scripting Guys really weren't all that smart. We know that's hard to believe, but – what do you mean it's not that hard to believe? And what do you mean it couldn't have been that long ago? Why, we'll have you know that – oh, OK. Obviously you saw one of the Scripting Guys authors trying to hook up a new TV the other day.

In his defence, though, this particular Scripting Guy would argue that hooking up a new TV isn't rocket science; actually it's way more complicated than rocket science. Just hooking up the TV itself involves three video cables and a pair of audio cables; on top of that you then have to throw in a VCR, a DVD player and an Xbox®, not to mention punching random sets of numbers into the 'universal' remote to try to get it to recognise all those devices. Compared to that, placing a satellite in orbit around the planet Mars is a piece of cake.

Remarkably, the Scripting Guy actually managed to get all of those connections exactly right. However, when he turned everything on nothing happened. Dutifully, he double- and triple-checked each connection. He read through all the user manuals and even checked online to see if there might be some information he was missing. Nothing. He was just about to sacrifice an MP3 player to the electronics god when he noticed that he hadn't taken the key step of plugging in the cable leading from the cable box to the cable outlet in the wall.

Note: You might be thinking that if it's this hard to connect all the Scripting Family's stuff then maybe the Scripting Family has too much stuff. Actually, this Scripting Guy would agree with that. However, any time the family votes on acquiring even more stuff this Scripting Guy is outvoted 2-0. And no, that shouldn't read 2-1; it's 2-0. For some reason, the Scripting Dad's vote never seems to count in family elections, and there's no paper trail.

In answer to your next question, yes, he has tried. But so far the United Nations has refused to send election monitors to the Scripting House.

Of course, now that we think about it, it wasn't so much that the Scripting Guys (well, the one with the new TV in particular) were dumb as they were naïve. For example, when they were planning the Microsoft Windows 2000 Scripting Guide, they solicited help from a number of...experts...when it was time to hammer out the book's chapters. One of the chapters that did not make the cut was a chapter on managing local users and groups. "Managing local users and groups?!?" said the experts. "Why is that even being considered? Who cares about managing local users and groups?"

Well, much to our chagrin,, we have since discovered that everybody (with the possible exception of our in-house 'experts') cares about local users and groups. Each week we get tons of e-mail from people asking how to do things like add a user to a local group, delete a user from a local group, or change the local Administrator password on a computer.

Granted, the Script Center Script Repository (microsoft.com/technet/scriptcenter/scripts) does have some sample scripts that show you how to perform these tasks, but most of these scripts work on just one computer at

Figure 1 **We finally figured it out**

```
On Error Resume Next

Set objExcel = CreateObject("Excel.Application")
Set objWorkbook = objExcel.Workbooks.Open("C:\Scripts\Test.xls")
objExcel.Visible = True

i = 2

Do Until objExcel.Cells(i, 1).Value = ""
    strComputer = objExcel.Cells(i, 1).Value
    Set objUser = GetObject("WinNT://" & strComputer & "/Administrator")
    If Err = 0 Then
         objUser.SetPassword "egTY634!alK2"
         objExcel.Cells(i, 2).Value = Now
    End If
    Err.Clear
    i = i + 1
Loop
```

a time. That's not good enough: people want to know how to do this on multiple machines and all at the same time. Change the local Administrator password on one lousy computer? Get in touch with the real world, Scripting Guys – we want to do that on all our computers, or on all the computers in an OU, or on all our mail servers, or – well, you get the idea.

Considering the fact that the Scripting Guide was published in 2003, that means we've spent nearly four years being besieged by these same e-mail messages. "Here's another reader asking how they can read computer names from an Excel spreadsheet and then change the local Administrator password on each of those computers," one of the Scripting Guys recently remarked. "What in the world can we do about that?"

Well, it took us four years, but we finally figured it out. **Figure 1** shows exactly what we can do about that.

Note: As you can see, it didn't take us four years to write this script because it was hard; it took us four years to write this script because it takes us that long to do pretty much anything. That's why you never want to miss the Scripting Guys' annual New Year's Eve parties; after all, you never know when, or if, the next annual event will come along.

What we have here is a script that pulls computer names from a spreadsheet and then changes the local Administrator password on each of those machines. To perform this magic, we start off easy, by assuming that you have a very simple little spreadsheet, one that looks something like **Figure 2**. (If not, create one right now.)

As you can see, there's nothing special here. In column 1 (column A), we list the names of all the computers that have passwords we need to change. In column 2 (column B), we simply keep track of the date and time that the Administrator password on each computer was last changed. That makes it easy for you to keep tabs on whether your Administrator passwords are all in sync.

Here is a script that pulls computer names from a spreadsheet and then changes the local Administrator password on each

As for the script itself, we kick things off with the On Error Resume Next statement. Usually we don't bother including error handling in our scripts. That isn't because we're opposed to error handling; rather, it's because we try to keep our scripts as short and easy to follow as possible. In this case, however, the On Error Resume Next statement is crucial. After all, our script is going to try to connect to scores of different computers. What happens if one of those computers is turned off? Well, with error handling, nothing will happen. Sure, an error will occur, but the script will be able to shrug it off and continue on. Without error handling you'll get pretty much the same thing the Scripting Guy got when he turned on the new TV for the first time: nothing.

