

WORKING WITH WINDOWS SHELL32..... 3

MANAGING DISK QUOTAS ON THE NTFS FILE SYSTEM	4
MICROSOFT.DIDISKQUOTAUSER OBJECT	4
DIDiskQuotaUser.AccountContainerName Property	5
DIDiskQuotaUser.AccountStatus Property	5
DIDiskQuotaUser.DisplayName Property	5
DIDiskQuotaUser.ID Property	6
DIDiskQuotaUser.LogonName Property	6
DIDiskQuotaUser.QuotaLimit Property	6
DIDiskQuotaUser.QuotaLimitText Property	6
DIDiskQuotaUser.QuotaThreshold Property	7
DIDiskQuotaUser.QuotaThresholdText Property	7
DIDiskQuotaUser.QuotaUsed Property	7
DIDiskQuotaUser.QuotaUsedText Property	7
DIDiskQuotaUser.Invalidate Method	8
DISKQUOTACONTROL OBJECT	8
DiskQuotaControl.DefaultQuotaLimit Property	8
DiskQuotaControl.DefaultQuotaLimitText Property	9
DiskQuotaControl.DefaultQuotaThreshold Property	9
DiskQuotaControl.DefaultQuotaThresholdText Property	10
DiskQuotaControl.LogQuotaLimit Property	10
DiskQuotaControl.LogQuotaThreshold Property	10
DiskQuotaControl.QuotaFileIncomplete Property	11
DiskQuotaControl.QuotaFileRebuilding Property	11
DiskQuotaControl.QuotaState Property	11
DiskQuotaControl.UserNameResolution Property	12
DiskQuotaControl.AddUser Method	12
DiskQuotaControl.DeleteUser Method	12
DiskQuotaControl.FindUser Method	13
DiskQuotaControl.GiveUserNameResolutionPriority Method	14
DiskQuotaControl.Initialize Method	14
DiskQuotaControl.InvalidateSidNameCache Method	15
DiskQuotaControl.ShutdownNameResolution Method	15
DiskQuotaControl.TranslateLogonNameToSID Method	16
SHELL32.FOLDER AND SHELL32.FOLDER2 OBJECTS	16
Shell32.Folder.ParentFolder Property	17
Shell32.Folder.Title Property	17
Shell32.Folder.CopyHere Method	18
Shell32.Folder.GetDetailsOf Method	19
Shell32.Folder.Items Method	21
Shell32.Folder.MoveHere Method	21
Shell32.Folder.NewFolder Method	22
Shell32.Folder.ParseName Method	23
Shell32.Folder2.OfflineStatus Property	24
Shell32.Folder2.Self Property	24
Shell32.Folder2.DismissedWebViewBarricade Method	25
Shell32.Folder2.Synchronize Method	25
SHELL32.FOLDERITEM OBJECT	25

Shell32.FolderItem.GetFolder Property	25
Shell32.FolderItem.GetLink Property	26
Shell32.FolderItem.IsBrowsable Property	26
Shell32.FolderItem.IsFileSystem Property	27
Shell32.FolderItem.IsFolder Property	27
Shell32.FolderItem.IsLink Property	27
Shell32.FolderItem.ModifyDate Property	27
Shell32.FolderItem.Name Property	27
Shell32.FolderItem.Parent Property	28
Shell32.FolderItem.Path Property	28
Shell32.FolderItem.Size Property	28
Shell32.FolderItem.Type Property	28
Shell32.FolderItem.InvokeVerb Method	29
Shell32.FolderItem.Verbs Method	30
SHELL32.FOLDERITEMS, SHELL.FOLDERITEMS2, SHELL.FOLDERITEMS3 OBJECTS	32
Shell32.FolderItems.Count Property	32
Shell32.FolderItems.Item Method	32
Shell32.FolderItems2.InvokeVerbEx Method	33
Shell32.FolderItems3.Verbs Property	33
Shell32.FolderItems3.Filter Method	33
SHELL32.FOLDERITEMVERB OBJECT	35
Shell32.FolderItemVerb.Name Property	35
Shell32.FolderItemVerb.DoIt Method	36
SHELL32.FOLDERITEMVERBS OBJECT	36
Shell32.FolderItemVerbs.Count Property	37
Shell32.FolderItemVerbs.Item Method	37
SHELL32.ISHELLDISPATCH INTERFACE AND SHELL OBJECT	37
Shell32.Shell.Parent Property	37
Shell32.AddToRecent Method	37
Shell32.CanStartStopService Method	38
Shell32.CascadeWindows Method	38
Shell32.ControlPanelItem Method	39
Shell32.EjectPC Method	40
Shell32.Explore Method	40
Shell32.ExplorerPolicy Method	41
Shell32.FileRun Method	42
Shell32.FindComputer Method	42
Shell32.FindFiles Method	43
Shell32.FindPrinter Method	44
Shell32.GetSetting Method	45
Shell32.GetSystemInformation Method	45
Shell32.Help Method	46
Shell32.IsRestricted Method	47
Shell32.IsServiceRunning Method	48
Shell32.MinimizeAll Method	48
Shell32.Namespace Method	49
Shell32.Open Method	49
Shell32.RefreshMenu Method	50
Shell32.ServiceStop Method	50
Shell32.ShellExecute Method	50
Shell32.SetTime Method	51

Shell32.ShowBrowserBar Method	52
Shell32.ShutdownWindows Method	53
Shell32.TileHorizontally Method	53
Shell32.TileVertically Method	54
Shell32.ToggleDesktop Method	54
Shell32.TrayProperties Method	55
Shell32.UndoMinimizeAll Method	56
Shell32.Windows Method	56
Shell32.WindowsSecurity Method	56
SHELL32.SHELLLINKOBJECT OBJECT	57
Shell32.ShellLinkObject.Target Property	57
SHELL32.SHELLFOLDERITEM OBJECT	57
Shell32.ShellFolderItem.ExtendedProperty Method	57
Q&A	58
HOW TO LIST ITEMS IN THE ADMINISTRATIVE TOOLS FOLDER?	58
HOW TO LIST ITEMS IN THE "MY MUSIC" FOLDER?	58
HOW TO RETRIEVE THE PATH TO "COMMON DESKTOP" FOLDER?	58
HOW TO LIST THE LOCAL COMPUTER INFORMATION?	60
HOW TO ADD A WEB SITE TO THE FAVORITES MENU?	60
APPENDIX 14.A – SHELL32 CONSTANTS	61
QuotaStateConstants Values	61
UserNameResolutionConstants Values	61
QuotaStateConstants Values	61
Shell32.OfflineFolderStatus Values	61
SHFILEOPSTRUCT Values	62
Details Options Values	62
Shell32.ShellSpecialFolderConstants Values	63
Shell32.ShellFolderViewOptions Values	66
SHCONTF Enumerated Type Values	66
SHGetSetSettings Enumerated Type Values	66
wProcessorArchitectureType SYSTEM_INFO Values	66
SHCOLUMNID Values	66

Working with Windows Shell32

Microsoft® Windows® Script Host (**WSH**) is a language-independent scripting host for **Windows Script** compatible scripting engines. It brings simple, powerful, and flexible scripting to the Windows 32-bit platform, allowing you to run scripts from both the Windows desktop and the command prompt.

Windows Script Host is ideal for non-interactive scripting needs, such as logon scripting, administrative scripting, and machine automation.

This section describes the Microsoft Windows objects implemented by the Shell32.

Object	Description
DIDiskQuotaUser	The DiskQuotaControl object allows a client to manage an NTFS volume's global disk quota settings. This object makes the essential functionality of the DIDiskQuotaUser interface
DiskQuotaControl	The NTFS file system allows an administrator to manage disk usage on a shared volume by allocating a specified amount of disk space, or quota

	limit, to each user.
Folder	The Folder object represents a Shell folder. It contains properties and methods that allow you to retrieve information about the folder.
Folder2	The Folder2 object extends the Folder object to support offline folders.
FolderItem	The FolderItem object represents an item in a Shell folder. It contains properties and methods that allow you to retrieve information about the item.
FolderItems	The FolderItems object represents the collection of items in a Shell folder. It contains properties and methods that allow you to retrieve information about the collection.
FolderItems2	The FolderItems2 object extends the FolderItems object. It supports one additional method.
FolderItems3	The FolderItems3 object extends the FolderItems2 object. It supports an additional method and property.
FolderItemVerb	The FolderItemVerb object represents a single verb available to an item.

Managing Disk Quotas on the NTFS File System

Disk quotas are an integral part of the **NTFS** file system. When a file or a folder is created on a volume formatted with **NTFS**, that item is assigned an owner (typically the user who created the item). **NTFS** obtains the **user ID** of the file owner and stores that information in the file or folder's Standard Information attribute. This attribute tallies all the disk space allocated to the file or folder. **NTFS** then locates the quota entry for that user and determines whether the new allocation of disk space causes the user to exceed the assigned quota. If it does, **NTFS** then takes the appropriate steps, which can include logging an entry in the System event log or preventing the user from creating the file or folder. As the file or folder changes size, **NTFS** updates the quota control entry to reflect the total disk space used by the user.

Disk quotas are not configured on a computer-wide basis, but are instead tied to individual **NTFS** volumes. Each drive has separate quota settings, and the actions you take on one volume do not affect the other volumes.

When managing disk quotas on a computer, the actions you take on one volume do not affect the other volumes in any way. If you allocate User **A** 50 MB of disk space on drive **C**, this does not also give User **A** 50 MB of disk space on drives **D** and **E**. If you disable disk quotas on drive **D**, quotas remain enabled on drives **C** and **E**.

Microsoft.DIDiskQuotaUser Object

The **DiskQuotaControl** object allows a client to manage an **NTFS** volume's global disk quota settings. This object makes the essential functionality of the **DIDiskQuotaUser** interface available to scripting and **Microsoft Visual Basic**-based applications.

Each user on the volume that is managed by the **DiskQuotaControl** object has a **DIDiskQuotaUser** object associated with it. This object allows a client to manage an individual user's settings. There are several ways to obtain a user's **DIDiskQuotaUser** object:

- The **DIDiskQuotaUser** objects for all users with quotas on the volume are exposed as a collection and can be enumerated. A discussion of how to enumerate **DIDiskQuotaUser** objects is found below.
- When you add a new user, the **AddUser** method returns the user's **DIDiskQuotaUser** object.
- If you have the user's name, the **FindUser** method returns the user's **DIDiskQuotaUser** object.

The **DIDiskQuotaUser** objects for all users with a quota on the volume are exposed as a collection. The **DiskQuotaControl** object exports a standard enumerator method that allows you to enumerate the collection of **DIDiskQuotaUser** objects.

DIDiskQuotaUser.AccountContainerName Property

Description

The **AccountContainerName** property retrieves the name of the user's account container.

Data Type

String value that is set to the user's account container name.

Note

- For Microsoft Windows NT 4.0 accounts or other accounts without directory services information, this property contains the domain name.
- For accounts with directory services information, this property contains a canonical name with the terminating object name removed.

