Microsoft

OFFICIAL MICROSOFT LEARNING PRODUCT

2433B

Microsoft® Visual Basic®, Scripting Edition and Microsoft Windows® Script Host Essentials

Companion Content

Product Number: 2433B

Released: 10/2007

MICROSOFT LICENSE TERMS

OFFICIAL MICROSOFT LEARNING PRODUCTS COURSEWARE – STUDENT EDITION – Pre-Release and Final Versions

These license terms are an agreement between Microsoft Corporation and you. Please read them. They apply to the licensed content named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft

- updates,
- supplements,
- Internet-based services, and
- support services

for this licensed content, unless other terms accompany those items. If so, those terms apply.

By using the licensed content, you accept these terms. If you do not accept them, do not use the licensed content.

If you comply with these license terms, you have the rights below.

1. **OVERVIEW.**

   Licensed Content. The licensed content includes software, printed materials, academic materials (online and electronic), and associated media.

   License Model. The licensed content is licensed on a per copy per device basis.

2. **INSTALLATION AND USE RIGHTS.**

   a. **Licensed Device.** The licensed device is the device on which you use the licensed content. You may install and use one copy of the licensed content on the licensed device.

   b. **Portable Device.** You may install another copy on a portable device for use by the single primary user of the licensed device.

   c. **Separation of Components.** The components of the licensed content are licensed as a single unit. You may not separate the components and install them on different devices.

   d. **Third Party Programs.** The licensed content may contain third party programs. These license terms will apply to your use of those third party programs, unless other terms accompany those programs.

3. **PRE-RELEASE VERSIONS.** If the licensed content is a pre-release ("beta") version, in addition to the other provisions in this agreement, then these terms also apply:

   a. **Pre-Release Licensed Content.** This licensed content is a pre-release version. It may not contain the same information and/or work the way a final version of the licensed content will. We may change it for the final, commercial version. We also may not release a commercial version. You will clearly and conspicuously inform any Students who participate in an Authorized Training Session and any Trainers who provide training in such Authorized Training Sessions of the foregoing; and, that you or Microsoft are under no obligation to provide them with any further content, including but not limited to the final released version of the Licensed Content for the Course.

   b. **Feedback.** If you agree to give feedback about the licensed content to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software, licensed content, or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

   c. **Confidential Information.** The licensed content, including any viewer, user interface, features and documentation that may be included with the licensed content, is confidential and proprietary to Microsoft and its suppliers.

      i. **Use.** For five years after installation of the licensed content or its commercial release, whichever is first, you may not disclose confidential information to third parties. You may disclose confidential information only to your employees and consultants who need to know the information. You must have written agreements with them that protect the confidential information at least as much as this agreement.

      ii. **Survival.** Your duty to protect confidential information survives this agreement.

iii. **Exclusions**. You may disclose confidential information in response to a judicial or governmental order. You must first give written notice to Microsoft to allow it to seek a protective order or otherwise protect the information. Confidential information does not include information that

- becomes publicly known through no wrongful act;
- you received from a third party who did not breach confidentiality obligations to Microsoft or its suppliers; or
- you developed independently.

d. **Term**. The term of this agreement for pre-release versions is (i) the date which Microsoft informs you is the end date for using the beta version, or (ii) the commercial release of the final release version of the licensed content, whichever is first ("beta term").

e. **Use**. You will cease using all copies of the beta version upon expiration or termination of the beta term, and will destroy all copies of same in the possession or under your control.

f. **Copies**. Microsoft will inform Authorized Learning Centers if they may make copies of the beta version (in either print and/or CD version) and distribute such copies to Students and/or Trainers. If Microsoft allows to such distribution, you will follow any additional terms that Microsoft provides to you for such copies and distribution.

4. **ADDITIONAL LICENSING REQUIREMENTS AND/OR USE RIGHTS.**

a. Media Elements and Templates. You may use images, clip art, animations, sounds, music, shapes, video clips and templates provided with the licensed content solely for your personal training use. If you wish to use these media elements or templates for any other purpose, go to www.microsoft.com/permission to learn whether that use is allowed.

b. Academic Materials. If the licensed content contains academic materials (such as white papers, labs, tests, datasheets and FAQs), you may copy and use the academic materials. You may not make any modifications to the academic materials and you may not print any book (either electronic or print version) in its entirety. If you reproduce any academic materials, you agree that:

- The use of the academic materials will be only for your personal reference or training use
- You will not republish or post the academic materials on any network computer or broadcast in any media;
- You will include the academic material's original copyright notice, or a copyright notice to Microsoft's benefit in the format provided below:

Form of Notice:

© 2008 Reprinted for personal reference use only with permission by Microsoft Corporation. All rights reserved.

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the US and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

c. Distributable Code. The licensed content may contain code that you are permitted to distribute in programs you develop if you comply with the terms below.

i. Right to Use and Distribute. The code and text files listed below are "Distributable Code."

- REDIST.TXT Files. You may copy and distribute the object code form of code listed in REDIST.TXT files.
- Sample Code. You may modify, copy, and distribute the source and object code form of code marked as "sample."
- Third Party Distribution. You may permit distributors of your programs to copy and distribute the Distributable Code as part of those programs.

ii. Distribution Requirements. For any Distributable Code you distribute, you must

- add significant primary functionality to it in your programs;
- require distributors and external end users to agree to terms that protect it at least as much as this agreement;
- display your valid copyright notice on your programs; and
- indemnify, defend, and hold harmless Microsoft from any claims, including attorneys' fees, related to the distribution or use of your programs.

iii. **Distribution Restrictions.** You may not

- alter any copyright, trademark or patent notice in the Distributable Code;
- use Microsoft's trademarks in your programs' names or in a way that suggests your programs come from or are endorsed by Microsoft;
- distribute Distributable Code to run on a platform other than the Windows platform;
- include Distributable Code in malicious, deceptive or unlawful programs; or
- modify or distribute the source code of any Distributable Code so that any part of it becomes subject to an Excluded License. An Excluded License is one that requires, as a condition of use, modification or distribution, that
  - the code be disclosed or distributed in source code form; or
  - others have the right to modify it.

5. **INTERNET-BASED SERVICES.** Microsoft may provide Internet-based services with the licensed content. It may change or cancel them at any time. You may not use these services in any way that could harm them or impair anyone else's use of them. You may not use the services to try to gain unauthorized access to any service, data, account or network by any means.
6. **SCOPE OF LICENSE.** The licensed content is licensed, not sold. This agreement only gives you some rights to use the licensed content. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the licensed content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the licensed content that only allow you to use it in certain ways. You may not
   - disclose the results of any benchmark tests of the licensed content to any third party without Microsoft's prior written approval;
   - work around any technical limitations in the licensed content;
   - reverse engineer, decompile or disassemble the licensed content, except and only to the extent that applicable law expressly permits, despite this limitation;
   - make more copies of the licensed content than specified in this agreement or allowed by applicable law, despite this limitation;
   - publish the licensed content for others to copy;
   - transfer the licensed content marked as 'beta' or 'pre-release' to any third party;
   - allow others to access or use the licensed content;
   - rent, lease or lend the licensed content; or
   - use the licensed content for commercial licensed content hosting services.
   - Rights to access the server software that may be included with the Licensed Content, including the Virtual Hard Disks does not give you any right to implement Microsoft patents or other Microsoft intellectual property in software or devices that may access the server.
7. **BACKUP COPY.** You may make one backup copy of the licensed content. You may use it only to reinstall the licensed content.
8. **TRANSFER TO ANOTHER DEVICE.** You may uninstall the licensed content and install it on another device for your personal training use. You may not do so to share this license between devices.
9. **TRANSFER TO A THIRD PARTY.** You may not transfer those versions marked as 'beta' or 'pre-release' to a third party. For final versions, these terms apply: The first user of the licensed content may transfer it and this agreement directly to a third party. Before the transfer, that party must agree that this agreement applies to the transfer and use of the licensed content. The first user must uninstall the licensed content before transferring it separately from the device. The first user may not retain any copies.
10. **EXPORT RESTRICTIONS.** The licensed content is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the licensed content. These laws include restrictions on destinations, end users and end use. For additional information, see www.microsoft.com/exporting.
11. **NOT FOR RESALE SOFTWARE/LICENSED CONTENT.** You may not sell software or licensed content marked as "NFR" or "Not for Resale."

12. **ACADEMIC EDITION.** You must be a "Qualified Educational User" to use licensed content marked as "Academic Edition" or "AE." If you do not know whether you are a Qualified Educational User, visit www.microsoft.com/education or contact the Microsoft affiliate serving your country.

13. **ENTIRE AGREEMENT.** This agreement, and the terms for supplements, updates, Internet-based services and support services that you use, are the entire agreement for the licensed content and support services.

14. **APPLICABLE LAW.**

    a. **United States.** If you acquired the licensed content in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.

    b. **Outside the United States.** If you acquired the licensed content in any other country, the laws of that country apply.

15. **LEGAL EFFECT.** This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the licensed content. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.

16. **DISCLAIMER OF WARRANTY. THE LICENSED CONTENT IS LICENSED "AS-IS." YOU BEAR THE RISK OF USING IT. MICROSOFT GIVES NO EXPRESS WARRANTIES, GUARANTEES OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT EXCLUDES THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**

17. **LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO U.S. $5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.**

    This limitation applies to

    - anything related to the licensed content, software, services, content (including code) on third party Internet sites, or third party programs; and
    - claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

    It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

Please note: As this licensed content is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.

Remarque : Ce le contenu sous licence étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.

EXONÉRATION DE GARANTIE. Le contenu sous licence visé par une licence est offert « tel quel ». Toute utilisation de ce contenu sous licence est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection dues consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit locale, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES. Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 $ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.

Cette limitation concerne:

- tout ce qui est relié au le contenu sous licence , aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers ; et
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

EFFET JURIDIQUE. Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

Module 1

Overview of Windows Scripting Technologies

Contents:

Lab Answer Keys

Lab: Configuring and Using WSH

Exercise 1: Configuring WSH

▶ Task 1: To list the script settings for the CScript.exe host

1. On the 2433B-VISTA-CL1-01 virtual machine, click **Start**, point to **All Programs**, click **Accessories**, and then click **Command Prompt.**
2. At the command prompt, type **CScript** and then press ENTER.
3. Review the list of switches for the CScript host.

▶ Task 2: To run a script file

1. At the command prompt, type e: and then press ENTER.
2. At the command prompt, type **cd labfiles\starter** and then press ENTER.
3. At the command prompt, type **Scr1.vbs** and then press ENTER.

   The WScript host launches the script.
4. In the **Windows Script Host** dialog box, click **OK** three times.
5. At the command prompt, type **CScript.exe Scr1.vbs** and then press ENTER.

   The CScript host launches the script.

▶ Task 3: To change the default script host

1. At the command prompt, type **CScript.exe //H:Cscript** and then press ENTER.

   You are notified that the default host has been set to CScript.exe.
2. At the command prompt, type **Scr1.vbs** and then press ENTER.

   The CScript host launches the script.

▶ Task 4: To create a WSH file for Scr1.vbs

1. Click **Start**, and then click **Computer**.
2. In the Computer folder, double-click **AllFiles (E:),** double-click **Labfiles,** and then double-click **Starter**.
3. Right-click **Scr1.vbs**, and select **Properties**.
4. In the **Properties** dialog box, click the **Script** tab.
5. Select **Stop script after specified number of seconds**, and then click **OK**.

   A new file named Scr1.wsh is created in the same folder as the .vbs file.

▶ Task 5: To open the WSH file for Scr1.vbs

1. Click **Start**, point to **All Programs**, point to **Accessories**, and then click **Notepad**.
2. In Notepad, on the **File** menu, click **Open**.
3. In the **Open** dialog box, navigate to the **E:\Labfiles\Starter** folder.

4. In the **File name** box, type ***.*** and click **Open**.
5. In the **Open** dialog box, click **Scr1.wsh**, and then click **Open**.

▶ Task 6: To modify the WSH file for Scr1.vbs

1. Modify the line that reads **Timeout=1**, so that the line reads **Timeout=5**.
2. On the **File** menu, click **Save**.
3. Close Notepad.
4. At the command prompt, type **WScript Scr1.wsh** and then press ENTER.
5. Wait for five seconds. Do not click OK or press ENTER in the message box. Wait for the script to be forcibly shut down by the host.
6. Close Windows Explorer.