Next up, we use the following block of code to create an instance of the Excel.Application object, make that instance of Excel visible onscreen, and open the spreadsheet C:\Scripts\Test.xls:

```
Set objExcel = CreateObject _
    ("Excel.Application")
Set objWorkbook = objExcel.Workbooks.Open _
    ("C:\Scripts\Test.xls")
objExcel.Visible = True
```

And then last, but surely not least, we assign the value 2 to a counter variable named i. We'll use this variable to keep track of our current row in the spreadsheet.

Note: So then why is i assigned the value 2? That's easy: if you look closely at the spreadsheet you'll see that our data actually begins in row 2. Row 1 is just a header row.

See? We might not be that good at connecting YPbPr1 cables, but we do know something about scripting.

Well, sometimes, anyway.

Now we're ready to get to work. First of all, we set up a Do Until loop that runs until we encounter an empty cell in column 1. And yes, that means that, no matter what else you do, you can't leave any blank rows in column 1. If your script encounters a blank row, it will assume that it has reached the end of the data so the loop (and the script) will end. And while there are ways to work around that, it's a lot simpler just to avoid having blank rows anywhere in your spreadsheet.

Inside the loop, we begin by assigning the value found in cell row i, column 1 to a variable named strCom-

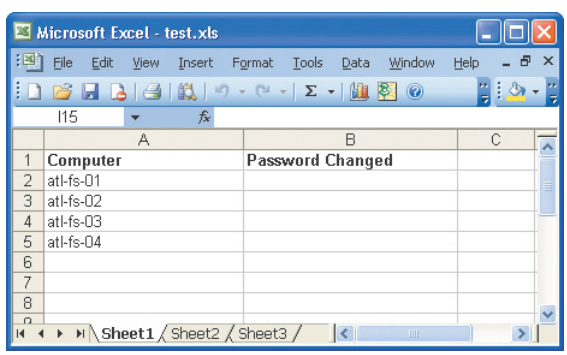


Figure 2 **Obligatory list of computers**

puter. (Remember, the first time through the loop i is equal to 2; thus we'll be working with the value in cell row 2, column 1.) We then use this line of code to create an object reference (objUser) to the local Administrator account on the computer represented by the variable strComputer:

```
Set objUser = GetObject("WinNT://" & _
    strComputer & "/Administrator")
```

If anything is going to go wrong with this script, it's going to blow up right here. For example, suppose we're having network problems, suppose the computer has been turned off, or suppose one of the Scripting Guys has hooked up his DVD player incorrectly and blacked out the entire western United States. If any of those things happen, we're not going to be able to make a connection to the computer. That's why we immediately check the value of the VBScript Err object:

```
If Err = 0 Then
```

If the Err object is equal to 0, that means that we were able to connect to the computer (and to the Administrator account) without any problem. If Err is equal to anything other than 0, that can mean only one thing: our connection attempt failed. In that case, we won't even bother to try to change the password on that computer; needless to say, if we can't make the connection, we're not going to be able to change the password.

Assuming all goes well and that no error occurs, we then execute the following two lines of code:

```
objUser.SetPassword "egTY634!alK2"
objExcel.Cells(i, 2).Value = Now
```

In line 1, we use the SetPassword method to assign a new password (egTY634!alK2) to the local Administrator's account. (And yes, we know: you're not supposed to use your child's name as a password.) In line 2, we then write the current date and time (using the VBScript Now function) to cell row i, column 2. Notice that we update the Password Changed column only if we're actually able to connect to the remote computer and change the password. If we can't connect to a computer then its corresponding Password Changed field will never get updated; that's how we can tell which operations were successful and which ones weren't.

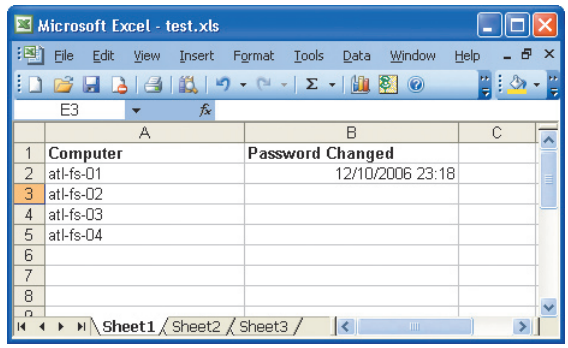


Figure 3 **List of computers after resetting the first password**

What does all that mean? That means that, the first time through the loop, our spreadsheet will look something like **Figure 3**.

