

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### ABSTRACT

The following information is presented in this programmer's reference:

- About Microsoft® Windows® operating system Scripting Host
- Using Windows Scripting Host
- Windows Scripting Host Object Reference

### CONTENTS

- INTRODUCTION
- ACTIVEX SCRIPTING HOSTS
- OVERVIEW OF THE WINDOWS SCRIPTING HOST OBJECT MODEL
- USING WINDOWS SCRIPTING HOST
  - Writing a Scripting Engine for Windows Scripting Host
  - How Windows Scripting Host Uses the Registry
  - Registering a Scripting Engine for Windows Scripting Host
  - Running Scripts Using Wscript.Exe
  - Creating .WSH Files to Record Script Options
  - Using .WSH Files to Run Scripts
  - Running Scripts Using Cscript.exe
  - A Cscript Example
  - Document Conventions
- WINDOWS SCRIPTING HOST OBJECT REFERENCE
  - Objects provided by Wscript.exe
  - Objects provided by WSHom.Ocx
  - Wscript Object
  - Wscript.Application

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

- Wscript.Arguments
- Wscript.FullName
- Wscript.Name
- Wscript.Path
- Wscript.ScriptFullName
- Wscript.ScriptName
- Wscript.Version
- Wscript.CreateObject
- Wscript.DisconnectObject
- Wscript.Echo
- Wscript.GetObject
- Wscript.Quit
- WshArguments Object
- WshArguments.Item
- WshArguments.Count
- WshArguments.length
- WshShell Object
- WshShell.Environment
- WshShell.SpecialFolders
- WshShell.CreateShortcut
- WshShell.ExpandEnvironmentStrings
- WshShell.Popup
- WshShell.RegDelete
- WshShell.RegRead

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

- WshShell.RegWrite
- WshShell.Run
- WshNetwork Object
- WshNetwork.ComputerName
- WshNetwork.UserDomain
- WshNetwork.UserName
- WshNetwork.AddPrinterConnection
- WshNetwork.EnumNetworkDrives
- WshNetwork.EnumPrinterConnections
- WshNetwork.MapNetworkDrive
- WshNetwork.RemoveNetworkDrive
- WshNetwork.RemovePrinterConnection
- WshNetwork.SetDefaultPrinter
- WshCollection Object
- WshCollection.Item
- WshCollection.Count
- WshCollection.length
- WshEnvironment Object
- WshEnvironment.Item
- WshEnvironment.Count
- WshEnvironment.length
- WshEnvironment.Remove
- WshShortcut Object
- WshShortcut.Arguments

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

- WshShortcut.Description
- WshShortcut.FullName
- WshShortcut.Hotkey
- WshShortcut.IconLocation
- WshShortcut.TargetPath
- WshShortcut.WindowStyle
- WshShortcut.WorkingDirectory
- WshShortcut.Save
- WshSpecialFolders Object
- WshSpecialFolders.Item
- WshSpecialFolders.Count
- WshSpecialFolders.length
- WshUrlShortcut Object
- WshUrlShortcut.FullName
- WshUrlShortcut.TargetPath
- WshUrlShortcut.Save

- FOR MORE INFORMATION

### INTRODUCTION

The Microsoft® Windows® operating system Scripting Host is a language-independent scripting host for ActiveX™ scripting engines on 32-bit Windows platforms. Windows Scripting Host will be integrated into Windows 98, Windows NT® Workstation operating system version 5.0, and Windows NT Server version 5.0.

Both Microsoft Visual Basic® development system Scripting Edition (VBScript) and Microsoft JScript™ development system scripting engines are provided with Windows Scripting Host. Other software companies will provide ActiveX scripting engines for languages such as Perl, TCL, REXX, Python, and others. Windows Scripting Host can be run from either the Windows-based host (Wscript.exe), or the command-shell-based host (Cscript.exe).

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### ACTIVEX SCRIPTING HOSTS

ActiveX scripting architecture supports powerful scripting, using languages such as Visual Basic Scripting Edition, JScript, Perl, and others. Microsoft currently provides three hosts for running these scripting languages across the Windows platform:

- Microsoft Internet Explorer. Internet Explorer can execute scripts on client computers from within HTML pages.
- Internet Information Server (IIS). IIS supports Active Server Pages, so scripts can run on Web servers. In other words, IIS supports server-side scripting over the Internet or an intranet.
- Windows Scripting Host. Windows Scripting Host provides a low-memory scripting host so scripts can execute directly on the Windows desktop or Command Prompt window; the scripts do not need to be embedded in an HTML document. Windows Scripting Host is an ideal host for noninteractive scripts that perform logon and administrative tasks.

In addition to having a smaller footprint than the other two scripting hosts, the Windows Scripting Host does not rely on an HTML SCRIPT tag or LANGUAGE attribute to identify a script engine. Instead, it uses the extension of the script file to determine what script engine to use. As a result, the scriptwriter need not obtain a script engine ProgID. The scripting host itself maintains a mapping of script extensions to ProgIDs and uses the Windows association model to launch the appropriate engine for a given script.

For more information about how Windows Scripting Host works, see:  
<http://www.microsoft.com/management/wsh.htm>.

### OVERVIEW OF THE WINDOWS SCRIPTING HOST OBJECT MODEL

The Windows Scripting Host object model provides two main categories of ActiveX interfaces:

1. Script execution and troubleshooting: Properties and methods that are directly related to script execution. This set of interfaces allows scripts to manipulate Windows Scripting Host, display messages on the screen, and perform basic functions such as CreateObject and GetObject.
2. Helper functions: Properties and methods that map network drives, connect to printers, retrieve and modify environment variables, and manipulate registry keys. These functions are provided so administrators can use Windows Scripting Host to create simple logon scripts.

In addition to the object interfaces provided by Windows Scripting Host, administrators can use any ActiveX controls that expose Automation interfaces to perform various tasks on the Windows platform. For example, administrators can write scripts that use the Active Directory to manage the Windows NT Directory Service.

### USING WINDOWS SCRIPTING HOST

The following section describes how you can use Windows Scripting Host.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### Writing a Scripting Engine for Windows Scripting Host

The principal requirement for a third-party scripting engine is that it support the ActiveX IActiveScriptParse COM interface.

The Windows Scripting Host reads and passes the contents of a specified script file to a registered script engine using the script engine's IActiveScriptParse::ParseScriptText method.

A scripting engine can also call Wshom.ocx, an ActiveX control that supports the IDispatch COM interface. If the scripting engine supports ActiveX control creation in a script, Wshom.ocx can be called from within the script.

If the scripting engine itself calls Wshom.ocx, you should obtain lwshom.idl, lwshom.h and lwshom.iid from the Platform SDK. With these files, you can call methods in Wshom.ocx using the normal COM calling sequence. Wshom.ocx supports a dual interface, aggregation, and the IErrorInfo COM interface.

