

# From MCPmag.com: Feature Article

 [print article](#)

## Automate Your Administration

*Speed up and simplify your work by applying these 10 scripts to the job.*

by Chris Brooke



There's an old joke that goes, "How many computer programmers does it take to screw in a light bulb?" Answer: "Can't be done. That's a hardware problem." While that may have been true "back in the day" (back when it was funny), it no longer truly represents life in IT. The lines have blurred. Developers now take an active role in choosing and assembling hardware for their applications. By the same token, network administrators have found themselves in the unenviable position of having to hone software-development skills in order to speed and simplify their administrative tasks, lest they end up spending every night at the office. As the father of a 1-year-old, believe me when I say that I would rather be home bouncing said munchkin on my knee than in the server room waiting for a backup to complete so that I can finish the rest of the server maintenance.

### Related Editorial

- [Broaden Your Sites: The Site Server 3.0 Story](#)
- [A Course on Site Server](#)
- [Search the World Over](#)

### Want more?

See the back issues vault for Premier area members only.

Enter administrative scripting.

Whether you're a network administrator looking to speed up certain mundane tasks or a closet code-monkey who likes to automate everything you possibly can, scripting can simplify your life and help ensure you get plenty of quality munchkin-time. To that end, I've trudded through mountains of scripts and narrowed them down to 10 that you're sure to find indispensable. As part of my search criteria, I looked for scripts that not only accomplished a vital task, but also could be easily modified so that you can extend them as your scripting skills evolve. Along with each script represented here, I've highlighted relevant sections and included sample use-case information.

One final caveat before we dive in to these scripts: I've left them in their original form, whether or not copyright was explicitly specified. As such, you'll notice differences in how variables are declared—with or without some form of Hungarian Notation, formatting, comments and so on. Most are well commented and you should be able to interpret them for future editing. Where necessary, I've provided additional explanation in the text discussing the script.

### Tip

You can download the complete set of scripts by right-clicking on this link and saving the file to disk: [Scripting.zip \(29KB\)](#)  
*Problems downloading this script?* Send e-mail to [michael.domingo@mcpmag.com](mailto:michael.domingo@mcpmag.com); put "Scripting.zip" on subject line of message.

### DumpUsers.vbs by Tim Hill

A few years ago, in my column, "Windows Utilities Inside Out," I highlighted a utility called DumpACL. It exports NT user and group information, along with all NTFS Access Control Lists (ACLs) for a given computer. No column before or since has generated as many inquiries. To this day, I get readers e-mailing me requesting it. This script was published by Tim Hill in his book, *Windows 2000 Windows Script Host*. It uses Active Directory Services Interface (ADSI) to perform half of the DumpACL operation—namely: dumping users. The benefit of this script is that it dumps extensive user information, including the User Flags, into a nicely formatted Excel spreadsheet. Below I've highlighted some of the relevant code, which exists almost entirely in the MAIN( ) function.

```
'////////////////////////////////////////
'$Workfile: DumpUsers.vbs $ $Revision: 2 $ $Date: 7/19/99 12:19a $
'$Archive: /TimH/Projects (SCC)/Books & Articles/Macmillan/Windows 2000
'$ System Scripting (1999)/Scripts/DumpUsers.vbs $
```

```

' Copyright (c) 1998 Tim Hill. All Rights Reserved.
////////////////////////////////////
' Dump user accounts to an Excel spreadsheet
... <snip>
Function Main
Trace 1, "+++Main"
Dim sAdsPath, sExcelPath, oSheet, oExcel, oUser, oAdsObj
...<snip>
End If
sAdsPath = "WinNT://" & Wscript.Arguments(0)
sExcelPath = g_oFSO.GetAbsolutePathName(Wscript.Arguments(1))

' Prepare ADSI computer/domain object
On Error Resume Next
Set oAdsObj = GetObject(sAdsPath)
If Err.Number <> 0 Then
Wscript.Echo sAdsPath & ": not found (0x" & Hex(Err.Number) & ")"
Wscript.Quit(Err.Number)
End If
On Error Goto 0

...<snip>
' Enumerate all users in the computer/domain
oAdsObj.Filter = Array("User") ' Filter user accounts only
ix = 0
For Each oUser In oAdsObj ' For each user account...
DumpAccount oSheet, ix, oUser ' Go add to sheet
ix = ix + 1 ' Bump index
Next

... <snip>
' Return value is passed to Wscript.Quit as script exit code
Main = 0
End Function

```

After all the “snipping,” we can see the main logic of the script. It uses the WScript.Arguments collection to retrieve the computer/domain that you want to work with. Indeed, it speeds things up if you specify the type of object you want to work when starting the script, but it’s by no means required. For example, at the command line, you can enter:

```

C:\cscript dumpusers mycomputer,computer
c:\myspreadsheet.xls

```

Or simply type this:

```

C:\cscript dumpusers mycomputer c:\myspreadsheet.xls

```

and let the script figure it out for itself.