So much for computer 1. Now we just need to do a little housekeeping before we move on:

```
Err.Clear
i = i + 1
```

Short as they might be, both these lines of code are very important. (As if the Scripting Guys would write any lines of code that weren't very important!) The first line resets the Err object to 0. If no error occurred, that's redundant; after all, in that case Err will already be 0. If an error did take place, however, then Err will be equal to a value other than 0. In that case, it's crucial that we manually reset the Err object. Why? Because this is the Error object: it only keeps track of errors, it doesn't keep track of successes. For example, let's assume a problem occurred with computer 1 and we were unable to make a connection. For the sake of argument, let's say that caused the Error object to be set to 99.

Now, let's further assume that we loop around and successfully connect to computer 2. What do you think the value of Err will be now? That's right: like it or not, Err will still be set to 99. That's because the Err object value gets changed only when an error occurs; if no error occurs, however, then the Err object will maintain its current value indefinitely.

Does that really matter to us? Well, it should. After all, technically we don't change the password on a computer if we make a successful connection; instead, we change the password only if Err is equal to 0. And – hold on a second while we bring up Calculator – yes, we were right: 99 is not equal to 0 (except, of course, in Scripting Family elections, when 99 Scripting Dad votes do add up to 0). And that matters to us because, even though we made the connection to computer 2, Err is not equal to 0. And because Err is not equal to 0, we don't even try to change the local Administrator password.

In other words, if the value of Err ever changes, you'll need to reset that value to 0; otherwise from then on your script will believe that an error has occurred. And how do you reset the Err object back to 0? You got it: just call Err.Clear.

Next we increment the value of i by 1. Why? Well, the first time through the loop the variable i was equal to 2; that's because we wanted to connect to the computer listed in row 2, column 1. The second time through the loop we want to connect to the computer listed in row 3, column 1. Seeing as how i is the variable that indicates the row we want to work with we thus need to make i equal to 3. And, as ev-

eryone knows, 2 + 1 = 3. (Good thing we still had Calculator running, isn't it?) Only then do we loop around and repeat the process with the next computer in the spreadsheet.

Now, that's pretty cool, but – seeing as how we're no longer as naïve as we used to be – we're well aware that this script, by itself, will barely put a dent in the e-mail we'll get. "That Excel thing was nice," people will say, "but I still need to change the local Administrator password for all the computers in an OU, or for all the computers listed in a text file. Or maybe I'd like to pop up a list of computers and be able to select one from a listbox. Or maybe what I'd really like to do is ..."

Relax. Your wish is the Scripting Guys' command. (Assuming you're willing to wait four years for that wish to come true, that is.) As it turns out, we have scores of multiple computer templates posted on the Script Center, templates that make it extremely easy to run a script against multiple machines. For example, suppose you do want to change the local Administrator password for all the computers in an OU. Hey, no problem. The template for running a script against all the computers in an OU looks like **Figure 4**.

See the section in the template labelled 'Insert your code here'? All you need to do is delete the sample code found in that section and replace it with the code (which you can copy from this column) for changing the local Administrator password. In other words, replace the sample code with the following code:

```
Set objUser = GetObject("WinNT://" & _
    strComputer & "/Administrator")
If Err = 0 Then
    objUser.SetPassword "egTY634!alK2"
End If
Err.Clear
```

This particular example won't log the results for you, but you can easily add that code in yourself. Replace the sample code, change the connection string LDAP://OU=Finance,dc=fabrikam,dc=com to point to one of your OUs, and you're in business.

> Technically we don't change the password on a computer if we make a successful connection

To tell you the truth, we actually posted these templates over a year ago, figuring that they would be one of the most popular items we ever put up in the Script Center. We were wrong about that: they hardly ever get used, even though people write to us pretty much every day asking how to run scripts against multiple machines. We're assuming that no one knew that these templates existed. But now you know. Or at least you'll know if we tell you where to find them: microsoft.com/technet/scriptcenter/scripts/templates. Hmmm, maybe that's why no one ever used these templates...

So now that the Scripting Guys aren't as naïve as they used to be, does that also mean that they aren't as dumb and as klutzy as they used to be? Let's put it this way: not too long ago one of the Scripting Guys decided to spend the weekend fixing up a few things around the house. When he came back to work on Monday, he had both a smashed thumb that wouldn't stop bleeding and a big gash in his leg.

Which, come to think of it, really wasn't all that bad. At least not for a Scripting Guy. ■

Figure 4 **Running a script against an OU**

```
On Error Resume Next

Set objOU = GetObject("LDAP://OU=Finance,dc=fabrikam,dc=com")
objOU.Filter = Array("Computer")

For Each objComputer in objOU
    strComputer = objComputer.CN

    '
======================================================================
    ' Insert your code here
    '
======================================================================

    Set objComputer = GetObject("WinNT://" & strComputer & "")
    objComputer.Filter = Array("User")
    For Each objUser in objComputer
        Wscript.Echo objUser.Name
    Next

    '
======================================================================
    ' End
    '
======================================================================

Next
```

The Scripting Guys *work for – well, are employed by – Microsoft. When not playing/coaching/watching baseball (and various other activities) they run the TechNet Script Center. Check it out at www.scriptingguys.com.*