### Example

```
#include <IWSHom.h>
HRESULT
Use_WSHShell() {
    HRESULT hr;
    IWSHShell* pShell;

    hr = ::CoCreateInstance(CLSID_IWSHShell,
        NULL,
        CTX_INPROC_SERVER,
        IID_IWSHShell,
        (LPVOID*) &pShell );
    if (FAILED(hr)) return hr;
    hr = pShell-Run(bstrCommand,
        &varWindowStyle,
        &varWaitOnReturn,
        &nExitCode );
    pShell-Release();
    return hr;
}
```

### How Windows Scripting Host Uses the Registry

An ActiveX scripting engine is registered for use with Windows Scripting Host by adding registry entries under the HKEY\_CURRENT\_USER (HKCU) key. Windows Scripting Host determines which scripting engine to use to execute a script by examining the ScriptEngine key. For example, if a user types the following command in a Command Prompt window:

#### **cscript chart.vbs**

To look up the scripting engine for the Chart.vbs file, Windows Scripting Host:

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

- Searches HKCU\.vbs to obtain a script identifier, such as VBSfile.
- Searches HKCU\VBSFile\ScriptEngine to get a scripting engine identifier, such as VBScript.
- Finds the CLSID from the scripting engine identifier; in this example, the CLSID is located in HKCU\VBScript\CLSID.
- Calls CoCreateInstance with the CLSID and IUnknown.
- Calls QueryInterface to get the IActiveScriptParse COM interface.

### Registering a Scripting Engine for Windows Scripting Host

If you have written an ActiveX scripting engine with the following characteristics:

- Script engine identifier is FooScript
- File extension for script is .FOO
- Script identifier is FooFile

The following table shows the registry entries that are necessary to register this engine (none of these has a value name).

Key	Value Name	Type	Value
.FOO	none	REG_SZ	FooFile
FooFile	none	REG_SZ	FooScript Script File
FooFile\ScriptEngine	none	REG_SZ	FooFile
FooFile\Shell\Open	none	REG_SZ	&Open
FooFile\Shell\Open\Command	none	REG_EXPAND_SZ	%systemroot%\system32\wscript "%1" %*
FooFile\Shell\Open2	none	REG_EXPAND_SZ	Open &with command console
FooFile\Shell\Open2\Command	none	REG_SZ	%systemroot%\system32\wscript "%1" %*
FooFile\ShellEx\PropertySheetHandlers\WSHProps	none	REG_SZ	{60254CA5-953B-11CF-8C96-00AA00B8708C}

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

For Windows 98, use REG\_SZ instead of REG\_EXPAND\_SZ and change the path of Wscript.exe and Cscript.exe as shown in the following table.

Windows NT		Windows 98
Wscript	%SystemRoot%\System32\Wscript.exe	C:\Windows\Wscript.exe
Cscript	%SystemRoot%\System32\Cscript.exe	C:\Windows\Command\Cscript.exe

### Running Scripts Using Wscript.Exe

Using Wscript.exe, you can run scripts under Windows in three ways:

- Double-clicking files or icons. These can be files or icons listed in My Computer, Windows Explorer, the Find window, the Start menu, or on the Desktop.
- Entering a script name at the Run command on the Start Menu. Press the Start button, select Run, and enter the full name of the script you want to run, including file extension and any necessary path information.
- Entering Wscript.exe followed by a script name at the Run command. Press the Start button, select Run, and enter Wscript, followed by the full name and path of the script you want to run.

If you double-click a script file whose extension has not yet been associated with Wscript.exe, an Open With dialog box appears asking you what program you would like to use to open the file you double-clicked. After choosing Wscript, if you check the Always use this program to open this file check box, Wscript is registered as the default application for all files having the same extension as the one you double-clicked. For example, if you check this check box when you run Chart.vbs, Wscript.exe becomes the default application for all files having the .vbs extension.

Wscript.exe has a properties page associated with it that provides the following options:

Property	Function	CSCRIPT equivalent
Stop script after nn seconds.	Specifies a maximum number of seconds that a script can run (the default is no limit).	//T:nn
Display logo when scripts executed in a command console.	Display a banner before running the script (this is the default the opposite is //nologo).	//logo or //nologo

By using the Wscript.exe Properties page, you can set global scripting options for all scripts that

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Wscript runs on the local machine. It is also possible to set options for individual scripts, using a .wsh file, as described in the next section.

### Creating .WSH Files to Record Script Options

You can record specific settings for each individual script you run by creating a control file for that script with a .wsh file extension.

A .wsh file is a text file that uses a format similar to that of .ini files. It is created automatically when you set the properties for a supported script file.

To create a .wsh file for a given script

1. Right-click the given script file in Windows Explorer.
2. Select Properties on the context menu that appears.
3. Choose the settings you want for the script on the property page.
4. Choose OK or Apply.

A .wsh file is automatically created using the same name as the script file that you right-clicked. A sample .wsh file might contain the following text:

```
[ScriptFile]  
Path=C:\WINNT\Samples\WSH\showprop.vbs
```

```
[Options]  
Timeout=0  
DisplayLogo=1  
BatchMode=0
```

The Path setting in the [ScriptFile] section identifies the script file that this .wsh file is associated with. The keys in the [Options] section correspond to settings in the Script tab within the Properties dialog box.

### Using .WSH Files to Run Scripts

A .wsh file is analogous to the .pif files used to run older 16-bit applications. It can be treated as if it were an executable or batch file.

For example, suppose you have a script named Myscript.vbs for which you have created a .wsh file named Myscript.wsh. To run Myscript.vbs with the options recorded in Myscript.wsh, you can either double-click Myscript.wsh in Windows Explorer, or pass Myscript.wsh as a script argument to CScript.exe or WScript.exe. For example, you could enter the following command at the command prompt:

```
cscript Myscript.wsh
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Administrators in particular will find .wsh files useful. By creating multiple .wsh files for a given script, an administrator can tailor the way that script runs to the needs of specific groups or even individuals within an organization.

The same logon script, for example, might serve two very different groups if invoked by two different .wsh files containing different settings and parameters.

### Running Scripts Using Cscript.exe

To use Cscript.exe, open a Command Prompt window and type a Cscript command line. CScript uses the following syntax:

```
cscript [script name] [host options...] [script options]
```

- Script name is the name of the script file, complete with extension and any necessary path information, such as: d:\admin\vbscripts\chart.vbs.
- Host options enable or disable various Windows Scripting Host features. Host options are always preceded by two slashes (//).
- Script options are passed to the script. Script parameters are always preceded by only one slash (/).

Each parameter is optional; however, you cannot specify script options without specifying a script name. If you do not specify parameters, Cscript displays the Cscript syntax and the valid host parameters.

Cscript.exe supports the host parameters shown in the following table.\

Parameter	Description
//I	Interactive Mode: allows display of user prompts and script errors (this is the default, and the opposite of //B).
//B	Batch Mode: suppresses command-line display of user prompts and script errors.
//T:nn	Enables time-out: the maximum number of seconds the script can run. The default is no limit. (See the text following this table for more information on this parameter.)
//logo	Displays a banner (default; opposite of //nologo).
//nologo	Prevents display of an execution banner at run time.
//H:Cscript or //H:Wscript	Registers Cscript.exe or Wscript.exe as the default application for running scripts. If neither is specified, Wscript.exe is assumed as the

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

	default.
//S	Saves the current command-line options for this user.
//?	Shows command usage (same as with parameters).

The //T parameter prevents excessive execution of scripts; it does this by setting a timer. When execution time exceeds the specified value, Cscript interrupts the scripting engine using the IActiveScript::InterruptThread method and terminates the process.

### A Cscript Example

Several sample scripts are installed when you install the final release of the Windows Scripting Host. These are also available for download at: <http://www.microsoft.com/management/wsh.htm>

Suppose, for the purposes of this example, that you have copied the Chart.vbs sample script to the following folder on your computer:

c:\sample scripts\chart.vbs

You can run the script using with and without a logo, as follows:

1. Choose the MS-DOS® operating system Command Prompt from Programs on the Start menu.
2. Enter the following commands at the command prompt (if your sample scripts are located in a different folder, these lines should be modified accordingly):

