VBScript, WMI, and ADSI

Using VBScript, WMI, and ADSI to Automate Windows® Administration
## Contents at a Glance

<table>
<thead>
<tr>
<th>Part</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Introduction to Windows Administrative Scripting</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Scripting Concepts and Terminology</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Running Scripts</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>The Components of a Script</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>Designing a Script</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td><strong>VBScript Tutorial</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Functions, Objects, Variables, and More</td>
<td>75</td>
</tr>
<tr>
<td>6</td>
<td>Input and Output</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>Manipulating Numbers</td>
<td>115</td>
</tr>
<tr>
<td>8</td>
<td>Manipulating Strings</td>
<td>129</td>
</tr>
<tr>
<td>9</td>
<td>Manipulating Other Types of Data</td>
<td>145</td>
</tr>
<tr>
<td>10</td>
<td>Controlling the Flow of Execution</td>
<td>155</td>
</tr>
<tr>
<td>11</td>
<td>Built-In Scripting Objects</td>
<td>173</td>
</tr>
<tr>
<td>12</td>
<td>Working with the File System</td>
<td>193</td>
</tr>
<tr>
<td>13</td>
<td>Putting It All Together: Creating Your First Script from Scratch</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td><strong>Windows Management Instrumentation and Active Directory Services Interface</strong></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Working with ADSI Providers</td>
<td>247</td>
</tr>
<tr>
<td>15</td>
<td>Manipulating Domains</td>
<td>257</td>
</tr>
<tr>
<td>16</td>
<td>Manipulating Users and Groups</td>
<td>271</td>
</tr>
<tr>
<td>17</td>
<td>Understanding WMI</td>
<td>283</td>
</tr>
<tr>
<td>18</td>
<td>Querying Basic WMI Information</td>
<td>299</td>
</tr>
<tr>
<td>19</td>
<td>Querying Complex WMI Information</td>
<td>317</td>
</tr>
<tr>
<td>20</td>
<td>Putting It All Together: Your First WMI/ADSI Script</td>
<td>335</td>
</tr>
<tr>
<td>21</td>
<td>Testing and Debugging WMI and ADSI Queries</td>
<td>357</td>
</tr>
<tr>
<td></td>
<td><strong>Advanced Scripting Techniques</strong></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Modular Script Programming</td>
<td>373</td>
</tr>
<tr>
<td>23</td>
<td>Scripts Packaging and Protection</td>
<td>389</td>
</tr>
</tbody>
</table>
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>Who Should Read This Book?</td>
<td>2</td>
</tr>
<tr>
<td>How to Use This Book</td>
<td>2</td>
</tr>
<tr>
<td>Part I: Introduction to Windows Administrative Scripting</td>
<td>3</td>
</tr>
<tr>
<td>Part II: VBScript Tutorial</td>
<td>3</td>
</tr>
<tr>
<td>Part III: Windows Management Instrumentation and Active Directory</td>
<td>5</td>
</tr>
<tr>
<td>Part IV: Advanced Scripting Techniques</td>
<td>6</td>
</tr>
<tr>
<td>Part V: Ready-to-Run Examples</td>
<td>7</td>
</tr>
<tr>
<td>Appendix</td>
<td>8</td>
</tr>
<tr>
<td>Preparing to Use This Book</td>
<td>8</td>
</tr>
<tr>
<td>Typographical Elements</td>
<td>8</td>
</tr>
<tr>
<td>Sample Scripts</td>
<td>9</td>
</tr>
<tr>
<td>Sample Scripts—Explained</td>
<td>9</td>
</tr>
</tbody>
</table>

## Part I Introduction to Windows Administrative Scripting

1. **Scripting Concepts and Terminology** 13
   - Overview of Scripting 13
   - Script Hosts 14
   - ActiveX Scripting Languages 16
   - The Component Object Model (COM) 18
   - Critical Scripting Security Issues 18
   - VBScript Versus Windows PowerShell 19
   - Summary 20

2. **Running Scripts** 21
   - Windows Script Host 21
   - Command-Line Scripts 23
   - Notepad and Script Editors 25
     - Bare Bones: Notepad 26
     - A Step Up: Programmer’s File Editor 26
   - Script-Specific: VBScript Editors 27
   - All-Inclusive: VBScript IDEs 29
Contents

Functions Defined ................................................................. 85
  Input Parameters ................................................................. 85
  Output Values ................................................................. 87
  Intrinsic Versus Custom Functions ........................................... 88
Statements and Subroutines Defined ........................................ 92
  Functions, Without the Output .............................................. 92
Objects Defined ........................................................................ 94
  Properties ........................................................................ 95
  Methods ......................................................................... 95
  Collections ..................................................................... 96
  A Sample Object ................................................................ 97
  Scripting with Objects ....................................................... 98
Summary .................................................................................. 99

6 Input and Output .................................................................. 101
  Displaying Messages .......................................................... 101
    The MsgBox Statement and Function .................................. 102
    More Complex Messages ................................................. 105
    MsgBox Best Practices .................................................. 106
    Go Generic with WScript.Echo ....................................... 107
  Asking for Input ............................................................... 108
    Graphical Input ............................................................ 108
    Command-Line Input .................................................... 109
  Command-Line Parameters as Input ....................................... 110
    Running Command-Line Scripts ..................................... 111
    Parsing Parameters ....................................................... 111
Summary .................................................................................. 114

7 Manipulating Numbers ......................................................... 115
  Numbers in VBScript .......................................................... 115
  Basic Arithmetic ................................................................ 116
  Advanced Arithmetic ........................................................ 118
  Boolean Math .................................................................... 121
  Converting Numeric Data Types .......................................... 126
  Converting Other Data Types to Numeric Data ...................... 127
Summary .................................................................................. 127

8 Manipulating Strings ............................................................ 129
  Strings in VBScript ............................................................ 129
  Working with Substrings ..................................................... 131
### 9 Manipulating Other Types of Data

**Working with Dates and Times**
- Dates and Times in VBScript 145
- Getting the Date or Time 146
- Converting Date and Time Data 147
- Working with Past and Future Dates 147

**Working with Arrays**
- Arrays in VBScript 149
- Creating and Manipulating Arrays 149
- Working with Array Data 151

**Working with Bytes**
- Bytes in VBScript 152
- Converting Byte Data 152

**Summary** 153

### 10 Controlling the Flow of Execution

**Conditional Execution**
- If/Then 156
- Select/Case 161

**Loops**
- Do While/Loop and Do/Loop While 162
- Do Until/Loop and Do/Loop Until 163
- For/Next 164
- For Each/Next 165
- Exiting Loops 166

**Putting It All Together** 167

**Summary** 170

### 11 Built-In Scripting Objects

**The WScript Object** 173
**The Network Object**
- Methods and Properties 175
- Practical Application 178
**The ADSystemInfo Object** 179
<table>
<thead>
<tr>
<th>Contents</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Shell Object</td>
<td>180</td>
</tr>
<tr>
<td>Methods and Properties</td>
<td>180</td>
</tr>
<tr>
<td>The Shortcut Object</td>
<td>187</td>
</tr>
<tr>
<td>Methods and Properties</td>
<td>187</td>
</tr>
<tr>
<td>Practical Application</td>
<td>188</td>
</tr>
<tr>
<td>Objects in Detail</td>
<td>189</td>
</tr>
<tr>
<td>Summary</td>
<td>191</td>
</tr>
<tr>
<td>12 Working with the File System</td>
<td>193</td>
</tr>
<tr>
<td>The FileSystemObject Library</td>
<td>193</td>
</tr>
<tr>
<td>Working with Drives</td>
<td>194</td>
</tr>
<tr>
<td>Working with Drive Objects</td>
<td>195</td>
</tr>
<tr>
<td>Working with Folders</td>
<td>197</td>
</tr>
<tr>
<td>Working with Folder Objects</td>
<td>198</td>
</tr>
<tr>
<td>Folder Attributes</td>
<td>200</td>
</tr>
<tr>
<td>Properties That Are Objects</td>
<td>202</td>
</tr>
<tr>
<td>Working with Files</td>
<td>203</td>
</tr>
<tr>
<td>Working with File Objects</td>
<td>204</td>
</tr>
<tr>
<td>Reading and Writing Text Files</td>
<td>206</td>
</tr>
<tr>
<td>Other FSO Methods and Properties</td>
<td>209</td>
</tr>
<tr>
<td>Creating a Log File Scanner</td>
<td>211</td>
</tr>
<tr>
<td>Summary</td>
<td>218</td>
</tr>
<tr>
<td>13 Putting It All Together: Creating Your First Script from Scratch</td>
<td>219</td>
</tr>
<tr>
<td>Designing the Script</td>
<td>219</td>
</tr>
<tr>
<td>Gathering Facts</td>
<td>220</td>
</tr>
<tr>
<td>Defining Tasks</td>
<td>221</td>
</tr>
<tr>
<td>Outlining the Script</td>
<td>222</td>
</tr>
<tr>
<td>Writing Functions and Subroutines</td>
<td>223</td>
</tr>
<tr>
<td>Identifying Candidate Modules</td>
<td>223</td>
</tr>
<tr>
<td>Writing the Filename Formatting Function</td>
<td>223</td>
</tr>
<tr>
<td>Writing the Main Script</td>
<td>228</td>
</tr>
<tr>
<td>Identifying Potential Errors</td>
<td>231</td>
</tr>
<tr>
<td>Testing the Script</td>
<td>234</td>
</tr>
<tr>
<td>Analyzing the Results</td>
<td>236</td>
</tr>
<tr>
<td>Adding Debug Code</td>
<td>236</td>
</tr>
<tr>
<td>Modifying the Script</td>
<td>240</td>
</tr>
<tr>
<td>Completing the Script</td>
<td>241</td>
</tr>
<tr>
<td>Summary</td>
<td>244</td>
</tr>
</tbody>
</table>
## Contents xi

Using the WMI Tools ........................................................................ 291
  Scriptomatic ........................................................................ 292
The Easy Way to Write WMI Scripts ................................................. 293
  Find the Class ...................................................................... 293
  Write the Query, Test the Query .............................................. 294
  Write the Script .................................................................... 295
  Test the Script ...................................................................... 297
Summary .................................................................................... 297

18 Querying Basic WMI Information ................................................... 299
  The WMI Query Language (WQL) .............................................. 299
    WMI Query Basics ................................................................ 300
    Boolean Operators ................................................................ 301
    Comparison Operators......................................................... 302
    Associators, References, and Keys ........................................ 303
  Determining What to Query .................................................... 304
  Testing the Query ................................................................... 306
  Writing the Query in VBScript .................................................. 308
  Using the Query Results ............................................................ 312
  Alternative Methods .................................................................. 314
Summary .................................................................................... 316

19 Querying Complex WMI Information ............................................ 317
  Understanding WMI Relationships .............................................. 317
  Associating WMI Instances ....................................................... 321
    ASSOCIATORS OF ................................................................... 321
    REFERENCES OF .................................................................... 323
    Using WHERE with ASSOCIATIONS OF and REFERENCES OF ... 324
  Writing the Query .................................................................... 325
  Testing the Query .................................................................... 325
  Writing the Query in VBScript .................................................. 326
  Another Example ....................................................................... 330
Summary .................................................................................... 333

20 Putting It All Together: Your First WMI/ADSI Script ...................... 335
  Designing the Script .................................................................. 335
  Writing Functions and Subroutines .............................................. 340
  Writing the Main Script .............................................................. 343
  Testing the Script .................................................................... 351
Summary .................................................................................... 355
Contents

26 Debugging Tips, Tools, and Techniques 421
Types of Bugs ........................................... 421
Preventing Syntax Errors ................................. 422
Use a VBScript Editor—and Pay Attention! .... 422
Use Option Explicit ..................................... 423
Reuse Your Code ......................................... 424
Debugging Logic Errors .................................... 425
Do Not Spaghetti-Debug! ............................... 425
Break It Down ............................................ 427
Get Inside the Script ...................................... 428
Get Out of the Script ..................................... 432
What About a Debugger? .............................. 434
Summary .................................................. 435

27 Logon and Logoff Scripts 439
NT and Active Directory Logon Scripts ........ 440
Active Directory–Specific Logon Scripts ....... 446
Active Directory Logoff Scripts ................. 451
Summary .................................................. 453

28 Windows and Domain Administration Scripts 455
Automating User Creation ............................ 455
Finding Inactive Users .................................. 464
Collecting System Information .................... 469
Templates .................................................. 474
Summary .................................................. 480

29 Network Administration Scripts 481
Shutting Down Remote Computers ............... 481
Listing Remote Shares ................................. 484
### WMI and ADSI Scripts

<table>
<thead>
<tr>
<th>Script Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The All-Purpose WMI Query Script</td>
<td>497</td>
</tr>
<tr>
<td>The All-Purpose WMI Update Script</td>
<td>499</td>
</tr>
<tr>
<td>The All-Purpose ADSI Object Creation Script</td>
<td>500</td>
</tr>
<tr>
<td>The All-Purpose ADSI Object Query Script</td>
<td>502</td>
</tr>
<tr>
<td>The All-Purpose ADSI Object Deletion Script</td>
<td>502</td>
</tr>
<tr>
<td>Mass Password Changes with ADSI</td>
<td>503</td>
</tr>
<tr>
<td>Updating Service Accounts</td>
<td>505</td>
</tr>
<tr>
<td>Summary</td>
<td>508</td>
</tr>
</tbody>
</table>

### Appendix

**Administrator’s Quick Script Reference**  
509

**Index**  
523
About the Author

**Don Jones** is an internationally recognized scripting guru, speaker, and author. He serves as the Director of Projects and Services for SAPIEN Technologies, where his primary job is to drive the development of new products and services for Windows administrative scripting. Don is the founder of ScriptingAnswers.com, the web’s friendliest community for Windows scripting. Don has written more than 30 books on information technology, including *Managing Windows with VBScript and WMI* (Addison-Wesley; the first edition of this book), *Windows Administrator’s Automation Toolkit* (Microsoft Press), *Advanced VBScript for Windows Administrators* (Microsoft Press), and *Windows PowerShell: TFM™* (SAPIEN Press). Don heads SAPIEN Technologies’ Las Vegas office, speaks at a half-dozen technical conferences each year, and contributes monthly content to Microsoft *TechNet Magazine*. 
Dedication

To Alex and Ferdinand: Thanks for having me.

Acknowledgments

Book projects always go more smoothly with an experienced team—and of the major publishing houses, let me tell you that Pearson (Addison-Wesley and Sams) has consistently had the best teams. I’m indebted to them for the opportunity to produce this new, revised edition of my original scripting book.

I would also like to thank my technical review panel made up of Dan Cazzulino, Jim Christopher, Doug Ellis, Jeffery Hicks, Bob Reselman, and Rob van der Woude. Without their valuable commentary and feedback, the book would not have been what it is today.

Support on the home front is important, too: Thanks to Chris for being wonderfully patient, and thanks to Alex, Ferdinand, Spoon, and Margaret for giving me the time to work on this lengthy revision. Thanks also to my ferrets, Pepper, Patch, and Nutmeg, who were wonderfully understanding—not—when “daddy” couldn’t play right then because he was typing.

And thanks, perhaps most of all, to Microsoft, for realizing at long last how important VBScript is to the community of Windows administrators, for deciding to continue to include it in future versions of Windows, and for giving the TechNet “Scripting Guys” full-time permission to produce samples, answer questions, and, in general, really promote administrative scripting to the world.