Exercise 2: Reviewing the WSH and VBScript Documentation

▶ Task 1: To review the WSH documentation

1. On the desktop, double-click **Windows Script 5.6 Documentation.chm.**
2. In the Windows Scripting Technologies help file, click the **Contents** tab.
3. On the **Contents** tab, expand **Microsoft Windows Script Technologies**.
4. Under **Microsoft Windows Script Technologies**, expand **Windows Script Host**.
5. Under **Windows Script Host**, expand **Windows Script Host** Basics.
6. Under **Windows Script Host Basics**, expand **What Is WSH**?
7. Review the information provided.

▶ Task 2: To review the VBScript documentation

1. In the Windows Scripting Technologies help file, click the **Contents** tab.
2. On the **Contents** tab, expand **VBScript**.
3. Under **VBScript**, expand **VBScript User's Guide**.
4. Click **What Is VBScript?**
5. Review the information provided.

▶ Task 3: To search the Windows Scripting Technologies documentation

1. In the Windows Scripting Technologies help file, click the **Index** tab.
2. In the **Type in the keyword to find** box, type **Echo**
3. Click **Display**.
4. Review the information provided about the **Echo** method.
5. Close the Windows Scripting Technologies help file.

Exercise 3: Using Windows Script Files

▶ Task 1: To use WSF files

1. Click **Start**, and then click **Computer**.
2. In the Computer folder, double-click **AllFiles (E:),** double-click **Labfiles**, and then double-click **Starter**.
3. Double-click **WSFInAction.wsf**.
4. Note the name of the job that runs, and then click **OK**.
5. At the command prompt, make sure that you are still in the E:\Labfiles\Starter folder.
6. At the command prompt, type **Cscript WSFInAction.wsf //Job:Job2** and then press ENTER.

   Notice that a different script runs, and then click **OK**.
7. Repeat Step 6, but run **Job4**.

   Notice that two scripts run in the same job.

Exercise 4: Using an IDE to Edit Script Files

▶ Task 1: To prepare the Primal Script IDE

1. Click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom.**
2. In the **Welcome** dialog box, click **Next**.
3. In the **Select your role** dialog box, select **VBScript System/Network Administrator**, and then click **Next**.
4. In the **Select which Nexus windows to show** dialog box, click **Next**.
5. In the **Select which file types you use** dialog box, click **Next**.
6. In the **Finished** dialog box, click **Finish**.
7. In the **Windows Security Alert** dialog box, click **Unblock**.
8. On the **View** menu, click **Left Nexus Window** to close the Resources Browser.
9. On the **View** menu, click **Right Nexus Window** to close the Info Browser.
10. On the **Tools** menu, click **Options**.
11. In the **Options** dialog box, expand **Environment**, and then select **Directories**.
12. Click the ellipsis (…) next to **My Scripts**.
13. In the **Browse For Folder** dialog box, expand **Computer**, expand **AllFiles (E:),** click **Labfiles**, and then click **OK**.
14. Click **OK** to close the **Options** dialog box.

▶ Task 2: To open a script file in Primal Script

1. On the **File** menu, click **Open**.
2. In the **Open** dialog box, click **Starter**, and then click **Open**.
3. Click **Scr1.vbs**, and then click **Open**.

4. Review the contents of the file.

▶ Task 3: To use the auto-complete features of the IDE

1. Click at the end of line 5, and then press ENTER.
2. Type **WScript**

   Notice that the command that you have just typed is gray. This indicates that the IDE recognizes the word that you have typed.
3. Type a single period.

   Notice that a list of the available valid terms is displayed. These are the members of the **WScript** object.
4. In the list, double-click **Echo**.

   The IDE auto-complete feature enters the word into your script.
5. Press SPACEBAR.

   Note that some on-screen help is provided by the IDE.
6. Type **"Welcome to the VBScript and WSH course!"** (Including the quotation marks).

▶ Task 4: To save and run the script

1. On the **File** menu, click **Save**.
2. On the **Script** menu, click **Run Script**.
3. Scroll through the results shown in the **Output** window at the bottom of the screen.
4. On the **File** menu, click **Exit**.
5. At the command prompt, make sure that you are still in the E:\Labfiles\Starter folder.
6. At the command prompt, type **CScript.exe Scr1.vbs** and then press ENTER.

   The CScript host launches the script.
7. At the command prompt, type **WScript.exe Scr1.vbs** and then press ENTER.

   The WScript host launches the script.

Lab: Question and Answers

Lab: Configuring and Using WSH

Exercise 1: Configuring WSH

Question: What are the script host executable files?

Answer: CScript.exe and WScript.exe.

Question: Which host is the default host?

Answer: WScript.exe.

Exercise 2: Reviewing the WSH and VBScript Documentation

Question: What is the Visual Basic, Scripting Edition equivalent of the **Debug.Print** command in Visual Basic?

Answer: WScript.Echo.

Question: What is the main difference between Visual Basic, Scripting Edition and WSH?

Answer: Visual Basic, Scripting Edition is the scripting language that you use to code your scripts, and WSH is the script host environment.

Exercise 3: Using Windows Script Files

Question: Which script engine will the Myscript.jse file invoke?

Answer: The JScript engine.

Question: Which script engine will the Myscript.wsf file invoke?

Answer: The script engine used will depend on the language used to code the particular jobs in the .wsf file.

Exercise 4: Using an IDE to Edit Script Files

Question: What are some advantages of using Notepad as your script editor?

Answer: Possible answers include: Notepad is installed on all Windows-based computers by default; Notepad supports cut-and-paste and other text-editing operations; and Notepad has low memory and processor requirements.

Question: What are some advantages to using an IDE to develop scripts?

Answer: Possible answers include: An IDE may include an object browser, context-sensitive help, syntax checking and formatting, and auto-complete features.

Module 2

Objects in VBScript and WSH

Contents:

Lab Answer Keys

Lab: Objects in VBScript and WSH

Exercise 1: Manipulating the Scripting Object Model

▶ Task 1: To instantiate scripting objects

1. Click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalSCRIPT 4.1 Classroom.**
2. If a **Windows Security Alert** dialog box appears, click **Unblock**.
3. On the **File** menu, click **Open**.
4. In the **Open** dialog box, browse to **E:\Labfiles\Starter**.
5. Click **ScriptingOM.vbs**, and then click **Open**.
6. Locate the comment **TODO Instantiate FileSystemObject**.
7. Uncomment the following code.

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
```

8. Locate the comment **TODO Bind to the Programs File folder**.
9. Uncomment the following code.

```
Set objFolder = objFSO.GetFolder("C:\Program Files")
```

10. 10. Locate the comment **TODO Get the collection of subfolders in the Program Files folder**.
11. Uncomment the following code.

```
Set colSubFolders = objFolder.Subfolders
```

▶ Task 2: To retrieve scripting object properties

1. Locate the comment **TODO Display some of the properties of each subfolder**.
2. Uncomment the following code.

```
WScript.Echo "The Folder's name is: " & objFolder.Name
WScript.Echo "The Folder was created: " & objFolder.DateCreated
WScript.Echo "The Folder was last changed on: " & objFolder.DateLastModified
WScript.Echo "The Folder was last accessed on: " & objFolder.DateLastAccessed
WScript.Echo "The Folder size is: " & objFolder.Size
WScript.Echo "The Folder type is: " & objFolder.Type
Wscript.Echo "The Folder has " & objFolder.SubFolders.count & " subfolders"
```

   You have accessed the **Folder** object's **Name, DateCreated, DateLastModified, DateLastAccessed**, **Size**, and **Type** properties and concatenated each value with an introductory phrase. Additionally, you have accessed the **Count** property of the **Folder** object's **SubFolders** collection and concatenated it with two string literals.

   Each value and the literal string you concatenated with it are passed as a single parameter to the **WScript.Echo** method.

3. On the **File** menu, click **Save**.

4. On the **Script** menu, click **Run Script**.
5. In the output pane, review the messages as the script runs.

   You have now successfully manipulated the scripting object model.

Exercise 2: Manipulating the WSH Object Model

▶ Task 1: To instantiate and manipulate the Shell object

1. On the **File** menu, click **Open**.
2. In the **Open** dialog box, browse to **E:\Labfiles\Starter**.
3. Click **WSHShell.vbs**, and then click **Open**.
4. Locate the comment **TODO Instantiate the Shell object**.
5. Uncomment the following code.

```
Set objShell=WScript.CreateObject("Wscript.Shell")
```

6. Locate the comment **TODO Set a string to the SendTo special folder**.
7. Uncomment the following code.

```
strSendToFolder = objShell.SpecialFolders("SendTo")
```

8. Locate the comment TODO Set a string to Notepad's default location on the OS.
9. Uncomment the following code.

```
strPathToNotepad = objShell.ExpandEnvironmentStrings _
("%SystemRoot%/system32/notepad.exe")
```

10. Locate the comment **TODO Create the shortcut**.
11. Uncomment the following code.

```
Set objShortcut = objShell.CreateShortcut(strSendToFolder & _
"\notepad.lnk")
```

12. Locate the comment **TODO Set the shortcut's target path, and then save the shortcut**.
13. Uncomment the following code.

```
objShortcut.TargetPath = strPathToNotepad
objShortcut.Save
```

    This script adds Notepad to the **Send To** menu when you right-click a file.

14. On the **File** menu, click **Save**.
15. On the **Script** menu, click **Run Script**.

    You are ready to test whether the script succeeded.

16. Click **Start**, right-click **Computer**, and then click **Explore**.
17. Browse to **E:\Labfiles\Solution**.
18. Right-click **WSHShell.vbs**, point to **Send To**, and then click **Notepad**.

The script file opens in Notepad.

▶ Task 2: To instantiate and manipulate the Network object

1. Switch to PrimalScript.
2. On the **File** menu, click **Open**.
3. In the **Open** dialog box, browse to **E:\Labfiles\Starter**.
4. Click **WSHNetwork.vbs**, and then click **Open**.
5. Locate the comment **TODO Instantiate WSH Network object**.
6. Uncomment the following code.

```
Set objNetwork = WScript.CreateObject("WScript.Network")
```

7. Locate the comment **TODO Map a drive to a folder on a server**.
8. Uncomment the following code.

```
objNetwork.MapNetworkDrive "F:", "\\LON-DC1\LabShare"
```

9. On the **File** menu, click **Save**.
10. On the **Script** menu, click **Run Script**.
11. Review the message boxes in the output pane as the script runs.
12. Using Windows Explorer, verify that drive F has been successfully mapped to the LabShare folder on the server computer.

    You have now successfully manipulated the WSH object model.

Exercise 3: Automating Microsoft Office Word

▶ Task 1: To automate Microsoft Office Word

1. Switch to PrimalScript.
2. On the **File** menu, click **Open**.
3. In the **Open** dialog box, browse to **E:\Labfiles\Starter**.
4. Click **CreateDoc.vbs**, and then click **Open**.
5. Locate the comment **TODO Instantiate a Word application object**.
6. Uncomment the following code.

```
Set objWord=WScript.CreateObject("Word.Application")
```

7. Locate the comment **TODO Ensure the Word doc is visible**.
8. Uncomment the following code.

```
objWord.Visible = True
```

9. Locate the comment **TODO Specify a document object for the application object and a selection range**.

10. Uncomment the following code.

```
Set objDoc = objWord.Documents.Add()
Set objSelection = objWord.Selection
```

11. Locate the comment **TODO Assign property values to the selection and type text and a paragraph**.
12. Uncomment the following code.

```
objSelection.Font.Name = "Arial"
objSelection.Font.Size = "18"
objSelection.TypeText "Network Adapter Report"
objSelection.TypeParagraph()
```

13. Locate the comment **TODO Set the font size for the next paragraph and enter the date and a paragraph**.
14. Uncomment the following code.

```
objSelection.Font.Size = "14"
objSelection.TypeText "" & Date()
objSelection.TypeParagraph()
```

    This script writes two lines of text. The first is a heading that displays the text "Network Adapter Report." The second line of text displays today's date.

15. On the **File** menu, click **Save**.
16. On the **Script** menu, click **Run Script**.

    When the Office Word document opens, you will see the text that you defined appear.

Lab: Question and Answers

Lab: Objects in VBScript and WSH

Exercise 1: Manipulating the Scripting Object Model

Question: What is missing from this line of script?

```
MyFSO = CreateObject("Scripting.FileSystemObject")
```

Answer: The **Set** keyword at the beginning of the line.

Question: What ADSI provider is used to access Active Directory?

Answer: LDAP.

Exercise 2: Manipulating the WSH Object Model

Question: What object model provides access to the Windows **Shell** and **Network** objects?

Answer: The WSH object model.

Exercise 3: Automating Microsoft Office Word

Question: To which protocols does the Cdosys.dll allow scriptable access?

Answer: SMTP and NNTP.

Module 3

Script Logic

Contents:

Lab Answer Keys

Lab: Script Logic

Exercise 1: Creating Script Templates

▶ Task 1: To create a PrimalScript template

1. On the 2433B-VISTA-CL1-03 virtual machine, click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
2. On the **File** menu, point to **New**, and then click **File**.
3. In the **New File** dialog box, in the **Categories** list, click **Script Files**, in the **Templates** list, click **VBScript** and then click **OK**.
4. Edit the standard template, so that it looks like the following example.