```
cscript //logo c:\sample scripts\chart.vbs  
cscript //nologo c:\sample scripts\chart.vbs
```

When Windows Scripting Host ships with Windows NT 5.0, you will no longer need to provide the script file's extension. Instead, you can type the script name alone, or double-click the script in Windows Explorer.

### Document Conventions

Variable prefixes in parameter listings and code examples have the following meanings.

Prefix	Description
any	Any type
b	Boolean
int	Integer

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

nat	Natural number or nonnegative integer
obj	Object (Idispatch interface)
str	String

Scripting examples are written in the Microsoft Visual Basic Scripting Edition (VBScript).

Methods written in Microsoft Jscript are case-sensitive. In general, you should use the same case as the syntax examples shown in this document.

Parameters in brackets (for example, [anyValue]) are optional.

### WINDOWS SCRIPTING HOST OBJECT REFERENCE

#### Objects provided by Wscript.exe

Wscript	Exposed to scripting engine as "Wscript."
WshArguments	Not exposed; accessed through the Wscript.Arguments property.

#### Objects provided by WSHom.Ocx

WshShell	Automation object, ProgID "Wscript.WshShell."
WshNetwork	Automation object, ProgID "Wscript.WshNetwork."
WshShortcut	Not exposed; accessed through WshShell.CreateShortcut method.
WshUrlShortcut	Not exposed; accessed through WshShell.CreateShortcut method.
WshCollection	Not exposed; accessed through WshNetwork.EnumNetworkDrives or WshNetwork.EnumPrinterConnection.
WshEnvironment	Not exposed; accessed through WshShell.Environment property.
WshSpecialFolders	Not exposed; accessed through WshShell.Folder property.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### Wscript Object

ProgID	N/A
Filename	Wscript.exe or Cscript.exe
CLSID	60254CA2-953b-11CF-8C96-00AA00B8708C
IID	60254CA1-953b-11CF-8C96-00AA00B8708C

The following table describes the properties associated with the Wscript object.

Property	Description
Application	The IDispatch interface for Wscript.
Arguments	A parameters collection object.
FullName	Full path to the host executable.
Name	Friendly name of Wscript (the default property).
Path	Name of the directory where Wscript.exe or Cscript.exe resides.
ScriptFullName	Full path to the script being run by the Windows Scripting Host.
ScriptName	File name of the script being run by the Windows Scripting Host.
Version	A version string for the Windows Scripting Host.

The following table describes the methods associated with the Wscript object.

Method	Description
CreateObject	Creates an object and establishes its event handling.
DisconnectObject	Disconnects a previously connected object from Windows Scripting Host.
Echo	Displays parameters in a window or at a command prompt in a Command Prompt window.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

GetObject	Retrieves an Automation object from a file.
Quit	Quits execution with a specified error code.

### **Wscript.Application**

The Application property provides the IDispatch interface on the Wscript object.

#### **Syntax**

```
Wscript.Application = objWscript
```

### **Wscript.Arguments**

The Arguments property provides a WshArguments collection object.

#### **Syntax**

```
Wscript.Arguments = objArguments
```

#### **Example**

```
' Display all command-line parameters  
Set objArgs = Wscript.Arguments  
For l = 0 to objArgs.Count - 1  
Wscript.Echo objArgs(l)  
Next
```

### **Wscript.FullName**

The FullName property provides a string containing the full path to the host executable.

#### **Syntax**

```
Wscript.FullName = strFullName
```

#### **Example**

```
Wscript.FullName = C:\WinNT\System32\wscript.exe
```

#### **See Also**

Wscript.Path property.

### **Wscript.Name**

The Name property provides a string containing the friendly name of the Wscript object. This is the default property.

#### **Syntax**

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Wscript.Name = strName

### Example

Wscript.Name = Windows Scripting Host  
Wscript = Windows Scripting Host

### Wscript.Path

The Wscript.Path property provides a string containing the name of the directory where Wscript.Exe or Cscript.Exe resides.

### Syntax

Wscript.Path = strPath

### Example

Wscript.Path = C:\WinNT\System32

### See Also

Wscript.FullName property

### Wscript.ScriptFullName

The ScriptFullName property provides the full path to the script being run by the Windows Scripting Host.

### Syntax

Wscript.ScriptFullName = strScriptFullName

### Example

Wscript.ScriptFullName = C:\Users\Default\WSH\Sample.VBS

### See Also

Wscript.ScriptName property

### Wscript.ScriptName

The ScriptName property provides the file name of the script being run by the Windows Scripting Host.

### Syntax

Wscript.ScriptName = strScriptName

### Example

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Wscript.ScriptName = Sample.VBS

### See Also

Wscript.ScriptFullName property

### Wscript.Version

The Version property provides a version string for the Windows Scripting Host.

### Syntax

Wscript.Version = strVersion

### Example

Wscript.Version = 5.0

### Wscript.CreateObject

The CreateObject method creates an object specified by the strProgID parameter. If the parameter strPrefix is specified, Windows Scripting Host connects the object's outgoing interface to the script file after creating the object. When the object fires an event, Windows Scripting Host calls a subroutine named strPrefix and the event name.

For example, if strPrefix is "MYOBJ\_" and the object fires an event named "OnBegin," Windows Scripting Host calls the "MYOBJ\_OnBegin" subroutine located in the script.

### Syntax

Wscript.CreateObject(strProgID, [strPrefix]) = objObject

### See Also

Wscript.GetObject method, Wscript.DisconnectObject method

### Wscript.DisconnectObject

The DisconnectObject method disconnects a previously connected object (obj) from Windows Scripting Host. If the specified object is not already connected to Windows Scripting Host, this method does nothing.

### Syntax

Wscript.DisconnectObject obj

### Example

```
' This code fragment instantiates a fictitious object and  
' connects it to the script file. The script then calls the  
' "SomeMethod" method on the object.
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

```
' If an error called "Event" occurs in the object, then the
' MyEvent_Event subroutine is called.
Set MyObject = Wscript.CreateObject("SomeObject", "MyEvent")
MyObject.SomeMethod
Sub MyEvent_Event(strName)
Wscript.Echo strName
End Sub
' When done with the object, disconnect it from the script
' and then set it to Nothing
Wscript.DisconnectObject MyObject
Set MyObject = Nothing
```

### See Also

Wscript.CreateObject method, Wscript.GetObject method

### Wscript.Echo

The Echo method displays parameters in a window (in Wscript.exe) or at the command prompt in a Command Prompt window (in Cscript.exe).