Don Jones
SAPIEN Technologies
May 2007
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write me directly to let me know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book’s title and author as well as your name and phone or email address. I will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: consumer@samspublishing.com
Mail: Mark Taber
      Associate Publisher
      Sams Publishing
      800 East 96th Street
      Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at www.samspublishing.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
Introduction

Microsoft introduced Visual Basic, Scripting Edition—commonly known as VBScript—in the mid-1990s, positioning it as a native replacement for Windows’ aging command-line batch language, which was based on Microsoft’s earliest operating system, MS-DOS. VBScript was intended to be easy to learn, powerful, and flexible. The language was included as an add-on to Windows 95 and Windows NT 4.0, was an optional installation component included in Windows 98, and was included in all editions of Windows Me, Windows 2000, Windows XP, and Windows Server 2003.

Software developers immediately seized upon VBScript for web programming, particularly in Active Server Pages, Microsoft’s rapid-development programming framework for the web. However, Windows administrators—one of VBScript’s initial target audiences—were left cold. VBScript seemed to be much more complicated than administrators’ beloved MS-DOS-based batch language, and many didn’t see the need to learn an entirely new batch language.

When Windows 2000 and Active Directory came along, however, administrators found that Windows administration had become a great deal more complex. Suddenly, administrators were searching for Resource Kits and other utilities that offered automated administration, especially for repetitive tasks. Active Directory enabled the use of VBScript for logon and logoff scripts, which seemed to promise more advanced-use environment manipulation. At around the same time, Microsoft’s naiveté in releasing a powerful language like VBScript with absolutely no security controls resulted in a huge wave of high-impact VBScript-based viruses, forcing administrators to lock down their environments and remove VBScript as an option both for viruses and for administrative tools.

As a regular speaker at some of the country’s top technical conferences that focus on Windows technologies, including TechMentor, the past few years I’ve given half- and full-day sessions on VBScripting for Windows administrators, and the sessions have been incredibly popular. In these sessions, I try to provide just enough VBScript experience to make scripting possible, and then concentrate on accomplishing common administrative tasks with VBScript. I also cover the security concerns of VBScript and provide administrators with the means for safely using VBScript in their environments. This book is essentially a written form of those sessions, greatly expanded with more coverage of Windows Management Instrumentation and other advanced topics, and with more coverage of VBScript security issues and resolutions.

I’m not out to turn you into a programmer. In fact, one of the real successes of VBScript is that you don’t need to be a programmer to use it. Most of what you’ll be doing in this book involves using VBScript to tell Windows to do things for you; you’ll be able to ignore much of VBScript’s complexity, using it as a sort of electronic glue to combine various operating system functions.
It’s been four years since the original edition of this book, published as Managing Windows with VBScript and WMI. At the time, Windows administrators were really just discovering scripting and its potential to automate administrative tasks; since then, scripting and automation have taken off in a big way. Managers—not just administrators—realize that automation makes better use of skilled technical professionals, freeing them up from boring, repetitive tasks for new projects. That realization has led to the word scripting being added to many a high-end job description, and scripting is emerging as one of the most important differentiators between entry-level technicians and experienced professionals.

In the past four years, I’ve done a tremendous amount of work to promote scripting and education. I produced more than a dozen free Webcasts for Microsoft TechNet, launched a web community for administrative scripting called www.ScriptingAnswers.com, created two complete series of training videos for scripting (viewable at http://www.ScriptingTraining.com), and wrote nearly a half-dozen books on Windows scripting and automation (including two free ones from http://www.Realtimepublishers.com). Scripting is here to stay.

Who Should Read This Book?
The only assumption I have about you is that you already know how to administer some version of Microsoft Windows. You’ll find that most of the material in this book is suitable for Windows NT, Windows 2000, Windows Server 2003, and (as it’s known as of this writing) Windows “Longhorn” Server environments (that includes the client versions of these operating systems, such as Windows XP and Windows Vista), and it will continue to be useful through future versions of Windows. I do not assume that you have any background in programming, and I’m not going to give you a programming background.

You should have a desire to learn how to use what I call “the batch language of the twenty-first century” and a desire to move away from clumsier—and often more complex—batch files based on the MS-DOS batch language. Although some folks like to refer to batch files as scripts, I don’t; and when you see how easy and flexible VBScript is, you’ll understand why!

How to Use This Book
You can read this book in order from the Introduction to the Appendix. However, if you already have some experience with VBScript, or if you just want to dive right in to the more complete sample scripts, you can skip around as much as you want. This book is organized in the same way that I organize my live VBScripting sessions at conferences, so you might feel that it’s some time before you really get into the meat of scripting. I assure you, though, that each example in this book—starting in Chapter 1—is focused on Windows administration. You’ll get your feet wet right away!
To help you decide where to start, the following sections provide a brief overview of each chapter.

**Part I: Introduction to Windows Administrative Scripting**

Part I serves as an introduction to the world of scripting and provides you with a methodology for approaching administrative tasks from a scripting standpoint. One of the most difficult parts about producing new scripts from scratch is the “Where do I start?” factor, and this part provides you with a framework for figuring that out every time.

**Chapter 1: Scripting Concepts and Terminology**

As implied previously, administrative scripting isn’t hard-core programming. Instead, it’s using VBScript as a sort of electronic glue to secure various bits of the Windows operating system together. This chapter introduces you to those various bits and sets the stage with some basic terminology that you’ll use throughout this book.

**Chapter 2: Running Scripts**

Writing a script isn’t much fun if you can’t run the script! This chapter focuses on the technologies used to execute scripts. You might be surprised to learn how many different Microsoft products support scripting. This chapter shows you how far your scripting skills can really take you and also introduces you to some scripting tools that can make writing and debugging scripts a bit easier.

**Chapter 3: The Components of a Script**

This chapter presents a complete administrative script and then breaks it down line-by-line to explain its various components. Although this chapter isn’t necessary to learning administrative scripting, it will help you write scripts that are more reliable and easier to troubleshoot.

**Chapter 4: Designing a Script**

As mentioned previously, one of the toughest aspects about scripting can be figuring out where to start. This chapter provides you with a framework that you can use as a starting point for every new scripting project. This chapter also introduces you to some concepts that many scripting books ignore, such as planning for errors and creating a useful “resource kit” of script components that you can reuse throughout your scripting projects.

**Part II: VBScript Tutorial**

Part II serves as your official crash course to the VBScript language: just enough to make administration via script a possibility! The best part is that this part doesn’t use the trite “Hello, world” examples that books for software developers often start out with. Instead, every example is useful to you as a Windows administrator. This means you’ll produce simple, useful scripts at the same time you’re learning VBScript. What could be better?
Chapter 5: Functions, Objects, Variables, and More
This chapter shows you the basic building blocks of any script and introduces you to some sample scripts that use each building block in a particular administrative task. This is really the meat of administrative scripting, and you'll be able to write useful scripts when you're finished with this chapter.

Chapter 6: Input and Output
You can make your scripts more flexible by adding the ability to dynamically change computer, user, and domain names, along with other information. This chapter shows you how your script can collect information it needs to run and dynamically alter itself to take advantage of that information.

Chapter 7: Manipulating Numbers
This chapter explains how scripts can manipulate numbers, making it easier to create scripts that work with numeric data, such as user account data. It also introduces you to VBScript’s numeric data handling and conversion commands, putting you on the path to some great scripting techniques.

Chapter 8: Manipulating Strings
Strings—a fancy word for text data—are at the heart of most scripting tasks. This chapter shows you how VBScript deals with strings and how you can easily integrate them into your scripts.

Chapter 9: Manipulating Other Types of Data
Aside from text and numbers, your scripts might need to deal with dates, times, bytes, and other forms of data to accomplish specific administrative tasks. This chapter shows you how VBScript handles these other data types and how you can use them in your own scripts.

Chapter 10: Controlling the Flow of Execution
The best administrative scripts can respond to changing conditions with internal logic, called control-of-flow. This chapter shows you how your scripts can be made to evaluate various conditions and respond accordingly, perform repetitive tasks, and much more.

Chapter 11: Built-in Scripting Objects
Much of VBScript’s power comes from its capability to join various operating system objects, and this chapter introduces you to your first set of those objects. You’ll learn how to manipulate network information, map drives, and much more—pretty much everything you need to write effective logon scripts.

Chapter 12: Working with the File System
A common use of scripting is to manipulate files and folders, and this chapter introduces you to the VBScript FileSystemObject, which provides a complete object model for working with the file system. You’ll learn to build a utility that scans Internet Information Services (IIS) log files for error messages, a useful script for any environment!
Chapter 13: Putting It All Together: Creating Your First Script from Scratch
This is where you put everything from Part II together. You’ll create a script that rotates IIS log files, keeping the past 30 days worth of files in a special archive folder. This chapter guides you through the complete process of designing, writing, testing, and troubleshooting the script. In fact, it deliberately introduces some logic errors into the script so that you can see the debugging process in action.

Part III: Windows Management Instrumentation and Active Directory Services Interface
With the glue of VBScript under your belt, this part dives into the two most powerful technologies for administering Windows: Windows Management Instrumentation (WMI) and the Active Directory Services Interface (ADSI). These technologies provide administrative access to, and control over, nearly every aspect of the Windows operating system, from Windows NT to Windows Server 2003.

Chapter 14: Working with ADSI Providers
Despite its name, ADSI isn’t just for Active Directory. This chapter shows you how ADSI can be used to interface with NT, Active Directory, Novell NDS, Exchange Server, and other types of directory services. This chapter provides some basic examples of the types of tasks you can perform with ADSI to get you started.

Chapter 15: Manipulating Domains
With the ADSI basics out of the way, this chapter focuses on manipulating domain information in a script. You’ll learn how to query domain information, modify domain policies such as password length, and much more.

Chapter 16: Manipulating Users and Groups
This chapter shows you how to write scripts that query and modify user and group information. This is one of the most common tasks you’ll perform with VBScript, and this chapter includes plenty of useful examples.

Chapter 17: Understanding WMI
WMI provides a hook into just about every portion of the Windows operating system, making it an incredibly useful tool for administrative scripts. This chapter introduces you to WMI and shows you a preview of what you can use it for in your environment.

Chapter 18: Querying Basic WMI Information
Do you want to find out which users in your organization have a Pentium 4 computer? This chapter shows you how to write your own basic WMI queries, including those that involve remote machines. You’ll also learn basic WMI manipulation, which lets you modify local and remote machine settings from within a script.
Chapter 19: Querying Complex WMI Information
Some WMI queries are more complex, such as querying the IP addresses from multiple
network adapters in multiple remote computers. This chapter provides clear examples of
these more complex WMI tasks, helping you learn to write enterprise management
scripts.

Chapter 20: Putting It All Together: Your First WMI/ADSI Script
This is where it all comes together. This chapter walks you through the process of design-
ing, writing, testing, and debugging a complete WMI/ADSI script from scratch. You’ll
finish this chapter with a concrete example of the administrative capabilities of these
technologies, and then you’ll be ready to start writing your own scripts.

Chapter 21: Testing and Debugging WMI and ADSI Queries
Getting the perfect WMI or ADSI query is critical to the success of your scripts, so this
chapter focuses on tools you can use to develop those queries more interactively, test your
queries, and have them fully refined before pasting them into your scripts.

Part IV: Advanced Scripting Techniques
As you become a more experienced scripter, you’ll be ready to start saving time and be
more secure, with advanced techniques like script encryption, scripting components,
script security, and so forth. This part of the book gives you a comprehensive look at
each of these technologies and shows you how to put them into use in your own
environment.

Chapter 22: Modular Script Programming
If you find yourself cutting and pasting code—or worse, retyping it—this is the chapter
for you. This chapter introduces you to modular scripting concepts, which make it easier
to reuse code between various scripts, saving you time and effort! By way of example, this
chapter starts with a complex script that contains lots of useful code and then breaks it
down into easily reused modules.

Chapter 23: Scripts Packaging and Protection
Are you worried that others will peek into your scripts and steal your ideas? Script packag-
ing and other techniques help protect your scripts from both Peeping Toms and potential
misuse, so this chapter shows you how to set up, deploy, and use script packages within
your environment.

Chapter 24: Scripting Security
Some folks think Microsoft made a huge mistake when it included VBScript in the
Windows operating system, but others disagree. Properly configured, scripting can be as
safe as any other type of application. This chapter explains scripting security concepts and
introduces you to the tools that can make scripting a safe and valuable part of any
computing environment.
Chapter 25: Introduction to HTML Applications
HTML Applications, or HTAs, provide a way to mix VBScript and Hypertext Markup Language (HTML) code to produce graphical scripts that look almost like full Windows applications. They’re a great way to produce tools that you plan to share with less-experienced users or administrators. This chapter gives you a quick start in building HTAs, along with an explanation of how they differ from more traditional VBScript projects.

Chapter 26: Debugging Tips, Tools, and Techniques
By now, you’ll have seen your fair share of script bugs, and so this chapter shows you how to prevent them from happening, find them quickly when they do happen, and squash them just as quickly so that you can get on with your scripting.

Part V: Ready-to-Run Examples
This part is a great way to wrap up the book—with a whole section on ready-made sample scripts that you can start using in your own environment. In addition, these scripts—like every other script in this book—have complete, line-by-line explanations, making them a perfect reference guide as you start to create your own scripts from scratch.

Chapter 27: Logon and Logoff Scripts
This chapter presents more complex logon and logoff scripts and gives you some ideas for how scripting can make these important scripts more effective. Of course, the line-by-line explanations make each script a useful reference for customizing your own scripts.

Chapter 28: Windows and Domain Administration Scripts
Automating domain administration is probably one of the big reasons you started looking at scripting in the first place, so this chapter presents a number of examples of tasks that scripts can perform. The detailed explanations with each script will help you rip them apart and customize them for your own use.

Chapter 29: Network Administration Scripts
Network administration is ideally suited for scripting, and this chapter provides a handful of examples that show you what’s possible. The line-by-line explanations make it easy to put these into use in your own environment.

Chapter 30: WMI and ADSI Scripts
These can be the toughest scripts to write because of the complexity and flexibility of WMI and ADSI. This chapter provides you with several ready-to-use scripts for common tasks, such as querying WMI, creating users and groups, and more. These scripts can be easily modified and incorporated into your own scripts, saving you scripting time!
Appendix

Appendix: Administrator’s Quick Script Reference
One of the toughest parts about VBScript is that it contains so much functionality. It’s usually pretty easy to figure out what you want a script to do; the tough part is often figuring out how to make VBScript do it! This appendix provides an alphabetical list of common tasks and gives the VBScript commands that perform each task. You can use this reference along with the VBScript documentation to make designing and writing scripts much easier.

Preparing to Use This Book

Before you dive in, you should make sure that your computers are ready for VBScript. Fortunately, any computer with Windows 2000 or later is ready to go out of the box, and this book assumes that you’re doing your development work on either a Windows 2000–, Windows XP–, or Windows Server 2003–based computer.

Typographical Elements

Books on programming can benefit a great deal from easy-to-understand typestyles and elements like the ones explained here. These typestyles and elements are designed to make the text easier to follow and to call your attention to special concerns.

Monospaced type will be used to set off material that should be typed into the computer. For example, “select Run from the Start menu, type wbemtest, and click OK” sets off the menu selection, and what you need to type onscreen.

Blocks of code and code lines that appear within the text appear in a monospaced font, as in, “To change the contents of a variable, you can use Var1 = Trim(Var1).”

TIP

Tips provide shortcuts and other “insider advice” about scripting that you’ll find valuable.

NOTE

Notes provide cautions and other clarifications that will help you avoid problems or further clarify complex concepts.
You’ll also be directed to material that more thoroughly explains particular concepts, VBScript commands, and so forth. Although you might not be a big fan of flipping back and forth through a book, these cross-references allow you to remain focused within each chapter and guide you to more detailed explanations, when appropriate.

Finally, there are times when it is necessary to present an extended explanation of something that isn’t critical to the task at hand. In those cases, a sidebar is included. A sidebar is a cue that the information is useful, but it’s not really key to the main text; you’re welcome to skip the sidebar and come back to it later.