```
'==========================================================================
'
' VBScript Source File
'
' NAME:
'
' AUTHOR:
' DATE : (Leave as default)
'
' COMMENT:
'
' KEYWORDS:
'
'==========================================================================
'==========================================================================
'Variable Declarations
'==========================================================================
'==========================================================================
' Main Body
'==========================================================================
'==========================================================================
' Procedures
'==========================================================================
```

▶ Task 2: To save the script template

1. On the **File** menu, click **Save As**.
2. In the **Save As** dialog box, navigate to the **C:\Program Files\SAPIEN\PrimalScript**
3. **Classroom\Templates\File Templates\Script Files**.
4. In the **File name** box, type **Fourthcoffee Script.vbs**
5. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.
6. On the **File** menu, click **Close**.

▶ Task 3: To create a new script based on your template

1. On the **File** menu, point to **New**, and then click **File**.
2. In the **New File** dialog box, in the **Categories** list, click **Script Files**, in the **Templates** list, click **Fourthcoffee Script**, and then click **OK**.

A new document is created that is based on your template. Note that the text is colorcoded and formatted as Visual Basic, Scripting Edition. Changes that you make to this new document will not affect the Fourthcoffee Script template.

3. Leave the script open, because you will use it in the next exercise.

Exercise 2: Using and Creating Code Snippets

▶ Task 1: To use a code snippet

1. Click to place the cursor under **Main Body**.
2. On the **View** menu, click **Right Nexus Window**.
3. Click the **Snippets Browser** tab at the bottom of the Right Nexus, to open the Snippets Browser.
4. In the Snippets Browser, expand **VBScript**.
5. Double-click **IfElse** to create an empty **If…Then…Else** statement.

▶ Task 2: To create a new code snippet

1. Modify the **If…Then…Else** statement so that it includes an **ElseIf** snippet.

```
'=====================================================================
' Main Body
If <condition> Then
<statements>
ElseIf <condition> Then
<statements>
End If
'=====================================================================
```

2. Highlight the code for the **If…Then…ElseIf** statement, right-click, and then click **Copy**.
3. In the Snippets Browser, right-click **VBScript**, and then click **New Snippet**.
4. Type **IfElseIf** as the snippet name, and press ENTER.
5. In the IfElseIf.snippet window, right-click, and then click **Paste**.
6. On the **File** menu, click **Save**.
7. On the **File** menu, click **Close** to close IfElseIf.snippet.
8. On the **File** menu, click **Close** to close Untitled.vbs, and then click **No**.

Exercise 3: Adding Constants, Variables, Loops, and Conditional Structures to Scripts

▶ Task 1: Task 1: To complete the script information header

1. On the **File** menu, point to **New**, and then click **File**.
2. In the **New File** dialog box, in the **Categories** list, click **Script Files**, in the **Templates** list, click **Fourthcoffee Script**, and then click **OK**.
3. Modify the script information header so that it reads as shown in the following example.

```
'=====================================================================
'
' VBScript Source File
'
```

```
' NAME: Validate.vbs
'
' AUTHOR: ADD YOUR NAME HERE
' DATE : (Leave as default)
'
' COMMENT: Validates administrative users
'
' KEYWORDS: Loops, variables, constants
'
'=====================================================================
```

4. Notice that PrimalScript automatically adds today's date; edit this date into your preferred format if necessary.

▶ Task 2: To declare constants and variables

1. Under **Variable Declarations**, declare the following variables:
   - iLoopCount
   - sUserName
   - iAnswer
2. In the same section, declare a constant named ADMIN_1, with a value of ADRIAN LANNIN. This constant must be entered in uppercase for the script to work.
3. Declare another constant named ADMIN_2, with a value of JO BERRY.

   The complete section should resemble the following code example.

```
Dim iLoopCount, sUserName, iAnswer
Const ADMIN_1 = "ADRIAN LANNIN"
Const ADMIN_2 = "JO BERRY"
```

▶ Task 3: To use a decision-making structure

1. Under **Main Body**, add the following line, and notice that PrimalScript will autocomplete iLoopCount.

```
iLoopCount = iLoopCount + 1
```

2. Click at the end of the new line, and then press ENTER. Make sure that the cursor is in the new line.
3. In the Snippets Browser, double-click **IfElseIf**, to create an empty **If…Then…ElseIf** statement.
4. Edit the **If…Then…ElseIf** statement to resemble the following code.

```
If iLoopCount > 3 then
WScript.Echo "Maximum attempts exceeded!" & vbCrLf
& "This script will now end..."
WScript.Quit
ElseIf iLoopCount > 1 then
iAnswer = MsgBox("You must be an Administrator to run this script." _
& vbCrLf & "Do you want to try again?", vbYesNo)
If iAnswer = vbNo then
WScript.Quit
End If
End If
```

> **Tip:** PrimalScript will auto-complete iLoopCount, .Echo, .Quit, iAnswer and MsgBox. PrimalScript will auto-format WScript, vbCrLf, vbYesNo, and vbNo. Use the Snippets Browser to insert the nested **If…Then** statement, using the **IfThen** code example.
>
> Note also that in the code text above, the line-continuation characters ( ) are used to denote that you should type the subsequent code on the same line as the preceding code. The text has simply wrapped around in the printed document.

5. Immediately below the final **End If** statement, add the following line to gather user input and assign it to the variable sUserName, and notice that PrimalScript will autocomplete **sUserName** and **InputBox**.

```
sUserName = InputBox ("Please enter your name (Firstname Lastname)")
```

6. On the **File** menu, click **Save As**.
7. Navigate to **E:\Labfiles\Starter**.
8. In the **File name** box, type **Validate.vbs** and then click **Save**.

▶ Task 4: To use a Do…Loop

1. Place the cursor immediately after the comment **Main Body**, and then press ENTER to create a new line.
2. In the Snippets Browser, double-click **LoopWhile**, to create an empty **Do…Loop While** statement.
3. Delete the highlighted **<statement>**, because you have already entered the code that will be executed within the **Do…Loop** statement.
4. Highlight the **Loop While <condition>** code, and on the **Edit** menu, click **Cut**.
5. Place the cursor immediately after the **InputBox** statement, and then press ENTER to create a new line.
6. On the **Edit** menu, click **Paste**.
7. Edit the **Loop While** statement, which controls the termination of the loop, to resemble the following code.

```
Loop While UCase(sUserName) <> ADMIN_1 And UCase(sUserName) <> ADMIN_2
```

> **Tip:** PrimalScript will auto-complete uCase, sUserName, ADMIN_1, and ADMIN_2.

8. Immediately below the **Loop** statement, type the text shown in the following example.

```
WScript.Echo "You have been validated!" & vbCrLf _
    & "The script will now continue..."
```

The Main Body of the script should now look like the following example.

```
'=====================================================================
' Main Body
Do
iLoopCount = iLoopCount + 1
If iLoopCount > 3 Then
WScript.Echo "Maximum attempts exceeded!" & VbCrLf & "This script will
 now end .."
```

```
WScript.Quit
ElseIf iLoopCount > 1 Then
iAnswer = MsgBox("You must be an Administrator to run this script."_
& VbCrLf & "Do you want to try again?", vbYesNo)
If iAnswer = vbNo Then
WScript.Quit
End If
End If
sUserName = InputBox("Please enter your name (Firstname Lastname)")
Loop While UCase(sUserName)<> ADMIN_1 And UCase(sUserName)<>ADMIN_2
WScript.Echo "You have been validated!" & VbCrLf _
& "The script will now continue..."
'========================================================================
```

9. On the **File** menu, click **Save**.

▶ Task 5: To test the script

1. On the **Script** menu, click **Run Script**.
2. In the input box, type **Yan Li,** and click **OK**.

   Yan Li is not one of the administrators. You will be notified of this with a message box.
3. In the message box, click **No**.

   The script ends.
4. Repeat Steps 1 and 2.
5. Click **Yes** to try again.
6. In the input box, type **Yan Li**, and click **OK** again.
7. Click **Yes** to try again.
8. In the input box, type **Yan Li**, and click **OK** one more time.

   You are notified that you have exceeded the maximum number of attempts. The script then ends. Note that, because this message is displayed using **WScript.Echo** instead of **MsgBox**, you will need to use the PrimalScript Output Window window to see the message.
9. On the **Script** menu, click **Run Script**.
10. In the input box, type **Jo Berry**, and click **OK**.
11. In PrimalScript Output Window, verify that you have been notified of a successful validation and that the script will continue.

Exercise 4: Adding Procedures to Scripts

▶ Task 1: To create a Validate function

1. Place the cursor immediately after the comment **Procedures**, and then press ENTER to create a new line.
2. In the Snippets Browser, double-click **Function** to create an empty **Function** statement.
3. Edit the first line of the **Function** statement to resemble the following code.

```
Function Validate(sUser)
```

4. Select the <**statements**> line and, in the Snippets Browser, double-click **IfElse** to replace <**statements**> with an **If...Then...Else** statement.
5. Edit the **If...Then...Else** statement to resemble the following code.

```
If UCase(sUser) = ADMIN_1 Or UCase(sUser) = ADMIN_2 then
Validate = True
Else
Validate = False
End if
End Function
```

6. Under **Main Body**, modify the **Loop** statement so that it uses the function, as the following example shows.

```
Loop While Not Validate(sUserName)
```

The **Main Body** and **Procedures** sections of the script should now look like the following example.

```
'=========================================================================
' Main Body
Do
iLoopCount = iLoopCount + 1
If iLoopCount > 3 Then
WScript.Echo "Maximum attempts exceeded!" & VbCrLf _
& "This script will now end .."
WScript.Quit
ElseIf iLoopCount > 1 Then
iAnswer = MsgBox("You must be an Administrator to run this script."_
& VbCrLf & "Do you want to try again?", vbYesNo)
If iAnswer = vbNo Then
WScript.Quit
End If
End If
sUserName = InputBox("Please enter your name (Firstname Lastname)")
Loop While Not Validate(sUserName)
WScript.Echo "You have been validated!" & VbCrLf _
& "The script will now continue..."
'=========================================================================
'=========================================================================
' Procedures
Function Validate(sUser)
If UCase(sUser) = ADMIN_1 Or UCase(sUser) = ADMIN_2 Then
Validate=True
Else
Validate=False
End If
End Function
'=========================================================================
```

7. On the **File** menu, click **Save**.
8. On the **Script** menu, click **Run Script**.
9. In the input box, type **Yan Li**, and click **OK**.

   Yan Li is not one of the administrators. You will be notified accordingly in a message box.
10. In the message box, click **Yes**.
11. In the input box, type **Adrian Lannin**, and click **OK**.

12. Use the PrimalScript Output Window window to verify that you have been notified of a successful validation and that the script will continue.
13. Close PrimalScript.

Lab: Question and Answers

Lab: Script Logic

Exercise 1: Creating Script Templates

Question: Which feature of Visual Basic, Scripting Edition is case-sensitive?

Answer: Visual Basic, Scripting Edition compares strings in a case-sensitive manner.

Question: What is wrong with the following code?

```
WScript.Echo "Message to _
& vUserName _
& Please make sure that you remember _
& to log off or lock your workstation _
& if you leave it unattended, Thanks"
```

Answer: If you split a long string across multiple lines, you must manage the string data on each line as a separate string. You must enclose each string with double quotes, and then concatenate each string to the previous line by using the string-concatenation character (&).

Exercise 2: Using and Creating Code Snippets

Question: When assigning data to a variable, how do you specify that the data represents a date or time?

Answer: The data must be enclosed by the number sign (#).

Question: The following script causes a compilation error. What is the problem?