DIDiskQuotaUser.AccountStatus Property

Description

The **AccountStatus** property retrieves the status of the user's account.

Data Type

Integer – one of the AccountStatus Values in Table 3 on page 61

Note

- The property is read-only. The property has no default value.

DIDiskQuotaUser.DisplayName Property

Description

The **DisplayName** property retrieves the status of the user's account.

Data Type

String value that is set to the user's display name.

 **Note**

- This property contains the user's "friendly name." Its value is not necessarily defined.

DIDiskQuotaUser.ID Property

 **Description**

The **ID** property retrieves an identifier (ID) that uniquely identifies the user.

Data Type

Integer value that uniquely identifies the user's **DIDiskQuotaUser** object within a particular **DiskQuotaControl** process.

 **Note**

- The property is read-only. The property has no default value.

DIDiskQuotaUser.LogonName Property

 **Description**

The **LogonName** property retrieves the user's logon account name.

Data Type

String value that is set to the user's account logon name

 **Note**

- The property is read-only. The property has no default value.

DIDiskQuotaUser.QuotaLimit Property

 **Description**

The **QuotaLimit** property sets or retrieves the user's current quota limit, in bytes.

Data Type

Integer value that specifies or receives the user's current quota limit, in bytes.

 **Note**

- The property is read/write. The property has no default value.

DIDiskQuotaUser.QuotaLimitText Property

 **Description**

The **QuotaLimitText** property sets or retrieves the user's current quota limit, in bytes.

Data Type

String value that contains the user's current quota limit.

 **Note**

- The property is read-only. The property has no default value.

DIDiskQuotaUser.QuotaThreshold Property

Description

The **QuotaThreshold** property sets or retrieves the user's warning threshold, in bytes.

Data Type

Integer value that specifies or receives the user's warning threshold. If a user's disk usage exceeds this value and the **LogQuotaThreshold** property is set to **TRUE**, the system generates an event log entry.

Note

- The property is read/write. The property has no default value.

DIDiskQuotaUser.QuotaThresholdText Property

Description

The **QuotaThresholdText** property retrieves the user's warning threshold as a text string.

Data Type

String value that contains the user's warning threshold. If a user's disk usage exceeds this value and the **LogQuotaThreshold** property is set to **TRUE**, the system generates an event log entry.

Note

- The property is read-only. The property has no default value.

DIDiskQuotaUser.QuotaUsed Property

Description

The **QuotaUsed** property retrieves the user's current disk usage, in bytes.

Data Type

Integer value that is set to the amount of disk space currently in use. If **NTFS** file compression is enabled, **QuotaUsed** reflects the amount of disk space that the data would require in an uncompressed state.

Note

- The property is read-only. The property has no default value.

DIDiskQuotaUser.QuotaUsedText Property

Description

The **QuotaUsedText** property retrieve the user's current disk usage as a text string.

Data Type

String value that is set to the amount of disk space currently in use. If **NTFS** file compression is enabled, this property reflects the amount of disk space that the data would require in an uncompressed state.

 **Note**

- The property is read-only. The property has no default value.

DI DiskQuotaUser.Invalidate Method

 **Description**

The **Invalidate** method clears the object's cached user information.

Syntax

```
object.Invalidate ()
```

Return Value

No return value.

 **Note**

- This method clears the user information stored in the object's cache. The next time a request is made for quota-related information, the object retrieves the information from the **NTFS** volume and refreshes the cache.

DiskQuotaControl Object

The **NTFS** file system allows an administrator to manage disk usage on a shared volume by allocating a specified amount of disk space, or quota limit, to each user. The **DiskQuotaControl** object allows an administrator to manage a volume's disk quota properties. For instance, you can use this object to set the default quota limit that will be automatically assigned to all new users.

- Manage the folder structure to make files easy for users to locate.
- Ensure that the proper versions of specific files are installed and updated when necessary.
- Track files and folders, periodically culling files and folders that are no longer used.
- Move files and folders from one location to another as circumstances dictate.
- Create and manage shared folders to provide access to files from anywhere within the organization.

DiskQuotaControl.DefaultQuotaLimit Property

 **Description**

The **DefaultQuotaLimit** property sets or retrieves the default quota limit, in bytes.

Data Type

Integer value that specifies or receives the default quota limit for new users, in bytes.

Example

```
Option Explicit
Dim oShell, oSpFoldersColl
Set oShell = CreateObject("WScript.Shell")
Set oSpFoldersColl = oShell.SpecialFolders
MsgBox oSpFoldersColl.Count
```

DiskQuotaControl.DefaultQuotaLimitText Property**Description**

The **DefaultQuotaLimitText** property the default quota limit as a text string.

Data Type

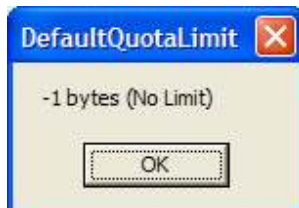
String value that contains the default quota limit for new users of the volume. For example, if the default quota is 10.5 MB, the value of the property is "10.5 MB". If the volume has no default quota, the property is set to "No Limit" or the localized equivalent.

Note

- The property is read-only. The property has no default value.

Example

```
Option Explicit
Dim oDQCtrl
Dim nDefQuotaLimit, sDefQuotaLimitText
Dim bReadWrite : bReadWrite = False
'--- Creating the MS DiskQuotaControl object from DiskQuotaTypeLibrary
Set oDQCtrl = CreateObject("Microsoft.DiskQuota.1")
oDQCtrl.Initialize "C:\", bReadWrite
'--- Retrieving the default quota limit (-1 no limit)
nDQLimit = oDQCtrl.DefaultQuotaLimit
sDQLimitText = oDQCtrl.DefaultQuotaLimitText
MsgBox nDQLimit & " bytes (" & sDQLimitText & ")", 0, "DefaultQuotaLimit"
'--- Cleaning objects
Set oDQCtrl = Nothing
```

**DiskQuotaControl.DefaultQuotaThreshold Property****Description**

The **DefaultQuotaThreshold** property sets or retrieves the default quota threshold, in bytes

Data Type

Integer value that is set to the default warning threshold for new users, in

bytes.

 **Note**

- The property is read/write. The property has no default value.
- The default quota threshold is applied automatically to new users of the volume.
- If a user's disk usage exceeds this value and the **LogQuotaThreshold** property is set to **TRUE**, the system generates an event log entry. For example, if the default threshold is 10.0 MB, the value of the property is "10.0 MB". If the volume has no default threshold, the property is set to "No Limit" or the localized equivalent.

DiskQuotaControl.DefaultQuotaThresholdText Property

 **Description**

The **DefaultQuotaThresholdText** property retrieves the default quota threshold as a text string.

Data Type

String value that contains the default quota threshold for the volume.

 **Note**

- The property is read-only. The property has no default value.
- The default quota threshold is applied automatically to new users of the volume.
- If a user's disk usage exceeds this value and the **LogQuotaThreshold** property is set to **TRUE**, the system generates an event log entry. For example, if the default threshold is 10.0 MB, the value of the property is "10.0 MB". If the volume has no default threshold, the property is set to "No Limit" or the localized equivalent.

DiskQuotaControl.LogQuotaLimit Property

 **Description**

The **LogQuotaLimit** property sets or retrieves a **Boolean** value that indicates whether a system event log entry will be made when a user exceeds his or her assigned quota limit.

Data Type

This property is set to **TRUE** if a system event log entry is made when the user exceeds his or her quota limit, or **FALSE** otherwise.

 **Note**

- The property has no default value.

DiskQuotaControl.LogQuotaThreshold Property

 **Description**

The **LogQuotaThreshold** property sets or retrieves a **Boolean** value that

indicates whether a system event log entry will be made when a user exceeds his or her assigned quota threshold.

Data Type

This property is set to **TRUE** if a system event log entry is made when the user exceeds his or her quota warning threshold, or **FALSE** otherwise.

Note

- The property has no default value.

DiskQuotaControl.QuotaFileIncomplete Property

Description

The **QuotaFileIncomplete** property retrieves a **Boolean** value that indicates whether the quota file for the volume is complete.

Data Type

This property is set to **TRUE** if the quota file is incomplete, or **FALSE** otherwise.

Note

- The property is read-only. The property has no default value.

DiskQuotaControl.QuotaFileRebuilding Property

Description

The **QuotaFileRebuilding** property retrieves a **Boolean** value that indicates whether the quota file for the volume is currently being rebuilt.

Data Type

This property is set to **TRUE** if the quota file is being rebuilt, or **FALSE** otherwise.

Note

- The property is read-only. The property has no default value.
- The quota file is automatically rebuilt when quotas are enabled on the system or when one or more user entries are marked for deletion.

DiskQuotaControl.QuotaState Property

Description

The **QuotaState** property sets or retrieves the state of the volume's disk quotas.

Data Type

Integer, The property can be set to one of the [QuotaStateConstants](#) defined in Table 1 on page 61

Note

- The property is read/write. The property has no default value

DiskQuotaControl.UserNameResolution Property

Description

The **UserNameResolution** property sets or retrieves a value that controls how user **Security IDentifier (SID)** is resolved to user names.

Data Type

Integer, The property can be set to one of the **UserNameResolutionConstants** defined in Table 2 on page 71

Note

- The property is read/write. The property has no default value.
- This property affects the enumeration of **DIDiskQuotaUser** objects, and the **AddUser** and **FindUser** methods.

DiskQuotaControl.AddUser Method

Description

The **AddUser** method assigns a nondefault disk quota to a new user.

Syntax

```
object.AddUser (sLogonName)
```

Arguments

Parameter	Description
<i>sLogonName</i>	Required. String value that contains the user's logon name. Use the UserNameResolution property to specify how the name is to be resolved.

Return Value

No return value.

Note

- The **NTFS** file system automatically creates a user quota entry when a user first writes to the volume.
- Entries that are created in this way are assigned the default warning threshold and hard quota limit values for the volume.
- This method allows you to create a user quota entry before a user writes information to the volume. It returns a **DIDiskQuotaUser** object that can be used to assign a warning threshold or quota limit value that differs from the default settings for the volume.
- If the user already exists, no new entry is created. The method returns the **DIDiskQuotaUser** object associated with the existing entry.

DiskQuotaControl.DeleteUser Method

Description

The **DeleteUser** method deletes a user from the volume.

Syntax

```
object.DeleteUser (oUser)
```

Arguments

Parameter	Description
<i>oUser</i>	Required. Object expression that evaluates to the user's DDiskQuotaUser object.

Return Value

No return value.

Note

- This method fails if the user owns any storage on the volume. Before you delete a user from a volume, all storage for that user must be deleted, be moved to another volume, or have its ownership transferred to another user.the default settings for the volume.

DiskQuotaControl.FindUser Method**Description**

The **FindUser** method finds a user's entry, by name, in the volume's quota file.

Syntax

```
object.FindUser (sLogonName)
```

Arguments

Parameter	Description
<i>sLogonName</i>	Required. String value that contains the user's logon name.