**Sidebars**
Sidebars make it easier to cover slightly off-topic information without distracting you from the main text.

**Sample Scripts**
Obviously, a book on scripting is going to have many code listings. To make these as useful as possible, each sample script is presented in a listing by itself with no comments.

**LISTING P1 A Sample Script**

```
'Get the user's name
sName = InputBox("What is your name?")

'Display the user's name
MsgBox "Your name is " & sName
```

After each script, any changes you might need to make to get the script running in your environment, such as changing computer or domain names, are presented.

**Sample Scripts—Explained**
For each script in this book, a line-by-line explanation of the script is included, so that you understand exactly what's going on. For example:

First, the sample script displays a dialog box where the user can type his name. By default, this dialog box includes an OK and Cancel button; this script does not provide any way to detect the Cancel button, so it is assumed the user will type something and click OK.

```
'Get the user's name
sName = InputBox("What is your name?")
```
Finally, the script uses the `MsgBox` statement to redisplay the user's name. Notice the use of the ampersand operator (&) to tack on the contents of the variable `sName`, which stores whatever the user typed into the input box.

```vbnet
'Display the user's name
MsgBox "Your name is " & sName
```

Walk-throughs like this one will help you become more familiar with VBScript, what each command does, and exactly how each sample script works.
By now, you should have a good idea of what WMI and ADSI can do for you. In this chapter, I’ll walk you through the complete design process for an entirely new script. This time, I’ll use both WMI and ADSI in the same script. The script’s job will be to check in on every computer in an Active Directory or NT domain and query some information about its operating systems. I want the script to output this information to a text file on a file server. The information I want to collect includes operating system version, service pack level, number of processors in the machine, maximum physical memory in the machine, and so forth. This is a useful way to quickly inventory a network and see what machines might need to be upgraded before deploying a new application, or to see what machines don’t have the latest service pack applied.

**Designing the Script**

My script is a reasonably complex undertaking, so it helps to break it down into manageable tasks. I need the script to do three things:

1. Query a list of computers from the domain.
2. Query information from each computer.
3. Write information out to a text file.

The last bit is probably the easiest. I can use the FileSystemObject to open a text file, write information to it, and then close the text file. Something like the following would work:
Dim oFSO, oFile
Set oFSO = CreateObject("Scripting.FileSystemObject")
Set oFile = oFSO.CreateTextFile("output.txt")
oFile.Write "Information"
oFile.Close

For more information on using the FileSystemObject, refer to Chapter 12, “Working with the File System.”

Querying a list of computers from the domain shouldn’t be too hard, either. If I want the script to work with both NT and Active Directory domains, I need to use the WinNT ADSI provider because only that provider works with both domains. I can query all of the objects in the domain, and then use an If/Then construct to work with only the computer objects. Code such as the following should do the trick:

Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

'is this object a computer?
If oObject.Class = "Computer" Then

'yes – do something with it

End If
Next

For more information on querying domains by using ADSI, see Chapter 14, “Working with ADSI Providers,” and Chapter 15, “Manipulating Domains.”

Pulling the operating system (OS) information is tougher. WMI seems like the way to go, but WMI has about three gazillion classes. Which one do I need? Fortunately, I have a way to cheat. My script editor includes a WMI Script Wizard.

Running the wizard displays the dialog box shown in Figure 20.1. The left side of the dialog box shows a list of every WMI class that my computer knows about. Scrolling through the list, I find that there’s a class named Win32_OperatingSystem. That seems like a good place to start.

Clicking the Win32_OperatingSystem class changes the dialog box to look like the one shown in Figure 20.2. Here, the wizard has filled in a sample script capable of querying information from the selected class. I see things like service pack level and operating system version, so this is probably the class I want. The wizard offers an Insert button to immediately insert this code into my script, and a Copy button to copy the code to the clipboard. Listing 20.1 shows the complete wizard code.
NOTE

I’ve added line breaks and line continuation characters (\_) to Listing 20.1 so that it will fit in this book.

FIGURE 20.1  The WMI Wizard starts with a list of all available WMI classes.

FIGURE 20.2  The wizard generates sample code to query the selected class.

LISTING 20.1  WizardCode.vbs. This code queries the Win32_OperatingSystem class and outputs all of the classes’ attributes and their values.

On Error Resume Next
Dim strComputer
Dim objWMIService
Dim colItems

strComputer = "."
Set objWMIService = GetObject("winmgmts:\" & _
   strComputer & "\root\cimv2")
Set colItems = objWMIService.ExecQuery( _
   "Select * from Win32_OperatingSystem",,48)
For Each objItem in colItems
   WScript.Echo "BootDevice: " & objItem.BootDevice
   WScript.Echo "BuildNumber: " & objItem.BuildNumber
   WScript.Echo "BuildType: " & objItem.BuildType
   WScript.Echo "Caption: " & objItem.Caption
   WScript.Echo "CreationClassName: " & objItem.CreationClassName
   WScript.Echo "CSCreationClassName: " & _
      objItem.CSCreationClassName
   WScript.Echo "CSDVersion: " & objItem.CSDVersion
   WScript.Echo "CSName: " & objItem.CSName
   WScript.Echo "CurrentTimeZone: " & objItem.CurrentTimeZone
   WScript.Echo "Debug: " & objItem.Debug
   WScript.Echo "Description: " & objItem.Description
   WScript.Echo "Distributed: " & objItem.Distributed
   WScript.Echo "EncryptionLevel: " & objItem.EncryptionLevel
   WScript.Echo "ForegroundApplicationBoost: " & _
   objItem.ForegroundApplicationBoost
   WScript.Echo "FreePhysicalMemory: " & _
      objItem.FreePhysicalMemory
   WScript.Echo "FreeSpaceInPagingFiles: " & _
      objItem.FreeSpaceInPagingFiles
   WScript.Echo "FreeVirtualMemory: " & objItem.FreeVirtualMemory
   WScript.Echo "InstallDate: " & objItem.InstallDate
   WScript.Echo "LargeSystemCache: " & objItem.LargeSystemCache
   WScript.Echo "LastBootUpTime: " & objItem.LastBootUpTime
   WScript.Echo "LocalDateTime: " & objItem.LocalDateTime
   WScript.Echo "Locale: " & objItem.Locale
   WScript.Echo "Manufacturer: " & objItem.Manufacturer
   WScript.Echo "MaxNumberOfProcesses: " & objItem.MaxNumberOfProcesses
   WScript.Echo "MaxProcessMemorySize: " & objItem.MaxProcessMemorySize
   WScript.Echo "Name: " & objItem.Name
   WScript.Echo "NumberOfLicensedUsers: " & objItem.NumberOfLicensedUsers
   WScript.Echo "NumberOfProcesses: " & objItem.NumberOfProcesses
   WScript.Echo "NumberOfUsers: " & objItem.NumberOfUsers
   WScript.Echo "Organization: " & objItem.Organization
The wizard's code pulls more information than I want, and it's displaying the information in message boxes, rather than writing them to a file, but the code makes a great place to start. I can easily modify it to meet my needs.

The script is designed! I identified the three major tasks that the script needs to be able to complete, and I've created some prototype code that can be adapted to the script's exact requirements. In short, I now know how to do everything I need; I just need to rearrange it and customize it.
What, No Wizard?

If you’re not using PrimalScript, there are some other tools you can use to make WMI scripting easier. In Chapter 18, “Querying Basic WMI Information,” for example, I introduced Microsoft’s Scriptomatic tool, which performs a similar function to the PrimalScript WMI Wizard. You can also dive into the WMI documentation in the MSDN Library (http://msdn.microsoft.com/library), which documents each WMI class and includes some scripting examples. Newer versions of PrimalScript do include an ADSI Wizard, but it doesn’t produce boilerplate code like the WMI Wizard does.

Writing Functions and Subroutines

The one bit of functionality that seems to be standalone is the code generated by the wizard, which will do my WMI querying for me. I might need to use that code in another script someday, and I’ll definitely be using it over and over in the script I’m writing now, so it makes sense to write it as a function.

I want the function to accept a computer name, query that computer for specific operating system information, and then compile all that information into a neatly formatted string. The function should return the string to the main script, which can then write it to a file or whatever.

Adapting the wizard’s code isn’t too difficult. Listing 20.2 shows my new GetOSInfo() function. Note that this isn’t intended to be run as a standalone script; as a function, it must be called by another script, which must provide the name of the computer to connect to as the function’s input parameter.

LISTING 20.2 GetOSInfo.vbs. This function queries a computer’s operating system information and returns the results in a string.

Function GetOSInfo(sComputer)

  'declare variables
  Dim objWMIService
  Dim colItems
  Dim strOutput

  'get WMI service
  Set objWMIService = GetObject("winmgmts:\" & _
    strComputer & ":\root\cimv2")

  'get item collection
  Set colItems = objWMIService.ExecQuery( _
    "Select * from Win32_OperatingSystem",,48)

  'init output string
  sOutput = String(70,"-") & vbCrLf

  For Each item in colItems
    sOutput = sOutput & item.String & vbCrLf
  Next item

  'return result
  Set objWMIService = Nothing
  Set colItems = Nothing
  WScript.Echo sOutput

End Function
sOutput = sOutput & sComputer & vbCrLf

'append info to output string
For Each objItem in colItems
    strOutput = strOutput & "BuildNumber: " & _
        objItem.BuildNumber & vbCrLf
    strOutput = strOutput & "BuildType: " & _
        objItem.BuildType & vbCrLf
    strOutput = strOutput & "Caption: " & _
        objItem.Caption & vbCrLf
    strOutput = strOutput & "EncryptionLevel: " & _
        objItem.EncryptionLevel & vbCrLf
    strOutput = strOutput & "InstallDate: " & _
        objItem.InstallDate & vbCrLf
    strOutput = strOutput & "Manufacturer: " & _
        objItem.Manufacturer & vbCrLf
    strOutput = strOutput & "MaxNumberOfProcesses: " & _
        objItem.MaxNumberOfProcesses & vbCrLf
    strOutput = strOutput & "MaxProcessMemorySize: " & _
        objItem.MaxProcessMemorySize & vbCrLf
    strOutput = strOutput & "Name: " & _
        objItem.Name & vbCrLf
    strOutput = strOutput & "NumberOfLicensedUsers: " & _
        objItem.NumberOfLicensedUsers & vbCrLf
    strOutput = strOutput & "NumberOfProcesses: " & _
        objItem.NumberOfProcesses & vbCrLf
    strOutput = strOutput & "NumberOfUsers: " & _
        objItem.NumberOfUsers & vbCrLf
    strOutput = strOutput & "OSProductSuite: " & _
        objItem.OSProductSuite & vbCrLf
    strOutput = strOutput & "OSType: " & _
        objItem.OSType & vbCrLf
    strOutput = strOutput & "OtherTypeDescription: " & _
        objItem.OtherTypeDescription & vbCrLf
    strOutput = strOutput & "Primary: " & _
        objItem.Primary & vbCrLf
    strOutput = strOutput & "ProductType: " & _
        objItem.ProductType & vbCrLf
    strOutput = strOutput & "RegisteredUser: " & _
        objItem.RegisteredUser & vbCrLf
    strOutput = strOutput & "SerialNumber: " & _
        objItem.SerialNumber & vbCrLf
strOutput = strOutput & _
I didn’t have to do much to adapt the script. First, I deleted all the lines that I didn’t want in my script. I changed all the WScript.Echo commands to strOutput = strOutput & &, which appends the information into a string rather than displays it in a message box. I also added & vbCrLf to the end of each line, which adds a carriage return and linefeed character. Those help keep the final output file looking nice.

I also dressed up the code at the beginning of the function.

I added some comments to document the code—PrimalScript isn’t so good about that—and I initialized my sOutput variable. I also started sOutput off to contain a line of 70 hyphens, and the name of the computer I’m querying. These extra touches help make the final output file easier to read and more useful.
Writing the Main Script

The function was probably the toughest part to write; with that out of the way, I can adapt my prototype code to create the main script, shown in Listing 20.3.

LISTING 20.3  MainScript.vbs. Queries the domain, creates the output file, and calls the custom function I already wrote.

```vbs
Dim sDomain
sDomain = InputBox("Enter domain to inventory")

'connect to domain and retrieve 'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)

'get the filesystemobject
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")

'open an output file
Dim oOutput
Set oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")

'run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

'is this object a computer?
If oObject.Class = "Computer" Then

'yes - get computer name
sComputerName = oObject.Name

'get OS info
sDetails = GetOSInfo(sComputerName)

'write info to the file
oOutput.Write sDetails

End If
Next

'close the output file
oOutput.Close
```
Listing 20.3  Continued

' release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \server1\public\output.txt"

I'll provide my usual walk-through of this script in a bit; for now, try to pick out the adapted pieces of prototype code. Notice where I'm querying the domain, opening and writing to the text file, closing the text file, and calling the GetOSInfo() function.

Inventorying the Domain

Listing 20.4 shows the complete, ready-to-run script. Get this ready to run, but don't execute it just yet. In the next section, I'll cover testing and troubleshooting this script.

Listing 20.4  InventoryDomain.vbs. The complete domain inventory script.

' get domain name
Dim sDomain
sDomain = InputBox("Enter domain to inventory")

' connect to domain and retrieve
'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)

' get the filesystemobject
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")

' open an output file
Dim oOutput
oOutput = oFSO.CreateTextFile("\server1\public\output.txt")

' run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

' is this object a computer?
If oObject.Class = "Computer" Then

' yes - get computer name
LISTING 20.4  Continued

sComputerName = oObject.Name

'get OS info
sDetails = GetOSInfo(sComputerName)

'write info to the file
oOutput.Write sDetails

End If
Next

'close the output file
oOutput.Close

'release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = Nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \server1\public\output.txt"

Function GetOSInfo(sComputer)

'declare variables
Dim objWMIService
Dim colItems
Dim strOutput

'get WMI service
Set objWMIService = GetObject("winmgmts:" & _
   strComputer & "\root\cimv2")

'get item collection
Set colItems = objWMIService.ExecQuery(_
   "Select * from Win32_OperatingSystem",,48)

'init output string
sOutput = String(70,"-") & vbCrLf
sOutput = sOutput & sComputer & vbCrLf

'append info to output string
For Each objItem in colItems
strOutput = strOutput & "BuildNumber: " & __
objItem.BuildNumber & vbCrLf
strOutput = strOutput & "BuildType: " & __
objItem.BuildType & vbCrLf
strOutput = strOutput & "Caption: " & __
objItem.Caption & vbCrLf
strOutput = strOutput & "EncryptionLevel: " & __
objItem.EncryptionLevel & vbCrLf
strOutput = strOutput & "InstallDate: " & __
objItem.InstallDate & vbCrLf
strOutput = strOutput & "Manufacturer: " & __
objItem.Manufacturer & vbCrLf
strOutput = strOutput & "MaxNumberOfProcesses: " & __
objItem.MaxNumberOfProcesses & vbCrLf
strOutput = strOutput & "MaxProcessMemorySize: " & __
objItem.MaxProcessMemorySize & vbCrLf
strOutput = strOutput & "Name: " & __
objItem.Name & vbCrLf
strOutput = strOutput & __
"NumberOfLicensedUsers: " & __
objItem.NumberOfLicensedUsers & vbCrLf
strOutput = strOutput & __
"NumberOfProcesses: " & __
objItem.NumberOfProcesses & vbCrLf
strOutput = strOutput & __
"NumberOfUsers: " & __
objItem.NumberOfUsers & vbCrLf
strOutput = strOutput & __
"OSProductSuite: " & __
objItem.OSProductSuite & vbCrLf
strOutput = strOutput & __
"OSType: " & __
objItem.OSType & vbCrLf
strOutput = strOutput & __
"OtherTypeDescription: " & __
objItem.OtherTypeDescription & vbCrLf
strOutput = strOutput & __
"Primary: " & __
objItem.Primary & vbCrLf
strOutput = strOutput & __
"ProductType: " & __
objItem.ProductType & vbCrLf
strOutput = strOutput & __
"RegisteredUser: " & __
objItem.RegisteredUser & vbCrLf
strOutput = strOutput & __
"SerialNumber: " & __
objItem.SerialNumber & vbCrLf
strOutput = strOutput & __
"ServicePackMajorVersion: " & __
objItem.ServicePackMajorVersion & vbCrLf
strOutput = strOutput & __
"ServicePackMinorVersion: " & __
LISTING 20.4  Continued

   =objItem.ServicePackMinorVersion & vbCrLf
    strOutput = strOutput & "Version: " & _
    objItem.Version & vbCrLf
    strOutput = strOutput & "WindowsDirectory: " & _
    objItem.WindowsDirectory & vbCrLf
Next

' return results
GetOSInfo = sOutput
End Function

You need to change where this script puts its output file before using it in your environment. The script prompts for the domain name, so you won’t have to make any changes there.