```
If MyVar = 5 Then
WScript.Echo "MyVar = 5"
If MyVar = 7 Then
WScript.Echo "MyVar = 7"
End If
```

Answer: There are two **Block If** structures, but only one **End If** statement.

Exercise 3: Adding Constants, Variables, Loops, and Conditional Structures to Scripts

Question: What is the primary data type used in Visual Basic, Scripting Edition?

Answer: Variant.

Question: What is the difference between a constant and a variable?

Answer: The value assigned to a constant cannot change during the execution of a script.

Exercise 3: Adding Procedures to Scripts

Question: What would be the result of the following?

(10 – 5 + 10) * 2 – (4 * 5 / 2)

Answer: 20

Question: What are the main benefits of using procedures in scripts?

Answer: Answers may vary. One possible answer is: Procedures are used to encapsulate blocks of code that are reused several times within a script, or that may be required in more than one script.

Module 4

Error Handling and Debugging

Contents:

Lab Answer Keys

Lab: Error Handling and Debugging

Exercise 1: Trapping Run-Time Errors

▶ Task 1: To execute a script that can cause run-time errors

1. On the 2433B-VISTA-CL1-04 virtual machine, use Windows Explorer to navigate to the E:\Labfiles\Starter folder.
2. Double-click **ExamAverage.vbs**.
3. In the **Enter the Total points scored** box, type **1284** and then click **OK**.
4. In the **Enter the number of exams** box, type **0** and then click **OK**.

   A Division by Zero run-time error occurs.
5. In the **Windows Script Host** dialog box, click **OK**.
6. Repeat Steps 2 and 3.
7. In the **Enter the number of exams** box, type **banana** and then click **OK**.

   A type-mismatch run-time error occurs.
8. In the **Windows Script Host** dialog box, click **OK**.

▶ Task 2: To write error-handling code

1. On the **Start** menu, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalSCRIPT 4.1 Classroom**.
2. In PrimalScript Enterprise, on the **File** menu, click **Open**.
3. In the **Open** dialog box, browse to **E:\Labfiles\Starter**.
4. Click **ExamAverage.vbs**, and then click **Open**.
5. Locate the comment **'TODO Set On Error Resume Next for script**.
6. Uncomment the following code.

```
On Error Resume Next
```

7. Locate the comment **' TODO Handle Errors**.
8. Uncomment the following code.

```
Select Case Err.Number
Case 0
if CInt(intNumberOfTests) <> CDbl(intNumberOfTests) then
strError = "Number of tests must be a whole number (no fractions)"
elseIf CInt( intNumberOfTests ) < 0 then
strError = "Number of tests cannot be negative"
else
wScript.echo "Average score = " & numAverage
end if
Case 11
strError = "Cannot divide by zero"
Case 13
strError = "Only numeric data is allowed"
```

```
Case else
strError = "An unexpected error occurred"
End Select
```

9. Locate the comment **'TODO Write error to host**.
10. Uncomment the following code.

```
wscript.echo strError
```

11. Locate the comment **TODO Exit Gracefully**.
12. Uncomment the following code.

```
If len(strError) > 0 then
wScript.Echo strError
intAnswer = MsgBox("Do you want to try again?", vbYesNo)
if intAnswer = vbNo then
wScript.Quit
end if
```

13. Locate the comment **'TODO Clear Error**.
14. Uncomment the following code.

```
err.Clear
end if 'len(strError)
```

15. On the **File** menu, click **Save**.

▶ Task 3: To test your error-handling code regarding division by zero

1. On the **Script** menu, click **Run Script**.
2. In the **Enter the Total points scored** box, type **1284** and then click **OK**.
3. In the **Enter the number of exams** box, type **0** and then click **OK**.

   The error is trapped. In the output pane, a message tells you that there is a division by zero error.

   You are prompted as to whether you want to try again.
4. Click **Yes**.

▶ Task 4: To test your error-handling code regarding type mismatch

1. In the **Enter the Total points scored** box, type **1284** and then click **OK**.
2. In the **Enter the number of exams** box, type **banana** and then click **OK**.

   The error is trapped. In the output pane, a message tells you that only numeric data is allowed.

   You are prompted as to whether you want to try again.
3. Click **Yes**.

▶ Task 5: To test the recoverability of your code

1. In the **Enter the Total points scored** box, type **1284** and then click **OK**.
2. In the **Enter the number of exams** box, type **20** and then click **OK**.

   The script runs successfully. In the output pane, you are informed that the average score is 64.2.

You have now successfully handled run-time errors for this script.

Exercise 2: Debugging VBScript

▶ Task 1: To port a script to the server

1. On the **File** menu, click **Open**.
2. Browse to **E:\Labfiles\Starter**, click **Debug_Exercise.vbs**, and then click **Open**.
3. On the **File** menu, click **Save As**.
4. In the **Save As** dialog box, in the **File name** box, type **\\LONDC1\Labshare\DebugExercise.vbs** and then click **Save**.
5. Close PrimalScript Enterprise.

▶ Task 2: To test the script for run-time errors

1. Switch to the 2433B-LON-DC1 virtual machine.
2. Click **Start**, click **MyComputer**, double-click **Local Disk**, and then double-click **Labshare**.
3. Double-click **DebugExercise.vbs**.
4. In the **How many days did you work last week** box, type **5** and then click **OK**.
5. In the **Enter hours for day 0** box, type **7.5** and then click **OK**.
6. Repeat Step 5 for each input box.
7. In the message box that informs you how many hours were worked in total, click **OK**.
8. In the message box that informs you of the average daily shift, click **OK**.

   The script does not have any run-time errors. However, it prompts you for details about the hours worked for more than five days, which can be considered a logic error.

▶ Task 3: To debug the script

1. Click **Start**, point to **All Programs**, point to **Accessories**, and then click **Command Prompt**.
2. At the command prompt, type the following command, and then press ENTER.

   ```
   cd..\..\Labshare
   ```

3. At the command prompt, type the following command, and then press ENTER.

   ```
   cscript debugexercise.vbs //x
   ```

   The Microsoft Script Debugger appears. Execution is halted on the first executable line of code.
4. On the **View** menu, click **Command Window**.

   The Command window appears.
5. On the **Debug** menu, click **Step Into**, and then repeat.

   The VBScript input box appears. Note that it may be hidden behind the debugger window, so switch to the input box if necessary by using the Windows taskbar.
6. Enter **5** as the number of days worked last week, and then click **OK**.

7. Switch to the Command window, type **?intDays** and then press ENTER.

   The Command window displays the current value of the intDays variable. Note that the intDays variable is correctly set to five at this point in the code.

8. From the **Debug** menu, click **Step Into**.

   Note that the highlighted code is about to redimension the **intHours()** array. If this line of code executes, the array contains six elements.

   You have found the first logic error. The array should contain five elements rather than six. It should contain one element for each day of the week that was worked. Remember that arrays are zero-based in Visual Basic, Scripting Edition.

▶ Task 4: To fix the logic errors in your code

1. On the **Debug** menu, click **Stop Debugging**.
2. Close the Microsoft Script Debugger.
3. At the command prompt, type the following command, and then press ENTER.

```
notepad debugexercise.vbs
```

4. In Notepad, find the line of code that redimensions the **intHours()** array so that it reads as follows.

```
ReDim intHours(intDays-1)
```

5. Modify the subsequent line so that it reads as follows.

```
For intLoop = 0 to intDays – 1
```

6. Modify the next line so that it reads as follows.

```
intHours(intLoop) = CSng(InputBox("Enter hours for day " & intLoop + 1))
```

7. On the **File** menu, click **Save**.

▶ Task 5: To test your fixed code

1. Switch to the command prompt.
2. At the command prompt, type the following command, and then press ENTER.

```
Cscript debugexercise.vbs
```

3. After the message box opens, enter **5** as the number of days worked last week, and then click **OK**.
4. Enter **7.5** as the number of hours worked, and then click **OK**.
5. Enter the following data for the other input boxes:
   - Day 2: **8**
   - Day 3: **8**
   - Day 4: **7**
   - Day 5: **7**

6. Verify that you are now only prompted for hours worked on five days.
7. Close Notepad.

You have now successfully used the script debugger to debug a script that was written in Visual Basic, Scripting Edition.

Lab: Question and Answers

Lab: Error Handling and Debugging

Exercise 1: Trapping Run-Time Errors

Question: What is the name of the object that holds the details about run-time errors?

Answer: The **Err** object.

Question: Name two commonly used properties of the **Err** object.

Answer: Number and **Description**.

Exercise 2: Debugging VBScript

Question: What are the three stepping options that the Microsoft Script Debugger supports?

Answer: Step Into, **Step Over**, and **Step Out**.

Module 5

ADSI

Contents:

Lab Answer Keys

Lab A: ADO Search

Exercise 1: Performing Searches by Using ADSI and ADO

▶ Task 1: To create a new script by using a template

1. On the 2433B-VISTA-CL1-05 virtual machine, click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
2. On the **File** menu, click **Open**.
3. In the **Open** dialog box, navigate to the E:\Labfiles\Starter folder.
4. In the **Open** dialog box, click **ADOTemplate.vbs** and click **Open**.
5. Modify the script information header so that it includes your name and today's date.
6. On the **File** menu, click **Save As**.
7. In the **File name** box, type **ADOSearch.vbs**
8. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.

▶ Task 2: To investigate the ADO Search template

1. The following lines of code declare the variables that are used in the script. The first three variables are used to manipulate ADO objects and are used to instantiate and manipulate a **Connection** object, a **Command** object, and a **Recordset** object, respectively.

```
Option Explicit
Dim aConnection, aCommand, aResult, sResultText
```

2. The following lines of code instantiate the **Connection** and **Command** objects.

```
Set aConnection = CreateObject("ADODB.Connection")
Set aCommand = CreateObject("ADODB.Command")
```

3. The following lines of code manipulate the **Connection** and **Command** objects.

```
aConnection.Provider = "ADsDSOObject"
aConnection.Open
aCommand.ActiveConnection = aConnection
aCommand.CommandText = "<LDAP://OU=OU-name,
DC=domain-name,DC=domain-suffix>;" _
      & "(objectClass=class-type);" _
      & "object-attribute1,object-attribute2,object-attribute3,
object-attribute4;" _
      & "subTree"
```

4. The following line of code creates the **Recordset** object.

```
Set aResult = aCommand.Execute()
```

5. The following lines of code loop through the **Recordset** object and build a string from its contents.

```
Do While Not aResult.EOF
       sResultText = sResultText _
```

```
              & aResult.Fields("object-attribute1") _
              & ", " & aResult.Fields("object-attribute2") _
              & ", " & aResult.Fields("object-attribute3") _
              & ", " & aResult.Fields("object-attribute4") _
              & VbCrLf
       aResult.MoveNext
Loop
```

6. The last line of the code completes the script by displaying the string to the user.

```
WScript.Echo sResultText
```

▶ Task 3: To edit the ADO Search template

1. Specify that you want to search the **2433Users** OU, by editing the line of code that contains the <LDAP://...> string, so that it resembles the following code.

```
aCommand.CommandText = "<LDAP://OU=2433Users,DC=fourthcoffee,DC=com>;" _
```

2. Specify the class type as **user**, by editing the line of code that contains the **objectClass** statement, so that it resembles the following code.

```
      & "(objectClass=user);" _
```

3. Specify the attributes to return from the search, by editing the line of code that contains the object-attribute statements, so that it resembles the following code.

```
      & " sAMAccountname,givenName,sN,telephoneNumber;" _
```

4. Specify the attributes to display, by editing the lines of code that contain the **aResult.Fields** statements, so that they resemble the following code.

```
Do While Not aResult.EOF
       sResultText = sResultText _
              & aResult.Fields("sAMAccountname") _
              & ", " & aResult.Fields("givenName") _
              & ", " & aResult.Fields("sN") _
              & ", " & aResult.Fields("telephoneNumber") _
              & VbCrLf
       aResult.MoveNext
Loop
```

5. On the **File** menu, click **Save**.
6. On the **Script** menu, click **Run Script**.
7. Review the details that are displayed in the PrimalScript Output window.
8. Minimize PrimalScript.
9. Click **Start**, and then click **Computer**.
10. In the **Computer** folder, double-click **AllFiles (E:)**, double-click **Labfiles**, and then double-click **Starter**.
11. In the **Starter** folder, double-click **ADOSearch.vbs**.
12. Review the details that are displayed in the message box, and click **OK**.

Exercise 2: Searching for User and Computer Information by Using ADSI and ADO

▶ Task 1: To search for additional user information

1. Edit your ADOSearch.vbs script, to include additional **object-attribute** statements. Choose the extra user object attributes from the examples that are listed in the following table.

UI label	Active Directory attribute
First Name	givenName
Last Name	sn
Initials	initials
Display Name	displayName
Office	physicalDeliveryOfficeName
Telephone Number	telephoneNumber
E-Mail	mail
Web Page	wwwHomePage
UserLogon Name	userPrincipalName
User logon name (pre-Windows 2000)	sAMAccountname
Street	streetAddress
City	l (lowercase "L" as in Locale)
State/Province	st
Zip/Postal Code	postalCode
Country/Region	c, co, and countryCode
Title	title
Department	department
Company	company

Tip: A complete list of attributes is given in the Active Directory User Interface Mappings section of the Platform Software Development Kit (SDK) for Windows Server® 2003 R2.

2. Edit your ADOSearch.vbs script, so that the **aResult.Fields** statements include the same user object attributes as you specified in Step 1.
3. On the **File** menu, click **Save As**.
4. In the **File name** box, type **ADOSearch-Users.vbs**
5. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.

6. On the **Script** menu, click **Run Script**.
7. Review the details that are displayed in the PrimalScript Output window.
8. On the **File** menu, click **Close**.
9. Minimize PrimalScript.
10. Click **Start**, and then click **Computer**.
11. In the **Computer** folder, double-click **AllFiles (E:)**, double-click **Labfiles**, and then double-click **Starter**.
12. Double-click **ADOSearch-Users.vbs**.
13. Review the details that are displayed in the message box, and click **OK**.

▶ Task 2: To search for computer information

1. Switch to PrimalScript.
2. On the **File** menu, click **Open**.
3. In the **Open** dialog box, navigate to the E:\Labfiles\Starter folder.
4. In the **Open** dialog box, click **ADOTemplate.vbs**, and then click **Open**.
5. Modify the script information header so that it includes your name and today's date.
6. On the **File** menu, click **Save As**.
7. In the **File name** box, type **ADOSearch-Computers.vbs**
8. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.
9. Specify that you want to search the whole fourthcoffee.com domain, by editing the line of code that contains the <LDAP://…> string, so that it resembles the following code.