It then uses ADSI and applies a filter so that only users will be dumped (I’m seeing great possibilities already! With just a little editing, we could dump groups, computers in a domain and so on) It then uses another SUB, “DumpAccount,” to place the information in the spreadsheet. Figure 1 shows the spreadsheet I received when I ran the script on my computer.

From **Windows 2000  
Windows Script Host**,  
by Tim Hill  
ISBN 1578701392, \$35  
Copyright 2000;  
reproduced by  
permission of New  
Riders

Name	User ID	User Type	Description
Administrator	00000000000000000000000000000000	Administrator	Administrators have full control over the system and can perform any task.
Guest	00000000000000000000000000000000	Guest	Guests have limited access to the system and can only view public information.
System	00000000000000000000000000000000	System	System accounts are used for system processes and services.
...	...	...	...

**Figure 1.** DumpACL generates an Excel report of users. (Click image to view larger version.)

This script offers significant advantages over standard GUI tools or even the DumpACL utility. For instance, with very little editing, you could modify this script to take users from a spreadsheet and add them to a computer or domain (a task that, depending upon the size of your organization, you, no doubt, perform quite often).

### AddUsers.vbs by Microsoft

Just in case you don't have the time to edit (or simply don't feel like editing) the previous script to add users, I've included this script from the MSDN Scripting Web site. It's also included in the Windows NT Server 4.0 Resource Kit. It reads a list of users from an Excel spreadsheet and adds them using ADSI. This script is somewhat shorter than Dump Users.vbs, as there's no spreadsheet formatting to include. It comes with a spreadsheet already configured in the proper format—all you have to do is add the names. Also, this script is designed to add users to Active Directory using the LDAP provider, rather than into an NT domain using the WinNT provider. If you're still using NT domains (and, according to the latest numbers, there are a lot of you!), this script can also be easily edited to work with the WinNT provider.

```
' Windows Script Host Sample Script
'
'-----
' Copyright (C) 1996 Microsoft Corporation
'
...<snip>
' The sample uses the directory root
' "LDAP://DC=ArcadiaBay,DC=Com,O=Internet"
' Change the directory path in the EXCEL spreadsheet to match your DS
' before running this sample.

... <snip>

Do While oXL.activecell.Value <> ""

' if the requested OU is a new one...
If oXL.activecell.Value <> ou Then
' Pick up the OU name...
ou = oXL.activecell.Value
' Compose the ADSI path...
s = "LDAP://" + ou + "," + root

' Show it to the user...
WScript.Echo s

' And get the object
Set c = GetObject(s)
End If

' Compose the user common name name from first and last names...
uname = "CN=" + oXL.activecell.offset(0, 1).Value + "" +
oXL.activecell.offset(0, 2).Value

' Create the new user object...
Set u = c.Create("user", uname)
```

```
' Set the properties of the new user
```

```
u.Put "givenName", oXL.activecell.offset(0, 1).Value ' givenName
u.Put "sn", oXL.activecell.offset(0, 2).Value ' sn
u.Put "mail", oXL.activecell.offset(0, 3).Value ' Email
u.Put "sAMAccountName", oXL.activecell.offset(0, 4).Value ' Sam Acct
u.Put "telephoneNumber", oXL.activecell.offset(0, 5).Value ' Phone
```

```
' Enable the account, must change pw @ logon
```

```
u.Put "userAccountControl",16
```

```
' ... and update the DS
```

```
u.SetInfo
```

```
' Done with this object, discard it
```

```
Set u = Nothing
```

```
' Step to the next user...
```

```
oXL.activecell.offset(1, 0).Activate ' Next row
```

```
Loop
```

```
' ... rest of script
```

This script doesn't use any user-defined SUBS or FUNCTIONS. Instead, the Do... Loop shown contains the bulk of the script logic. The script uses automation to open the spreadsheet, making it a simple matter to step through the cells and acquire the user data. This is then transferred to the ADSI object and entered as user information. Once the loop has reached the end of the list, script execution terminates.