Note

- This method returns a **DDiskQuotaUser** object even if there is no entry for the user in the quota file. The returned user object has warning threshold and hard quota limits set to the volume's default values.
- The string returned from **TranslateLogonNameToSID** may be passed in place of the *sLogonName* parameter.
- When FindUser receives a **SID** string it uses the corresponding **SID** for direct lookup of the user's quota record on the volume. This bypasses the **SID**-name cache. In cases where **FindUser** fails due to a mismatch in format (**SAM**-compatible vs. **UPN**) of the logon name provided and the logon name cached, the logon name can be translated to a **SID** string using **TranslateLogonNameToSID** then passed again to **FindUser**.

Example

the logon name can be translated to a **SID** string using **TranslateLogonNameToSID** then passed again to **FindUser**.

```
Function Find(ByVal oDQC, ByVal sName)
    On Error Resume Next
```

```

Set Find = oDQC.FindUser(sName)
If Err.Number <> 0 Then
    Err.Clear
    Set Find = oDQC.FindUser(dqc.TranslateLogonNameToSID(sName))
End If
End Function
Set oDQC = Nothing

```

DiskQuotaControl.GiveUserNameResolutionPriority Method

Description

The **GiveUserNameResolutionPriority** method places the specified user object next in line for name resolution.

Syntax

```
object.GiveUserNameResolutionPriority (oUser)
```

Arguments

Parameter	Description
<i>oUser</i>	Required. Object expression that evaluates to the user's DDiskQuotaUser object.

Return Value

No return value.

Note

- If asynchronous name resolution is enabled, user objects are placed in a queue. By default, they are serviced in the order they are placed in the queue. The **GiveUserNameResolutionPriority** method moves an object to the front of the queue so that it is next in line to be serviced.
- Use the **UserNameResolution** property to enable asynchronous name resolution.

DiskQuotaControl.Initialize Method

Description

The **GiveUserNameResolutionPriority** method places the specified user object next in line for name resolution.

Syntax

```
object.Initialize (sPath, bReadWrite)
```

Arguments

Parameter	Description
<i>sPath</i>	Required. String value that contains the fully qualified path of the volume to be initialized.
<i>bReadWrite</i>	Required. Boolean value that specifies how the volume is to be opened.

	Set <i>bReadWrite</i> to TRUE for read/write access or FALSE for read-only access.
--	--

Return Value

No return value.

DiskQuotaControl.InvalidateSidNameCache Method

Description

The **InvalidateSidNameCache** method invalidates the security identifier (ID) user name cache.

Syntax

```
object.InvalidateSidNameCache
```

Return Value

No return value.

Note

- Users' names and associated security IDs are stored in a cache. You can clear this cache by calling **InvalidateSidNameCache**.
- If you subsequently create a new user object, the information will have to be obtained from the domain controller, and the cache will have to be reestablished.

DiskQuotaControl.ShutdownNameResolution Method

Description

The **ShutdownNameResolution** method shuts down the user name resolution thread.

Syntax

```
object.ShutdownNameResolution
```

Return Value

No return value.

Note

- The security identifier (**SID**) name resolver translates **SID** to user names on a background thread.
- This thread is shut down automatically when the associated quota control object is destroyed. However, there are some cases when the thread is no longer needed, but the object is not yet ready to be destroyed.
- A typical example is when no further processing is taking place, but clients are still holding references to the object.
- The **ShutdownNameResolution** method allows you to terminate the resolver thread and free the associated resources without destroying the quota control object.
- When you shut down the resolver thread, asynchronous name resolution

immediately stops. If calls are subsequently made to methods such as **AddUser** or **FindUser**, the quota object might re-create the resolver thread.

DiskQuotaControl.TranslateLogonNameToSID Method

Description

The **TranslateLogonNameToSID** method places the specified user object next in line for name resolution.

Syntax

```
object.TranslateLogonNameToSID (logonname)
```

Arguments

Parameter	Description
<i>logonname</i>	Required. String value that specifies the user's logon name.

Return Value

No return value.

Note

- The returned **SID** string can be passed to the **FindUser** method in place of a logon name.
- When a call to the **FindUser**(*logonname*) method fails, it could be due to a mismatch between the form (**Security Account Manager [SAM]** compatible vs. **User Principal Name [UPN]**) of the logon name provided and the form stored in the **SID**-name cache. In such cases, the logon name can be converted to a **SID** and the call to **FindUser** repeated. **FindUser** recognizes a **SID** string and will bypass the **SID**-name cache lookup.
- Name-to-**SID** translation can be a slow process when compared to lookups in the **SID**-name cache. Therefore, it is recommended that **FindUser** first be called with a logon name.

Example

the logon name can be translated to a **SID** string using **TranslateLogonNameToSID** then passed again to **FindUser**.

```
Function Find(ByVal oDQC, ByVal sName)
    On Error Resume Next
    Set Find = oDQC.FindUser(sName)
    If Err.Number <> 0 Then
        Err.Clear
        Set Find = oDQC.FindUser(dqc.TranslateLogonNameToSID(sName))
    End If
End Function
Set oDQC = Nothing
```

Shell32.Folder and Shell32.Folder2 Objects

The **Folder** object represents a **Shell** folder. It contains properties and methods

that allow you to retrieve information about the folder.

The **Folder2** object extends the **Folder** object to support offline folders.

Shell32.Folder.ParentFolder Property

Description

The **ParentFolder** property contains the parent **Folder** object.

Data Type

An object reference to the **ParentFolder** object.

Note

- Note Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder (CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Example

The following code implements the **ParentFolder** property to retrieve the **Folder** object.

```
Option Explicit
Private Const ssfPROGRAMS = 2
Dim oShell32, oFolder, oParentFolder
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.NameSpace(ssfPROGRAMS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) Then
    '--- Retrieving the parent folder object
    Set oParentFolder = oFolder.ParentFolder '--- usually "Start Menu".
    MsgBox oParentFolder.Title
End If
Set oParentFolder = Nothing : Set oFolder = Nothing : Set oShell32 = Nothing
```

Shell32.Folder.Title Property

Description

The **ParentFolder** property contains the title of the folder.

Data Type

An object reference to the **ParentFolder** object.

Note

- Note Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder (CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Example

The following example uses **Title** to retrieve the title of the folder holding the

user's program groups (usually "Programs").

```

Option Explicit
Private Const ssfPROGRAMS = 2
Dim oShell32, oFolder
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.NameSpace(ssfPROGRAMS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) Then
    '--- Retrieving the folder title
    MsgBox oFolder.Title
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing

```

Shell32.Folder.CopyHere Method

Description

The **CopyHere** method copies an item or items to a folder.

Syntax

```
object.CopyHere (vItem, vOptions)
```

Arguments

Parameter	Description
<i>vItem</i>	Required. Specifies the item or items to copy. This can be a string that represents a file name, a FolderItem object, or a FolderItems object.
<i>vOptions</i>	Optional. Specifies options for the copy operation. This value can be zero or a combination of the of the SHFILEOPSTRUCT structure values described in Table 5 on page 62

Return Value

No return value.

Note

- Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder (CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Example

The following example uses **CopyHere** to copy the Autoexec.bat file from the root directory to the C:\Windows directory

```

Option Explicit
Private Const FOF_FILESONLY = 128
Private Const FOF_NOCONFIRMATION = 16
Dim oShell32, oFolder
'--- Creating the shell application file
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = ShellApp.NameSpace("C:\Windows")
If Not oFolder Is Nothing Then
    '--- Copy to folder all files under temp with
    '--- flag for use regular expression and
    '--- flag to avoid confirmation dialog
    oFolder.CopyHere "C:\temp\*.*", FILESONLY + FOF_NOCONFIRMATION
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing

```

Shell32.Folder.GetDetailsOf Method

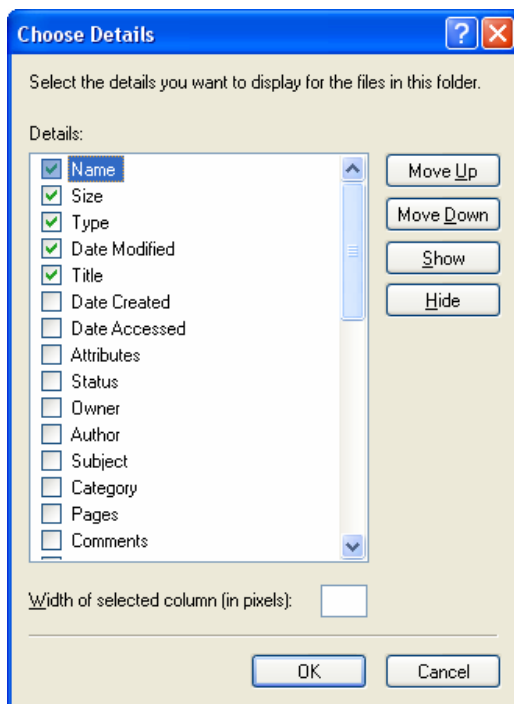


Figure 1 - Details options

Description

The **GetDetailsOf** method retrieves details about an item in a folder. For example, its size, type, or the time of its last modification.

Syntax

```
object.GetDetailsOf(vItem, iColumn)
```

Arguments

Parameter	Description
-----------	-------------

<i>vItem</i>	Required. Specifies the item or items to copy. This can be a string that represents a file name, a FolderItem object, or a FolderItems object.
<i>iColumn</i>	Required. An Integer value that specifies the information to be retrieved. The information available for an item depends on the folder in which it is displayed. This value corresponds to the zero-based column number that is displayed in a Shell view. For an item in the file system, this can be one of the Details Options Values in Table 6 on page 63

Return Value

String containing the retrieved detail.

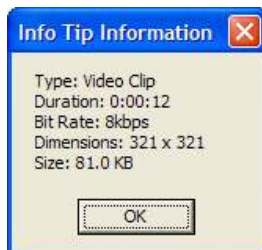
Note

- Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder (CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Example

The following example uses **GetDetailsOf** to retrieve the type of the file named Clock.avi.

```
Option Explicit
Private Const colInfoTip = -1
Dim oShell32, oFolder, oFolderItem
Dim sInfo
'--- Creating the shell application file
Set oShell32 = CreateObject("Shell.Application")
'--- Retrieving a folder object using the namespace method
Set oFolder = oShell32.Namespace("C:\Windows")
'--- Checking Folder validation
If not oFolder is nothing then
    '--- Retrieving the file item object
    Set oFolderItem = oFolder.ParseName("clock.avi")
    '--- Checking file item validation
    If (Not oFolderItem Is Nothing) then
        '--- Retrieving the info tip information for the item.
        sInfo = oFolder.GetDetailsOf(oFolderItem, colInfoTip)
        MsgBox sInfo, 0, "Info Tip Information"
    End if
    Set oFolderItem = Nothing
End if
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing : Set oFolderItem = Nothing
```



Shell32.Folder.Items Method

Description

The **Items** method retrieves a **FolderItems** object that represents the collection of items in the folder.