**Inventorying the Domain—Explained**  The script starts by prompting for the domain name. This allows the script to be used in a multidomain environment. The domain name is stored in a string variable.

' get domain name
Dim sDomain
sDomain = InputBox("Enter domain to inventory")

Next, the script uses ADSI to connect to the domain and retrieve a list of all domain objects. This might be a lengthy operation in a large domain because computer, user, and all other objects are included in the results.

' connect to domain and retrieve 'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)

The script creates a new FileSystemObject and assigns it to a variable.

' get the filesystemobject
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")

The script now creates a new text file by using the FileSystemObject’s CreateTextFile method. The method returns a TextStream object, which is assigned to the variable oOutput.

' open an output file
Dim oOutput
oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")
oDomain now represents all of the objects in the domain; I’ll use a For Each/Next loop to iterate through each object in turn. Within the loop, oObject will represent the current object.

'run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

Because oDomain contains more than just computers, I need to check each object to see if its Class property equals “Computer.” That way, I can just work with the computer objects and skip the rest.

'is this object a computer?
If oObject.Class = "Computer" Then

For objects that are a computer, I pull the computer name into a variable. Then, I assign the results of GetOSInfo() to variable sDetails. Finally, I write sDetails to the output text file using the TextStream object's Write method. I then close up the loop with Next to move on to the next object in the domain.

'yes - get computer name
sComputerName = oObject.Name

'get OS info
sDetails = GetOSInfo(sComputerName)

'write info to the file
oOutput.Write sDetails

End If
Next

When I'm done with all the objects, I close the output file, release all the objects I created by setting them equal to Nothing, and then display a simple completion message.

'close the output file
oOutput.Close

'release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = Nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \server1\public\output.txt"
Here’s that function I wrote earlier. It starts with a basic variable declaration.

```vbscript
Function GetOSInfo(sComputer)
    'declare variables
    Dim objWMIService
    Dim colItems
    Dim strOutput

    Next is pure wizard code, which uses GetObject to connect to the specified computer's WMI service.
    'get WMI service
    Set objWMIService = GetObject("winmgmts:\" & 
        strComputer & ":\root\cimv2")

    After I am connected, I execute a query to retrieve the Win32_OperatingSystem class.
    'get item collection
    Set colItems = objWMIService.ExecQuery( 
        "Select * from Win32_OperatingSystem",,48)

    I set up my output string to include a line of hyphens and the current computer name.
    'init output string
    sOutput = String(70,"-") & vbCrLf
    sOutput = sOutput & sComputer & vbCrLf

    Finally, I append the WMI information to the output string.
    'append info to output string
    For Each objItem in colItems
        strOutput = strOutput & "BuildNumber: " & 
            objItem.BuildNumber & vbCrLf
        strOutput = strOutput & "BuildType: " & 
            objItem.BuildType & vbCrLf
        strOutput = strOutput & "Caption: " & 
            objItem.Caption & vbCrLf
        strOutput = strOutput & "EncryptionLevel: " & 
            objItem.EncryptionLevel & vbCrLf
        strOutput = strOutput & "InstallDate: " & 
            objItem.InstallDate & vbCrLf
        strOutput = strOutput & "Manufacturer: " & 
            objItem.Manufacturer & vbCrLf
        strOutput = strOutput & "MaxNumberOfProcesses: " & 
            objItem.MaxNumberOfProcesses & vbCrLf
        strOutput = strOutput & "MaxProcessMemorySize: " & 
            objItem.MaxProcessMemorySize & vbCrLf
    Next objItem
End Function
```
strOutput = strOutput & "Name: " & objItem.Name & vbCrLf
strOutput = strOutput & "NumberOfLicensedUsers: " & objItem.NumberOfLicensedUsers & vbCrLf
strOutput = strOutput & "NumberOfProcesses: " & objItem.NumberOfProcesses & vbCrLf
strOutput = strOutput & "NumberOfUsers: " & objItem.NumberOfUsers & vbCrLf
strOutput = strOutput & "OSProductSuite: " & objItem.OSProductSuite & vbCrLf
strOutput = strOutput & "OSType: " & objItem.OSType & vbCrLf
strOutput = strOutput & "OtherTypeDescription: " & objItem.OtherTypeDescription & vbCrLf
strOutput = strOutput & "Primary: " & objItem.Primary & vbCrLf
strOutput = strOutput & "ProductType: " & objItem.ProductType & vbCrLf
strOutput = strOutput & "RegisteredUser: " & objItem.RegisteredUser & vbCrLf
strOutput = strOutput & "SerialNumber: " & objItem.SerialNumber & vbCrLf
strOutput = strOutput & "Version: " & objItem.Version & vbCrLf

Next

With the main script finished, I return the output string as the function’s result.

' return results
GetOSInfo = sOutput

End Function

There you have it—a nice, easy-to-use administrative script that uses both WMI and ADSI to accomplish a useful task.
Testing the Script

If you jumped ahead and already tried to execute the final script, you realize that it’s flawed. If you haven’t, go ahead and give it a whirl now. Take a few minutes to see if you can track down the problem. There are actually three errors, and here are some hints:

- One is a simple typo.
- One is a sort of logic error, where something isn’t being used properly for the situation.
- The last one is a typo, and could have been avoided if I had followed my own advice from earlier in the book.

Can you find them all? The first one is an easy mistake: I simply forgot a closing parenthesis.

`connect to domain and retrieve`  
' a list of member objects  
Dim oDomain  
Set oDomain = GetObject("WinNT://" & sDomain

The correct code should be `Set oDomain = GetObject("WinNT://" & sDomain). The next one’s a bit trickier.

' open an output file  
Dim oOutput  
oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")

Can you see it? I’m using `oOutput` to represent an object, but I forgot to use the `Set` keyword when making the assignment. VBScript requires `Set` whenever you’re assigning an object to a variable. The corrected code looks like this:

' open an output file  
Dim oOutput  
Set oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")

The last error is tricky, too. It’s in the `GetOSInfo()` function.

Function GetOSInfo(sComputer)

  ' declare variables  
  Dim objWMIService  
  Dim colItems  
  Dim strOutput

  ' get WMI service  
  Set objWMIService = GetObject("winmgmts:\" & _  
    strComputer & "\root\cimv2")
Did you find it? The problem is that I used the wizard-generated code, which uses “str” as a prefix for string variables. I’m in the habit of using the shorter prefix “s” for string variables, and that’s where my problem lies. In the function definition, I declared sComputer, but in the line of code that connects to the WMI service, I used strComputer. I continued using sComputer elsewhere, so strComputer is wrong. Here’s the corrected code snippet:

Function GetOSInfo(sComputer)
    ‘declare variables
    Dim objWMIService
    Dim colItems
    Dim strOutput

    ‘get WMI service
    Set objWMIService = GetObject("winmgmts:\" & _
        sComputer & "\root\cimv2")

The problem with this error is that it doesn’t cause a problem for the script; the script will execute just fine. You just won’t get any results because the script would try to connect to a computer named “”. I mentioned that I could have avoided this problem by following my own advice. Had I included Option Explicit, VBScript would have produced an error on the offending line of code because strComputer wasn’t declared. sComputer, on the other hand, is implicitly declared because it’s part of a function declaration. You’ll notice that I did the same thing with strOutput and sOutput, meaning they’ll have to be corrected, too.

Just to make sure you’ve got it all, Listing 20.5 includes the complete, corrected script. Remember that this script is also available in the book’s downloads at http://www.ScriptingAnswers.com/books.asp.

LISTING 20.5  InventoryDomain2.vbs. This corrected script produces the expected results.

'get domain name
Dim sDomain
sDomain = InputBox("Enter domain to inventory")

'connect to domain and retrieve
'a list of member objects
Dim oDomain
Set oDomain = GetObject("WinNT://" & sDomain)

'get the filesystemobject
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")

'open an output file
Dim oOutput
Set oOutput = oFSO.CreateTextFile("\\server1\public\output.txt")

'run through the objects
Dim oObject, sComputerName, sDetails
For Each oObject In oDomain

'is this object a computer?
If oObject.Class = "Computer" Then

'yes - get computer name
sComputerName = oObject.Name

'get OS info
sDetails = GetOSInfo(sComputerName)

'write info to the file
oOutput.Write sDetails

End If
Next
'close the output file
oOutput.Close

'release objects
Set oOutput = Nothing
Set oFSO = Nothing
Set oObject = nothing
Set oDomain = Nothing

'display completion message
WScript.Echo "Output saved to \server1\public\output.txt"

Function GetOSInfo(sComputer)

'declare variables
Dim objWMIService
Dim colItems
Dim strOutput

'get WMI service
Set objWMIService = GetObject("winmgmts:\" & _
  sComputer & "+\root\cimv2")

'get item collection
Listing 20.5  Continued

Set colItems = objWMIService.ExecQuery( _
    "Select * from Win32_OperatingSystem",,48)

' init output string
strOutput = String(70,".") & vbCrLf
strOutput = strOutput & sComputer & vbCrLf

' append info to output string
For Each objItem in colItems
    strOutput = strOutput & "BuildNumber: " & objItem.BuildNumber & vbCrLf
    strOutput = strOutput & "BuildType: " & objItem.BuildType & vbCrLf
    strOutput = strOutput & "Caption: " & objItem.Caption & vbCrLf
    strOutput = strOutput & "EncryptionLevel: " & objItem.EncryptionLevel & vbCrLf
    strOutput = strOutput & "InstallDate: " & objItem.InstallDate & vbCrLf
    strOutput = strOutput & "Manufacturer: " & objItem.Manufacturer & vbCrLf
    strOutput = strOutput & "MaxNumberOfProcesses: " & objItem.MaxNumberOfProcesses & vbCrLf
    strOutput = strOutput & "MaxProcessMemorySize: " & objItem.MaxProcessMemorySize & vbCrLf
    strOutput = strOutput & "Name: " & objItem.Name & vbCrLf
    strOutput = strOutput & "NumberOfLicensedUsers: " & objItem.NumberOfLicensedUsers & vbCrLf
    strOutput = strOutput & "NumberOfProcesses: " & objItem.NumberOfProcesses & vbCrLf
    strOutput = strOutput & "NumberOfUsers: " & objItem.NumberOfUsers & vbCrLf
    strOutput = strOutput & "OSProductSuite: " & objItem.OSProductSuite & vbCrLf
    strOutput = strOutput & "OSType: " & objItem.OSType & vbCrLf
    strOutput = strOutput & "OtherTypeDescription: " & objItem.OtherTypeDescription & vbCrLf
    strOutput = strOutput & "Primary: " & objItem.Primary & vbCrLf
    strOutput = strOutput & "ProductType: " & objItem.ProductType & vbCrLf
LISTING 20.5  Continued

```vbscript
strOutput = strOutput & "RegisteredUser: " & vbCrLf
strOutput = strOutput & "SerialNumber: " & vbCrLf
strOutput = strOutput & "ServicePackMajorVersion: " & vbCrLf
strOutput = strOutput & "ServicePackMinorVersion: " & vbCrLf
strOutput = strOutput & "Version: " & vbCrLf
strOutput = strOutput & "WindowsDirectory: " & vbCrLf
Next

'return results
GetOSInfo = strOutput
End Function
```

Testing a large script like this is much easier with the Script Debugger. You can spot lines that are causing trouble just by following the execution path.

For more information on the Script Debugger, see Chapter 13, “Putting It All Together: Creating Your First Script from Scratch.” You can also read up on the Script Debugger in the VBScript documentation at http://msdn.microsoft.com/scripting.

**Summary**

Pulling together ADSI and WMI into a single script offers some powerful functionality. More important, though, the example in this chapter should make you feel more comfortable with the sometimes-daunting task of creating a script from scratch. Just break down the tasks that need to be completed, and then develop some prototype code for each task. Use wizards, examples from the web, or samples from this book to help create prototype code. After all, there’s no sense reinventing the wheel when there’s a large library of samples on the web and in this book to work with!

With your task list and prototype out of the way, you can start assembling the script. Write functions and subs to perform repetitive tasks, or tasks that you might want to reuse in future scripts. Write the main script, and then start testing. With this methodology in mind, most scripts can be whipped together quickly!
This page intentionally left blank
This page intentionally left blank
Symbols

0: Unknown drive type, 196
1: Removable drive type, 196
2: Fixed drive type, 196
3: Network drive type, 196
4: CD-ROM drive type, 196
5: RAM Disk drive type, 196

A

Abort button, 107
Abs( ) function, 126
ACEs (access control entries), WMI (Windows Management Instrumentation), 300
Active Directory Services Interface (ADSI). See ADSI (Active Directory Services Interface)
Active Server Pages (ASPs), 384, 410
ActiveX Data Objects (ADO). See ADO (ActiveX Data Objects)
ActiveX scripting languages, 16-18
AD (Active Directory)
   attributes, 260
   classes, 260
   OUs (organizational units)
      creating, 260-261
      deleting, 263-264
      modifying, 261-263
      querying, 263
   users and computers, targeting, 477-479
AD (Active Directory) scripts
   logoff scripts, 451-454
   logon scripts, 440-446, 453-454
      specific logon scripts, 446-451
AddPrinterConnection method (Network object), 177

addresses
  IP addresses, third octet, 63
  local IP addresses, attaining, 61-63

AddUsers.vbs listing (28.1), 456-463
AddUsersFromXLS.vbs listing (3.2), 49-50
AddUsersFromXLS.vbs listing (3.3), 50-53
AddWindowsPrinterConnection method (Network object), 177
ADLogon1.vbs listing (27.3), 446-448
ADLogon2.vbs listing (27.4), 448-451

administrative scripts
  domains, 455, 480
  inactive users, finding, 464-469
  network administrative scripts, 481, 495
    listing hot fixes, 492-495
    listing software, 492-495
    open file monitoring, 486-488
    remote computer shut downs, 481-483
    remote MSI package uninstalls, 489-492
    remote share listings, 484-486
  system information, collecting, 469-474
  templates, 474-479
  user creation, automating, 455-463
ADO (ActiveX Data Objects), 384, 455
  Excel spreadsheets, queries, 455-463
Adobe Dreamweaver, 410
ADOs (ActiveX Data Objects), 422
ADSI (Active Directory Services Interface), 168, 173, 247, 257, 335, 355, 497, 508
  domains
    manipulating, 269
    OUs (organizational units), 260-264
    preloading, 264-269
    querying, 257-259
    reconfiguring, 259
  functions, writing, 340-342
mass password changes, executing, 503-505
object creation scripts, creating, 500-502
object deletion scripts, creating, 502-503
object query scripts, creating, 502
objects, 247-249
provider names, 248
providers, 254-255
  GC (Global Catalog), 254
  IIS, 254
  LDAP, 248-254
  NDS (NetWare Directory Services), 254
  NWCOMPAT, 254
  OLE DB, 254
  WinNT, 248-252
queries, debugging, 366-370
remote shares, listing, 484-486
scripts
  debugging, 357-358
  designing, 335-340
  testing, 351-355
  writing, 343-350
subroutines, writing, 340-342
ADSIDebug.vbs listing (21.2), 368-370
ADSystemInfo object, 179-180
advanced arithmetic, VBScript, 118-121
Advanced VBScript for Windows Administrators, 409, 420
American National Standards Institute (ANSI), 299
And Boolean operator, 156
ANSI (American National Standards Institute), 299
antivirus software, 406
AppActivate method (Shell object), 185
applications
graphical applications, 410
HTAs (HTML Applications), 409-410, 420
building, 410
event handlers, 410, 418-420
event-driven scripting, 410
functional code, 411-412
HTML code conversion, 414-416
Mshta.exe, 411
scheduled tasks, 410
script code, 416-418
security context, 409
user interface design, 412-413
writing, 411-420