```
aCommand.CommandText = "<LDAP://DC=fourthcoffee,DC=com>;" _
```

10. Specify the class type as **computer**, by editing the line of code that contains the **objectClass** statement, so that it resembles the following code.

```
        & "(objectClass=computer);" _
```

11. Specify the attributes to return from the search, by editing the line of code that contains the **object-attribute** statements, so that it resembles the following code.

```
        & "sAMAccountname,dNSHostName,operatingSystem,operatingSystemVersion;" _
```

12. Specify the attributes to display, by editing the lines of code that contain the **aResult.Fields** statements, so that they resemble the following code.

```
            & aResult.Fields("sAMAccountname") _
            & ", " & aResult.Fields("dNSHostName") _
            & ", " & aResult.Fields("operatingSystem") _
            & ", " & aResult.Fields("operatingSystemVersion") _
```

13. On the **File** menu, click **Save**.
14. On the **Script** menu, click **Run Script**.

15. Review the details that are displayed in the PrimalScript Output window.
16. Minimize PrimalScript.
17. Click **Start**, and then click **Computer**.
18. In the **Computer** folder, double-click **AllFiles (E:)**, double-click **Labfiles**, and then double-click **Starter**.
19. Double-click **ADOSearch-Computers.vbs**.
20. Review the details that are displayed in the message box, and click **OK**.

Lab B: Scripting Administrative Tasks by Using ADSI

Exercise 1: Retrieving Properties by Using ADSI

▶ Task 1: To create a new script by using a template

1. If PrimalScript is not running, click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
2. On the **File** menu, click **Open**.
3. In the **Open** dialog box, navigate to the E:\Labfiles\Starter folder.
4. In the **Open** dialog box, click **ListObjectsTemplate.vbs**, and then click **Open**.
5. Modify the script information header so that it includes your name and today's date.
6. On the **File** menu, click **Save As**.
7. In the **File name** box, type **ListUsers.vbs**
8. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.

▶ Task 2: To investigate the List Objects template

1. The following lines of code configure the error behavior and declare the constant used in the script to specify that the search scope is **subtree**.

```
On Error Resume Next
Const ADS_SCOPE_SUBTREE = 2
```

2. The following lines of code instantiate the **Connection** and **Command** objects.

```
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
```

3. The following lines of code manipulate the **Connection** and **Command** objects.

```
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection
```

4. The following lines of code specify that the script must return a maximum of 1000 records, and that the script must search the whole Active Directory tree.

```
objCommand.Properties("Page Size") = 1000
objCommand.Properties("Searchscope") = ADS_SCOPE_SUBTREE
```

5. The following line of code creates the query, and specifies the type of object to return.

```
objCommand.CommandText = _
"SELECT Name FROM 'LDAP://dc=domain-name,dc=domain-suffix' WHERE
objectCategory='object-category'"
```

6. The following line of code creates the **Recordset** object.

```
Set objRecordSet = objCommand.Execute
```

7. The following lines of code loop through the **Recordset** object, build a string from its contents, and display the string to the user.

```
objRecordSet.MoveFirst
Do Until objRecordSet.EOF
     Wscript.Echo objRecordSet.Fields("Name").Value
     objRecordSet.MoveNext
Loop
```

▶ Task 3: To edit the List Objects template

1. Specify that you want to search the fourthcoffee.com domain, and that you want to return a list of users, by editing the line of code that contains the 'LDAP://dc= …' string, so that it resembles the following code.

```
"SELECT Name FROM 'LDAP://dc=fourthcoffee,dc=com' WHERE objectCategory='user'"
```

2. On the **File** menu, click **Save**.
3. On the **Script** menu, click **Run Script**.
4. Review the details that are displayed in the PrimalScript Output window.
5. Minimize PrimalScript.
6. Click **Start**, and then click **Computer**.
7. In the **Computer** folder, double-click **AllFiles (E:)**, double-click **Labfiles**, and then double-click **Starter**.
8. Double-click **ListUsers.vbs**.
9. Review the details that are displayed in the message box, and click **OK**.
10. Repeat step 9 for each message box.

▶ Task 4: To create scripts to list computers, groups, and OUs

1. Switch to PrimalScript.
2. On the **File** menu, click **Open**.
3. In the **Open** dialog box, navigate to the E:\Labfiles\Starter folder.
4. In the **Open** dialog box, click **ListObjectsTemplate.vbs**, and then click **Open**.
5. Modify the script information header so that it includes your name and today's date.
6. On the **File** menu, click **Save As**.
7. In the **File name** box, type **ListComputers.vbs**
8. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.
9. Edit the line of code that contains the 'LDAP://dc= …' string, so that it resembles the following code.

```
SELECT Name FROM 'LDAP://dc=fourthcoffee,dc=com' WHERE
➥objectCategory='computer'"
```

10. On the **File** menu, click **Save**.

11. On the **Script** menu, click **Run Script**.
12. Review the details that are displayed in the PrimalScript Output window.
13. Repeat Steps 3–12, so that the **objectCategory** is **group** and the script name is **ListGroups.vbs**.
14. Repeat Steps 3–12, so that the **objectCategory** is **organizationalunit** and the script name is **ListOUs.vbs**.
15. Close PrimalScript.

Exercise 2: Creating OUs, Users, and Groups by Using ADSI

▶ Task 1: To install ADSI Scriptomatic

1. Click **Start**, and then click **Computer**.
2. In the **Computer** folder, double-click **AllFiles (E:)**, double-click **Labfiles**, and then double-click **Starter**.
3. Double-click **EZADScriptomatic.exe**.
4. In the **WinZip Self-Extractor** dialog box, in the **Unzip to folder** box, type **E:\Labfiles\Starter\ADSI Scriptomatic**
5. In the **WinZip Self-Extractor** dialog box, click **Unzip**.
6. In the **WinZip Self-Extractor** message box, click **OK**.
7. In the **WinZip Self-Extractor** dialog box, click **Close**.

▶ Task 2: To use ADSI Scriptomatic to produce a "create user" script

1. In the E:\Labfiles\Starter folder, double-click **ADSI Scriptomatic**.
2. Double-click **EZADScriptomatic.hta**.
3. In the **EzAD Scriptomatic** dialog box, in the **Select a task** list, select **Create an Object**.
4. In the **EzAD Scriptomatic** dialog box, in the **Select a class** list, select **user**.
5. Edit the first two lines of the EzAD Scriptomatic code, so that they resemble the following code.

```
strContainer = "ou=2433Users"
strName = "TestUser"
```

6. In the **EzAD Scriptomatic** dialog box, click **Run**.
7. Close the command prompt window.
8. Switch to the 2433B-LON-DC1 virtual machine.
9. Click **Start**, point to **Administrative Tools**, and then click **Active Directory Users and Computers**.
10. In the left pane, expand the **fourthcoffee.com** domain.
11. Select the **2433Users** container, and verify that the user account **TestUser** has been created. Notice that this account is disabled.

▶ Task 3: To use PrimalScript to investigate the ADSI Scriptomatic code

1. Switch to the 2433B-VISTA-CL1-05 virtual machine.

2. In the **EzAD Scriptomatic** dialog box, click **Save**.
3. In the file name box, type **E:\Labfiles\Starter\CreateUser.vbs** and then click **OK**.
4. In the **EzAD Scriptomatic** dialog box, click **Quit**.
5. Click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
6. On the **File** menu, click **Open**.
7. In the **Open** dialog box, navigate to the E:\Labfiles\Starter folder.
8. In the **Open** dialog box, click **CreateUser.vbs**, and then click **Open**.
9. The following lines of code declare the variables that are used in the script.

```
strContainer = "ou=2433Users"
strName = "TestUser"
```

10. The following lines of code instantiate the **Connection** object by using either the **rootDSE** object if strContainer is left blank, or a path if a container is specified in

```
strContainer.
Set objRootDSE = GetObject("LDAP://rootDSE")
If strContainer = "" Then
Set objContainer = GetObject("LDAP://" & _
objRootDSE.Get("defaultNamingContext"))
Else
Set objContainer = GetObject("LDAP://" & strContainer & "," & _
objRootDSE.Get("defaultNamingContext"))
End If
```

11. The following lines of code create the **user** object, write the **sAMAccountname** property, and then call the **SetInfo** method to ensure that the changes are written to Active Directory.

```
Set objUser = objContainer.Create("user", "cn=" & strName)
objUser.Put "sAMAccountName", strName
objUser.SetInfo
```

▶ Task 4: To use PrimalScript to modify the ADSI Scriptomatic code

1. To specify a new user account name and additional account properties, edit the opening lines of the script so that they resemble the following code.

```
strContainer = "ou=2433Users"
strName = "Kathie Flood"
strFirstName = "Kathie"
strLastName = "Flood"
strDepartment = "MIS"
strPassword = "TempPa$$w0rd"
```

2. To write the account properties, and call the **SetInfo** method to ensure that the changes are written to Active Directory, edit the final lines of the script so that they resemble the following code:

```
Set objUser = objContainer.Create("user", "cn=" & strName)
objUser.Put "sAMAccountName", strName
objUser.Put "givenName", strFirstName
objUser.Put "sn", strLastName
```

```
objUser.Put "department", strDepartment
objUser.SetInfo
```

> **Tip:** The user object attribute names are listed in the table provided in Lab A.

3. You cannot enable a new user account until it has been created, so add the following lines at the end of the script to enable the account and set a temporary password.

```
objUser.SetPassword strPassword
objUser.AccountDisabled = False
objUser.SetInfo
```

> **Important:** In the above code, **SetInfo** is called a second time. The first **SetInfo** creates the user account, and the second **SetInfo** updates the account with the "Enabled" status and a password.

4. Your completed script should now look like the following example.

```
strContainer = "ou=2433Users"
strName = "Kathie Flood"
strFirstName = "Kathie"
strLastName = "Flood"
strDepartment = "MIS"
strPassword = "TempPa$$w0rd"
'*********************************************
'* Connect to a container *
'*********************************************
Set objRootDSE = GetObject("LDAP://rootDSE")
If strContainer = "" Then
Set objContainer = GetObject("LDAP://" & _
objRootDSE.Get("defaultNamingContext"))
Else
Set objContainer = GetObject("LDAP://" & strContainer & "," & _
objRootDSE.Get("defaultNamingContext"))
End If
'*********************************************
'* End connect to a container *
'*********************************************
Set objUser = objContainer.Create("user", "cn=" & strName)
objUser.Put "sAMAccountName", strName
objUser.Put "givenName", strFirstName
objUser.Put "sn", strLastName
objUser.Put "department", strDepartment
objUser.SetInfo
objUser.SetPassword strPassword
objUser.AccountDisabled = False
objUser.SetInfo
```

5. On the **File** menu, click **Save**.
6. On the **Script** menu, click **Run Script**.
7. Review the details that are displayed in the PrimalScript Output window.
8. Switch to the 2433B-LON-DC1 virtual machine.

9. Right-click the **2433Users** container, and click **Refresh**.
10. Verify that the user account **KathieFlood** has been created. Notice that this account is enabled.
11. Right-click **KathieFlood**, and then click **Properties**.
12. On the **General** tab, verify that the object attributes for **First name** and **Last name** have been written.
13. Click the **Organization** tab, and verify that the object attribute for **Department** has been written

▶ Task 5: To use ADSI Scriptomatic to create a "create group" script

1. Switch to the 2433-VISTA-CL1-05 virtual machine.
2. Click **Start**, and then click **Computer**.
3. In the **Computer** folder, double-click **AllFiles (E:)**, double-click **Labfiles**, doubleclick **Starter**, and then double-click **ADSI Scriptomatic**.
4. Double-click **EZADScriptomatic.hta**.
5. In the **EzAD Scriptomatic** dialog box, in the **Select a task** list, select **Create an Object**.
6. In the **EzAD Scriptomatic** dialog box, in the **Select a class** list, select **group**.
7. Edit the sixth and seventh lines of the EzAD Scriptomatic code, so that they resemble the following code.