Syntax

```
object.Items ()
```

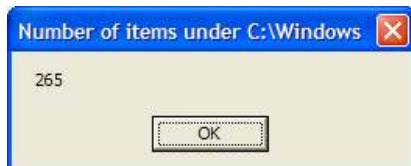
Note

- Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder (CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Example

The following example uses **Items** to determine the number of items in the C:\Program Files folder.

```
Option Explicit
Dim oShell32, oFolder, oFolderItem
Dim nCount
'--- Creating the shell application file
Set oShell32 = CreateObject("Shell.Application")
'--- Retrieving a folder object using the namespace method
Set oFolder = oShell32.Namespace(CSIDL_PROGRAM_FILES)
'--- Checking Folder validation
If not oFolder is nothing then
    '--- Retrieving all files items object
    Set oFolderItems = oFolder.Items
    '--- Checking file item validation
    If (Not oFolderItems Is Nothing) then
        '--- Retrieving the count information.
        nCount = oFolderItems.Count
        MsgBox nCount, 0, "Number of items under C:\Windows"
    End if
    Set oFolderItems = Nothing
End if
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```



Shell32.Folder.MoveHere Method

Description

The **MoveHere** method copies an item or items to a folder.

Syntax

```
object.MoveHere (vItem, vOptions)
```

Arguments

Parameter	Description
<i>vItem</i>	Required. Specifies the item or items to copy. This can be a string that represents a file name, a FolderItem object, or a FolderItems object.
<i>vOptions</i>	Optional. Specifies options for the copy operation. This value can be zero or a combination of the of the SHFILEOPSTRUCT structure values described in Table 5 on page 62

Note

- Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder (CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Example

The following example uses MoveHere to move the all the files in MyTemp from the root directory of the C:\ drive to the C:\MyTempFolder folder, with a progress bar.

```
Option Explicit
Private Const FOF_FILESONLY = 128
Private Const FOF_SIMPLEPROGRESS = 256
Dim sPath
Dim oShell32, oFolder
'--- Creating the shell application file
Set oShell32 = CreateObject("Shell.Application")
'--- Retrieving a folder object using the namespace method
Set oFolder = ShellApp.Namespace("C:\MyTempFolder")
'--- Checking Folder validation
If Not oFolder Is Nothing Then
    '--- Moving to oFolder
    sPath = "C:\Program Files\Mercury Interactive\MyTemp\*.*"
    oFolder.MoveHere sPath, FOF_FILESONLY + FOF_SIMPLEPROGRESS
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```

Shell32.Folder.NewFolder Method

Description

The **NewFolder** method creates a new folder.

Syntax

```
object.NewFolder (sName, vOptions)
```

Arguments

Parameter	Description
<i>sName</i>	Required. A string that specifies the name of the new folder.
<i>vOptions</i>	Optional. This value is not currently used.

Return Value

No return value.

Note

- Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder (CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Example

The following example uses **NewFolder** to create the new folder C:\TestFolder.

```
Option Explicit
Dim oShell32, oFolder
'--- Creating the shell application file
Set oShell32 = CreateObject("Shell.Application")
'--- Retrieving a folder object using the namespace method
Set oFolder = ShellApp.Namespace("C:\Windows")
'--- Checking Folder validation
If not oFolder is nothing then
    '--- Creating a new folder
    oFolder.NewFolder("TestFolder")
End if
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```

Shell32.Folder.ParseName Method

Description

The **ParseName** method creates and returns a **FolderItem** object that represents a specified item.

Syntax

```
object.ParseName (sName)
```

Arguments

Parameter	Description
<i>sName</i>	Required. A string that specifies the name of the item.

Return Value

An object reference to the **FolderItem** object.

Note

- ParseName** should not be used for virtual folders such as *My Documents*.
- Not all methods are implemented for all folders. For example, the **ParseName** method is not implemented for the Control Panel folder

(CSIDL_CONTROLS). If you attempt to call an unimplemented method, a 0x800A01BD (decimal 445) error is raised.

Shell32.Folder2.OfflineStatus Property

Description

The **OfflineStatus** property contains Contains the offline status of the folder.

Data Type

Integer that is set to one of the [Shell32.OfflineFolderStatus](#) values described in Table 4 on page 61

Note

- The property is read-only. The property has no default value.
- Offline Files must be enabled through **Folder** Options for **OfflineStatus** to work correctly. If the Offline Files option is not enabled, the property returns **OFS_INACTIVE**.

Example

The following example shows the proper use of **OfflineStatus**

```
Option Explicit
Dim oShell32, oFolder
Dim nReturn
'--- Creating the shell application object
Set oShell32 = CreateObject("Shell.Application")
'--- Retrieving a folder object using the namespace method
Set oFolder = oShell32.Namespace("\\server\share\folder")
'--- Checking Folder validation
If Not oFolder Is Nothing then
    nReturn = oFolder.OfflineStatus
End if
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```

Shell32.Folder2.Self Property

Description

The **Self** property contains Contains the offline status of the folder.

Data Type

Integer that is set to one of the [Shell32.OfflineFolderStatus](#) values described in Table 4 on page 61

Note

- The property is read-only. The property has no default value.
- Offline Files must be enabled through **Folder** Options for **OfflineStatus** to work correctly. If the Offline Files option is not enabled, the property returns **OFS_INACTIVE**.

Example

The following example uses **Self** to retrieve the **FolderItem** for the

C:\Windows folder.

```

Option Explicit
Private Const CSIDL_WINDOWS = 36
Dim oShell32, oFolder, oFolderItem
'--- Creating the shell application file
Set oShell32 = CreateObject("Shell.Application")
'--- Retrieving a folder object using the namespace method
Set oFolder = oShell32.Namespace(CSIDL_WINDOWS)
'--- Checking Folder validation
If Not oFolder Is Nothing Then
    Set oFolderItem = oFolder.Self
    If (Not oFolderItem Is Nothing) Then
        'Add code here.
    End if
End if
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing : Set oFolderItem = Nothing

```

Shell32.Folder2.DismissedWebViewBarricade Method

Description

The **DismissedWebViewBarricade** method Specifies that the Web view barricade has been dismissed by the user.

Syntax

```
object.DismissedWebViewBarricade ()
```

Note

- Call this method after the Web view barricade is dismissed by the user

Shell32.Folder2.Synchronize Method

Description

The **Synchronize** method synchronizes all offline files.

Syntax

```
object.Synchronize ()
```

Shell32.FolderItem Object

The **FolderItem** object represents an item in a **Shell** folder. It contains properties and methods that allow you to retrieve information about the item.

Shell32.FolderItem.GetFolder Property

Description

The **GetFolder** property contains the item's **Folder** object, If the item is a folder.

Note

- The property is read-only. The property has no default value.

Example

The following example uses **GetFolder** to retrieve the **Folder** object for the *System32* folder.

```
Option Explicit
Private Const CSIDL_WINDOWS = 36
Dim oShell32, oFolder, oFolderItem
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.NameSpace(CSIDL_WINDOWS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) Then
    '--- Retrieve the System32 folder item
    Set oFolderItem = oFolder.ParseName("system32")
    '--- Validation...
    If (Not oFolderItem Is Nothing) Then
        '--- Set Folder item to folder to activate folder methods
        Set oFolder = oFolderItem.GetFolder
        If (Not oFolder Is Nothing) Then
            '--- Add code here
        End If
        Set oFolder = Nothing
    End If
    Set oFolderItem = Nothing
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```

Shell32.FolderItem.GetLink Property

Description

The **GetLink** property contains the item's **ShellLinkObject** object, If the item is a shortcut

Note

- The property is read-only. The property has no default value.

Shell32.FolderItem.IsBrowsable Property

Description

The **IsBrowsable** property indicates if the item can be browsed.

Note

- The property is read-only. The property has no default value.
- Receives true if the item can be browsed or false if not.

Shell32.FolderItem.IsFileSystem Property

Description

The **IsFileSystem** property indicates if the item is part of the file system.

Note

- The property is read-only. The property has no default value.
- Receives true if the item is part of the file system or false if not.

Shell32.FolderItem.IsFolder Property

Description

The **IsFolder** property indicates if the item is a folder.

Data Type

Boolean that receives true if the item is a folder or false if not.

Note

- The property is read-only. The property has no default value.

Shell32.FolderItem.IsLink Property

Description

The **IsLink** property indicates if the item is a shortcut.

Data Type

Boolean that receives true if the item is a shortcut or false if not.

Note

- The property is read-only. The property has no default value.

Shell32.FolderItem.ModifyDate Property

Description

The **ModifyDate** property sets or retrieves the date and time that a file was last modified. **ModifyDate** can be used to retrieve the data and time that a folder was last modified, but cannot set it.

Data Type

Date that specifies or receives the date and time that the item was last modified.

Note

- The property is read/write. The property has no default value.

Shell32.FolderItem.Name Property

Description

The **Name** property sets or retrieves the item's name.

Data Type

String that specifies or receives the item's name.

 **Note**

- The property is read/write. The property has no default value.

Shell32.FolderItem.Parent Property

 **Description**

The **Parent** property contains the item's parent object.

Data Type

Object that receives the **Parent** object.

 **Note**

- The property is read-only. The property has no default value.

Shell32.FolderItem.Path Property

 **Description**

The **Path** property contains the item's full path and name.

Data Type

String that receives the item's full path and name.

 **Note**

- The property is read-only. The property has no default value.

Shell32.FolderItem.Size Property

 **Description**

The **Size** property contains the item's size, in bytes.

Data Type

Integer that receives the item's size.

 **Note**

- The property is read-only. The property has no default value.

Shell32.FolderItem.Type Property

 **Description**

The **Type** property contains a string representation of the item's type.

Data Type

String that receives the item's type.

 **Note**

- The property is read-only. The property has no default value.

Shell32.FolderItem.InvokeVerb Method

Description

The **InvokeVerb** method executes a verb on the item.

Syntax