“Are you sure?” dialog boxes, 104
Argument method (Shortcut object), 187
Arithmetic, VBScript, 116-118
advanced arithmetic, 118-121
Boolean math, 121-126
arrays, 145, 148, 152
creating, 149-151
dynamic arrays, 149
manipulating, 149-151
multidimensional arrays, 149
numeric variables, array elements, 151-152
VBScript, 149
Asc( ) function, 118, 130
ASPs (Active Server Pages), 384, 410
assigning data to variables, 82-83
associations
WMI (Windows Management Instrumentation), 319
instances, 321-325
WQL (WMI Query Language), 317
associators, WQL (WMI Query Language), 303
ASSOCIATORS OF query, WMI instances, associating, 321-325
Atn( ) function, 118
attributes, AD (Active Directory), 260
Attributes property (Folder objects), 200-202
Attributes property (FSO File object), 204
automating
tasks, domains, 455, 480
user creation, domains, 455-456, 459-463
AvailableSpace property (Drive object), 195
B
batch files, 2, 13, 20
best practices, MsgBox statement, 106-107
Boolean math, VBScript, 121-126
Boolean operators, 156-157
WQL (WMI Query Language), 301-302
bugs, scripts, 421-422, 435
logic errors, debugging, 425-435
prevention, 31
syntax errors, preventing, 422-425
building HTAs (HTML Applications), 410
BuildPath function, FSO (FileSystemObject), 209-210
built-in commands, 75
built-in functions, 42
custom functions, compared, 88-92
built-in functions. See intrinsic functions
built-in subroutines. See statements
buttons
Abort, 107
dialog boxes, Cancel button, 105
Ignore, 107
message boxes, 103
Retry, 107
bytes, 145, 152
converting, 152
VBScript, 152

How can we make this index more useful? Email us at indexes@samsPublishing.com
C

Calculator, Scientific mode, 124
Cancel button, dialog boxes, 105
candidate modules, identifying, 223
CBool( ) function, 126
CByte( ) function, 152
CCur( ) function, 126
CD-ROM drive type, 196
CDbl( ) function, 126
certificate rules, 400
characters
  double quotation marks, strings, 129-130
  line continuation characters (_), 79, 337
  underscore (_) character, 79, 106
CheckFolder.vbs listing (12.3), 200-202
CheckGroupMembership.vbs listing (16.1), 276-277
checking group membership, 61
CheckRoutes.vbs listing (17.2), 297
code
  bugs, 421-422, 435
  logic errors, 425-435
  syntax errors, 422-425
debuge code, adding, 236, 239
debugging
  logic errors, 425-435
  syntax errors, 422-425
derror-handling code, 70
functional code, HTAs (HTML Applications), 411-412
HTML code, converting for HTA (HTML Applications), 414-416
  inline code, 45
  pseudocode, 56
  reusable code, 31
reusing, 424-425
script code, adding to HTA (HTML Applications), 416, 418
code wizards, 32
coercion, numeric data types, 126-127
collections, COM objects, 96
CollectSysInfo.vbs listing (28.3), 469-474
COM (Component Object Model) objects, 18, 20, 76, 94-98, 439
collections, 96
methods, 95-96
properties, 95
sample object (Tree object), 97-98
scripting with, 98-99
command-line input, message boxes, asking for, 109-110
command-line parameters as input, 110-114
command-line scripts, 23-25
  execution of, 110
  parameters, parsing, 111-114
running, 111
command-line utilities, 110
CMEM (Common Information Model), WMI
  (Windows Management Information), 284
Clsn( ) function, 126
classes
  AD (Active Directory), 260
  finding, WMI (Windows Management Instrumentation), 293-294
WMI (Windows Management Instrumentation), 284-285, 305
  associations, 319-325
  interrelated classes, 300, 317-320
  Win32_OperatingSystem class, 336-337, 349, 482-483
WQL (WMI Query Language), interrelated classes, 317
Classid parameter (WSCs), 383
CLng( ) function, 126
commands
   built-in commands, 75
   DatePart, 224
   Format, 223
   Len, 225
   MapPrinter, 67
   net, 60
   NET TIME, 445
   Option Explicit, 81
   Right, 224
   Sleep, 186
   WScript.Echo, 213-214
comments, 48-53
Common Information Model (CIM), WMI (Windows Management Instrumentation), 284
comparison operators, WQL (WMI Query Language), 302-303
complex information
   WMI (Windows Management Instrumentation), 300, 317-319
   WQL (WMI Query Language), 317
complex information queries
   testing, 325-326
   writing, 325
   VBscript, 326-333
Component Object Model (COM) objects. See COM (Component Object Model) objects
CompressAll.vbs listing (19.2), 330-333
car programming, 75
ComputerName method (Network object), 178
computers, minimum requirements, 8
concatenating, strings, 135-139
conditional errors, 70
conditional execution statements
   If/Then, 156-161, 170-171
   Select/Case, 161-162
conditional executions, scripts, 155-162
   If/Then conditional execution statement, 156-161, 170-171
   Select/Case conditional execution statement, 161-162
connections, remote computers, WMI (Windows Management Instrumentation), 287-289
Const statement, 80
constants, 466
   variables, compared, 80
constructs
   If/Then, 69
   Select/Case, 48
consumers, WMI (Windows Management Instrumentation), 285-286
control-of-flow statements, 155
converting
   bytes, 152
   data types
      to numeric data, 127
      to strings, 142-143
   date and time data, 147
   HTML code, HTAs (HTML Applications), 414-416
      numeric data types, 126-127
Copy method (Folder objects), 198
Copy method (FSO File objects), 204
CopyFolder method (FSO), 197
copying-and-pasting, script, 373-377
   WSCs (Windows Script Components), 378, 382-384, 387
Cos( ) function, 118
cosines, 115
Create method, 260-261
Create method (objDomain), 248
CreateFolder method (FSO), 197
CreateObject, 247-248
CreateShortcut method (Shell object), 183, 187
CreateTextFile method (Folder objects), 198, 347
critical icon, 107
CScript.exe, 23-25, 110, 114
    command-line parameters, 114
CScript.exe executable, deleting, 405-406
CStr( ) function, 130, 143
currency number types, 116
custom functions
    intrinsic functions, compared, 48, 88-92
    main script, 45-46
custom statements. See subroutines
custom subroutines
    creating, 92-94
    main script, 45-46
    statements, compared, 48
data coercion, 83-85
data types, 82, 145
    assumptions, 127
    date and time, 145-148
        converting, 147
        numeric data, converting to, 127
    numeric data types
        converting, 126-127
        converting to, 127
        strings, converting to, 142-143
Database Browser, 32
date and time data type, 145-148
    converting, 147
Date( ) function, 42, 85, 146, 227
DateAdd( ) function, 147-148, 223
DateCreated property (FSO File object), 204
DateDiff( ) function, 87, 467
DateLastAccessed property (FSO File object), 204
DateLastModified property (FSO File object), 204
DatePart command, 224
DatePart( ) function, 118, 136, 146
dates
    future dates, working with, 147-148
    getting, 146-147
    past dates, working with, 147-148
    VBScript, 145-148
Day( ) function, 146
debug code, adding, 236, 239
debuggers, 434-435
debugging
    ADSI queries, 366-370
    first script, 35-36
    scripts, 357-358, 435
    logic errors, 425-435
    spaghetti debugging, 425-426
    WMI queries, 358-370
declaring variables, 79-81
default script extensions, Windows XP, 399
defaults, message boxes, 103
defining function input/output, 223
Delete method, 263
Delete method (Folder objects), 198
Delete method (FSO File objects), 204
Delete method (objDomain), 249
DeleteFolder method (FSO), 197
DeleteMail method, 96
deleting OUs (organizational units), 263-264
deployment, scripts, 17
Description method (Shortcut object), 187
Description parameter (WSCs), 383
designing
scripts, 55, 219-220, 335-340
    fact gathering, 220-221
    script outlines, 222
    task definitions, 221-222
    user interfaces, HTAs (HTML Applications), 412-413
designing scripts
    modules, creating, 63-69
    task lists, creating, 56-58
    tools, selecting, 58-63
Desktop Management Task Force (DMTF), WMI (Windows Management Instrumentation), 283
DHCP (Dynamic Host Configuration Protocol), 325
dialog boxes
    “Are you sure?”, 104
    Cancel buttons, 105
    creating, 101
    Script Package, 395
    Starting, 22
dialog builders, 101
digitally signing scripts, 402-405
Dim keyword, 89, 149
Dim statements, 80, 98
Dim sVariable( ) function, 149
directories, LDAP, fully qualified domain names, 253
DisableUser.vbs listing (5.1), 77-79
DisableUser2.vbs listing (5.2), 90-92
display options, MsgBox statement, 103
displaying messages, 59-60, 101-108
    complex messages, 105-106
    WScript.Echo, 107-108
DLLs (dynamic link libraries), 16, 76
DMTF (Desktop Management Task Force), WMI (Windows Management Instrumentation), 283
Do Until statement, 424
Do Until/Loop construct, 163-164
Do While/Loop construct, 162-163
Do/Loop Until construct, 163-164
Do/Loop While construct, 162-163
documentation, VBScript, downloading, 59
documentation (scripts), 48-53
domain attributes, LDAP provider, 258
domains, 257
    inactive users, finding, 464-469
    inventorying script for, 343-355
    lists of computers, targeting, 474-477
    lockout policies, changing, 259
    manipulating, 269
    names, FQDNs (fully qualified domain names), 271
    OUs (organizational units), 260
        creating, 260-261
        deleting, 263-264
        modifying, 261-263
        querying, 263
    passwords, changing, 259
    preloading, 264-269, 277-281
    querying, 257-259
    system information, collecting, 469-474
    tasks, automating, 455, 480
    time zones, 472
    user creation, automating, 455-456, 459-463
    users, targeting, 477-479
DoObject subroutine, 476-477
DoObject_User subroutine, 479
double quotation marks, strings, 129-130
doubles, 115
downloading VBScript documentation, 59
Dreamweaver, 410
Drive object (FSO), 194-197
    drive types, 196
    properties, 195-196

How can we make this index more useful? Email us at indexes@sams publishing.com
Drive property (FSO File object), 204

drive types, Drive object (FSO), 196

DriveLetter property (Drive object), 195
drives, 196

mapping, 60-61

DriveType property (Drive object), 195
debugging scripts, syntax errors, 422-425

Duncan, Larry, 61
dynamic arrays, 149

Dynamic Host Configuration Protocol (DHCP), 325
dynamic link libraries (DLLs). See DLLs (dynamic link libraries)

E

ECMAScript, 382

ElseIf.vbs listing (10.1), 160-161

Encapsulation, functions and subroutines, 373

encoded script, 389
detriment, 394

encrypted code, compared, 390

running, 393

writing, 390-393

encoding script, 397

encrypted code, encoded code, compared, 390

EnumNetworkDrive method (Network object), 176

EnumPrinterConnection method (Network object), 177

Environment method (Shell object), 183-184

error-handling code, 70

ErrorMsg Subroutine listing (5.3), 93

errors

conditional errors, 70

logic errors, 70, 219

planning for, 70-72

scripts, identifying, 231-233

syntax errors, 70

errors, scripts, 421-422, 435

logic errors, debugging, 425-435

syntax errors, preventing, 422-425

Escape( ) function, 130

ESP (Evolved Script Package), 394-397
event handlers, 410

HTAs (HTML Applications), wiring up, 418-420
event-driven programs, 410
event-driven scripting, HTAs (HTML Applications), 410

Evolved Script Package (ESP), 394-397

Excel spreadsheets, queries, ADO (Active Directory Objects), 455-463

exclamation mark icon, 106

Exec method (Shell object), 180-182

ExecQuery( ) method, 285, 422

Execute( ) method, 423

executions, scripts, 155, 167-170

conditional executions, 155-162

loops, 162-167

exiting loops, 166-167

Exp( ) function, 119

ExpandEnvironmentStrings method (Shell object), 184

Expression, 410

Extensible Markup Language (XML). See XML (Extensible Markup Language)
extensions (file), WScript.exe associations, removing, 405
file extensions, WScript.exe associations, removing, 405

File object (FSO), 194

File objects, FSO (FileSystemObject), 203-206
  methods, 204-206
  properties, 204-206

file servers, connecting to, WinNT provider, 249

file systems, FSO (FileSystemObject), 193, 209, 218
  BuildPath function, 209-210
  Drive object, 194-197
  File object, 194
  File objects, 203-206
  Folder object, 194
  Folder objects, 198-203
  folders, 197-198
  GetTempName method, 210
  log file scanner creation, 211-218
  object libraries, 193-194
  TaxStream object, 194
  TextStream objects, 206-209, 212, 216

File Transfer Protocol (FTP). See FTP (File Transfer Protocol)

FileProperties.vbs listing (12.4), 204

files
  batch files, 13, 20
  CScript.exe executable, deleting, 405-406
  FSO (FileSystemObject), 209, 218
    BuildPath function, 209-210
    File objects, 203-206
    GetTempName method, 210
    log file scanners, 211-218
    TextStream objects, 206-209, 212, 216
  open files, monitoring, 486-488
  text files, 14
  WScript.exe executable
    deleting, 405-406
    renaming, 406

FileSystem property (Drive object), 195

FileSystemObject (FSO), 193, 209, 218, 247, 335, 347, 509
  BuildPath function, 209-210
  Drive object, 194-197
    drive types, 196
    properties, 195-196
  File objects, 194, 203-206
    methods, 204-206
    properties, 204-206
  Folder objects, 194, 198-199
    Attributes property, 200-202
    methods, 198
    properties as objects, 202-203
  folders, 197-198
    methods, 197
    GetTempName method, 210
  log file scanners, creating, 211-218
  object libraries, 193-194
  TaxStream object, 194
  TextStream objects, 206-209, 212, 216