Parameters are delimited by one space. Under Cscript.exe, this method outputs a carriage-return/line feed pair (CR LF) after the last parameter displayed.

### Syntax

```
Wscript.Echo [anyArg...]
```

### Example

```
Wscript.Echo
Wscript.Echo 1, 2, 3
Wscript.Echo "Windows Scripting Host is cool."
```

### Wscript.GetObject

The GetObject method retrieves an Automation object from a file, or an object specified by strProgID parameter. Use the GetObject method when there is a current instance of the object, or if you want to create the object from a file that is already loaded. If there is no current instance and you don't want the object started from a file that is already loaded, use the CreateObject method.

### Syntax

```
Wscript.GetObject(strPathname [,strProgID] ), [strPrefix]) = objObject
```

### Parameters strPathname

The full path and the name of the file containing the object to retrieve. The parameter strPathname is required.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

strProgID

A string representing the program identifier (ProgID) of the object.

strPrefix

If the parameter strPrefix is specified, Windows Scripting Host connects the object's outgoing interface to the script file after creating the object. When the object fires an event, Windows Scripting Host calls a subroutine named strPrefix and the event name.

For example, if strPrefix is "MYOBJ\_" and the object fires an event named "OnBegin," Windows Scripting Host calls the "MYOBJ\_OnBegin" subroutine located in the script.

### Return

objObject

The Automation object retrieved.

### Remarks

If an object has registered itself as a single-instance object (for example, the Word.Basic object in Microsoft Word 7.0), only one instance of the object is created, no matter how many times CreateObject is executed. In addition, with a single-instance object, GetObject always returns the same instance when called with the zero-length string syntax (""), and it causes an error if the path parameter is omitted. You can't use GetObject to obtain a reference to a Visual Basic class created with Visual Basic 4.0 or earlier.

GetObject works with all COM classes, independent of the language used to create the object.

### Examples

```
Dim MyObject As Object
Set MyObject = GetObject("C:\CAD\SCHEMA.CAD")
= MyCAD.Application
```

When this code is executed, the application associated with the specified strPathname is started, and the object in the specified file is activated.

If strPathname is a zero-length string (""), GetObject returns a new object instance of the specified type. If the strPathname parameter is omitted entirely, GetObject returns a currently active object of the specified type. If no object of the specified type exists, an error occurs.

Some applications allow you to activate part of a file. To do this, add an exclamation point (!) to the end of the file name and follow it with a string that identifies the part of the file you want to activate. For information on how to create this string, see the documentation for the application that created the object. For example, in a drawing application you might have multiple layers to a drawing stored in a file. You could use the following code to activate a layer within a drawing file called schema.cad:

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

```
Set LayerObject = GetObject("C:\CAD\SCHEMA.CAD!Layer3")
```

If you do not specify the object's class, COM determines the application to start and the object to activate according to the file name you provide. Some files, however, may support more than one class of object. For example, a drawing might support three different types of objects: an application object, a drawing object, and a toolbar object, all of which are part of the same file. To specify which object in a file you want to activate, use the optional class parameter. For example:

```
Dim MyObject As Object  
Set MyObject = GetObject("C:\DRAWINGS\SAMPLE.DRW", "FIGMENT.DRAWING")
```

In the above example, FIGMENT is the name of a drawing application and DRAWING is one of the object types it supports.

Once an object is activated, you reference it in code using the object variable you defined. In the above example, you access properties and methods of the new object using the object variable MyObject. For example:

```
MyObject.Line 9, 90  
MyObject.InsertText 9, 100, "Hello, world."  
MyObject.SaveAs "C:\DRAWINGS\SAMPLE.DRW"
```

### See Also

Wscript.CreateObject method, Wscript.DisconnectObject method

### Wscript.Quit

The Quit method quits execution with a specified error code.

### Syntax

```
Wscript.Quit [intErrorCode]
```

### Parameters

intErrorCode

If this parameter is included, Wscript returns it as the process exit code. Otherwise, if intErrorCode is omitted, Wscript returns zero (0) as the process exit code.

### Example

```
Wscript.Quit(1)
```

### WshArguments Object

This object is not exposed directly; access it using the Wscript.Arguments property.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

ProgID	N/A
Filename	Wscript.Exe or Cscript.Exe
CLSID	60254CA4-953b-11CF-8C96-00AA00B8708C
IID	60254CA3-953b-11CF-8C96-00AA00B8708C

The following table describes the properties associated with the WshArguments object.

Property	Description
Item	The nth command-line parameter (the default property).
Count	The number of command-line parameters.
length	The number of command-line parameters (JScript).

### **WshArguments.Item**

The Item property contains the natIndexth command-line parameter as a string. It is the default property.

### **Syntax**

```
Arguments(natIndex)  
Arguments.Item(natIndex) = strArgument
```

**Example** Set oArgs = Wscript.Arguments.  
Wscript.Echo oArgs(0)  
Wscript.Echo oArgs.Item(0)

### **See Also**

Wscript.Arguments property

**WshArguments.Count** The Count property provides the number of command-line parameters.

### **Syntax**

```
Arguments.Count = natNumberOfArguments
```

### **See Also**

Wscript.Arguments property, WshArguments.length property

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### WshArguments.length

The length property provides the number of command-line parameters. This property provides the same functionality as the Count property and is provided for Microsoft JScript compatibility.

### Syntax

Arguments.length = natNumberOfArguments

### See Also

Wscript.Arguments property, WshArguments.Count property

### WshShell Object

ProgID	Wscript.WshShell
Filename	WSHom.Ocx
CLSID	F935DC22-1CF0-11d0-ADB9-00C04FD58A0B
IID	F935DC21-1CF0-11d0-ADB9-00C04FD58A0B

The following table describes the properties associated with the WshShell object.

Property	Description
Environment	Returns the WshEnvironment collection object.
SpecialFolders	Provides access to Windows shell folders such as the desktop folder, start menu folder, and personal document folder, using the WshSpecialFolders object.

The following table describes the methods associated with the WshShell object.

Method	Description
CreateShortcut	Creates a WshShortcut object and returns it.
ExpandEnvironmentStrings	Expands a PROCESS environment variable and returns the result string.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Popup	Pops up a message box window containing a specified message.
RegDelete	Deletes, from the registry, a specified key or value.
RegRead	Returns a specified key or value from the registry.
RegWrite	Sets a specified key or value in the registry.
Run	Creates a new process that executes a specified command with a specified window style.

### WshShell.Environment

The Environment property returns the WshEnvironment object.