Adding users is a common, yet tedious, administrative task. This script simplifies and speeds this process as-is. Also, you could tweak it a bit to perform such tasks as group maintenance, modification of user rights and so on.

### The Future of Scripting: ASP.NET

Although the scripts highlighted here were, for the most part, written for the Windows Script Host, they can be easily adapted to provide their functionality to a Web page via Active Server Pages. Consequently, ASP pages provide a wealth of functionality from which you can extract scripts. Once all references to intrinsic ASP objects have been removed, you're left with plain ol' VBScript. At least, that's how it has been until now.

The Microsoft .NET framework has introduced a new Web platform called ASP.NET. These Web pages are differentiated from standard ASP pages by their extension "ASPX." ASP.NET pages require that the .NET Framework is installed on the Web server. They can coexist side-by-side with ASP, but a different engine handles processing for ASPX files. Why the difference? ASP.NET is fully integrated with the .NET Framework. Rather than relying on VBScript or JScript (which are just subsets of larger, more complete development languages), ASP.NET pages can be developed in any .NET language such as Visual Basic.NET, C# or even Visual C++ .NET. This gives Web pages the full power of the underlying development languages.

You may ask, "How does this affect me?" Well, maybe not at all—or perhaps a great deal. If your duties as a beloved "server-jockey" include some Web development, it would be in your best interest to start learning at least one of the .NET languages now. If you currently use mostly JScript in your scripts, I recommend J# or C#. If you have become accustomed to VBScript, take a look at Visual Basic.NET.

I'm not trying to convert any of you into bonafide "code monkeys" (perish the thought!), but the times, they are a-changin'. In this era of volatile technology stocks and an uncertain job market, the ones with the most skills win. See ya at the finish line!

—Chris Brooke

#### Shortcut.vbs by Microsoft

This script almost didn't make the first cut when I was trudging through the volumes of possible programs to feature here. In the first place, it's short and doesn't do much—only puts a shortcut on the desktop. However, the more I thought about it, the more I realized it's likely to end up being one of those little scripts that you'll use all the time. Originally, it was one of the samples included as part of the Windows Script Host documentation download. Here's the bulk of it:

```
' Windows Script Host Sample Script
'
'-----
' Copyright (C) 1996-1997 Microsoft Corporation
'
... <snip>

' Read desktop path using WshSpecialFolders object
DesktopPath = WSHShell.SpecialFolders("Desktop")

' Create a shortcut object on the desktop
Set MyShortcut = WSHShell.CreateShortcut(DesktopPath & "\Shortcut to notepad.Ink")

' Set shortcut object properties and save it
MyShortcut.TargetPath = _
    WSHShell.ExpandEnvironmentStrings("%windir%\notepad.exe")
MyShortcut.WorkingDirectory = _
    WSHShell.ExpandEnvironmentStrings("%windir%")
MyShortcut.WindowStyle = 4
MyShortcut.IconLocation = _
    WSHShell.ExpandEnvironmentStrings("%windir%\notepad.exe, 0")
MyShortcut.Save
```

WScript.Echo "A shortcut to Notepad now exists on your Desktop."

Not much on its own, but there's some real benefit to be found in combining this script with others. Just think for a minute about how many times you have to look for something, such as the log file for your backup software or perhaps the spreadsheet created by the "DumpUsers" script. Putting a shortcut on the desktop automatically is a real time-saver. Left a script running overnight and need to view the error log first thing in the morning? Put a shortcut to it on your desktop. Need everyone in your office to download the latest virus-checking software? Put a shortcut on their desktop! While the sample script has "Notepad.exe" hard-wired into it as the application started from the shortcut, this can be whatever you want. It can even be specified by arguments or as a persisted value in a text file or database.