FileSystemObject's GetFolder method, 99

FindOldUsers.vbs listing (28.2), 464-469

first script
  debugging, 35-36
  running, 35
  writing, 33-35

Fixed drive type, 196

Folder objects, FSO (FileSystemObject), 194, 198-199
  Attributes property, 200-202
  Folder objects, 198-203
    methods, 197-198
    properties as objects, 202-203
FolderExists method (FSO), 197
For Each/Next construct, 165-166
For/Next construct, 164-165
forceLogoff domain attribute (LDAP provider), 258
Format commands, 223
FormatCurrency( ) function, 130, 142
FormatDateTime( ) function, 130, 141-142
FormatLogFileName function, writing, 223-227
FormatLogFileName Function listing (13.1), 225-227
FormatNumber( ) function, 130, 142
FormatPercent( ) function, 130, 141-142
formatting strings, 141-142
FQDNs (fully qualified domain names), 271
fractional values, numbers, 115
FreeSpace property (Drive object), 195
FROM property (WQL query), 290, 300
FrontPage, 410
FSO (FileSystemObject), 193, 209, 218
  BuildPath function, 209-210
  Drive object, 194-197
    drive types, 196
    properties, 195-196
  File object, 194
  File objects, 203-206
    methods, 204-206
    properties, 204-206
  Folder object, 194
  Folder objects, 198-199
    Attributes property, 200-202
    methods, 198
    properties as objects, 202-203
  folders, 197-198
    methods, 197
  GetTempName method, 210
  log file scanners, creating, 211-218
  object libraries, 193-194
  TaxStream object, 194
  TextStream objects, 206-209, 212, 216
FTP (File Transfer Protocol), 32
FullName method (Shortcut object), 187
fully qualified domain names, objects, LDAP directories, 253
fully qualified domain names (FQDNs). See
FQDNs (fully qualified domain names)
Function keyword, 48
Function statement, 89, 118
functional code, HTAs (HTML Applications), writing, 411-412
functions, 39, 42-43, 76, 85, 99
  Abs( ), 126
  as statements, 94
  Asc( ), 118, 130
  Atn( ), 118
  BuildPath, 209-210
  CBool( ), 126
  CByte( ), 152
  CCur( ), 126
  CDbl( ), 126
  Chr( ), 130
  CInt( ), 126
  CLng( ), 126
  Cos( ), 118
  CStr( ), 130, 143
custom functions, 88-92
  Date( ), 42, 85, 146, 227
  DateAdd, 223
  DateAdd( ), 147-148
  DateDiff( ), 87, 467
  DatePart( ), 118, 136, 146
  Day( ), 146
  Dim sVariable( ), 149
encapsulation, 373
Escape( ), 130
Exp( ), 119
experimenting with, 139
FormatCurrency( ), 130, 142
FormatDateTime( ), 130, 141-142
FormatNumber( ), 130, 142
FormatPercent( ), 130, 141-142
GetIP( ), 44, 67-68
GetOSInfo( ), 340, 342, 344, 348, 351
GetSPVer( ), 418
Hour( ), 146
input, defining, 223
input parameters, 85-89
InputBox( ), 108-110, 114, 216
InStr( ), 44, 63, 67, 86, 131, 133-134, 143, 213
InStrRev( ), 131, 134
Int( ), 126
intrinsic functions, 42, 88-92
IsArray( ), 152
IsMember( ), 46
IsMemberOf function, 43
IsMemberOf function( ), 43
IsMemberOf( ), 43-44, 46, 67, 69
Join( ), 151
LCase( ), 131, 139, 217
Left( ), 131, 133, 143
Len( ), 131
Log, 468
Log( ), 119
LTrim( ), 131
main script
    custom functions, 45-46
    intrinsic functions, 47-48
MakePW, 117
Mid( ), 44, 63, 67, 118, 131, 143
Minute( ), 146
modular, 43
Month( ), 146
MonthName( ), 147
MsgBox( ), 94, 101-110, 114, 491
NameCase( ), 140
Now( ), 146
On Error Goto 0, 70
output, defining, 223
output values, 87-88
Randomize, 119
Replace( ), 131, 140-141, 143
Right( ), 131, 134, 143
Rnd( ), 119, 126
Round( ), 509
RTrim( ), 131
Second( ), 146
Sin( ), 119
Space( ), 131
Split( ), 150
Sqr( ), 119
StrComp( ), 131
string manipulation, 130-131
StrReverse( ), 131
Tan( ), 119
time( ), 146
Trim( ), 131
UCase( ), 131, 136, 139-140
Unescape( ), 131
Weekday( ), 146
WeekdayName( ), 147
when to use, 45
writing, 223-227, 340, 342
Year( ), 146

future dates, working with, 147-148

How can we make this index more useful? Email us at indexes@samspublishing.com
GC (Global Catalog) provider, ADSI, 254
Get method, 263
    user information, querying, 273-274
Get method (objDomain), 248
Get( ) method, 87
GetEx method, 263
GetFolder method (FSO), 197
GetInfo method, 263
GetInfo.vbs listing (6.1), 111-114
GetIP( ) function, 44, 67-68
GetMail( ) method, 96
GetObject statement, 248
GetOSInfo( ) function, 340-351
GetOSInfo.vbs listing (20.2), 340-342
GetParentFolderName method (FSO), 197
GetSpecialFolder method (FSO), 197
GetSPVer( ) function, 418
GetTempName method, FSO
    (FileSystemObject), 210
Go subroutine, 418
graphical applications, 410
graphical input, message boxes, asking for,
    108-109
Graphical User Interfaces (GUIs). See GUIs
    (Graphical User Interfaces)
Group Policy object (GPO) Editor, 406
groups, 271
    creating, 271
        LDAP provider, 273
        WinNT provider, 272
memberhsip, checking, 61, 275-277
GUIs (graphical user interfaces), 409
    HTA (HTML Applications), designing,
        412-413
H
hash rules, 400
Hicks, Jeffrey, 409
hierarchies, WMI (Windows Management Instrumentation), 283-285
    consumers, 285-286
    providers, 285-286
    versions, 286
hosts, script hosts, 14-16
hot fixes, listing, 492-495
HotKey method (Shortcut object), 187
Hour( ) function, 146
HTAs (HTML Applications), 23, 409-410, 420
    building, 410
    event handlers, 410
        wiring up, 418-420
    event-driven scripting, 410
    functional code, writing, 411-412
    HTML code, converting, 414-416
    Mshta.exe, 411
    scheduled tasks, 410
    script code, adding, 416, 418
    security context, 409
    user interface, designing, 412-413
    writing, 411-416, 418-420
HTML (Hypertext Markup Language), 23
HTML Applications (HTAs). See HTAs (HTML Applications)
    HTML code, HTAs (HTML Applications),
        converting, 414-416
Hungarian notation, 85
IconLocation method (Shortcut object), 187
icons
- critical icon, 107
- exclamation mark icon, 106
- message boxes, 103
- question mark icon, 106
Identifying possible errors listing (4.6), 71
IDEs (Integrated Development Environments), 29, 44
If/Then conditional execution statements, 156-161, 170-171
- nesting, 159-160
If/Then constructs, 69
Ignore button, 107
IIS (Internet Information Services), 219
IIS log rotation tool, creating, 219
- candidate modules, identifying, 223
  FormatLogFileName function, writing, 223-227
- functions, writing, 223-227
  main script, writing, 228-234
  script, testing, 234-243
  script design, 219-222
  subroutines, writing, 223-227
IIS provider, ADSI, 254
“impersonate” directives, queries, 315
inline code, 45
input
- command-line parameters, 110-114
  functions, defining, 223
  message boxes, asking for, 108-110
input parameters, functions, 85-89
InputBox( ) function, 108-110, 114, 216
installation
  Script Encoder, 390
  WMI (Windows Management Instrumentation), 290-291
instances, WMI instances, associating, 321-325
InStr( ) function, 44, 63, 67, 86, 131-134, 143, 213
InStrRev( ) function, 131, 134
Int( ) function, 126
integers, 115
Integrated Development Environments (IDEs).
  See IDEs (Integrated Development Environments)
interaction, objects, 75
interfaces, 193
Internet Information Services (IIS), 219
Internet shortcuts, creating, 188
Internet Zone rules, 401
interpretation, scripting, 17
interpreted languages, VBScript, 14
interrelated classes
  WMI (Windows Management Instrumentation), 300, 317-320
  WQL (WMI Query Language), 317
intrinsic functions, 42
  custom functions, compared, 48, 88-92
  main script, 47-48
intrinsic subroutines, 44
  main script, 47-48
inventories, domains, script for, 343-344, 347-352, 355
Inventory.vbs listing (29.6), 494-495
InventoryDomain.vbs listing (20.4), 344-350
InventoryDomain2.vbs listing (20.5), 352-355
IP addresses
  local IP addresses, attaining, 61-63
  third octet, attaining, 63
ipconfig command-line utility, 110
IsArray( ) function, 152
IsMember( ) function, 46
IsMemberOf( ) function, 43-46, 67, 69
IsReady property (Drive object), 195
Java Virtual Machines (JVMs), 17
JavaScript, 382
Join( ) function, 151
JScript, 382
JVMs (Java Virtual Machines), 17

keywords, 422
Dim, 89, 149
Function, 48
Preserve, 150
Sub, 48
WHERE, 136
WQL (WMI Query Language), 303
Kixtart, 439

languages
strongly typed languages, 82
weakly typed languages, 82

LCase( ) function, 131, 139, 217
LDAP (Lightweight Directory Access Protocol), 252
directories, fully qualified domain names, 253
paths, 252
LDAP provider, 271
ADSI, connections, 248
domain attributes, 258
domains, querying, 257-259
group membership, checking, 275-277
groups, creating, 273
user settings, changing, 274-275
users
creating, 273
information queries, 273-274
Left( ) function, 131-133, 143
Len command, 225
Len( ) function, 131
libraries
object libraries, FSO (FileSystemObject), 193-194
script libraries, creating, 72
See LDAP (Lightweight Directory Access Protocol)
line continuation characters, 34, 79
line continuation characters (\_), 337
listing
hot fixes, 492, 494-495
remote shares, ADSI, 484-486
software, 492-495
listings, 9
2.1 (WhoHas.vbs Displays the User or Users Who Have a File Open), 33
3.1 (LoginScript.vbs Sample Logon Script), 40
3.2 (AddUsersFromXLS.vbs. Creates users from an Excel spreadsheet), 49-50
3.3 (AddUsersFromXLS.vbs. Creates users from an Excel spreadsheet), 50-53
4.1 (Logon script task list), 56
4.2 (Logon script task list v2), 56
4.3 (Logon script task list v3), 57
4.4 (Retrieve IP Address.vbs), 62
4.5 (LogonScript.vbs), 64-69
4.6 (Identifying possible errors), 71
5.1 (DisableUser.vbs), 77-79
5.2 (DisableUser2.vbs), 90-92
5.3 (ErrorMsg Subroutine), 93
log rotation tool, creating, 219
  candidate modules, identifying, 223
  FormatLogFileName function, writing, 223-227
  functions, writing, 223-227
  main script, writing, 228-234
  script, testing, 234-236, 239-243
  script design, 219-222
  subroutines, writing, 223-227
Log Rotation.vbs listing (13.2), 228-231
Log( ) function, 119
LogEvent method (Shell object), 184
logic errors, 70, 219
LoginScript.vbs sample logon script, 39, 42
LoginScript.vbs Sample Logon Script listing (3.1), 40
logoff scripts, 439
  AD (Active Directory), 451-453
Logoff.vbs listing (27.5), 451-453
Logon script task list listing (4.1), 56
Logon script task list v2 listing (4.2), 56
Logon script task list v3 listing (4.3), 57
logon scripts, 439
  AD scripts, 440-446, 453-454
  specific logon scripts, 446-451
  NT domains, calling, 446
  NT scripts, 440-446, 453-454
Logon.vbs listing (11.1), 178-179
Logon1.vbs listing (27.1), 440-444
Logon2.vbs listing (27.2), 444-445
LogonScript.vbs listing (4.5), 64-69
LogRotation2.vbs listing (13.3), 232-234
LogRotation3.vbs listing (13.4), 234-236
LogRotation4.vbs listing (13.5), 237-239
LogRotation5.vbs listing (13.6), 241-243
long integers, 115
loops, 162-167
  Do Until/Loop construct, 163-164
  Do While/Loop construct, 162-163
  Do/Loop Until construct, 163-164
  Do/Loop While construct, 162-163
  exiting, 166-167
  For Each/Next construct, 165-166
  For/Next construct, 164-165
LTrim( ) function, 131
main script, 44-45
  custom functions and subroutines, 45-46
decision making in, 48
  intrinsic functions, 47-48
  statements, 47-48
testing, 234-243
  writing, 228-350
main scripts, errors, identifying, 231-233
MainScript.vbs listing (20.3), 343-344
MakePW function, 117
MakePW.vbs listing (7.1), 117-118
MakePW.vbs listing (7.2), 120-121
Managing Windows with VBScript and WMI, 2
MapDrive routine, 68
MapDrive subroutine, 44, 67
MapNetworkDrive method (Network object), 175-176
mapping
  drives, 60-61
  printers, 61
MapPrinter command, 67
mass password changes, ADSI, performing, 503-505
MassPass.vbs listing (30.1), 503-505
mathematical operators, 116-118
  Boolean operators, 121, 156-157
maxPwdAge domain attribute (LDAP provider), 258
Melissa virus, 399
membership, groups, checking, 61, 275-277
message boxes, 101
  buttons, 103
  command-line input, asking for, 109-110
  defaults, 103
  graphical input, asking for, 108-109
  icons, 103
  input, asking for, 108-110
  modality, 103
  system icons, 102
messages
  displaying, 59-60, 101-108
    complex messages, 105-106
    WScript.Echo, 107-108
  nonerror messages, 106
  nonwarning messages, 106
methods
  COM objects, 95-96
  Create, 260
  Create method, 261
  CreateShortcut, 187
  CreateTextFile, 347
  Delete, 263
  DeleteMail, 96
  ExecQuery( ), 285, 422
  Execute( ), 423
  FileSystemObject’s GetFolder, 99
  FSO (FileSystemObject)
    CopyFolder, 197
    CreateFolder, 197
  DeleteFolder, 197
  File objects, 204-206
  Folder objects, 198
  FolderExists, 197
  folders, 197
  GetFolder, 197
  GetParentFolderName, 197
  GetSpecialFolder, 197
  MoveFolder, 197
  Get, 263, 273-274
  Get( ), 87
  GetEx, 263
  GetInfo, 263
  GetMail( ), 96
Network object
  AddPrinterConnection, 177
  AddWindowsPrinterConnection, 177
  ComputerName, 178
  EnumNetworkDrive, 176
  EnumPrinterConnection, 177
  MapNetworkDrive, 175-176
  RemoveNetworkDrive, 176
  RemovePrinterConnection, 178
  SetDefaultPrinter, 178
  UserDomain, 178
  UserName, 178
  objDomain variable, 248
  OpenTextFile, 468
  Put, 263, 274-275
  PutEx, 275
  SetInfo, 260-263
Shell object
  AppActivate method, 185
  CreateShortcut method, 183
  Environment method, 183-184
  Exec method, 180-182
  ExpandEnvironmentStrings method, 184

How can we make this index more useful? Email us at indexes@samspublishing.com
LogEvent method, 184
Popup method, 187
RegDelete method, 184-185
RegRead method, 184
RegWrite method, 184-185
Run method, 180-182
SendKeys method, 185-187
SpecialFolders method, 182-183
Shortcut object, 187-188
Shutdown, 308, 311, 483
Uninstall, 492
Microsoft Expression, 410
Microsoft FrontPage, 410
Microsoft TechNet, 2
Microsoft’s MSDN Library, 420
Mid( ) function, 44, 63, 67, 118, 131, 143
minimum requirements, computers, 8
minPwdAge domain attribute (LDAP provider), 258
minPwdLength domain attribute (LDAP provider), 258
Minute( ) function, 146
modality, message boxes, 103
modifying scripts, 240-241
modular functions, 43
modularization, scripts, 373-377, 388
detriments, 377
reasons for, 377
modularization, scripts, WSCs (Windows Script Components), 378
XML (Extensible Markup Language), 378-384, 387
modules
creating, 63-69
searching for, 373-377
monospaced type, 8
Month( ) function, 146
MonthName( ) function, 147
Move method (Folder objects), 198
Move method (FSO File objects), 204
MoveFolder method (FSO), 197
MSDN Library (Microsoft), 420
MsgBox statement, 24-25, 84, 94, 101-108, 114, 170
best practices, 106-107
display options, 103
message boxes, displaying, 101-108
return values, 104
MsgBox( ) function, 94, 101-108, 114, 491
input asking for, 108-110
messages, displaying, 101-108
Mshta.exe, HTAs (HTML Applications), 411
MSI packages, remote MSI packages, uninstalling, 489-492
multidimensional arrays, 149