### Syntax

WshShell.Environment ( [strType]) = objWshEnvironment

### Remarks

If strType specifies where the environment variable resides, possible values are "System," "User," "Volatile," and "Process." If strType is not supplied, this method retrieves the system environment variables in Windows NT or the process environment variables in Windows 95. For Windows 95, only "Process" is supported in the strType parameter. The following variables are provided with the Windows operating system. Scripts can also get environment variables that were set by other applications.

Name	Description
NUMBER_OF_PROCESSORS	Number of processors running on the machine
PROCESSOR_ARCHITECTURE	Processor type of the user's workstation
PROCESSOR_IDENTIFIER	Processor ID of the user's workstation
PROCESSOR_LEVEL	Processor level of the user's workstation
PROCESSOR_REVISION	Processor version of the user's workstation
OS	Operating system on the user's workstation
COMSPEC	Executable for command Command Prompt (typically cmd.exe)

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

HOMEDRIVE	Primary local drive (typically the C drive)
HOMEPath	Default directory for users (on Windows NT this is typically \users\default)
PATH	PATH environment variable
PATHExt	Extensions for executable files (typically .com, .exe, .bat, or .cmd)
PROMPT	Command prompt (typically \$P\$G)
SYSTEMDRIVE	Local drive on which system directory resides (e.g., c:\)
SYSTEMROOT	System directory (e.g., c:\winnt). This is the same as WINDIR
WINDIR	System directory (e.g., c:\winnt). This is the same as SYSTEMROOT
TEMP	Directory for storing temporary files (e.g., c:\temp). User, Volatile
TMP	Directory for storing temporary files (e.g., c:\temp). User, Volatile

### Example

```
' Retrieve the NUMBER_OF_PROCESSORS system environment variable
Set WshShell = Wscript.CreateObject("Wscript.Shell")
Set WshSysEnv = WshShell.Environment("SYSTEM")
Wscript.Echo WshSysEnv("NUMBER_OF_PROCESSORS")
```

### See Also

WshEnvironment object

### WshShell.SpecialFolders

The SpecialFolders property provides the WshSpecialFolders object for accessing Windows shell folders such as the desktop folder, start menu folder, and personal document folder.

### Syntax

```
WshShell.SpecialFolders = objWshSpecialFolders
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### Example

```
' This code fragment shows how to access the desktop folder
Set WshShell = Wscript.CreateObject("Wscript.Shell")
MsgBox "Your desktop is " & WshShell.SpecialFolders("Desktop")
```

### See Also

WshSpecialFolders object

### WshShell.CreateShortcut

The CreateShortcut method creates a WshShortcut object and returns it. If the shortcut title ends with ".URL," a WshURLShortcut object is created.

### Syntax

```
WshShell.CreateShortcut(strPathname) = objShortcut
```

### Example

```
' This code fragment creates a shortcut
' to the currently executing script
Set WshShell = Wscript.CreateObject("Wscript.Shell")
Set oShellLink = WshShell.CreateShortcut("Current Script.lnk")
oShellLink.TargetPath = Wscript.ScriptFullName
oShellLink.Save
Set oUrlLink = WshShell.CreateShortcut("Microsoft Web Site.URL")
oUrlLink.TargetPath = "http://www.microsoft.com"
oUrlLink.Save
```

### See Also

WshShortcut object, WshUrlShortcut object

### WshShell.ExpandEnvironmentStrings

The ExpandEnvironmentStrings method expands the PROCESS environment variable in strString and returns the result string. Variables are enclosed by the "%" character.

The environment variable name is not case-sensitive.

### Syntax

```
WshShell.ExpandEnvironmentStrings(strString) = strExpandedString
```

### Example

```
MsgBox "Prompt is " & WshShell.ExpandEnvironmentStrings("%PROMPT%")
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### WshShell.Popup

The Popup method displays a popup message box window that contains the message contained in strText. The window title of this message box is specified by strTitle. If strTitle is omitted, the window title is "Windows Scripting Host."

### Syntax

```
WshShell.Popup(strText, [natSecondsToWait], [strTitle], [natType]) = intButton
```

### Remarks

If natSecondsToWait is supplied and its value is greater than zero, the message box window will be closed after natSecondsToWait seconds.

The meaning of natType is the same as in the Microsoft Win32® application programming interface MessageBox function. The following table shows the value and its meaning in natType. You can combine values in the following tables.

### Button Types

Value	Description
0	Show [OK] button
1	Show [OK] and [Cancel] buttons
2	Show [Abort], [Retry] and [Ignore] buttons
3	Show [Yes], [No] and [Cancel] buttons
4	Show [Yes] and [No] buttons
5	Show [Retry] and [Cancel] buttons

### Icon Types

Value	Description
16	Show "Stop Mark" icon
32	Show "Question Mark" icon
48	Show "Exclamation Mark" icon
64	Show "Information Mark" icon

The preceding two tables do not cover all values for natType. For a complete list, see the Win32

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

documentation.

The return value `intButton` denotes the number of the button that the user clicked. If the user does not click a button before `natSecondsToWait` seconds, `intButton` is set to -1.

Value	Description
1	[OK] button
2	[Cancel] button
3	[Abort] button
4	[Retry] button
5	[Ignore] button
6	[Yes] button
7	[No] button

### Example

```
Set WshShell = Wscript.CreateObject("Wscript.Shell") WshShell.Popup "Where do you want to go today?"
```

### See Also

Wscript.Echo method

### WshShell.RegDelete

The `RegDelete` method deletes from the registry the key or value named `strName`.

### Syntax

```
WshShell.RegDelete strName
```

### Parameters

`strName`

If `strName` ends with a backslash character (`\`), this method deletes the key instead of the value. The `strName` parameter must begin with one of following root key names:

Short	Long
-------	------

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKCR	HKEY_CLASSES_ROOT
	HKEY_USERS
	HKEY_CURRENT_CONFIG

### Example

```
Set WshShell = Wscript.CreateObject("Wscript.Shell")
WshShell.RegDelete "HKCU\ScriptEngine\Value" ' Delete value "Value"
WshShell.RegDelete "HKCU\ScriptEngine\Key\" ' Delete key "Key"
```

### See Also

WshShell.RegRead method, WshShell.RegWrite method

### WshShell.RegRead

The RegRead method returns the registry key or value named by strName.

### Syntax

```
WshShell.RegRead(strName) = strValue
```

### Parameters

strName

If strName ends with the backslash character (\), this method returns the key instead of the value. StrName must begin with one of following root key names:

Short	Long
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKCR	HKEY_CLASSES_ROOT
	HKEY_USERS
	HKEY_CURRENT_CONFIG

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### Remarks

The RegRead method supports only REG\_SZ, REG\_EXPAND\_SZ, REG\_DWORD, REG\_BINARY, and REG\_MULTI\_SZ data types. If the registry has other data types, RegRead returns DISP\_E\_TYPEMISMATCH.

### Example

```
Set WshShell = Wscript.CreateObject("Wscript.Shell")
WshShell.RegRead("HKCU\ScriptEngine\Val") ' Read from value "Val"
WshShell.RegRead("HKCU\ScriptEngine\Key") ' Read from key "Key"
```

### See Also

WshShell.RegDelete method, WshShell.RegWrite method

### WshShell.RegWrite

The RegWrite method sets the registry key or value named by strName.