From **Scripting Windows 2000**, by Jeffrey Honeyman ISBN 007212444X, \$44.99 Copyright 2000; reproduced by permission of McGraw-Hill/Osborne Media

### SvcStat.vbs by Jeff Honeyman

A large part of NT administration relates to services. Indeed, starting, stopping and querying service settings such as Log On Account, Startup Mode and so on are critical tasks that occur quite frequently. This script from Jeffrey Honeyman's *Scripting Windows 2000* uses Windows Management Instrumentation (WMI) to query the status of a service on a given computer. Is the Scheduler service running on the DC? Hmm... let's check!

```
' SvcStat.vbs
' Copyright 2000 by Jeffrey Honeyman
' Published in Scripting Windows 2000 by McGraw-Hill Professional Publishing
... <snip>
```

```
Function FindComputer(box)
    Set objBoxes=GetObject("LDAP://CN=computers,DC=domain,DC=com")
    For Each thing in objBoxes
        If ucase(thing.cn) = ucase(box) Then
            FindComputer=thing.ADsPath
        Exit Function
    End If
Next
End Function
```

```
Function FindDC(box)
    Set objBoxes=GetObject("LDAP://OU=Domain Controllers,DC=domain,DC=com")
    For Each thing in objBoxes
        If ucase(thing.cn) = ucase(box) Then
            FindDC=thing.ADsPath
        Exit Function
    End If
Next
End Function
```

... <snip>

```
Set objSys=GetObject(patth)
Set objSvc=objSys.GetWMI Services
Set objServs=objSvc.instancesOf("WIN32_Service")
For Each thing in objServs
    If ucase(thing.Properties_("DisplayName"))=ucase(serv) Then
        s="Properties for " & serv & " on " & box & ".:" & vbCRLF & vbCRLF
        For Each prop in thing.Properties_
            s=s & prop.name & " = " & prop.value & vbCRLF
        Next
    End If
Next
```

... <snip>

[msgbox s](#)

The approach here is straightforward. The functions FindComputer and FindDC are used to locate the specified computer, whether it's a domain controller or not. Once it has been located, WMI is used to get all the properties of the specified service by comparing every WIN32 service to the one specified on the command line. Once a match is found, the script builds and displays a msgbox. This script stands alone as a valuable tool to query service information.

### ProcStat.vbs by Jeff Honeyman

Another common administrative task is troubleshooting failed processes. Because many applications don't run as services (for instance Outlook, Word and Windows Messenger), it's useful to be able to query a running process. This script is similar to SvcStat.vbs (also from *Scripting Windows 2000*), except that it provides information about a particular process. The complete script available for download online includes a line that displays the correct usage, including what command-line arguments to use.

```
' ProcStat.vbs
' Copyright 2000 by Jeffrey Honeyman
' Published in Scripting Windows 2000 by McGraw-Hill/Osborne Media
...<snip>
```

```
box=WScript.Arguments(0)
proc=WScript.Arguments(1)
```

```
... <snip>
```

```
Set objSys=GetObject(patth)
Set objProc=objSys.GetWMIServices
Set objProcs=objSvc.instancesOf("WIN32_Process")
For Each thing in objProcs
  If ucase(thing.Properties_("name"))=ucase(proc) Then
    c=c+1
    s=s & "Properties for " & proc & " on " & box & ":" & vbCRLF & vbCRLF
    For Each prop in thing.Properties_
      s=s & prop.name & " = " & prop.value & vbCRLF
    Next
  End If
Next

If c=0 Then
  WScript.Echo "Process " & proc & " does not exist on computer " & box
  WScript.Quit (5558)
End If

If c>1 Then
  WScript.Echo "There are "& c & "instances of process "& proc * " on system" &
  box & vbCRLF & s
Else
  msgbox s
```

A significant difference between processes and services is that there can be more than one instance of a process running at any given time. As such, this script includes a counter variable "c" to count the number of instances for the specified process. If more than one is running, the output is echoed to the command line. If there's only one instance, it's placed in a msgbox just as it was for services.