```
strContainer = "ou=2433Users"
strName = "MISUsers"
```

8. In the **EzAD Scriptomatic** dialog box, click **Run**.
9. Close the command prompt window.
10. Switch to the 2433B-LON-DC1 virtual machine.
11. In **Active Directory Users and Computers**, in the left pane, expand the **fourthcoffee.com** domain.
12. Right-click the **2433Users** container, and click **Refresh**.
13. Verify that the group **MISUsers** has been created.

▶ Task 6: To use ADSI Scriptomatic and PrimalScript to add users to a group

1. Switch to the 2433B-VISTA-CL1-05 virtual machine.
2. In the **EzAD Scriptomatic** dialog box, in the **Select a task** list, select **Write an Object**.
3. In the **EzAD Scriptomatic** dialog box, in the **Select a class** list, select **group**.
4. In the **EzAD Scriptomatic** dialog box, click **Save**.
5. In the file name box, type **E:\Labfiles\Starter\AddUsersToGroup.vbs** and then click **OK**.
6. In the **EzAD Scriptomatic** dialog box, click **Quit**.
7. Switch to PrimalScript.
8. On the **File** menu, click **Open**.
9. In the **Open** dialog box, navigate to the E:\Labfiles\Starter folder.

10. In the **Open** dialog box, click **AddUsersToGroup.vbs**, and then click **Open**.
11. To specify the group name, edit the opening lines of the script so that they resemble the following code.

```
strContainer = "ou=2433Users"
strName = "MISUsers"
```

12. Edit the lines that contain **samAccountName**, **description**, and **mail**, so that they resemble the following code.

```
objItem.Put "samAccountName", "MISUsers"
objItem.SetInfo
objItem.Put "description", "MIS Users"
objItem.SetInfo
objItem.Put "mail", "misusers@fourthcoffee.com"
objItem.SetInfo
```

13. Edit the lines that contain member names, so that they resemble the following code.

```
objItem.PutEx ADS_PROPERTY_UPDATE, "member", _
Array("cn=Kathie Flood,ou=2433Users,dc=fourthcoffee,dc=com",
➥"cn=April Reagan,ou=2433Users,dc=fourthcoffee,dc=com",
➥"cn=David Junca,ou=2433Users,dc=fourthcoffee,dc=com")
objItem.SetInfo
```

> **Note:** You must use distinguished names when specifying users to add to the member list.

14. Edit the lines that contain managedBy, so that they resemble the following code.

```
objItem.Put "managedBy", "cn=Kathie Flood,ou=2433Users,dc=fourthcoffee,dc=com"
objItem.SetInfo
```

15. On the **File** menu, click **Save**.
16. On the **Script** menu, click **Run Script**.
17. Review the details that are displayed in the PrimalScript Output window.
18. Switch to the 2433B-LON-DC1 virtual machine.
19. Right-click the **2433Users** container, and click **Refresh**.
20. Right-click **MISUsers** and click **Properties**.

    Verify that the object attributes for **Description** and **E-mail** have been written, and that the Member list and Managed By information has been updated.

Lab: Question and Answers

Lab A: ADO Search

Exercise 1: Performing Searches by Using ADSI and ADO

Question: What is the main directory service provider used for Windows Server 2003?

Answer: LDAP.

Question: What is the main reason for using the ADO object model in ADSI scripts?

Answer: Answers may vary. One possible answer is: ADO supports efficient and powerful Active Directory search routines.

Exercise 2: Searching for User and Computer Information by Using ADSI and ADO

There are no questions and answers for this exercise.

Lab B: Scripting Administrative Tasks by Using ADSI

Exercise 1: Retrieving Properties by Using ADSI

Question: Before you can use ADSI to manipulate the objects in a directory, what action must your script perform with an ADSI provider?

Answer: Your script must bind to an ADSI provider.

Question: What is the preferred method of binding?

Answer: Serverless binding.

Exercise 2: Creating OUs, Users, and Groups by Using ADSI

Question: When you use the properties of an object, where are the updates stored?

Answer: Local properties cache.

Question: How are changes committed back to the directory from the local properties cache?

Answer: You must call the **SetInfo** method.

Module 6

Creating Logon Scripts

Contents:

Lab Answer Keys

Lab A: Creating Logon Scripts

Exercise 1: Mapping Drives

▶ Task 1: To map network drives

1. On the 2433B-VISTA-CL1-06 virtual machine, click **Start**, point to **All Programs**, point to **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
2. On the **File** menu, click **Open**.
3. In the **Open** dialog box, browse to **E:\Labfiles\Starter**.
4. Click **LogonScript1.vbs**, and then click **Open**.
5. Locate the comment **TODO Instantiate the Network object**.
6. Uncomment the following code.

```
Set objNetwork = CreateObject("WScript.Network")
```

7. Locate the comment **TODO Create the FileSystemObject and go through its drive collection**.
8. Uncomment the following code example.

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
For Each objDrive in ObjFSO.Drives
strFinalDrive = Left(objDrive, 1)
Next
```

9. Locate the comment **TODO Manipulate string to get the next drive letter**.
10. Uncomment the following code.

```
intChar = Asc(strFinalDrive)
strNewDrive = Chr(intChar + 1)
```

11. Locate the comment **TODO Map a drive to the LabShare folder using the next drive letter**.
12. Uncomment the following code.

```
objNetwork.MapNetworkDrive strNewDrive & ":", "\\LON-DC1\LabShare"
```

13. On the **File** menu, click **Save**.

Exercise 2: Creating Shortcuts

▶ Task 1: To create a shortcut to a text file

1. In PrimalScript, locate the comment **TODO Set up shortcuts. Start by instantiating the Shell object**.
2. Uncomment the following code.

```
Set objShell = CreateObject ("WScript.Shell")
```

3. Locate the comment **TODO Set a variable to the path to the desktop**.

4. Uncomment the following line of code.

```
strDesktop = objShell.SpecialFolders("Desktop")
```

5. Locate the comment **TODO Create the shortcut**.
6. Uncomment the following code.

```
Set objShortcut = objShell.CreateShortcut(strDesktop & "\readme.lnk")
objShortcut.TargetPath = "E:\Labfiles\Solution\Readme.txt"
objShortcut.Save
```

7. Locate the comment **TODO Send notice that the script has completed**.
8. Uncomment the following code.

```
WScript.Echo "Logon Script Complete".
```

▶ Task 2: To test the script

1. On the **File** menu, click **Save**.
2. On the **Script** menu, click **Run Script**.
3. After you receive the message "Logon Script Complete", minimize the PrimalScript window and verify that a **Readme** link exists on the desktop.
4. Click **Start**, right-click **Computer**, click **Explore**, and verify that there is a drive mapped to the LabShare folder on the LON-DC1 computer.

   You have now created and tested your logon script.
5. Click **Start**, right-click **Computer**, and then click **Disconnect Network Drive**.
6. In the **Disconnect Network Drives** dialog box, click **LabShare**, and then click **OK**.
7. On the desktop, right-click the **Readme** shortcut, and then click **Delete**.
8. In the **Delete File** dialog box, click **Yes**.

Exercise 3: Assigning Logon Scripts to a User

▶ Task 1: To assign the script to a user

1. Switch to PrimalScript.
2. On the **File** menu, click **Save As**.
3. In the **File name** box, type **\\LON-DC1\sysvol\fourthcoffee.com\scripts\LogonScript1.vbs** and then click **Save**.
4. Switch to the 2433B-LON-DC1 virtual machine.
5. Click **Start**, point to **All Programs**, point to **Administrative Tools**, and then click **Active Directory Users and Computers**.
6. Click **2433Users**.
7. In the objects pane, right-click **April Reagan**, and then click **Properties**.
8. In the **AprilReagan Properties** dialog box, on the **Profile** tab, in the **Logon script** box, type **.\LogonScript1.vbs** and then click **OK**.

9. Close Active Directory Users and Computers.

▶ Task 2: To verify that the script runs correctly during the logon process

1. Switch to the 2433B-VISTA-CL1-06 virtual machine.
2. Close all open windows and programs.
3. Click **Start**, click the arrow, and then click **Log Off**.
4. Press RIGHT ALT+DELETE.
5. Click **Switch User**, and then click **Other User**.
6. In the **User name** box, type **AprilReagan.**.
7. In the **Password** box, type **Pa$$w0rd** and then press ENTER.
8. Verify that you have a shortcut named **readme** on your desktop and that it opens the Readme.txt file from the E:\LabFiles\Solution\ folder.
9. Click **Start**, right-click **Computer**, and then click **Explore**.
10. Verify that the drive has been mapped as expected.

    You have now successfully created, assigned, and tested a logon script written in Microsoft® Visual Basic, Scripting Edition.

    Note: If the shortcut and mapped network drive do not appear, log off and log on again. When you log on, use the credentials from steps 6 and 7.

11. Close all open windows and programs.
12. Click **Start**, click the arrow, and then click **Log Off**.

Lab B: Assigning Logon Scripts

Exercise 1: Configuring Scripts by Using Group Policy

▶ Task 1: To create the script folder

1. On the 2433B-VISTA-CL1-06 virtual machine, click **Switch User**, and then click **Other User**.
2. In the **User name** box, type **FOURTHCOFFEE\Administrator**, in the **Password** box, type **Pa$$w0rd**, and then click **OK**.
3. After the logon process completes, click **Start**, right-click **Computer**, and then click **Explore**.
4. In Windows Explorer, go to the root of drive C.
5. On the **File** menu, point to **New**, and then click **Folder**.
6. In the **New Folder** box, type **LogonScripts** and then press ENTER.

▶ Task 2: To create the scripts

1. Click **Start**, point to **All Programs**, point to **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
2. On the **File** menu, point to **New**, and click **File**.
3. In the **New File** dialog box, click **Text**, and then click **OK**.
4. Type the following line of code.

```
MsgBox "Starting Up!"
```

5. On the **File** menu, click **Save As**.
6. In the **Save As** box, navigate to the C:\LogonScripts folder.
7. In the **File name** box, type **Startup.vbs** and then click **Save**.
8. Repeat Steps 2–7, to create the following files and message boxes:
   - Logon.vbs

```
MsgBox "Logging on!"
```

   - Logoff.vbs

```
MsgBox "Logging off!"
```

   - Shutdown.vbs

```
MsgBox "Shutting Down!"
```

9. Close PrimalScript.

> **Note:** The Shutdown.vbs and Startup.vbs files do not display a messages box on shutdown and startup because the scripts are not accessible to the computer at that stage of the shutdown and startup processes.

▶ Task 3: To open the Group Policy Microsoft Management Console

1. Click **Start**, point to **All Programs**, point to **Accessories**, and then click **Command Prompt**.
2. At the command prompt, type **MMC** and then press ENTER.
3. On the **File** menu, click **Add/Remove Snap-in**.
4. In the **Available snap-ins** list, click **Group Policy Object Editor**, and then click **Add**.
5. In the **Select Group Policy Object** dialog box, click **Finish**.
6. In the **Add or Remove Snap-ins** dialog box, click **OK**.

▶ Task 4: To assign the startup script

1. In the tree pane, expand **Local Computer Policy**, expand **Computer Configuration**, expand **Window Settings**, and then click **Scripts (Startup/Shutdown)**.
2. In the details pane, click **Startup**.
3. On the **Action** menu, click **Properties**.
4. In the **Startup Properties** dialog box, click **Add**.
5. In the **Add a Script** dialog box, click **Browse**.
6. Browse to the **C:\LogonScripts** folder, click **Startup.vbs**, and then click **Open**.
7. In the **Add a Script** dialog box, click **OK**.
8. In the **Startup Properties** dialog box, click **OK**.