### Syntax

```
WshShell.RegWrite strName, anyValue, [strType]
```

### Parameters

strName

If strName ends with a backslash character (\), this method sets the key instead of the value. StrName must begin with one of following root key names:

Short	Long
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKCR	HKEY_CLASSES_ROOT
	HKEY_USERS
	HKEY_CURRENT_CONFIG

### Remarks

RegWrite supports strType as REG\_SZ, REG\_EXPAND\_SZ, REG\_DWORD and REG\_BINARY. If another data type is passed as strType, RegWrite returns E\_INVALIDARG.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

RegWrite automatically converts anyValue to a string when strType is REG\_SZ or REG\_EXPAND\_SZ. If strType is REG\_DWORD, anyValue is converted to an integer. If strType is REG\_BINARY, anyValue must be an integer.

### Example

```
Set WshShell = Wscript.CreateObject("Wscript.Shell")
WshShell.RegWrite "HKCU\ScriptEngine\Value", "Some string value"
WshShell.RegWrite "HKCU\ScriptEngine\Key\", 1 "REG_DWORD"
```

### See Also

WshShell.RegDelete method, WshShell.RegWrite method

### WshShell.Run

The Run method creates a new process that executes strCommand with window style intWindowStyle.

### Syntax

```
WshShell.Run (strCommand, [intWindowStyle], [bWaitOnReturn])
```

### Parameters

strCommand

Environment variables within the strCommand parameter are automatically expanded.

bWaitOnReturn

If bWaitOnReturn is not specified or FALSE, this method immediately returns to script execution rather than waiting on the process termination.

If bWaitOnReturn is set to TRUE, the Run method returns any error code returned by the application.

If bWaitOnReturn is not specified or is FALSE, Run returns an error code of 0 (zero).

### Example

```
' This fragment launches Notepad with the current executed script
Set WshShell = Wscript.CreateObject("Wscript.Shell")
WshShell.Run ("notepad " & Wscript.ScriptFullName)
WshShell.Run ("%windir%\notepad" & Wscript.ScriptFullName)
```

```
' This fragment returns the error code from the executed application
Return = WshShell.Run("notepad " & Wscript.ScriptFullName, 1, TRUE)
```

### WshNetwork Object

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

ProgID	Wscript.WshNetwork
Filename	WSHom.Ocx
CLSID	F935DC26-1CF0-11d0-ADB9-00C04FD58A0B
IID	F935DC25-1CF0-11d0-ADB9-00C04FD58A0B

The following table describes the properties associated with the WshNetwork object.

Property	Description
ComputerName	A string representation of the computer name.
UserDomain	A string representation of the user's domain name.
UserName	A string representation of the user's name.

The following table describes the methods associated with the WshNetwork object.

Method	Description
AddPrinterConnection	Maps a remote printer to a local resource name.
EnumNetworkDrives	Returns the current network drive mappings.
EnumPrinterConnections	Returns the current network drive mappings.
MapNetworkDrive	Maps a share point to a local resource name.
RemoveNetworkDrive	Removes the current resource connection.
RemovePrinterConnection	Removes a current resource connection.
SetDefaultPrinter	Sets the default printer.

### **WshNetwork.ComputerName**

The ComputerName property provides a string representation of the computer name.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### Syntax

```
WshNetwork.ComputerName = strComputerName
```

### **WshNetwork.UserDomain**

The UserDomain property provides a string representation of the user domain name.

### Syntax

```
WshNetwork.UserDomain = strDomain
```

### **WshNetwork.UserName**

The UserName property provides a string representation of the user name.

### Syntax

```
WshNetwork.UserName = strName
```

### **WshNetwork.AddPrinterConnection**

The AddPrinterConnection method maps the remote printer specified by strRemoteName to the local resource name strLocalName.

### Syntax

```
WshNetwork.AddPrinterConnection strLocalName, strRemoteName, [bUpdateProfile], [strUser],  
[strPassword]
```

### Parameters

strLocalName

Local resource to map to.

strRemoteName Remote printer to map.

bUpdateProfile

If bUpdateProfile is supplied and its value is TRUE, this mapping is stored in the user profile. strUser, strPassword.

If you are mapping a remote printer using the credentials of someone other than current user, you can specify strUser and strPassword.

### Example

```
Set WshNetwork = Wscript.CreateObject("Wscript.Network")  
WshNetwork.AddPrinterConnection "LPT1", "\\Server\Print1"
```

### **WshNetwork.EnumNetworkDrives**

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

The EnumNetworkDrives method returns the current network drive mappings as a WshCollection object. Items in this collection are local names and remote names.

### Syntax

```
WshNetwork.EnumNetworkDrive = objWshCollection
```

### Example

```
Set WshNetwork = Wscript.CreateObject("Wscript.Network")
Set oDrives = WshNetwork.EnumNetworkDrives
Wscript.Echo oDrives.Item(0) = "Z:"
Wscript.Echo oDrives.Item(1) = "\\Server\Share"
```

**See Also** WshNetwork.MapNetworkDrive method, WshNetwork.RemoveNetworkDrive method, WshCollection object

### WshNetwork.EnumPrinterConnections

The EnumPrinterConnections method returns the current network drive mappings as a WshCollection object. Items in this collection are local names and remote names.

### Syntax

```
WshNetwork.EnumPrinterConnections = objWshCollection
```

### Example

```
Set WshNetwork = Wscript.CreateObject("Wscript.Network")
Set oPrinters = WshNetwork.EnumPrinterConnections
Wscript.Echo oPrinters.Item(0) = "LPT1:"
Wscript.Echo oPrinters.Item(1) "\\Server\Printer1"
```

### See Also

WshNetwork.AddPrinterConnection method, WshNetwork.RemovePrinterConnection method, WshCollection object.

### WshNetwork.MapNetworkDrive

The MapNetworkDrive method maps the share point specified by strRemoteName to the local resource name strLocalName.

### Syntax

```
WshNetwork.MapNetworkDrive strLocalName, strRemoteName, [bUpdateProfile], [strUser], [strPassword]
```

### Parameters

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

strLocalName

Local resource name to map to.

strRemoteName

Remote share to map.

bUpdateProfile

If bUpdateProfile is supplied and its value is TRUE, this mapping is stored in the user profile.

strUser, strPassword

If you are mapping the share point using the credentials of someone other than current user, you can specify strUser and strPassword.

**Example** Set WshNetwork = Wscript.CreateObject("Wscript.Network")

```
WshNetwork.MapNetworkDrive "Z:", "\\Server\Share"
```

```
WshNetwork.RemoveNetworkDrive
```

The RemoveNetworkDrive method removes the current resource connection denoted by strName.