### RemoteReboot.vbs/RemoteShutdown.vbs by Microsoft

These are actually two separate scripts, but with one common goal: to use WMI to perform that mainstay of troubleshooting skills—the reboot! How often do you have to leave your desk and trudge across the office (or perhaps even up some stairs) to reboot a server? Yes, Windows 2000 significantly reduced the number of reboots required when installing software and upgrading the OS, but they still happen. If you have the spare time and you like getting the extra exercise, feel

free to keep walking. If you're as busy as I am, try this script. It comes from the Microsoft WMI SDK downloaded at <http://download.microsoft.com/download/platformsdk/sdkx86/1.5/NT45/EN-US/wmisdk.exe>.

```
' Copyright (c) 1997-1999 Microsoft Corporation
'
*****
'
' WMI Sample Script - REMOTE system reboot (VBScript)
' ... <snip>
Set OpSysSet=GetObject(_
    "winmgmts:{(_
        RemoteShutdown)}//REMOTE_SYSTEM_NAME/root/cimv2").ExecQuery_
    ("select * from Win32_OperatingSystem where Primary=true")

For Each OpSys in OpSysSet
    OpSys.Reboot()
Next
```

And in case you just need to shut it down...

```
OpSys.Shutdown()
```

I suppose the next logical step would be to put both of these scripts into one .WSF file in two different jobs, so that you can select whether to reboot or shutdown when you run it. Keep a shortcut to this script on your desktop; you'll find yourself using it often.

#### BackupEventLog.vbs by Microsoft

One of the most useful troubleshooting tools an administrator has at his or her disposal is the NT Event Log. When "unpredictable" behavior occurs, the event log is usually the first place to look. The following script uses WMI to back up the application event log to a file. I downloaded this from the same site as the previous script.

```
' Copyright (c) 1997-1999 Microsoft Corporation
'
*****
'
' ... <snip>

Set LogFileSet=GetObject("winmgmts:{(Backup,Security)}").ExecQuery(_
    "select * from Win32_NTEventLogFile where LogfileName='Application'")

For Each Logfile in LogFileSet
    RetVal = Logfile.BackupEventlog("c:\BACKUP.LOG")
    if RetVal = 0 then WScript.Echo "Log Backed Up"
Next
```

This is another short but essential script for your toolbox, and one that can be easily modified for even more powerful behavior. For example, you could execute the script remotely and have it use automation to e-mail the log file to you (for, say, troubleshooting a location off-site). By simply changing the LogFileName in the GetObject line, we can dump the system or security logs, as well. Priceless.

#### SendMail.vbs by Anonymous

I bet I had you a bit worried in the last script when I mentioned sending an e-mail as if there were nothing to it, didn't I? Not to worry. This script, which comes from "anonymous" in the Windows Script Community at MSN.com, can be used with the above event log script, or anywhere you need to send an e-mail via automation. Let's face it: Windows doesn't always run perfectly. Errors occur, servers go offline. Stuff happens. The ability to notify the on-call administrator via an e-mail message will minimize down-time and make you look indispensable in the eyes of your boss. (If it looks familiar, I wrote a script almost identical to it for a previous



“Scripting for MCSEs” column before finding this one.)

```
' SendMail.vbs
' Published 2/27/01 - Windows Script Community, MSN.com

Set WSHShell = WScript.CreateObject("WScript.Shell")
Set appOutl = Wscript.CreateObject("Outlook.Application")

' Set a reference to the MailItem object.
Set maiMail = appOutl.CreateItem(0)
' Get an address from the user.
maiMail.Recipients.Add(InputBox("Enter name of message recipient"))
' Add subject and body text.
maiMail.Subject = "Testing mail by Automation"
maiMail.Body = "Message body text"

' Send the mail.
maiMail.Send
' Close object references.
Set appOutl = Nothing
Set maiMail = Nothing
Set recMessage = Nothing
```

Of course, you wouldn't want to prompt for the recipient's name in an automated script—this would be either “hard-wired” into the script or accessed from a file. Application automation is a wondrous thing. The sheer power of it all! (If you don't believe me, check out the “bonus” script at the end of this article. Talk about power!)

### ResetPassword.vbs by Chris Brooke

You didn't think you were going to get through this without at least one script written by yours truly, did you? In fact, this script was featured in my scripting column some time ago. I've included it here for those of you who (shame, shame!) haven't been following my column. Besides, we've already noted that a primary goal of scripting is to save time, and this one saves a bundle!

```
' ResetPW.vbs
' Copyright 2000, by Chris Brooke
```

```
Option Explicit
Dim objContainer, colUsers, IFlag
Set objContainer=GetObject("WinNT://domain")
```

```
ObjContainer.Filter=Array("User")
```

```
For Each colUsers in objContainer
  IFlag=colUsers.Get("UserFlags")
  If (IFlag AND &H10000) <> 0 Then
    colUsers.Put "UserFlags", IFlag XOR &H10000
  End If
  colUsers.Put "PasswordExpired", 1
  colUsers.SetInfo
Next
```

Small but mighty! It also checks to make sure that the “Password Never Expires” flag isn't set. No one should be exempt from the mandatory password policy. Not even the CEO. This script is vital whenever your organization has a personnel change or some other potential security hole that needs patching. Requiring users to reset their passwords between normal, scheduled password resets is something you've likely encountered several times before. I know I have—which is why I wrote the script!