```
object.InvokeVerb (vVerb)
```

Arguments

Parameter	Description
<i>vVerb</i>	Optional. A string that specifies the verb to be executed. It must be one of the values returned by the item's FolderItemVerb.Name property. If no verb is specified, the default verb will be invoked.

Return Value

No return value.

Note

- A verb is a string used to specify a particular action that an item supports. Invoking a verb is equivalent to selecting a command from an item's shortcut menu.
- Typically, invoking a verb launches a related application. For example, invoking the "open" verb on a .txt file opens the file with a text editor, usually Microsoft Notepad.
- The **FolderItemVerbs** object represents the collection of verbs associated with the item.
- The default verb may vary for different items, but it is typically "open".

Example1

The following example uses **InvokeVerb** to invoke the default verb ("open" in this case) on the Windows folder.

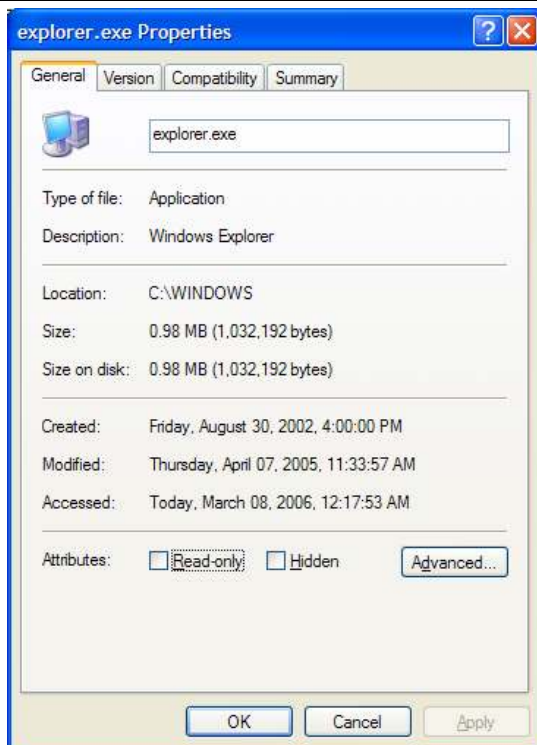
```
Option Explicit
Private Const CSIDL_WINDOWS = 36
Dim oShell32, oFolder, oFolderItem
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.Namespace(CSIDL_WINDOWS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) Then
    '--- Retrieve the FolderItem object
    Set oFolderItem = oFolder.Self
    '--- Validation...
    If (Not oFolderItem Is Nothing) Then
        '--- Invoking the default verb usually 'open'
        oFolderItem.InvokeVerb()
    End if
    Set oFolderItem = Nothing
End If
'--- Cleaning objects
```

```
Set oFolder = Nothing : Set oShell32 = Nothing
```

Example2

The following example uses **InvokeVerb** to invoke the verb "properties".

```
Option Explicit
Private Const CSIDL_WINDOWS = 36
Dim oShell32, oFolder, oFolderItem
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.NameSpace(CSIDL_WINDOWS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) Then
  '--- Retrieve the FolderItem object
  Set oFolderItem = oFolder.Self
  '--- Validation...
  If (Not oFolderItem Is Nothing) Then
    '--- Invoking the verb 'Properties'
    oFolderItem.InvokeVerb("Properties")
  End if
  Set oFolderItem = nothing
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```



Shell32.FolderItem.Verbs Method

Description

The **Verbs** method retrieves the item's **FolderItemVerbs** object. This object is

the collection of verbs that can be executed on the item.

Syntax

```
object.Verbs ()
```

Return Value

An object reference to the **FolderItemVerbs** object.

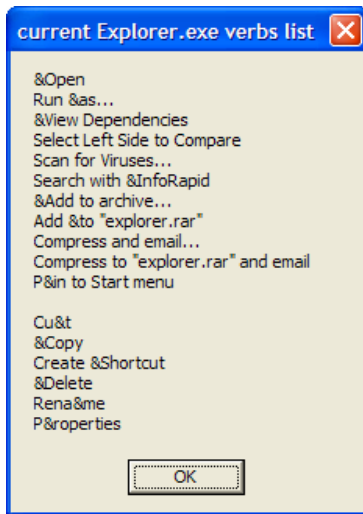
Note

- A verb is a string used to specify a particular action that an item supports. Invoking a verb is equivalent to selecting a command from an item's shortcut menu.
- Typically, invoking a verb launches a related application. For example, invoking the "open" verb on a .txt file opens the file with a text editor, usually Microsoft Notepad.
- The **FolderItemVerbs** object represents the collection of verbs associated with the item.
- The default verb may vary for different items, but it is typically "open".

Example

The following example uses Verbs to retrieve the **FolderItemVerbs** object representing the set of verbs that can be executed on the Windows folder.

```
Option Explicit
Private Const CSIDL_WINDOWS = 36
Dim oShell32, oFolder, oFolderItem, oFolderItemVerbs, oFolderItemVerb
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.NameSpace(CSIDL_WINDOWS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) then
    '--- Retrieve the FolderItem object
    Set oFolderItem = oFolder.ParseName("Explorer.exe")
    '--- Validation...
    If (Not oFolderItem Is Nothing) Then
        '--- invoking the context menu properties of the file
        Set oFolderItemVerbs = oFolderItem.Verbs
        '--- Looping the collection of oFolderItemVerbs
        For Each oFolderItemVerb In oFolderItemVerbs
            sVerbList = sVerbList & oFolderItemVerb.Name & vbNewLine
        Next
        '--- Cleaning objects
        Set oFolderItemVerb = Nothing : Set oFolderItemVerbs = Nothing
        MsgBox sVerbList,0, "current Explorer.exe verbs list"
    End if
    Set oFolderItem = Nothing
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```



Shell32.FolderItems, Shell.FolderItems2, Shell.FolderItems3 Objects

The **FolderItems** object represents the collection of items in a **Shell** folder. It contains properties and methods that allow you to retrieve information about the collection.

Shell32.FolderItems.Count Property

Description

The **Count** property contains the number of items in the collection.

Data Type

An Integer that contains a value for the **Count** property.

Note

- The property is read-only. The property has no default value.

Shell32.FolderItems.Item Method

Description

The **Item** method retrieves the **FolderItem** object for a specified item in the collection.

Syntax

```
object.Item(nIndex)
```

Arguments

Parameter	Description
<i>nIndex</i>	Optional. Specifies the zero-based index of the item to retrieve. This value must be less than the value of the Count property.

Return Value

An object reference to the **FolderItem** object.

Shell32.FolderItems2.InvokeVerbEx Method

Description

The **InvokeVerbEx** method executes a verb on a collection of **FolderItem** objects. This method is an extension of the **InvokeVerb** method, allowing additional control of the operation through a set of flags.

Syntax

```
object.InvokeVerbEx (vVerb, vArgs)
```

Arguments

Parameter	Description
<i>vVerb</i>	Optional. Variant with the verb string that corresponds to the command to be executed. If no verb is specified, the default verb is executed.
<i>vArgs</i>	Optional. Variant that consists of a string with one or more arguments to the command specified by <i>vVerb</i> . The format of this string depends on the particular verb.

Return Value

No return value.

 **Note**

- A verb is a string used to specify a particular action associated with an item or collection of items. Typically, calling a verb launches a related application.
- For example, calling the open verb on a .txt file normally opens the file with a text editor, usually Microsoft Notepad.

Shell32.FolderItems3.Verbs Property

 **Description**

The **Verbs** property retrieves the list of verbs common to all the folder items.

Data Type

Collection of **FolderItemVerbs** to be returned.

 **Note**

- The property is read-only. The property has no default value.

Shell32.FolderItems3.Filter Method

 **Description**

The **Filter** method sets a wildcard filter to apply to the items returned.

Syntax

```
object.Filter (grfFlags, bstrFilter)
```

Arguments

Parameter	Description
<i>grfFlags</i>	Required. This parameter can be one of the flags listed in SHCONTF Enumerated Type Values .
<i>bstrFilter</i>	Required. Contains a filter string.

Return Value

No return value.

Note

- For a list [SHCONTF Enumerated Type Values](#) of see Table 9 on page 66

Example

The following example uses Item to retrieve the **FolderItem** object representing the Windows folder and filtering the content

```
Option Explicit
Private Const CSIDL_WINDOWS = 36
Private Const SHCONTF_NONFOLDERS = 64
Dim oShell32, oFolder, oFolderItem, oFolderItems
Dim nCount
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.NameSpace(CSIDL_WINDOWS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) then
    '--- Retrieve the FolderItems collection object
    Set oFolderItems = oFolder.Items()
    '--- Validation...
    If (Not oFolderItems Is Nothing) Then
        nCount = oFolderItems.Count
        MsgBox "Items before Filter = " & nCount, 0, "Filter Method"
        oFolderItems.Filter SHCONTF_NONFOLDERS, "*.exe"
        nCount = oFolderItems.Count
        MsgBox "Items after Filter = " & nCount, 0, "Filter Method"
    End if
    Set oFolderItem = Nothing : Set oFolderItems = Nothing
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```



Shell32.FolderItemVerb Object

The **FolderItemVerb** object represents a single verb available to an item. The object contains properties and methods that allow you to retrieve information about the verb.

Each verb corresponds to the command that would be used to launch the application from a console window. The open verb is a good example, as it is commonly supported. For .exe files, open simply launches the application. However, it is more commonly used to launch an application that operates on a particular file. For instance, .txt files can be opened by Microsoft WordPad. The open verb for a .txt file would thus correspond to something like the following command:

```
"C:\Program Files\Windows NT\Accessories\Wordpad.exe" "%1"
```

When you use ShellExecute or ShellExecuteEx to open a .txt file, Wordpad.exe is launched with the specified file as its argument. Some commands can have additional arguments, such as flags, that can be added as needed to launch the application properly. For further discussion of shortcut menus and verbs, see

Shell32.FolderItemVerb.Name Property

Description

The **Name** property contains the verb's name.

Data Type

String that receives the **Name** property.

Note

- The property is read-only. The property has no default value.

Example

The following example uses **Name** property to retrieve the name of the first item in the collection of verbs to which the user's Program folder responds.

```
Option Explicit
Private Const CSIDL_PROGRAMS = 2
Dim oShell32, oFolder, oVerbs, oVerb
Dim sVerbList
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.Namespace(CSIDL_WINDOWS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) then
    '--- Retrieve the Verbs collection object
    Set oVerbs = oFolder.Self.Verbs
    If (Not oVerbs Is Nothing) then
        '--- This method implements the For each loop
        For Each oVerb in oVerbs
            sVerbList = sVerbList & oVerb.Name & vbNewLine
        Next
        MsgBox sVerbList, 0, "Verbs List"
```

```

'--- Resetting variable
sVerbsList = Empty
'--- This method implemets the For i to Count loop
For i = 0 To oVerbs.Count - 1
    sVerbsList = sVerbsList & oVerbs.Item(i).Name & vbNewLine
Next
Msgbox sVerbsList, 0, "Verbs List"
End If
Set oVerb = Nothing : Set oVerbs = Nothing
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing

```

Shell32.FolderItemVerb.DoIt Method

Description

The **DoIt** method Executes a verb on the **FolderItem** associated with the verb.

Syntax

```
object.DoIt ()
```

Return Value

No return value.

Shell32.FolderItemVerbs Object

The **FolderItemVerbs** object represents the collection of verbs for an item in a Shell folder. It contains properties and methods that allow you to retrieve information about the collection.

The verbs available for an object are essentially the items that you find on an object's shortcut menu. To find which verbs are available, look in the registry under

HKEY_CLASSES_ROOT\CLSID\{object_clsid}\Shell\verb

where `object_clsid` is the class identifier (**CLSID**) of the object, and `verb` is the name of the available verb. The `verb\command` subkey contains the data indicating what happens when that verb is invoked.

Commonly available verbs include:

Verb	Description
edit	Launches an editor and opens the document for editing.
find	Initiates a search starting from the specified directory.
open	If this file is not an executable file, its associated application is launched.
print	Prints the document file.
properties	Displays the object's properties.