### Syntax

```
WshNetwork.RemoveNetworkDrive strName, [bForce], [bUpdateProfile]
```

### Parameters

strName

The strName parameter can be either a local name or a remote name, depending on how the drive is mapped. If the drive has a mapping between a local name (drive letter) and a remote name, then strName must be set to the local name. If the network path does not have a local name (drive letter) mapping, then strName must be set to the remote name.

bForce

If bForce is supplied and its value is TRUE, this method removes the connections whether the resource is used or not.

bUpdateProfile

If bUpdateProfile is supplied and its value is TRUE, this mapping is stored in the user profile.

### Example

```
Set WshNetwork = Wscript.CreateObject("Wscript.Network")
```

```
' Local name mapped to remote share
```

```
WshNetwork.MapNetworkDrive "Z:", "\\Server\Share"
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

```
WshNetwork.RemoveNetworkDrive "Z:"
```

```
' No local name mapping, such as:
```

```
' NET USE \\Server\Share WshNetwork.RemoveNetworkDrive "\\Server\Share"
```

```
WshNetwork.MapNetworkDrive "\\Server\Share"
```

```
WshNetwork.RemoveNetworkDrive "\\Server\Share"
```

```
WshNetwork.RemovePrinterConnection
```

The RemovePrinterConnection method removes the current resource connection denoted by strName.

### Syntax

```
WshNetwork.RemovePrinterConnection strName, [bForce], [bUpdateProfile]
```

### Parameters

strName

The strName parameter can be either a local name or a remote name, depending on how the printer is connected. If the printer has a mapping between a local name (for example, LPT1) and a remote name, then strName must be set to the local name. If the network path does not have a local name mapping, then strName must be set to the remote name.

bForce

If bForce is supplied and its value is TRUE, this method removes the connection whether the resource is used or not.

bUpdateProfile

If bUpdateProfile is supplied and its value is TRUE, this mapping is stored in the user profile.

### Example

```
Set WshNetwork = Wscript.CreateObject("Wscript.Network")
```

```
' Local name mapped to remote share
```

```
WshNetwork.RemovePrinterConnection "LPT1:"
```

```
' No local name mapping. e.g. NET USE "\\Server\Printer1"
```

```
WshNetwork.RemovePrinterConnection "\\Server\Printer1"
```

```
WshNetwork.SetDefaultPrinter
```

The SetDefaultPrinter method sets default print to the remote printer specified by strPrinterName.

### Syntax

```
WshNetwork.SetDefaultPrinter strPrinterName
```

### Parameters

strPrinterName

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Remote printer name to set as default, such as "\\Server\Printer1". Note that strPrinterName cannot be a local name such as "LPT1:".

### WshCollection Object

The WshCollection object is not exposed directly. For access to it, use WshNetwork.EnumNetworkDrives or WshNetwork.EnumPrinterConnections.

ProgID	N/A
Filename	WSHom.Ocx
CLSID	F935DC24-1CF0-11d0-ADB9-00C04FD58A0B
IID	F935DC23-1CF0-11d0-ADB9-00C04FD58A0B

The following table describes the properties associated with the WshCollection object.

Property	Description
Item	Provides the nth enumerated item as a string.
Count	The number of enumerated items.
length	The number of enumerated items (JScript).

### WshCollection.Item

The Item property provides the natIndexth enumerated item as a string. It is the default property.

### Syntax

```
WshCollection(natIndex) = strEnumeratedItem  
WshCollection.Item(natIndex) = strEnumeratedItem
```

### Example

```
Set WshNetwork = Wscript.CreateObject("Wscript.Network")  
Set oDrives = WshNetwork.EnumNetworkDrives  
Wscript.Echo oDrives.Item(0) = "Z:"  
Wscript.Echo oDrives.Item(1) = "\\Server\Share"
```

### WshCollection.Count

The Count property provides the number of enumerated items.

### Syntax

```
WshCollection.Count = natNumberOfItemsI
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### See Also

WshCollection.length property

### WshCollection.length

The length property provides the number of enumerated items. This property provides the same functionality as the Count property and is provided for compatibility with JScript.

### Syntax

WshCollection.length = natNumberOfItems

### See Also

WshCollection.Count property

### WshEnvironment Object

The WshEnvironment object is not exposed directly. For access to it, use the WshShell.Environment property.

ProgID	N/A
Filename	WSHom.Ocx
CLSID	
IID	

The following table describes the properties associated with the WshEnvironment object.

Property	Description
Item	Gets or sets the value of a specified environment variable.
Count	The number of enumerated items.
length	The number of enumerated items (JScript).

The following table describes the method associated with the WshEnvironment object.

Method	Description
--------	-------------

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Remove	Deletes a specified environment variable.
--------	---

### **WshEnvironment.Item**

The Item property sets or returns the value for the strName environment variable. It is the default property.

### **Syntax**

```
WshEnvironment.Item("strName") = strValue  
WshEnvironment("strName") = strValue
```

### **Example**

```
' Get the value of NUMBER_OF_PROCESSORS environment variable  
Set WshShell = Wscript.CreateObject("Wscript.Shell")  
Set WshSysEnv = WshShell.Environment("SYSTEM")  
Wscript.Echo WshSysEnv("NUMBER_OF_PROCESSORS")
```

```
' Set the value of EXAMPLE volatile environment variable to A_VALUE  
Set WshShell = Wscript.CreateObject("Wscript.Shell")  
Set WshEnv = WshShell.Environment("VOLATILE")  
WshEnv("EXAMPLE")= "A_VALUE"
```

```
' List all system environment variables  
Set WshShell = Wscript.CreateObject("Wscript.Shell")  
For Each strVarName In WshShell.Environment("SYSTEM")  
MsgBox strVarName  
Next
```

### **See Also**

WshShell.Environment property

### **WshEnvironment.Count**

The Count property provides the number of enumerated items.

### **Syntax**

```
WshEnvironment.Count = natNumberOfItems
```

### **See Also**

WshEnvironment.length property

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### **WshEnvironment.length**

The length property provides the number of enumerated items. This property provides the same functionality as the Count property and is provided for compatibility with JScript.

### **Syntax**

```
WshEnvironment.length = natNumberOfItems
```

### **See Also**

WshEnvironment.Count property

### **WshEnvironment.Remove**

The Remove method deletes the environment variable specified by strName.

### **Syntax**

```
WshEnvironment.Remove(strName)
```

### **Example**

```
' Delete the EXAMPLE volatile environment variable
Set WshShell = Wscript.CreateObject("Wscript.Shell")
WshShell.Environment("VOLATILE").Remove("EXAMPLE")
```

```
' Delete multiple variables
Set WshUsrEnv = Wscript.Environment("User")
WshUsrEnv.Remove("EXAMPLE_1")
WshUsrEnv.Remove("EXAMPLE_2")
WshUsrEnv.Remove("EXAMPLE_3")
WshUsrEnv.Remove("EXAMPLE_4")
```

### **See Also**

WshShell.Environment property

**WshShortcut Object** This object is not exposed directly. To get the WshShortcut object, use the WshShell.CreateShortcut method.

ProgID	N/A
Filename	WSHom.Ocx
CLSID	F935DC28-1CF0-11d0-ADB9-00C04FD58A0B
IID	F935DC27-1CF0-11d0-ADB9-00C04FD58A0B

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

The following table describes the properties associated with the WshShortcut object.