N
Name property (FSO File object), 204
NameCase( ) function, 140
NameDrives.vbs listing (12.1), 196
names, variables, 227
native scripts, 17
NDS (NetWare Directory Services) provider, ADSI, 254
nesting If/Then constructs, 159-160
net command, 60
.NET Framework, 18
NET TIME command, 445
network administrative scripts, 481, 495
hot fixes, listing, 492-495
open files, monitoring, 486-488
remote computers, shutting down, 481, 483
remote MSI packages, uninstalling, 489-492
remote shares, listing, 484-486
software, listing, 492-495

Network drive type, 196

Network object (WSH), 173-175
  AddPrinterConnection method, 177
  AddWindowsPrinterConnection method, 177
  ComputerName method, 178
  EnumNetworkDrive method, 176
  EnumPrinterConnection method, 177
  MapNetworkDrive method, 175-176
  practical application, 178-179
  RemoveNetworkDrive method, 176
  RemovePrinterConnection method, 178
  SetDefaultPrinter method, 178
  UserDomain method, 178
  UserName method, 178

Nimda virus, 399

nonerror messages, 106
nonwarning messages, 106
notations, Hungarian notation, 85

Notepad, 219
  scripting with, 25-26

Nothing statement, 251

Now( ) functions, 146

NT domains, VBScript logon scripts, calling, 446

NT scripts, logon scripts, 440-446, 453-454

numbers
  currency number types, 116
doubles, 115
  fractional values, 115
  integers, 115
  singles, 115
  VBScript, 115-116
  whole numbers, 115

numeric data types, converting, 126-127
numeric variables, arrays, 151-152

NWCOMPAT provider, ADSI, 254

O

objDomain variable, 248
Object Browser (WMI), 291-292
object creation scripts, ADSI object creation script, 500-502
object deletion scripts, ADSI object creation script, 502-503
object interfaces, 76
object libraries, FSO (FileSystemObject), 193-194
object query scripts, ADSI object creation script, 502

object-oriented programming language, 75

objects, 99, 189-190
  ADSI (Active Directory Services Interface), 247-249
  ADSystem object, 179-180
  COM objects, 94-98
    collections, 96
    methods, 95-96
    properties, 95
    sample object (Tree object), 97-98
    scripting with, 98-99
  FSO (FileSystemObject), 209, 218
    Drive object, 194-197
    File object, 194
    File objects, 203-206
    Folder object, 194
    Folder objects, 197-203
    TextStream object, 194
    TextStream objects, 206-209, 212, 216
  fully qualified domain names, LDAP directories, 253
  interaction, 75
  names, 194
  Network object, 173-175
    AddPrinterConnection method, 177
    AddWindowsPrinterConnection method, 177
ComputerName method, 178
EnumNetworkDrive method, 176
EnumPrinterConnection method, 177
MapNetworkDrive method, 175-176
practical application, 178-179
RemoveNetworkDrive method, 176
RemovePrinterConnection method, 178
SetDefaultPrinter method, 178
UserDomain method, 178
UserName method, 178
Shell object, 173, 180
  AppActivate method, 185
  CreateShortcut method, 183
  Environment method, 183-184
  Exec method, 180-182
  ExpandEnvironmentStrings method, 184
  LogEvent method, 184
  Popup method, 187
  RegDelete method, 184-185
  RegRead method, 184
  RegWrite method, 184
  Run method, 180
  SendKeys method, 185-187
  SpecialFolders method, 182-183
Shortcut object, 173, 187
  methods, 187
  practical application, 188-189
  properties, 187
TextStream, 347
WScript, 247
WScript object, 173-174
objTS variable, 424
OLE DB provider, ADSI, 254
On Error Goto 0 function, 70
onClick event, 418
oops resilience, 32
open files, monitoring, WinNT provider, 486-488
OpenAsTextStream method (FSO File objects), 204
OpenTextFile method, 468
operators, 116-118
  Boolean operators, 121, 156-157
  WQL (WMI Query Language)
    Boolean operator, 301-302
    comparison operator, 302-303
Option Explicit, syntax errors, preventing, 423-424
Option Explicit command, 81
Or Boolean operator, 156
organizational units (OUs). See OUs (organizational units)
OUs (organizational units), 248, 257, 260, 271
  creating, 260-261
  deleting, 263-264
  modifying, 261-263
  querying, 263
oUser input parameter, 89
outlining scripts, 222
output, functions, defining, 223
output values, functions, 87-88

P
packaging script, 394-397
parameters
  command-line parameters as input, 110-114
  command-line scripts, parsing, 111-114
  input parameters, functions, 85-87
  MapNetworkDrive method, 175
ParentFolder property (FSO File object), 204
parsing parameters, command-line scripts, 111-114
Password parameter (MapNetworkDrive method), 175
passwords
  creating, 117-121
  domains, changing, 259
  mass password changes, ADSI, 503-505
  scripts, placing in, 176

past dates, working with, 147-148

Path property (Drive object), 195
Path property (FSO File object), 204
path rules, 401
paths, LDAP, 252

PCs (personal computers), minimum requirements, 8
PFE (Programmer's File Editor), 37
  scripting with, 26-27
ping command-line utility, 110

policies
  software restriction policies, 400-401
  WSH trust policy, implementing, 406

Popup method (Shell object), 187
PreLoad.vbs listing (15.1), 264-269
PreloadDomain2.vbs listing (16.2), 278-280
preloading domains, 264-269, 277-281
Preserve keyword, 150

PrimalScript, 29-31
  Live Syntax Checking, 422
  Snippets, 377
  syntax errors, preventing, 422-423
  TLB Browser, 189
  WMI Wizard, 336-337, 340

PrimalScript Wizard, WQL queries, writing, 308, 310-312

printers, mapping, 61
procedural programs, 410

programming books, 76
programming languages
  strongly typed languages, 82
  weakly typed languages, 82

programs, procedural programs, 410

properties
  COM objects, 95
  Drive object (FSO), 195-196
  FSO (FileSystemObject)
    File objects, 204-206
    objects, 202-203
  WQL (WMI Query Language), 300-301

providers
  ADSI, 254-255
    GC (Global Catalog) provider, 254
    IIS provider, 254
  LDAP provider, 248, 252-254
  NWCOMPAT provider, 254
  OLE DB provider, 254
  WinNT provider, 248-252

LDAP provider, 271
  ADSI, connections, 248
  domain attributes, 258
  domains, querying, 257-259
  group creation, 273
  group membership, checking, 275-277
  user creation, 273
  user settings, changing, 274-275

NDS (NetWare Directory Services) provider, 254

WinNT provider
  ADSI connections, 248-252
  domains, reconfiguring, 259
  group creation, 272
  LDAP 252-254
  open file monitoring, 486-488
  user creation, 272
  WMI (Windows Management Instrumentation), 285-286

How can we make this index more useful? Email us at indexes@samspublishing.com
provider names, ADSI, 248
pseudocode, 56
Put method, 263
    user settings, changing, 274-275
Put method (objDomain), 248
PutEx method, 275
pwdHistoryLength domain attribute (LDAP provider), 258

Q

queries
    ADSI queries, debugging, 366-370
    Excel spreadsheets, ADO (Active Directory Objects), 455-463
    “impersonate” directive, 315
testing, WMI (Windows Management Instrumentation), 294-295
WMI queries
    debugging, 358-370
testing, 325-326
writing, 325-333
WQL (WMI Query Language), 299-301
    associations, 317
    associators, 303
    Boolean operators, 301-302
    comparison operators, 302-303
    complex information, 317
determining, 304-305
    keywords, 303
properties, 300-301
    references, 303
results, 312-314
testing, 306-308
writing, 308-315

WQL queries
    ASSOCIATORS OF query, 321-325
    REFERENCES OF query, 323-325
    WMI (Windows Management Instrumentation), 289-290
    writing, WMI (Windows Management Instrumentation), 294-295
    query scripts, WMI (Windows Management Instrumentation), creating, 497-499
    querying
domains, 257-259
0Us (organizational units), 263
user information, Get method, 273-274
question mark icon, 106

R

RAM Disk drive type, 196
Randomize function, 119
RealTimePublishers.com, 2
ReDim statement, 149
references, WQL (WMI Query Language), 303
REFERENCES OF query, WMI instances, associating, 323-325
RegDelete method (Shell object), 184-185
RegRead method (Shell object), 184
RegWrite method (Shell object), 184
Remotable parameter (WSCs), 383
remote computers
    connecting to, WMI (Windows Management Instrumentation), 287-289
    shutting down, WMI (Windows Management Instrumentation), 481, 483
remote MSI packages, uninstalling, WMI (Windows Management Instrumentation), 489-492
Remote name parameter (MapNetworkDrive method), 175
remote shares, listing, ADSI, 484-486
Removable drive type, 196
RemoveNetworkDrive method (Network object), 176
RemovePrinterConnection method (Network object), 178
renaming WScript.exe, 406
Replace( ) function, 131, 140-141, 143
requirements, computers, 8
ResetPW.vbs listing (23.1), 391-393
results, script testing, analyzing, 236
Retrieve IP Address.vbs listing (4.4), 62
Retry button, 107
return values, MsgBox statement, 104
reusable code, 31
reusing code, 424-425
Right command, 224
Right( ) function, 131, 134, 143
Rnd( ) function, 119, 126
RootFiles listing (5.5), 98-99
RootFolder property (Drive object), 195
Round( ) function, 509
routines, subroutines
  encapsulation, 373
  event handlers, 410
  writing, 223-227, 340-342
RTrim( ) function, 131
rules, software, 400
Run method (Shell object), 180
running
  command-line scripts, 111
  encoded script, 393
  first script, 35
  scripts, WSH (Windows Script Host), 21-23
  signed scripts, 404-405

S
safe scripting, implementing, 405-406
SAM (Security Accounts Manager), 247, 271
SampleHTA.hta listing (25.1), 419-420
SAPIEN Primal Script
  syntax errors, preventing, 422-423
  TLB Browser, 189
ScanLog.vbs listing (12.6), 211-214
ScanLog2.vbs listing (12.7), 215-218
scheduled tasks, HTAs (HTML Applications), 410
Scientific mode, Calculator, 124
scripts
  query scripts, WMI (Windows Management Instrumentation), 497-499
  update scripts, WMI (Windows Management Instrumentation), 499-500
scr files, WScript.exe associations, removing, 405
script
  complex WMI queries, writing, 325-333
  encoded script, 389-390
    detrments, 394
    running, 393
    writing, 390-393
  encoding, 397
  encrypted code, 390
  script packaging, 394-397
script editors
  Notepad, 26
  PFE (Programmer’s File Editor), 26-27
  VBScript editors, 27-28
  VBScript IDEs, 29-33
  What-You-See-Is-What-You-Get editors, 410
Script Encoder, 389
  encoded script
    detrments, 394
    running, 393
    writing, 390-393
    installing, 390

How can we make this index more useful? Email us at indexes@samspublishing.com
script extensions, Windows XP, 399
script hosts, 14-16
script libraries, creating, 72
Script Package dialog box, 395
script packagers, 394-397
SCRIPT tags, 392
script-based viruses, 399
scripting, 13-14
  candidate modules, identifying, 223
dangers of, 399-400
dangers of, 399-400
errors, planning for, 70-72
event-driven scripting, HTAs (HTML Applications), 410
FormatLogFileName function, writing, 223-227
functions, writing, 223-227
interpretation, 17
main script, writing, 228-234
Notepad, 25-26
PFE (Programmer’s File Editor), 26-27
safe scripting, implementing, 405-406
security, 405-406
security issues, 18-19
strings, double quotation marks, 129-130
subroutines, writing, 223-227
typographical elements, 8-9
VBScript Editors, 27-28
VBScript IDEs, 29-33
Windows administrators, 13
  with objects, 98-99
scripting languages, ActiveX, 16-18
ScriptingAnswers.com, 2, 14, 420
ScriptingOutpost.com, 377
ScriptingTraining.com, 2
Scriptomatic, WQL queries, writing, 308-312
Scriptomatic tool, 340
Scriptomatic tool (WMI), 292-293

scripts, 2, 13
  administrative scripts, 455, 480
    inactive user discovery, 464-469
    system information collection, 469-474
    templates, 474-479
    user creation automation, 455-463
  ADSI mass password changes, performing, 503-505
  ADSI object creation scripts, creating, 500-502
  ADSI object deletion scripts, creating, 502-503
  ADSI object query scripts, creating, 502
  bugs, 421-422, 435
    logic errors, 425-435
    syntax errors, 422-425
  checking, 14
  command-line scripts, 23-25
    execution of, 110
    parsing parameters, 111-114
    running, 111
  comments, 48-53
  completing, 241
  debugging, 357-358
    logic errors, 425-435
    syntax errors, 422-425
  decision making in, 48
  deployment, 17
  designing, 55, 219-220, 335-340
    fact gathering, 220-221
    module creation, 63-69
    script outlines, 222
    task definitions, 221-222
    task lists, 56-58
    tool selection, 58-63
digitally signing, 402-405
documentation, 48-53
  errors, identifying, 231-233
executions, 155, 167-170
  conditional executions, 155-162
loops, 162-167
first script
  debugging, 35-36
  running, 35
  writing, 33-35
functions, 39-43
HTAs (HTML Applications)
  adding, 416-418
  writing, 411-420
logoff scripts, 439
  AD scripts, 451-454
logon scripts, 439
  AD scripts, 440-454
  NT scripts, 440-454
main script, 44-45
  custom functions and subroutines, 45-46
  intrinsic functions, 47-48
  statements, 47-48
  writing, 343-350
modifying, 240-241
modularization, 373-377, 388
  detriments, 377
  reason for, 377
  WSCs (Windows Script Components), 378-384, 387
modualrizing, 63-69
native scripts, 17
network administrative scripts, 481, 495
  listing hot fixes, 492-495
  listing software, 492-495
  open file monitoring, 486-488
  remote computer shutdowns, 481-483
  remote MSI package uninstalls, 489-492
  remote share listings, 484-486
passwords, placing in, 176
polishing, 243
pseudocode, 56
reusing, 424-425
running, WSH (Windows Script Host), 21-23
script-based viruses, 399
script hosts, 14-16
service accounts, updating, 505-508
shutdown scripts, 451
signed scripts, 402-405
  running only, 404-405
statements, 44
subroutines, 39, 43-44
testing, 234-236, 239-241-243, 351-352, 355
text-based scripts, 17
VBScript, typical scripts, 39, 42
Windows 9x scripts, execution, 440
WMI (Windows Management Instrumentation) scripts
  testing, 297
  writing, 293-297
WQL queries, writing, 308-312
sDate input parameter, 89
SearchScripting.com, 2
Second( ) function, 146
security
  antivirus software, 406
  scripting, 18-19
  dangers of, 399-400
  safe scripting, 405-406
  scripts
    digitally signing, 402-405
    signed scripts, 404-405
  software restriction policies, 400