Shell32.FolderItemVerbs.Count Property

Description

The **Count** property contains the number of items in the collection.

Data Type

Integer that receives the **Count** property.

Note

- The property is read-only. The property has no default value.

Shell32.FolderItemVerbs.Item Method

Description

The **Item** method retrieves the **FolderItemVerb** object for a specified item in the collection.

Syntax

```
object.Item (nIndex)
```

Arguments

Parameter	Description
<i>nIndex</i>	Required. Specifies the zero-based index of the item to retrieve. This value must be less than the value of the Count property

Return Value

Object that receives the **FolderItemVerb** object.

Shell32.IshellDispatch Interface and Shell Object

The **Shell** object represents the objects in the **Shell**. Methods are provided to control the **Shell** and to execute commands within the **Shell**. There are also methods to obtain other **Shell**-related objects.

Shell32.Shell.Parent Property

Description

The **Parent** property contains the object's parent object.

Data Type

Object that receives the **Parent** object.

Note

- The property is read-only. The property has no default value.

Shell32.AddToRecent Method

Description

The **AddToRecent** method determines if the current user can start and stop the named service.

Syntax

```
object.AddToRecent (varFile, bstrCategory)
```

Arguments

Parameter	Description
<i>varFile</i>	Required. Variant that specifies the file to add to the list of recently used documents.
<i>bstrCategory</i>	Optional. String that contains the name of the category in which to place the file.

Return Value

No return value..

Shell32.CanStartStopService Method

Description

The **CanStartStopService** method determines if the current user can start and stop the named service.

Syntax

```
object.CanStartStopService (sServiceName)
```

Arguments

Parameter	Description
<i>sServiceName</i>	Required. String that contains the name of the service.

Return Value

Returns **True** if the user can start and stop the service, or **False** otherwise.

Example

The following example shows the proper usage of CanStartStopService inside a public function

```
Public Function fCanStartStopService(ByVal sServiceName)
    Dim oShell32
    Set oShell32 = CreateObject("Shell.Application")
    fCanStartStopService = oShell32.CanStartStopService(sServiceName)
    Set oShell32 = nothing
End function
```

Shell32.CascadeWindows Method

Description

The **CascadeWindows** method Cascades all of the windows on the desktop. This method has the same effect as right-clicking the taskbar and selecting Cascade Windows.

Syntax

```
object.CascadeWindows ()
```

Return Value

No return value.

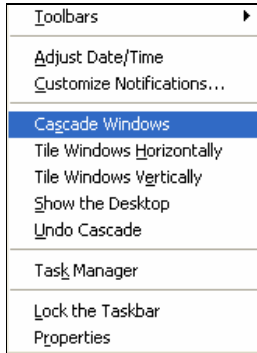


Figure 2 - Cascade Windows

Shell32.ControlPanelItem Method

Description

The **ControlPanelItem** method runs the specified **Control Panel** application. If the application is already open, it will activate the running instance.

Syntax

```
object.ControlPanelItem (sDir)
```

Arguments

Parameter	Description
<i>sDir</i>	Required. Specifies the Control Panel application's file name. All Control Panel applications have the .cpl extension.

Return Value

No return value.

Tip



Example

The following example uses **ControlPanelItem** to run the Control Panel's Display Properties item.

```
Option Explicit
Dim oShell32
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
oShell32.ControlPanelItem("desk.cpl")
'--- Cleaning objects
Set oShell32 = Nothing
```



Figure 3 - Display Properties item

Shell32.EjectPC Method

Description

The **EjectPC** method ejects the computer from its docking station. This is the same as clicking the Start menu and selecting Eject PC, if your computer supports this command.

Syntax

```
object.EjectPC()
```

Return Value

No return value.

Shell32.Explore Method

Description

The **Explore** method opens a specified folder in a Microsoft Windows Explorer window.

Syntax

```
object.Explore(vDir)
```

Arguments

Parameter	Description
<i>vDir</i>	Required. Specifies the folder to be displayed. This can be a string that specifies the path of the folder or one of the ShellSpecialFolderConstants

values.

Example

The following example shows Explore in use.

```

Option Explicit
Private Const CSIDL_PERSONAL = 5
Dim oShell32
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
oShell32.Explore(CSIDL_PERSONAL)
'--- Cleaning objects
Set oShell32 = Nothing

```

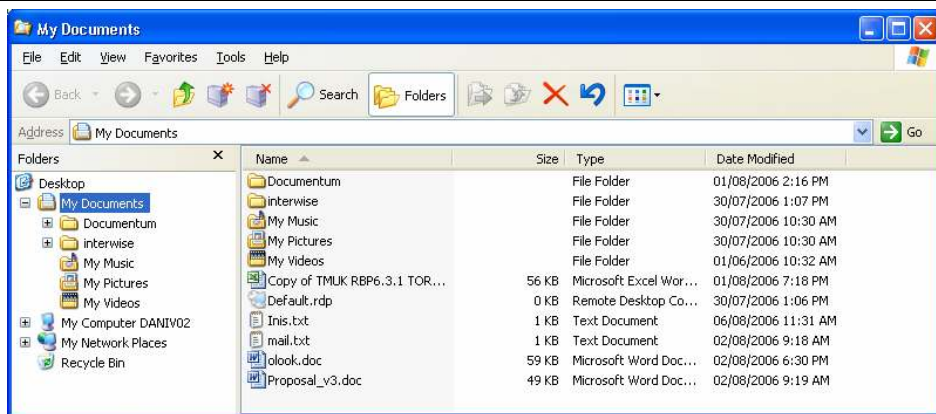


Figure 4 - Explore Window

Shell32.ExplorerPolicy Method

Description

The **ExplorerPolicy** method retrieves the value of a Microsoft Internet Explorer policy.

Syntax

```
object.ExplorerPolicy (bstrPolicyName)
```

Arguments

Parameter	Description
<i>bstrPolicyName</i>	Required. String that specifies the name of the Internet Explorer policy.

Return Value

Variant that receives the value of the specified Internet Explorer policy.

Note

- Network Administrators can control and manage the computing environment of their users by setting policies.

Shell32.FileRun Method

Description

The **Explore** method opens a specified folder in a Microsoft Windows Explorer window.

Syntax

```
object.FileRun ()
```

Return Value

No return value.

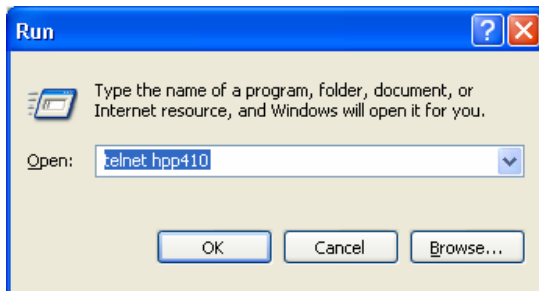


Figure 5 - File Run Dialog

Shell32.FindComputer Method

Description

The **FindComputer** method Displays the Search Results - Computers dialog box. The dialog box shows the result of the search for a specified computer.

Syntax

```
object.FindComputer ()
```

Return Value

No return value.

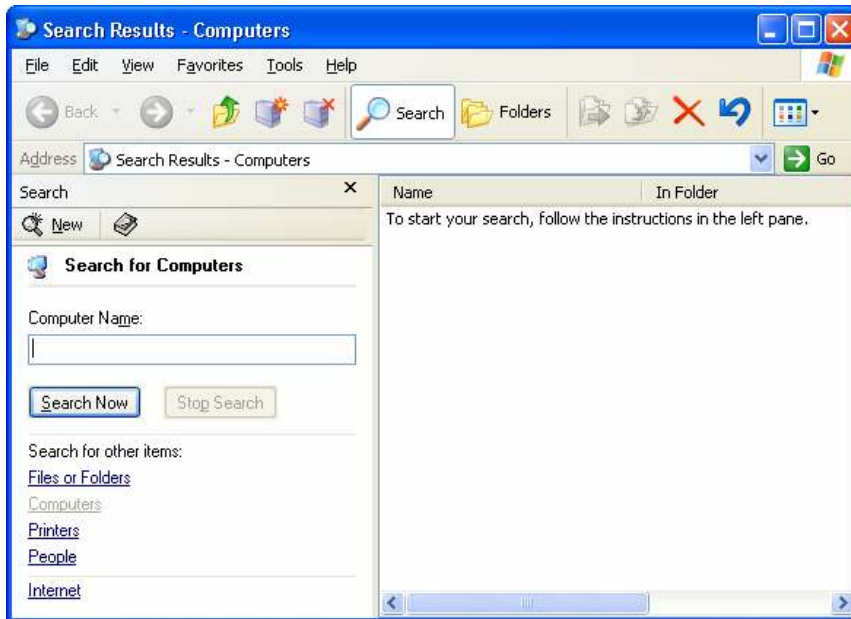


Figure 6 - Find Computer Dialog

Shell32.FindFiles Method

Description

The **FindFiles** method Displays the Find: All Files dialog box. This is the same as clicking the Start menu, selecting Find, and then selecting Files or Folders.

Syntax

```
object.FindFiles ()
```

Return Value

No return value.

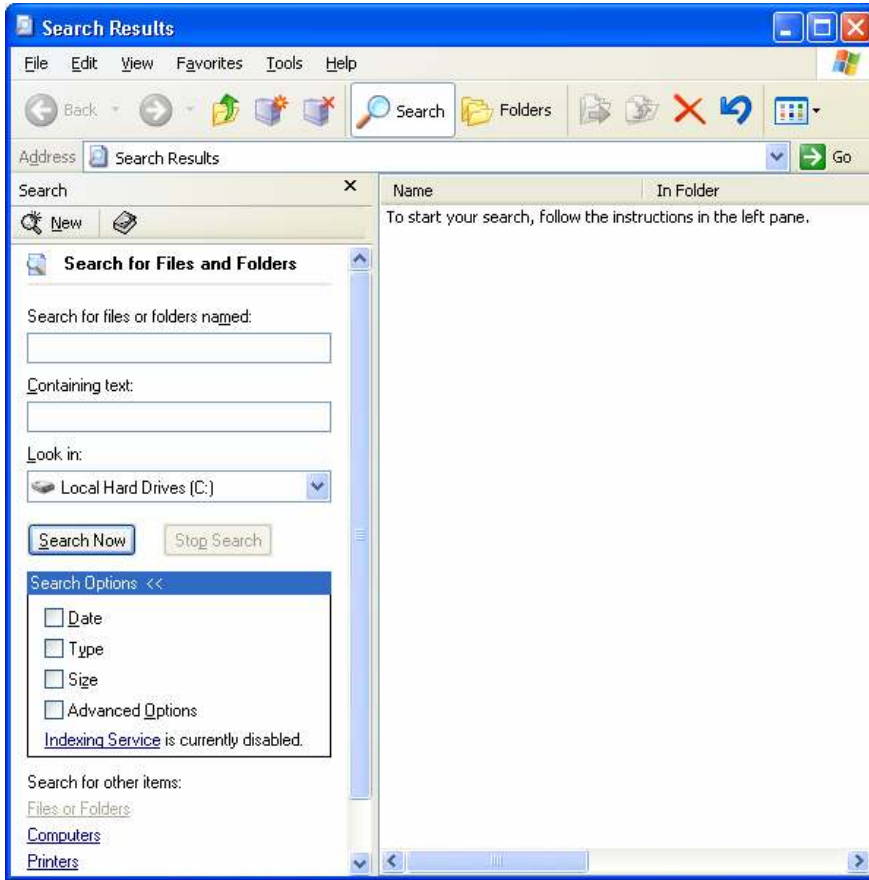


Figure 7 - Find Files Dialog

Shell32.FindPrinter Method

Description

The **FindPrinter** method displays the Find Printer dialog box to allow the user to find a printer.