Property	Description
Arguments	Parameters to a shortcut object.
Description	A description of a shortcut object.
Hotkey	The hotkey of a shortcut object.
IconLocation	The icon location of a shortcut object.
TargetPath	The target path of a shortcut object.
WindowStyle	The window style of a shortcut object.
WorkingDirectory	The working directory of a shortcut object.

The following table describes the method associated with the WshShortcut object.

Method	Description
Save	Saves a shortcut into a specified file system.

### **WshShortcut.Arguments**

The Arguments property provides parameters to a shortcut object.

### **Syntax**

WshShortcut.Arguments = strArguments

### **WshShortcut.Description**

The Description property provides a description of a shortcut object.

### **Syntax**

WshShortcut.Description = strDescription

### **WshShortcut.FullName**

The FullName property provides the full path of a shortcut object.

### **Syntax**

WshShortcut.FullName = strFullName

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### **WshShortcut.HotKey**

The HotKey property provides the hotkey of a shortcut object. A hotkey is a keyboard shortcut to start or switch to a program.

### **Syntax**

```
WshShortcut.HotKey = strHotKey
```

### **Remarks**

BNF syntax of strHotKey is as follows:

```
Hotkey ::= modifier* keyname  
modifier ::= "ALT+" | "CTRL+" | "SHIFT+" | "EXT+"  
keyname ::= "A" .. "Z" |  
           "0" .. "9" |  
           "Back" | "Tab" | "Clear" | "Return" |  
           "Escape" | "Space" | "Prior" | ...
```

Complete key names are found in WINUSER.H. Hotkey is case-insensitive.

Hotkeys can only activate shortcuts located on the Windows-based desktop or the Windows Start menu. The Windows Explorer does not accept ESC, ENTER, TAB, SPACE, PRINT SCREEN or BACKSPACE, even though WshShortcut.HotKey supports these in compliance with the Win32 API. It is therefore recommended that you not use these keys in your shortcut.

### **Example**

```
Set WshShell = Wscript.CreateObject("Wscript.WshShell")  
strDesktop = WshShell.SpecialFolders("Desktop")  
Set oMyShortcut = WshShell.CreateShortcut(strDesktop & "\a_key.lnk")  
oMyShortcut.TargetPath = "%windir%\notepad.exe"  
oMyShortcut.Hotkey = "ALT+CTRL+F"  
oMyShortcut.Save  
Wscript.Echo oMyShortcut.HotKey = "Alt+Ctrl+F"
```

### **See Also**

WshSpecialFolders object

### **WshShortcut.IconLocation**

The IconLocation property provides the icon location of a shortcut object. The format of the icon location should be "Path,index".

### **Syntax**

```
WshShortcut.IconLocation = strIconLocation
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

### **WshShortcut.TargetPath**

The TargetPath property provides the target path of a shortcut object.

#### **Syntax**

```
WshShortcut.TargetPath = strTargetPath
```

### **WshShortcut.WindowStyle**

The WindowStyle property provides the window style of a shortcut object.

#### **Syntax**

```
WshShortcut.WindowStyle = natWindowStyle
```

### **WshShortcut.WorkingDirectory**

The WorkingDirectory property provides the working directory of a shortcut object.

#### **Syntax**

```
WshShortcut.WorkingDirectory = strWorkingDirectory
```

### **WshShortcut.Save**

The Save method saves the shortcut object to the location specified by the FullName property.

#### **Syntax**

```
WshShortcut.Save
```

### **WshSpecialFolders Object**

This object is not exposed directly. To get the WshSpecialFolders object, use the WshShell.SpecialFolders property.

ProgID	N/A
Filename	WSHom.Ocx
CLSID	
IID	

The following table describes the properties associated with the WshSpecialFolders object.

Property	Description
Item	The full path of the specified special folder (default)

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Count	The number of enumerated items.
length	The number of enumerated items (JScript).

### **WshSpecialFolders.Item**

The Item property returns the full path for the special folder specified by strFolderName. It is the default property.

### **Syntax**

```
WshShell.SpecialFolders.Item("strFolderName") = strFolderPath  
WshShell.SpecialFolders("strFolderName") = strFolderPath
```

### **Remarks**

WshShell.SpecialFolders("strFolderName") returns NULL if the requested folder (strFolderName) is not available. For example, Windows 95 does not have an AllUsersDesktop folder and returns NULL if strFolderName = AllUsersDesktop

The following special folders are provided with the Windows 95 and Windows NT 4.0 operating systems:

- AllUsersDesktop
- AllUsersStartMenu
- AllUsersPrograms
- AllUsersStartup
- Desktop
- Favorites
- Fonts
- MyDocuments
- NetHood
- PrintHood
- Programs
- Recent
- SendTo
- StartMenu
- StartupB
- Templates

### **Example**

```
' This fragment returns the full path for the Windows Desktop folder  
Set WshShell = Wscript.CreateObject("Wscript.Shell")  
StrMyDesktop = WshShell.SpecialFolders("Desktop")
```

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

```
' List all special folders
For Each strFolder In WshShell.SpecialFolders
MsgBox strFolder
Next
```

### See Also

WshShell.SpecialFolders property

### WshSpecialFolders.Count

The Count property provides the number of enumerated items.

### Syntax

```
WshSpecialFolders.Count = natNumberOfItems
```

### See Also

WshSpecialFolders.length property

### WshSpecialFolders.length

The length property provides the number of enumerated items. This property provides the same functionality as the Count property and is provided for compatibility with Microsoft JScript.

### Syntax

```
WshSpecialFolders.length = natNumberOfItems
```

### See Also

WshSpecialFolders.Count property

### WshUrlShortcut Object

This object is not exposed directly. To get the WshUrlShortcut object, use the WshShell.CreateShortcut method.

ProgID	N/A
Filename	WSHom.Ocx
CLSID	
IID	

The following table describes the properties associated with the WshUrlShortcut object.

# MICROSOFT WINDOWS SCRIPTING HOST

## PROGRAMMER'S REFERENCE

Property	Description
FullName	The full path of a URL shortcut object.
TargetPath	The target path of a URL shortcut object.

The following table describes the method associated with the WshUrlShortcut object.

Property	Description
Save	Saves a shortcut into a specified file system.

### **WshUrlShortcut.FullName**

The FullName property provides the full path of a shortcut object.

#### **Syntax**

```
WshUrlShortcut.FullName = strFullName
```

### **WshUrlShortcut.TargetPath**

The TargetPath property provides the target path of a shortcut object.

#### **Syntax**

```
WshUrlShortcut.TargetPath = strTargetPath
```

### **WshUrlShortcut.Save**

The Save method saves a shortcut to the location specified by the FullName property.

#### **Syntax**

```
WshUrlShortcut.Save
```

## **FOR MORE INFORMATION**

For the latest information on Windows NT Server, check out our World Wide Web site at <http://www.microsoft.com/ntserver> or the Windows NT Server Forum on the Microsoft Network (GO WORD: MSNTS).