How can we make this index more useful? Email us at indexes@samspublishing.com
Windows Server 2003, improvements to, 400-402
Windows XP improvements to, 400-402
security, HTAs (HTML Applications), security context, 409
Security Accounts Manager (SAM). See SAM (Security Accounts Manager)
security identifiers (SIDs). See SIDs (security identifiers)
SELECT property (WQL), 300
SELECT property (WQL query), 290
Select/Case conditional execution statements, 161-162
Select/Case construct, 48
SelectCase.vbs listing (10.2), 161
SendKeys method (Shell object), 185-187
SerialNumber property (Drive object), 195
Server 2003 (Windows), security improvements to, 400-402
Server Message Blocks (SMBs), 445
service accounts, updating, 505-508
SetDefaultPrinter method (Network object), 178
SetDHCP.vbs listing (19.1), 328-330
SetInfo method, 260, 263
SetInfo method (objDomain), 249
settings, users, changing, 274-275
ShareName property (Drive object), 195
Shares.vbs listing (29.2), 484-485
Shell object, 173, 180
AppActivate method, 185
CreateShortcut method, 183
Environment method, 183-184
ExpandEnvironmentStrings method, 184
LogEvent method, 184
Popup method, 187
RegDelete method, 184-185
RegRead method, 184
RegWrite method, 184
Run method, 180-182
SendKeys method, 185-187
SpecialFolders method, 182-183
Shortcut object, 173, 187
methods, 187
practical application, 188-189
properties, 187
shortcuts, creating, 188
Shortcuts.vbs listing (11.2), 188-189
ShortName property (FSO File object), 204
ShowNIC.vbs listing (17.1), 285
ShowProperties.vbs listing (21.1), 366-368
Shutdown method, 308, 311, 483
shutdown scripts, 451
Shutdown.vbs listing (29.1), 482-483
shutting down, remote computers, WMI (Windows Management Instrumentation), 481-483
SIDs (security identifiers), 58
signed scripts, running only, 404-405
Signer.wsf. This script signs another one listing (24.1), 402-404
signing scripts, 402-405
Sin( ) function, 119
singles, 115
Size property (FSO File object), 204
Sleep command, 186
SMBs (Server Message Blocks), 445
SMS (Systems Management Server), system information, collecting, 469-474
Snippets, 377
snippets, reusing, 424-425
software
listing, 492-495
rules, 400
software restriction policies, 400-401
Space( ) function, 131
spaghetti debugging, 425-426
SpecialFolders method (Shell object), 182-183
Split( ) function, 150
SQL (Structured Query Language), 299
Sqr( ) function, 119
standard shortcuts, creating, 188
Starting dialog box, 22
statements, 44, 92
  conditional execution statements
    If/Then, 156-161, 170-171
    Select/Case, 161-162
Const, 80
control-of-flow statements, 155
custom subroutines, compared, 48
Dim, 80, 98
Do Until, 424
Function, 89, 118
functions, acting as, 94
GetObject, 248
main script, 47-48
MsgBox, 84, 94, 101-108, 114, 170
  best practices, 106-107
  display options, 103
  return values, 104
Nothing, 251
ReDim, 149
Sub, 93
WinNT://, 248
StdIn, command-line input, 109-110
StrComp( ) function, 131
strComputer variable, 418
strings, 129
  changing, 139-141
  concatenating, 135-139
  data types, converting to, 142-143
  double quotation marks, 129-130
  formatting, 141-142
  manipulating, 143-144
    functions, 130-131
    substrings, 131-134
  VBScript, 129-131
strName variable, 476
strongly typed languages, 82
strResult variable, 418
StrReverse( ) function, 131
Structured Query Language (SQL), 299
Sub keyword, 48
Sub statement, 93
subroutines, 39, 43-44, 90-92, 99
  custom subroutines, creating, 92-94
  DoObject, 476-477
  DoObject_User, 479
  encapsulation, 373
  event handlers, 410
    HTA (HTML Applications), 418-420
  intrinsic subroutines, 44
    main script
      custom subroutines, 45-46
      intrinsic subroutines, 47-48
    MapDrive, 44, 67
    when to use, 45
    WorkWithObject, 478
  writing, 223-227, 340, 342
substrings, 131-134
syntax, 75
syntax errors, 70
  preventing, 422-425
system icons, message boxes, 102
Systems Management Server (SMS). See SMS (System Management Server)

Tan( ) function, 119
tangents, 115
TargetingAD.vbs listing (28.5), 477-479
TargetingLists.vbs listing (28.4), 474-477
TargetPath method (Shortcut object), 187

task lists, creating, 56-58
tasks
  domains, automating, 455, 480
  scripts, defining, 221-222

TaxStream object (FSO), 194

TechNet (Microsoft), 2
templates, administrative scripts, 474-479
testing
  complex WMI queries, 325-326
  queries, WMI (Windows Management Instrumentation), 294-295
  scripts, 234-355
    WMI (Windows Management Instrumentation) scripts, 297
  WQL queries, 306-308
text files, 14
text-based scripts, 17

TextStream objects, 347
  FSO (FileSystemObject), 206-209, 212, 216

third octets, IP addresses, attaining, 63
time zones, domains, 472

Time( ) function, 146
times
  getting, 146-147
  VBScript, 145-148

TLB Browser, 189
tools
  Script Encoder, 390
  selecting, script design, 58-63
  WMI (Windows Management Instrumentation), 291-292
    Object Browser, 291-292
    Scriptomatic tool, 292-293

TotalSize property (Drive object), 196

TreeObject model script listing (5.4), 97

Trim( ) function, 131

trust policies, WSH trust policy, implementing, 406

Type property (FSO File object), 204

Types.vbs listing (12.2), 199
typographical elements, programming, 8-9

U

UCase( ) function, 131, 136, 139-140

UNC (Universal Naming Convention), 168

UNCs (universal naming conventions), 66

underscore (_) character, 79

underscore characters, 106

Unescape( ) function, 131

Uninstall method, 492

Uninstall.vbs listing (29.5), 489-492

uninstalling remote MSI packages, 489-492

Universal (Greenwich) time, 472

Universal Naming Convention (UNCs), 66, 168

Unknown drive type, 196

Update profile parameter (MapNetworkDrive method), 175

update scripts, WMI (Windows Management Instrumentation), creating, 499-500

UpdateServices.vbs listing (30.), 505-508

updating service accounts, WMI, 505-508

user input, validating, 69

user interfaces, HTAs (HTML Applications), designing, 412-413

UserDomain method (Network object), 178

UserName method (Network object), 178

Username parameter (MapNetworkDrive method), 175
users, 271
  creating, 271
  automating, 455-463
  LDAP provider, 273
  WinNT provider, 272
inactive users, finding, 464-469
information, querying, 273-274
settings, changing, 274-275
targeting, AD (Active Directory), 477-479

utilities, command-line utilities, 110

V

validating, user input, 69
values, constants, 466
variables, 76, 79, 99
  arrays, 145, 148, 152
  creating, 149-151
  manipulating, 149-151
  multidimensional arrays, 149
  numeric variables, 151-152
  VBScript, 149
bytes, 152
constants, compared, 80
data, assigning to, 82-83
data coercion, 83-85
data types, 82
  declaring, 79-81
dynamic arrays, 149
names, 227
objDomain, 248
objTS, 424
strComputer, 418
strName, 476
strResult, 418
vPasswd, 118

VBA (Visual Basic for Applications), 400
vbs files, WScript.exe associations, removing, 405
VBScript (Visual Basic, Scripting Edition), 1, 13, 16-18
  advanced arithmetic, 118-121
  arrays, 149
  as interpreted language, 14
  basic arithmetic, 116-118
  Boolean math, 121-126
  bytes, 152
  complex queries, writing, 326-333
  data types, converting, 127
  dates, 145-148
  documentation, downloading, 59
  input, asking for, 108-110
  interactive capabilities, limitations, 101
  introduction of, 1
  messages, displaying, 101-108
  numbers, 115-116
  numeric data types, converting, 126-127
  procedural programs, 410
  scripts, typical scripts, 39, 42
  strings, 129-131
  times, 145-148
  vulnerabilities, 1
  Windows PowerShell, compared, 19-20
  WQL queries, writing, 308-312
  WScript.exe, 37

VBScript Editor
  scripting with, 27-28
  syntax errors, preventing, 422-423
VBScript IDEs, scripting with, 29-33
VbsEdit, 27-28
Version parameter (WSCs), 383
versions, WMI (Windows Management Instrumentation), 286
virtual private network (VPN) adapters, 284
viruses
  antivirus software, 406
  script-based viruses, 399


Visual Basic for Applications (VBA), 400

Visual Studio, 101

VolumeName property (Drive object), 196

vPasswd variable, 118

VPN (virtual private network) adapters, 284

W

WBEM (Web-Based Enterprise Management), 286

Wbemtest.exe tool
  complex queries, testing, 325-326
  WMI (Windows Management Instrumentation) queries, 286-289
debugging, 358-366
  WQL queries, 289-290

weakly typed languages, 82

Web-Based Enterprise Management (WBEM), 286

Weekday( ) function, 146

WeekdayName( ) function, 147

What-You-See-Is-What-You-Get editors, 410

Where clause, WMI instances, associating, 324-325

WHERE keyword, 136

WHERE property (WQL query), 290, 301

WhoHas.vbs
  debugging, 35-36
  running, 35
  writing, 33-35

WhoHas.vbs listing (2.1), 33

WhoHas.vbs listing (29.4), 486-488

WhoHasFile.vbs listing (10.3), 167-170

whole numbers, 115

Win32 classes, WMI (Windows Management Instrumentation), 284

Win32_OperatingSystem class (WMI), 336-337, 349, 482-483

Windows 9x scripts, execution, 440

Windows administrators, scripting, 13

Windows Calculator, Scientific mode, 124

Windows domains
  inactive users, finding, 464-469
  lists of computers, targeting, 474-477
  system information, collecting, 469-474
tasks, automating, 455, 480
  user creation, automating, 455-456, 459-463
  users, targeting, 477-479

Windows Management Instrumentation (WMI). See WMI (Windows Management Instrumentation)

Windows PowerShell, VBScript, compared, 19-20

Windows PowerShell: TFM, 19

Windows Script Components (WSCs). See WSCs (Windows Script Components)

Windows Script Host (WSH). See WSH (Windows Script Host)

Windows Server 2003, security improvements to, 400-402

Windows XP
  default script extensions, 399
  security, rules, 400
  security improvements to, 400-402

WindowStyle method (Shortcut object), 187

WinNT, 271
WinNT provider
   ADSI, 249-252
   connections, 248
   file server connections, 249
   domains, reconfiguring, 259
   groups, creating, 272
   LDAP, 252-254
   open files, monitoring, 486-488
   users
      creating, 272
      information queries, 273-274
   WinNT:// statement, 248
WizardCode.vbs listing (20.1), 337, 339
wizards
   code wizard, 32
   WMI Wizard, 336-337
WMI (Windows Management Instrumentation), 14, 145, 173, 283, 286, 299, 335, 355, 497, 508
   ACEs (access control entries, 300
      associations, 319
         instances, 321-325
   CIM (Common Information Model), 284
   classes, 284-285, 305
      finding, 293-294
      interrelated classes, 300, 317-319
   Win32_OperatingSystem, 336-337, 482-483
   Win32_OperatingSystem class, 349
   complex information, 300, 317-319
   DMTF (Desktop Management Task Force), 283
   functions, writing, 340, 342
   hierarchy, 283-285
      consumers, 285-286
      providers, 285-286
      versions, 286
   installing, 290-291
queries
   debugging, 358-366, 370
   testing, 294-295, 325-326
   writing, 294-295, 325-333
   query scripts, creating, 497-499
   remote computers
      connecting to, 287-289
      shutting down, 481-483
   remote MSI packages, uninstalling, 489-492
scripts
   debugging, 357-358
   designing, 335-340
   testing, 297, 351-355
   writing, 293-297, 343-344, 347-350
   service accounts, updating, 505-508
   subroutines, writing, 340, 342
   tools, 291-292
      Object Browser, 291-292
      Scriptomatic tool, 292-293
   update scripts, creating, 499-500
   Wbemtest.exe tool, 286-289
   WQL queries, 289-290
   Win32, 284
   WQL (WMI Query Language), 299-301
      associations, 317
      associators, 303
      Boolean operators, 301-302
      classes, 317
      comparison operators, 302-303
      complex information, 317
      keywords, 303
      properties, 300-301
      queries, 304-315
      references, 303
   WMI Wizard, 336-337
WMIFunctions.wsc listing (22.1), 379-387

How can we make this index more useful? Email us at indexes@samspublishing.com
WorkingDirectory method (Shortcut object), 187
WorkWithObject subroutine, 478
WQL (WMI Query Language), 299-301
  associations, 317
  associators, 303
  Boolean operators, 301-302
  classes, interrelated classes, 317
  comparison operators, 302-303
  complex information, 317
  keywords, 303
  properties, 300-301
  queries
    debugging, 358-366
    determining, 304-305
    results, 312-314
    testing, 306-308, 325-326
    writing, 308-315, 325-333
  references, 303
  WMI instances, associating, 321-325
WQL queries, WMI (Windows Management Instrumentation), 289-290
writing
  complex WMI queries, 325
    VBScript, 326-333
    encoded script, 390-393
  first script, 33-35
  functions, 223-227, 340-342
    FormatLogFileName function, 223-227
  HTAs (HTML Applications), 411-420
  main script, 228-234, 343-350
  queries, WMI (Windows Management Instrumentation), 294-295
  scripts, WMI (Windows Management Instrumentation) scripts, 293-297
  subroutines, 223-227, 340-342
  WQL queries, 308-315
WScript objects, 173-174, 247
WScript.ADSSystemInfo object, 179-180
WScript.Echo, 114, 170, 213
  command-line output, 110
  message boxes, displaying, 107-108
WScript.Echo command, 213-214
WScript.exe (VBScript), 37
  deleting, 405-406
  file associations, removing, 405
  renaming, 406
WScript.Network object, 174-175
  AddPrinterConnection method, 177
  AddWindowsPrinterConnection method, 177
  ComputerName method, 178
  EnumNetworkDrive method, 176
  EnumPrinterConnection method, 177
  MapNetworkDrive method, 175-176
  practical application, 178-179
  RemoveNetworkDrive method, 176
  RemovePrinterConnection method, 178
  SetDefaultPrinter method, 178
  UserDomain method, 178
  UserName method, 178
WScript.Shell object, 180
  AppActivate method, 185
  CreateShortcut method, 183
  Environment method, 183-184
  Exec method, 180-182
  ExpandEnvironmentStrings method, 184
  LogEvent method, 184
  Popup method, 187
  RegDelete method, 184-185
  RegRead method, 184
  RegWrite method, 184
  Run method, 180
  SendKeys method, 185-187
  SpecialFolders method, 182-183
WScript.Shortcut object, 187
   methods, 187
   practical application, 188-189
   properties, 187
WScript.StdIn, command-line input, 109-110
WSCs (Windows Script Components), 378, 388
   registration, 383
   XML (Extensible Markup Language), 378, 382-384, 387
wsf files, WScript.exe associations, removing, 405
WSH (Windows Script Host), 15-16, 20-23, 173, 390
   ADSystemInfo object, 179-180
   Network object, 173-175
      AddPrinterConnection method, 177
      AddWindowsPrinterConnection method, 177
      ComputerName method, 178
      EnumNetworkDrive method, 176
      EnumPrinterConnection method, 177
      MapNetworkDrive method, 175-176
      practical application, 178-179
      RemoveNetworkDrive method, 176
      RemovePrinterConnection method, 178
      SetDefaultPrinter method, 178
      UserDomain method, 178
      UserName method, 178
   Shell object, 173, 180
      AppActivate method, 185
      CreateShortcut method, 183
      Environment method, 183-184
      Exec method, 180-182
      ExpandEnvironmentStrings method, 184
      LogEvent method, 184
      Popup method, 187
   RegDelete method, 184-185
   RegRead method, 184
   RegWrite method, 184
   run method, 180
   SendKeys method, 185-187
   SpecialFolders method, 182-183
   Shortcut object, 173, 187
   methods, 187
   practical application, 188-189
   properties, 187
   trust policy, implementing, 406
   WScript object, 173-174
WshNamed object, parameters, parsing, 111-114
X–Z
XML (Extensible Markup Language), 32
   WSCs (Windows Script Components), 378, 382-384, 387
Year( ) function, 146

How can we make this index more useful? Email us at indexes@samspublishing.com