Syntax

```
object.FindPrinter (sName, sLocation, sModel)
```

Arguments

Parameter	Description
<i>sName</i>	Optional. String that contains the printer name.
<i>sLocation</i>	Optional. String that contains the printer location.
<i>sModel</i>	Optional. String that contains the printer model.

Return Value

No return value.

Note

- If you assign strings to one or more of the optional parameters, they are

displayed as default values in the associated edit control when the Find Printer dialog box is displayed.

- The user can either accept or override these values. If no value is assigned to a parameter, the associated edit box is empty and the user must enter a value.

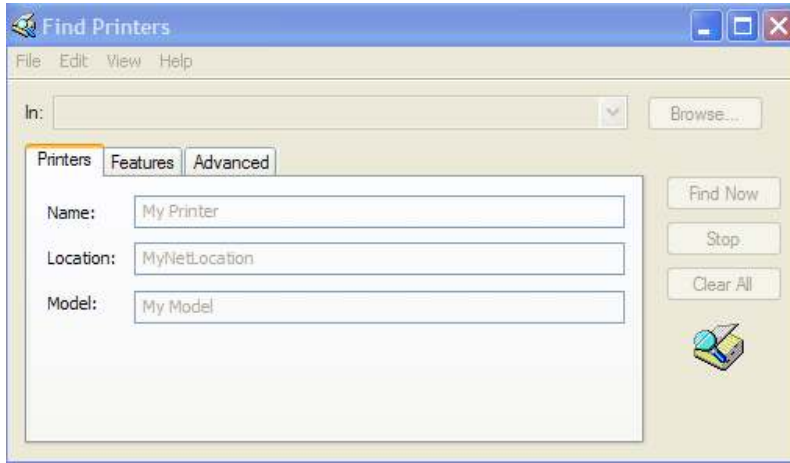


Figure 8 - Find Printer dialog

Shell32.GetSetting Method

Description

The **GetSetting** method retrieves a Shell global setting.

Syntax

```
object.GetSetting (nSetting)
```

Arguments

Parameter	Description
<i>nSetting</i>	Required. A value of type Integer that specifies the current Shell settings to retrieve. The SHGetSetSettings Enumerated Type described on Table 10 on page 66.

Return Value

Boolean that returns true if the setting exists, or false otherwise.

Shell32.GetSystemInformation Method

Description

The **GetSystemInformation** method retrieves a Shell global setting.

Syntax

```
object.GetSystemInformation (sName)
```

Arguments

Parameter	Description
-----------	-------------

<i>sName</i>	Required. String that specifies the system info that is being requested.
--------------	--

Return Value

Returns the requested system information value. The return type depends on what system information is requested.

Note

- This method can be used to request a number of system information values.
- The following table gives the *sName* value that is used to request the information and the type of the returned value.

sName	Return Type	Description
DirectoryServiceAvailable	Boolean	Set to true if directory service is available, or false otherwise.
DoubleClickTime	Integer	The double-click time, in milliseconds.
ProcessorLevel	Integer	The processor level. Returns 3, 4, or 5, for x386, x486, and Pentium-level processors, respectively.
ProcessorSpeed	Integer	The processor speed, in megahertz (MHz).
ProcessorArchitecture	Integer	The processor architecture. For details, see the discussion of the wProcessorArchitectureType SYSTEM_INFO Values on Table 11 on page 66
PhysicalMemoryInstalled	Integer	Amount of physical memory installed, in bytes.

Shell32.Help Method

Description

The **Help** method displays the Microsoft Windows Help and Support Center. This method has same effect as clicking the Start menu and selecting Help and Support.

Syntax

```
object.Help ()
```

Return Value

No return value.

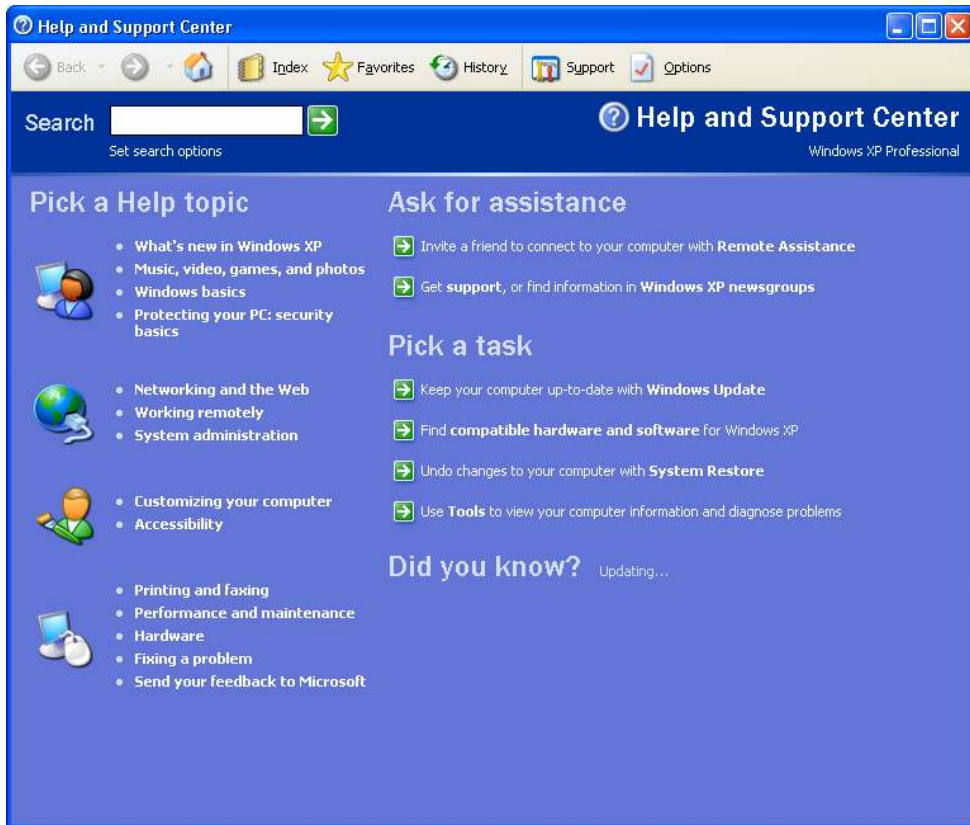


Figure 9 – Help Dialog

Shell32.IsRestricted Method

Description

The **IsRestricted** method retrieves the registry's data value for a given group's restriction value.

Syntax

```
object.IsRestricted (sGroup, sRestriction)
```

Arguments

Parameter	Description
<i>sGroup</i>	Required. The group (key) name under which to check for the restriction.
<i>sRestriction</i>	Required. The restriction whose data value is to be retrieved.

Return Value

The value of the restriction. If the specified restriction is not found, the return value is 0.

Note

- **IsRestricted** first looks for a registry key name matching *sGroup* under the following key.
 HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\policies

Shell32.IsServiceRunning Method

Description

The **IsServiceRunning** method returns a value that indicates whether a named service is running.

Syntax

```
object.IsServiceRunning (sServiceName)
```

Arguments

Parameter	Description
<i>sServiceName</i>	Required. String that contains the service's name.

Return Value

Returns true if the service specified by *sServiceName* is running, or false otherwise.

Example

The following example shows the proper usage of **IsServiceRunning** inside a public function

```
Public Function fIsServiceRunning (ByVal sServiceName)
    Dim oShell32
    Set oShell32 = CreateObject("Shell.Application")
    fIsServiceRunning = oShell32.IsServiceRunning(sServiceName)
    Set oShell32 = nothing
End function
```

Shell32.MinimizeAll Method

Description

The **Help** method Minimizes all of the windows on the desktop. This method has the same effect as right-clicking the taskbar and selecting Minimize All Windows on older systems or clicking the Show Desktop icon in the Quick Launch area of the taskbar in Microsoft Windows 2000 or Windows XP.

Syntax

```
object.MinimizeAll ()
```

Return Value

No return value.



Figure 10 – MinimizeAll Icon

Shell32.NameSpace Method

Description

The **NameSpace** method creates and returns a **Folder** object for the specified folder.

Syntax

```
object.NameSpace (vDir)
```

Arguments

Parameter	Description
<i>vDir</i>	Required. Specifies the folder for which to create the Folder object. This can be a string that specifies the path of the folder or one of the Shell32.ShellSpecialFolderConstants values in Table 7–Shell32.ShellSpecialFolderConstants Values on page 65.

Return Value

Object reference to the Folder object for the specified folder. If the folder is not successfully created, this value returns null.

Example

The following example shows **NameSpace** in use.

```
Public Function fGetShellNameSpace(ByVal vDir, ByRef sTitle)
    Dim oShell32, oFolder
    Set oShell32 = CreateObject("Shell.Application")
    Set oFolder = objShell.NameSpace(vDir)
    If (Not objFolder Is Nothing) then
        sTitle = oFolder.Title
    End If
    set oFolder = Nothing
    set oShell32 = Nothing
End function
```

Shell32.Open Method

Description

The **Open** method opens the specified folder.

Syntax

```
object.Open (vDir)
```

Arguments

Parameter	Description
<i>vDir</i>	Required. Specifies the folder for which to create the Folder object. This can be a string that specifies the path of the folder or one of the Shell32.ShellSpecialFolderConstants values in Table 7–Shell32.ShellSpecialFolderConstants Values on page 65.

Return Value

No return value.

Note

If *vDir* is set to one of the [Shell32.ShellSpecialFolderConstants](#) and the special folder does not exist, this function will create the folder.

Tip

Same as command-line "Explorer c:\Windows"

Shell32.RefreshMenu Method

Description

The **RefreshMenu** method refreshes the contents of the Start menu. Used only with systems preceding Microsoft Windows XP.

Syntax

```
object.RefreshMenu ()
```

Return Value

No return value.

Remarks

The functionality that RefreshMenu provides is handled automatically under Windows XP or later. Do not call this method under that operating system.

Shell32.ServiceStop Method

Description

The **ServiceStop** method stops a named service.