▶ Task 5: To assign the shutdown script

1. In the details pane, click **Shutdown**.
2. On the **Action** menu, click **Properties**.
3. In the **Shutdown Properties** dialog box, click **Add**.
4. In the **Add a Script** dialog box, click **Browse**.
5. Browse to the **C:\LogonScripts** folder, click **Shutdown.vbs**, and then click **Open**.
6. In the **Add a Script** dialog box, click **OK**.
7. In the **Shutdown Properties** dialog box, click **OK**.

▶ Task 6: To assign the logon script

1. In the tree pane, expand **User Configuration**, expand **Windows Settings**, and then click **Scripts (Logon/Logoff**).
2. In the details pane, click **Logon**.
3. On the **Action** menu, click **Properties**.
4. In the **Logon Properties** dialog box, click **Add**.
5. In the **Add a Script** dialog box, click **Browse**.
6. Browse to the **C:\LogonScripts** folder, click **Logon.vbs**, and then click **Open**.
7. In the **Add a Script** dialog box, click **OK**.

8. In the **Logon Properties** dialog box, click **OK**.

Task 7: To assign the logoff script

1. In the details pane, click **Logoff**.
2. On the **Action** menu, click **Properties**.
3. In the **Logoff Properties** dialog box, click **Add**.
4. In the **Add a Script** dialog box, click **Browse**.
5. Browse to the **C:\LogonScripts** folder, click **Logoff.vbs**, and then click **Open**.
6. In the **Add a Script** dialog box, click **OK**.
7. In the **Logoff Properties** dialog box, click **OK**.

   You have now assigned a script to each of the client-side extensions that are provided by Windows Vista.

Task 8: To save the Microsoft Management Console

1. On the **File** menu, click **Save As**.
2. In the **Save As** dialog box, in the **File name** box, type **GrpPol** and then click **Save**.
3. Close the Microsoft Management Console.

Task 9: To test the scripts for the client-side extensions

1. Click **Start**, click the arrow to the right of the **Start Search** box, and then click **Restart**.
2. In the **Logging Off!** message box, click **OK**.

   Note: No **Shutting Down!** message box appears. The script is not available to the computer at this stage of the shutdown process.

   Windows restarts.

   Note: No **Starting Up!** message box appears. The script is not available to the computer at this stage of the startup process.

3. Log on to the FourthCoffee domain as **AprilReagan**, and in the **Password** box, type **pa$$w0rd**
4. In the **Logging On!** message box, click **OK**.

   Note that your logon script, assigned in the previous lab, also runs.

5. Click **OK**.

   Note: Be aware that restarting the 2433B-VISTA-CL1-06 virtual machine can take from five to ten minutes.

Lab: Question and Answers

Lab A: Creating Logon Scripts

Exercise 1: Mapping Drives

Question: What mechanisms can you use to gather user input in a logon script?

Answer: Message boxes, popup boxes, and input boxes.

Exercise 2: Creating Shortcuts

Question: What does the command-line variable %LOGONSERVER% return?

Answer: The full path of the script file that is running.

Exercise 3: Assigning Logon Scripts to a User

Question: What object exposes the **MapNetworkDrive** method?

Answer: The **Network** object.

Lab B: Assigning Logon Scripts

Exercise 1: Configuring Scripts by Using Group Policy

Question: What are the four client-side extensions that are provided by Windows Server 2003?

Answer: Startup, Logon, Logoff, and Shutdown.

Question: What can you assign a logon script to?

Answer: You can assign a logon script directly to a user. Alternatively, you can use Group Policy to assign logon scripts to users, groups of users, and computers.

Module 7

Administrative Scripts

Contents:

Lab Answer Keys

Lab A: Administrative Scripts

Exercise 1: Passing Arguments to Scripts

▶ Task 1: To create a script that accepts file names as arguments

1. On the 2433B-VISTA-CL1-07 virtual machine, click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
2. On the **File** menu, point to **New**, and then click **File**.
3. In the **New File** dialog box, select **Script Files** in the **Categories** list, click **VBScript** in the **Templates** list, and then click **OK**.
4. Edit the template to declare the variables for this script, as the following example shows.

```
Option Explicit
Dim vArg, aArgs(), iCount
```

5. On the View menu, click Right Nexus Window.
6. In the **Right Nexus** window, click the **Snippets Browser** tab.
7. Edit the template, using auto-complete and the Snippets Browser, to insert an **If…Else** block to check for arguments passed to the script and a **For…Next** loop (labeled "ForTo" in the Snippets Browser). The code should resemble the following.

```
If WScript.Arguments.Count = 0 Then
       MsgBox "No Arguments Supplied"
       WScript.Quit
Else
       WScript.Echo "Argument count is: " & WScript.Arguments.Count
       ReDim aArgs(WScript.Arguments.Count - 1)
       For iCount = 0 to WScript.Arguments.Count - 1
              aArgs(iCount) = WScript.Arguments(iCount)
              MsgBox aArgs(iCount)
Next
End If
```

> **Note:** The above code stores the value of each argument in an array (for later use in the lab).

8. On the **File** menu, click **Save As**
9. In the **Save As** dialog box, navigate to the E:\Labfiles\Starter folder.
10. In the **File name** box, type **VBArgs.vbs**
11. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.

▶ Task 2: To run the script and use file names as arguments

1. Click **Start**, and then click **Computer**.
2. In the Computer folder, double-click **AllFiles (E:)**, double-click **Labfiles**, and then double-click **Starter**.

3. Drag **Text1.txt** and drop it onto the VBArgs.vbs script file.
4. In the message boxes, click **OK**.
5. Select **Text1.txt**, **Text2.txt**, and **Text3.txt.**
6. Drag **Text1.txt**, **Text2.txt**, and **Text3.txt** onto the VBArgs.vbs script file.
7. In the message boxes, click **OK**.

   Note that four message boxes appear: one for the argument count, and one for each file.
8. Double-click the **VBArgs.vbs** script to run it directly without any arguments. In the message box, click **OK**.

Exercise 2: Writing an Event to the Application Event Log

▶ Task 1: To write events to the Application Event Log

1. Switch to PrimalScript.
2. Add the following lines of code to the end of the VBArgs.vbs script to write entries to the Application Event Log. Make sure that you use auto-complete and the Snippets Browser to insert a **For…Next** loop (labeled "ForTo" in the Snippets Browser). The last seven lines of code should resemble the following code.

```
Dim oShell, sLogEntry
Set oShell = CreateObject("WScript.Shell")
sLogEntry = "Script was run with: "
For iCount = 0 to UBound(aArgs)
       sLogEntry = sLogEntry & vbCrLf & aArgs(iCount)
Next
oShell.LogEvent 4, sLogEntry
```

3. On the **File** menu, click **Save As**.
4. In the **Save As** dialog box, navigate to the E:\Labfiles\Starter folder.
5. In the **File name** box, type **VBArgsEvents.vbs**
6. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.
7. Switch to Windows Explorer.
8. Select **Text1.txt**, **Text2.txt**, and **Text3.txt**.
9. Drag **Text1.txt**, **Text2.txt**, and **Text3.txt** onto the **VBArgsEvents.vbs** script file.
10. Click **OK** for each of the message boxes.

▶ Task 2: To review the events in the Application Event Log

1. Click **Start**, point to **Control Panel**, and then click **Classic View**.
2. In **Control Panel**, double-click **Administrative Tools**.
3. In **Administrative Tools**, double-click **Event Viewer**.
4. In the scope pane, expand **Windows Logs**, and then select **Application**.
5. In the Application Log, review the entry at the top of the list.

Tip: The Source column will specify WSH for the entry generated by your script.

6. Close Event Viewer.
7. Close PrimalScript.

Lab B: File and E-mail Scripts

Exercise 1: Documenting Your Computer Drives and Folders

▶ Task 1: To create a script that documents the drives on your computer

1. On the 2433B-VISTA-CL1-07 virtual machine, click **Start**, point to **All Programs**, point to **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom**.
2. On the **File** menu, point to **New**, and then click **File**.
3. In the **New File** dialog box, select **Script Files** in the **Categories** list, click **VBScript** in the **Templates** list, and then click **OK**.
4. Edit the template to declare the variables for this script, as the following example shows.

```
Dim oFS, oDrive, oFileText, sOutPut
```

5. On the **File** menu, click **Insert**.
6. In the **Insert File** dialog box, navigate to the E:\Labfiles\Starter folder.
7. In the **Insert File** dialog box, click **CreateTextFile.snippet**, and then click **Open**.

> **Note:** The CreateTextFile.snippet is simply a text file that contains standard code. It is good practice to create and save these snippets for reuse in your scripts. If you save code snippets in the PrimalScript snippets folder, you can select them by using the Snippets Browser.

8. Edit the inserted snippet to resemble the following code.

```
Set oFS = CreateObject("Scripting.FileSystemObject")
'  Replace "Drive:\Path\Filename.extension" with your own data
Set oFileText = oFS.CreateTextFile("E:\Labfiles\Starter\DriveData.txt")
```

9. The code now needs to loop through the drives collection. Use the Snippets Browser to insert a **For…Each** loop, and edit the inserted loop (using auto-complete) to resemble the following code.

```
For Each oDrive In oFS.Drives
        <statements>
Next
```

10. Use auto-complete to replace the <statements> line with the following code.

```
For Each oDrive In oFS.Drives
        With oDrive
                sOutPut = "Drive: " & .DriveLetter
                sOutPut = sOutPut & " Type: " & GetDriveString(.DriveType)
          End With
Next
```

> **Note:** The above code uses a call to a function named **GetDriveString**. You will import this procedure as a code snippet later in this exercise.

11. Create a new line after the **GetDriveString** statement.

12. Use the Snippets Browser to insert an **If…Else** statement, and edit the inserted statement (using auto-complete) to add more information to the sOutPut string so that it resembles the following code.

```
If .IsReady Then
        sOutPut = sOutPut & " Format: " & .FileSystem
        sOutput = sOutput & " Label: " & .VolumeName
Else
        oFileText.WriteLine sOutPut & " not ready"
End If
```

13. Create a new line after the **.VolumeName** statement.
14. Use the Snippets Browser to insert an **If…Then** statement, and edit the inserted statement (using auto-complete) to check that the drive type is not "network" (Type 3) and to add more information to the sOutPut string so that it resembles the following code.

```
If .DriveType <> 3 Then
       oFileText.WriteLine sOutPut
      WriteFolder oDrive.RootFolder, " "
End If
```

> **Note:** The above code uses a call to a subprocedure named **WriteFolder**. You will import this procedure as a code snippet later in this exercise.

15. Create a new line after the final **Next** statement, and add the following code to write a status message and close the **FileSystemObject**.

```
WScript.Echo "All Done!!"
oFileText.Close
WScript.Quit
```

16. Create a new line after the final **WScript.Quit** statement.
17. On the **File** menu, click **Insert**.
18. In the **Insert File** dialog box, navigate to the E:\Labfiles\Starter folder.
19. In the **Insert File** dialog box, click **GetDriveStringFunction.snippet**, and then click **Open**.
20. Create a new line after the inserted function.
21. On the **File** menu, click **Insert**.
22. In the **Insert File** dialog box, navigate to the E:\Labfiles\Starter folder.
23. In the **Insert File** dialog box, click **WriteFolderSub.snippet**, and then click **Open**.
24. Your final script should look like the following example.

```
Dim oFS, oDrive, oFileText, sOutPut
Set oFS = CreateObject("Scripting.FileSystemObject")
' Replace "Drive:\Path\Filename.extension" with your own data
Set oFileText = oFS.CreateTextFile("E:\Labfiles\Starter\DriveData.txt")
For Each oDrive In oFS.Drives
      With oDrive
             sOutPut = "Drive: " & .DriveLetter
             sOutPut = sOutPut & " Type: " & GetDriveString(.DriveType)
             If .IsReady Then
```

```
                    sOutPut = sOutPut & " Format: " & .FileSystem
                    sOutput = sOutput & " Label: " & .VolumeName
                    If .DriveType <> 3 Then
                          oFileText.WriteLine sOutPut
                          WriteFolder oDrive.RootFolder, " "
            End If
      Else
               oFileText.WriteLine sOutPut & " not ready"
      End If
End With
Next
WScript.Echo "All Done!!"
oFileText.Close
WScript.Quit

Function GetDriveString(iDriveType)
Const CDRom = 4
Const Fixed = 2
Const RamDisk = 5
Const Remote = 3
Const Removable = 1
Const Unknown = 0
Const CDRomString = "CDROM"
Const FixedString = "Fixed"
Const RamDiskString = "RamDisk"
Const RemoteString = "Remote"
Const RemovableString = "Removable"
Const UnknownString = "Unknown"
Select Case iDriveType
Case CDROM
      GetDriveString = CDRomString
Case Fixed
      GetDriveString = FixedString
Case RamDisk
      GetDriveString = RamDiskString
Case Remote
      GetDriveString = RemoteString
Case Removable
      GetDriveString = RemovableString
Case Else
      GetDriveString = UnknownString
End Select
End Function
Sub WriteFolder(ByRef oFol, ByVal sSpaces)
Dim oFolder
On Error Resume Next
For Each oFolder in oFol.SubFolders
      sOutPut = sSpaces & "-" & oFolder.Name
      If Err.Number = 0 Then oFileTS.WriteLine sOutput
      Err.Clear
      WriteFolder oFolder, sSpaces & " "
Next
End Sub
```

25. On the **File** menu, click **Save As**.
26. In the **Save As** dialog box, navigate to the E:\Labfiles\Starter folder.
27. In the **File name** box, type **DriveReport.vbs**
28. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.