From *Windows  
NT/2000 ADSI  
Scripting for System*

### A Potpourri of Scriptlets by Thomas Eck

As our last featured script, I decided to include a flood of scriptlets from Thomas Eck's book, *Windows*

**Administration**, by  
Thomas Eck  
ISBN 1578702194, \$45  
Copyright 2000;  
reproduced by  
permission of [New  
Riders](#)

*NT/2000 ADSI Scripting for System Administration.*  
It's packaged as a text file and includes a plethora of  
script tidbits that you can add to your scripts. The  
functionality of each scriptlet is completely self-  
contained, and you can simply cut-and-paste them as  
you see fit. These scriptlets include everything from  
querying/setting AutoUnlockInterval for user accounts  
to reporting disabled accounts and much more. They

were all written to be integrated into ASP pages, so you'll have to change a few  
"Response.Write" lines to "WScript.Echo" (as well as other minor changes), but  
most are equally at home inside the Windows Script Host as in an ASP. Here are  
some examples:

#### ' Querying AutoUnlockInterval Using a VBScript Active Server Page

```
Dim Domain
Dim DomainName
DomainName = "Domain_Name_To_Manage"
Set Domain = GetObject("WinNT://" & DomainName)
Response.Write Domain.AutoUnlockInterval
```

#### ' Setting a New Value for MaxBadPasswordsAllowed ' Using a VBScript Active Server Page

```
Dim Domain
Dim DomainName
Dim NewValue
DomainName = "Domain_Name_To_Manage"
Set Domain = GetObject("WinNT://" & DomainName)
NewValue = 5
Domain.MaxBadPasswordsAllowed = NewValue
Domain.SetInfo
```

#### ' Querying MaxPasswordAge Using a VBScript Active Server Page

```
Dim Domain
Dim DomainName
DomainName = "Domain_Name_To_Manage"
Set Domain = GetObject("WinNT://" & DomainName)
Response.Write ((Domain.MaxPasswordAge) / 86400)
```


#### ' Removing a Computer Account Using a VBScript Active Server Page

```
Dim Container
Dim ContainerName
Dim ComputerToRemove
ContainerName = "Container_Name_To_Manage"
Set Container = GetObject("WinNT://" & ContainerName)
ComputerToRemove = "Computer_Account_To_Remove"
Call Container.Delete("Computer", ComputerToRemove)
```

### You What?! (A Bonus Script)

No discussion of scripting would be complete without mentioning arguably the  
most famous VBScript ever to be created: "I Love You.txt.vbs.". The "I Love You"  
worm was a textbook example of the power of VBScript. Because, arguably, all  
knowledge is morally neutral (only the application of knowledge can be good or  
evil), I'd recommend you study this script, if you have it available (I'd send it to  
you, but I'm pretty sure I'd get into serious trouble!), to gain insight into the  
potential power of application automation. Just don't send it to anyone!

### Now Get Out There and Automate!

I hope these scripts will find a place of honor in your administrative toolkit. If  
you're just now getting involved in scripting, these scripts will provide an essential  
launch pad to help you learn while doing. I also hope I've given you scripting  
veterans something of value, as well. If nothing else, you now have some ideas  
upon which to build even better scripts. I can see the bulb lighting up inside your  
head from here. Just don't call me when it's time to change it. 

*Chris Brooke, MCSE+Internet, is a contributing editor for MCP  
Magazine and product and technology editor for ComponentSource,  
an online component market place for professional developers and*

*technical decision-makers. He's been a practicing tech head for more than 14 years, specializing in development, integration services, and network/Internet administration. You can contact Chris about "Automate Your Administration" at [chrisb@componentsource.com](mailto:chrisb@componentsource.com).*

[back to previous page](#)

---

Copyright 1996-2002, 101communications LLC. See our Privacy Policy.  
For more information, e-mail [editor@mcpmag.com](mailto:editor@mcpmag.com).