▶ Task 2: To run the DriveReport script

1. On the **Script** menu, click **Run Script**.
2. The script execution is complete when the **All Done!!** message appears in the PrimalScript Output window.

> **Note:** The script may take a few seconds to run because it retrieves information about all of the folders on all of your drives.

3. Click **Start**, and then click **Computer**.
4. In the Computer folder, double-click **AllFiles (E:)**, double-click **Labfiles**, and then double-click **Starter**.
5. In the Starter folder, double-click **DriveData.txt**.
6. Review the file contents, and then close the DriveData.txt file.

Exercise 2: Sending a Message and Attachment

▶ Task 1: To create the SendReport script

1. Switch to PrimalScript.
2. On the **File** menu, point to **New**, and then click **File**.
3. In the **New File** dialog box, select **Script Files** in the **Categories** list, click **VBScript** in the **Templates** list, and then click **OK**.
4. Edit the template to declare the variables for this script, as the following example shows.

```
Dim oMessage
```

5. Use auto-complete to create the CDO message object, as the following example shows.

```
Set oMessage = CreateObject("CDO.Message")
```

6. Use auto-complete to create the message details, as the following example shows.

```
oMessage.Subject = "Drive Report"
oMessage.Sender = "adrianlannin@fourthcoffee.com"
oMessage.To = "administrator@fourthcoffee.com"
oMessage.TextBody = "The drive report is attached"
```

7. Use auto-complete to create the e-mail attachment, as the following example shows.
oMessage.AddAttachment "E:\Labfiles\Starter\DriveData.txt"

> **Note:** The file attachment is NOT set by using the equal sign (=).

8. Add the following line to send the message.

```
oMessage.Send
```

9. Finally, add a status message, as the following example shows.

```
WScript.Echo "Message Sent From : " & oMessage.Sender
WScript.Echo "To : " & oMessage.To
```

10. On the **File** menu, click **Save As**.
11. In the **Save As** dialog box, navigate to the E:\Labfiles\Starter folder.
12. In the **File name** box, type **SendReport.vbs**
13. In the **Save as** box, click **VBScript files (*.vbs)**, and then click **Save**.

▶ Task 2: To run the DriveReport script

1. On the **Script** menu, click **Run Script**.
2. The script execution is complete when the status message appears in the PrimalScript Output window.
3. Switch to the 2433B-LON-DC1 virtual machine.
4. Click **Start**, point to **All Programs**, and then click **Outlook Express**.
5. In Outlook Express, on the toolbar, click **Send/Recv**.
6. Verify that you have received an e-mail titled "Drive Report."
7. Open the e-mail.
8. Open the attachment, and verify that the attachment contains the drive report.

Lab: Question and Answers

Lab A: Administrative Scripts

Exercise 1: Passing Arguments to Scripts

Question: Why would you use the **On Error Resume Next** mechanism in a script?

Answer: By default, when a script encounters an error, the script stops execution at the line of code that contains the problem. By using **On Error Resume Next**, script execution continues after any error, and a specific error-handling routine can then be used to report the exact cause of the error.

Question: If an error occurs during a scheduled task, where does the scheduling service record the error?

Answer: In Windows Vista, Task Scheduler errors are recorded in the Task Scheduler Operational log, and can be viewed using Event Viewer. In Windows Server 2003, Windows XP, and Windows 2000, Task Scheduler errors are appended to %systemroot%\tasks\SchedLgU.txt.

Exercise 2: Writing an Event to the Application Event Log

Question: To which event log can you write by using the **LogEvent** method of the **Shell** object?

Answer: The Application Log.

Question: What is WevtUtil?

Answer: WevtUtil is a Windows Vista command-line tool used to read information from system event logs

Lab B: File and E-mail Scripts

Exercise 1: Documenting Your Computer Drives and Folders

Question: Which object model exposes **FileSystemObject**?

Answer: The Scripting Object Model.

Question: Why must you be careful when using scripts to manipulate the file system?

Answer: Because it is easy to delete every file and folder from a hard drive, and depending on the method used in your script, deleted files may not be placed in the Recycle Bin.

Exercise 2: Sending a Message and Attachment

Question: If no configuration information is set in a script when sending an e-mail message by means of CDO, where does the script get this information?

Answer: Either from the default MAPI profile or from the mail server settings.

Question: What function in Visual Basic, Scripting Edition can you use to help manage **REG_MULTI_SZ** values in the registry?

Answer: Join.

Module 8

WMI

Contents:

Lab Answer Keys

Lab: Writing WMI Scripts

Exercise 1: Using the PrimalScript WMI Wizard

▶ Task 1: To create a WMI script by using the PrimalScript WMI Wizard

1. Click **Start**, point to **All Programs**, click **SAPIEN Technologies, Inc**, and then click **PrimalScript 4.1 Classroom.**
2. In PrimalScript Enterprise, on the **Script** menu, point to **Wizards**, and then click **0 WMI Wizard**.
3. If a **Server Busy** dialog box appears, click **Retry**.
4. In the **WMI Wizard** dialog box, click **VBScript**.
5. In the **Classes** list, click **Win32_Desktop**, click **Copy**, and then click **Close**.
6. In PrimalScript Enterprise, on the **File** menu, point to **New**, and then click **File**.
7. In the **New File** dialog box, in the **Categories** box, click **Script Files**.
8. In the **Templates** box, click **VBScript**, and then click **OK**.
9. In PrimalScript Enterprise, press CTRL+V.
10. On the **File** menu, click **Save**.
11. In the **Save As** dialog box, browse to **E:\LabFiles\Starter**.
12. In the **File name** box, type **Desktop** and then click **Save**.
13. In the **Save As** dialog box, click **Yes**.
14. In PrimalScript Enterprise, on the **Script** menu, click **Run Script**.
15. After the script finishes, review the results in the output pane.
16. Close the output pane.

Exercise 2: Managing Remote Resources by Using WMI

▶ Task 1: To connect to the WMI service on a remote computer

1. On the **File** menu, click **Open**.
2. In the **Open** dialog box, browse to **E:\LabFiles\Starter**, click **WMIComputerManagement.vbs**, and then click **Open**.
3. Locate the comment **TODO 1) bind to WMI service. 2) bind to WMI on the Domain Controller**.
4. Uncomment the first line of code immediately after the comment in Step 3, and then change it to the following code.

```
strComputer = "LON-DC1"
```

5. Uncomment the following code.

```
Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\CIMV2")
```

▶ Task 2: To gather memory information from the remote computer by using WMI

1. Locate the comment TODO query Win32ComputerSystem WMI object for total physical memory.
2. Uncomment the following code.

```
Set colCSItems = objWMIService.ExecQuery("SELECT * FROM Win32_ComputerSystem")
For Each objCSItem In colCSItems
  WScript.Echo "Total Physical Memory: " & objCSItem.TotalPhysicalMemory
Next
```

3. Locate the comment TODO query the Win32_OperatingSystem WMI object for other memory values.
4. Uncomment the following code.

```
Set colOSItems = objWMIService.ExecQuery("SELECT * FROM Win32_OperatingSystem")
For Each objOSItem In colOSItems
  WScript.Echo "Free Physical Memory: " & objOSItem.FreePhysicalMemory
  WScript.Echo "Total Virtual Memory: " & objOSItem.TotalVirtualMemorySize
  WScript.Echo "Free Virtual Memory: " & objOSItem.FreeVirtualMemory
  WScript.Echo "Total Visible Memory Size: " & objOSItem.TotalVisibleMemorySize
Next
```

5. On the **File** menu, click **Save**.
6. On the **Script** menu, click **Run Script**.
7. After the script completes, review the results in the output pane.
8. Close the output pane.

Exercise 3: Limiting Event Log Data Returned to a Script

▶ Task 1: To establish the date range

1. On the **File** menu, click **Open**, browse to **E:\LabFiles\Starter**, click **WMIDateLimited.vbs**, and then click **Open**.
2. Locate the comment **TODO Create two WMI date time objects**.
3. Uncomment the following code.

```
Set dtmStartDate = CreateObject("WbemScripting.SWbemDateTime")
Set dtmEndDate = CreateObject("WbemScripting.SWbemDateTime")
```

4. Locate the comment **TODO Create date to check constant and the dates to be checked**.
5. Uncomment the following code.

```
DateToCheck = Date - 6
dtmEndDate.SetVarDate Date, True
dtmStartDate.SetVarDate DateToCheck, True
```

▶ Task 2: To connect to the WMI service on the remote computer

1. Locate the comment **TODO 1) connect to the WMI service on the computer. 2) change the computer to LON-DC1.**
2. Uncomment the first line of code immediately after the comment in Step 1, and then change it to the following code.

```
strComputer = "LON-DC1"
```

3. Uncomment the following code.

```
Set objWMIService = GetObject("winmgmts:" _
     & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
```

▶ Task 3: To query the Application event log and return property values

1. Locate the comment **TODO Query Application event log file for entries between start date and end date**.
2. Uncomment the following code.

```
Set colEvents = objWMIService.ExecQuery _
    ("Select * from Win32_NTLogEvent Where Logfile “_
        & “= 'Application' And TimeWritten >= '" _
        & dtmStartDate & "' and TimeWritten < '"_
        & dtmEndDate & "'")
```

3. Locate the comment **TODO Display the property values of each event log entry**.
4. Uncomment the following code.

```
For each objEvent in colEvents
    Wscript.Echo "Category: " & objEvent.Category
    Wscript.Echo "Computer Name: " & objEvent.ComputerName
    Wscript.Echo "Event Code: " & objEvent.EventCode
    Wscript.Echo "Message: " & objEvent.Message
    Wscript.Echo "Record Number: " & objEvent.RecordNumber
    Wscript.Echo "Source Name: " & objEvent.SourceName
    Wscript.Echo "Time Written: " & objEvent.TimeWritten
    Wscript.Echo "Event Type: " & objEvent.Type
    Wscript.Echo "User: " & objEvent.User
Next
```

5. On the **File** menu, click **Save**.
6. On the **Script** menu, click **Run Script**.
7. After the script completes, review the results in the output pane.
8. Close PrimalScript Enterprise.

Lab: Question and Answers

Lab: Writing WMI Scripts

Exercise 1: Using the PrimalScript WMI Wizard

Question: List five WMI providers that ship with Windows XP, Windows Vista, or Windows Server 2003.

Answer: Simple Network Management Protocol (SNMP), Active Directory, Event Viewer logs, PerfMon, and Win32.

Question: What date format do WMI objects use?

Answer: Coordinated Universal Time (UTC).

Exercise 2: Managing Remote Resources by Using WMI

Question: What WMI component determines whether the request involves static data stored in the Common Information Model (CIM) repository or dynamic data supplied by a provider?

Answer: Common Information Model Object Manager (CIMOM).

Exercise 3: Limiting Event Log Data Returned to a Script

Question: What two classes can you use to access event log information?

Answer: Win32_NTLogEvent and Win32_NTEventlogFile.

Send Us Your Feedback

You can search the Microsoft Knowledge Base for known issues at Microsoft Help and Support before submitting feedback. Search using either the course number and revision, or the course title.

Note Not all training products will have a Knowledge Base article – if that is the case, please ask your instructor whether or not there are existing error log entries.

Courseware Feedback

Send all courseware feedback to support@mscourseware.com. We truly appreciate your time and effort. We review every e-mail received and forward the information on to the appropriate team. Unfortunately, because of volume, we are unable to provide a response but we may use your feedback to improve your future experience with Microsoft Learning products.

Reporting Errors

When providing feedback, include the training product name and number in the subject line of your e-mail. When you provide comments or report bugs, please include the following:

- Document
- Page number or location
- Complete description of the error or suggested change

Please provide any details that are necessary to help us verify the issue.

Important All errors and suggestions are evaluated, but only those that are validated are added to the product Knowledge Base article.