Syntax

```
object.ServiceStop (sServiceName, vPersistent)
```

Arguments

Parameter	Description
<i>sServiceName</i>	Required. String that contains the service's name.
<i>vPersistent</i>	Required. Variant that is set to true to have the service started by the service control manager when ServiceStart is called. To leave the service configuration unchanged, set <i>vPersistent</i> to false.

Return Value

Returns true if successful, or false otherwise.

Shell32.ShellExecute Method

Description

The **ShellExecute** method stops a named service.

Syntax

```
object.ShellExecute (sFile, vArguments, vDirectory, vOperation, vShow)
```

Arguments

Parameter	Description
<i>sFile</i>	Required. String that contains the name of the file on which ShellExecute will perform the action specified by <i>vOperation</i> .
<i>vArguments</i>	Optional. Variant that contains the parameter values for the operation.
<i>vDirectory</i>	Optional. Variant that contains the fully qualified path of the directory that contains the file specified by <i>sFile</i> . If this parameter is not specified, the current working directory is used.
<i>vOperation</i>	Optional. Variant that specifies the operation to be performed. It should be set to one of the verb strings that is supported by the file. For a discussion of verbs, see the Remarks section. If this parameter is not specified, the default operation is performed.
<i>vShow</i>	Optional. Variant that recommends how the window that belongs to the application that performs the operation should be displayed initially. The application can ignore this recommendation.

Note

vShow can take one of the following values. If this parameter is not specified, the application uses its default value.

Value	Description
0	Open the application with a hidden window.
1	Open the application with a normal window. If the window is minimized or maximized, the system restores it to its original size and position.
2	Open the application with a minimized window.
3	Open the application with a maximized window.
4	Open the application with its window at its most recent size and position. The active window remains active.
5	Open the application with its window at its current size and position.
7	Open the application with a minimized window. The active window remains active.
10	Open the application with its window in the default state specified by the application.

Remarks

- This method is equivalent to launching one of the commands associated with a file's shortcut menu. Each command is identified by a verb string. The supported verbs vary from file to file. The most commonly supported verb is "open", which is also usually the default verb. Others might be supported only by certain types of files.

Shell32.SetTime Method

Description

The **SetTime** method Displays the Date and Time Properties dialog box. This

method has the same effect as right-clicking the clock in the taskbar status area and selecting Adjust Date/Time.

Syntax

```
object.SetTime ( )
```

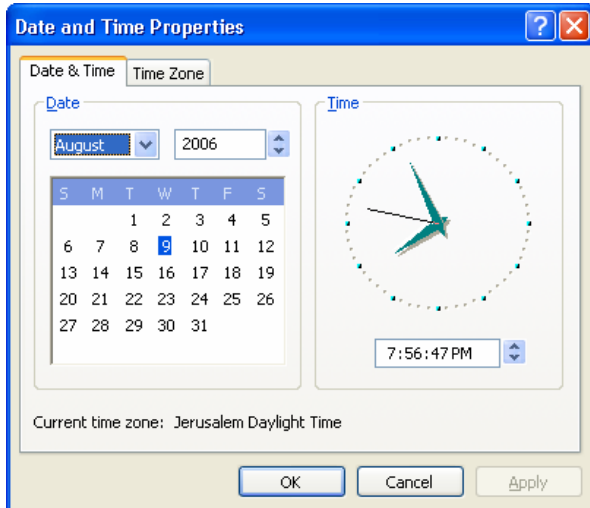


Figure 11 - Date and Time Properties Dialog

Shell32.ShowBrowserBar Method

Description Not Supported By QuickTest Professional

The **ShowBrowserBar** method stops a named service.

Syntax

```
object.ShowBrowserBar ( sCLSID, vShow )
```

Arguments

Parameter	Description
<i>sCLSID</i>	Required. String that contains the string form of the class identifier (CLSID) of the browser bar to be displayed. The object must be registered as an Explorer Bar object with a CATID_InfoBand component category.
<i>vShow</i>	Required. Variant that is set to true to show the browser bar, or false to hide it.

Return Value

Variant. Returns true if successful, or false otherwise.

Note

You can display one of the standard Explorer Bars by setting *sCLSID* to its

CLSID string. The standard Explorer Bars and their **CLSID** strings are as follows.

Explorer Bar	CLSID string
Favorites	{EFA24E61-B078-11d0-89E4-00C04FC9E26E}
Folders	{EFA24E64-B078-11d0-89E4-00C04FC9E26E}
History	{EFA24E62-B078-11d0-89E4-00C04FC9E26E}
Search	{30D02401-6A81-11d0-8274-00C04FD5AE38}

Shell32.ShutdownWindows Method

Description

Displays the Shut Down Windows dialog box. This is the same as clicking the Start menu and selecting Shut Down.

Syntax

```
object.ShutdownWindows ()
```

Return Value

No return value.



Figure 12 – Suht Down Windows Dialog

Shell32.TileHorizontally Method

Description

The **TileHorizontally** method Tiles all of the windows on the desktop horizontally. This method has the same effect as right-clicking the taskbar and selecting Tile Windows Horizontally.

Syntax

```
object.TileHorizontally()
```

Return Value

No return value.

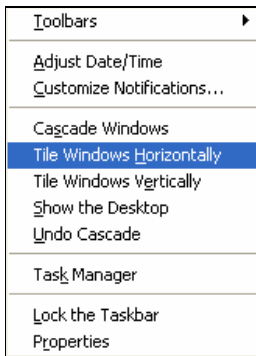


Figure 13 – Tile Windows Horizontally

Shell32.TileVertically Method

Description

The **TileHorizontally** method Tiles all of the windows on the desktop vertically. This method has the same effect as right-clicking the taskbar and selecting Tile Windows Vertically..

Syntax

```
object.TileVertically ( )
```

Return Value

No return value.

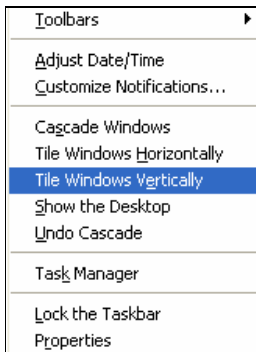


Figure 14 – Tile Windows Vertically

Shell32.ToggleDesktop Method

Description

The **ToggleDesktop** method raises and lowers the desktop.

Syntax

```
object.ShowBrowserBar ( )
```

Return Value

No return value.

Remarks

- This method behaves like the toggle desktop icon on the quick launch bar. It hides all open windows and shows the desktop, or hides the desktop and shows all open windows.
- The **ToggleDesktop** method does not display any user interface, it just invokes the toggle action.
- This method is similar to **MinimizeAll**



Figure 15 - Show Desktop

Shell32.TrayProperties Method

Description

The **TrayProperties** method Displays the Taskbar and Start Menu Properties dialog box. This method has the same effect as right-clicking the taskbar and selecting Properties.

Syntax

```
object.TrayProperties ()
```

Return Value

No return value.

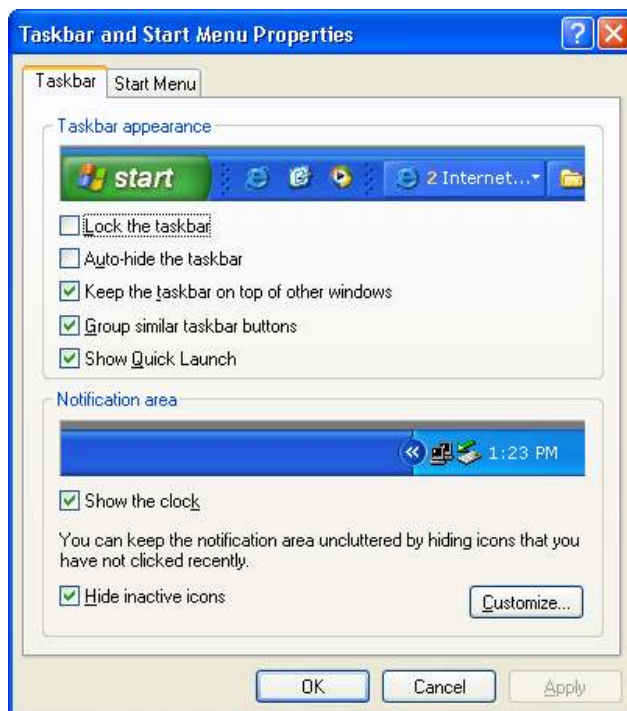


Figure 16 – Task

Shell32.UndoMinimizeAll Method

Description

The **UndoMinimizeAll** method Restores all of the windows on the desktop to the same state they were in before the last MinimizeAll command. This method has the same effect as right-clicking the taskbar and selecting Undo Minimize All Windows on older systems or a second clicking of the Show Desktop icon in the Quick Launch area of the taskbar in Microsoft Windows 2000 or Windows XP.

Syntax

```
object.UndoMinimizeAll ()
```

Return Value

No return value.

Shell32.Windows Method

Description

The **Windows** method Creates and returns a **ShellWindows** object. This object represents a collection of all of the open windows that belong to the Shell.

Syntax

```
object.Windows ()
```

Return Value

Object reference to the **ShellWindows** object.

Shell32.WindowsSecurity Method

Not Supported By QuickTest Professional

Figure 17 - Show Desktop

Description

The **WindowsSecurity** method displays the Windows Security dialog box.

Syntax

```
object.WindowsSecurity ( )
```

Return Value

No return value.

Remarks

- Using this method provides the same results as pressing CTRL+ALT+DELETE or using the security option that is on the Start menu when you connect using Microsoft Terminal Server.
- This method will work only when connected via a terminal session to

Microsoft Terminal Server.

Shell32.ShellLinkObject Object

The **Shell** object represents the objects in the **Shell**. Methods are provided to control the **Shell** and to execute commands within the **Shell**. There are also methods to obtain other **Shell**-related objects.

Shell32.ShellLinkObject.Target Property

Description

The **Target** property contains the link object's target.

Data Type

Object expression that evaluates to the target's **FolderItem** object.

Note

The property is read-only. The property has no default value.

Shell32.ShellFolderItem Object

This object extends the **FolderItem** object. In addition to the properties and methods supported by **FolderItem**, **ShellFolderItem** has two additional methods.

Shell32.ShellFolderItem.ExtendedProperty Method

Description

The **ExtendedProperty** method returns the value of a property from an item's property set. The property can be specified either by name or by the property set's format identifier (**FMTID**) and property identifier (**PID**).

Syntax

```
object.ExtendedProperty ( sPropName )
```

Arguments

Parameter	Description
<i>sPropName</i>	Required. String value that specifies the property. See the Remarks section for details.

Return Value

Variant. Returns the value of the property, if it exists for the specified item. The value will have full typing for example, dates will be returned as dates, not strings.

Remarks

- This method returns a zero-length string if the property is valid but does not exist for the specified item, or an **OLE** error code otherwise.
- There are two ways to specify a property. The first is to assign the property's

well-known name, such as "Author" or "Date", to *sPropName*. However, each property is a member of a **Component Object Model (COM)** property set and can also be identified by specifying its format ID (**FMTID**) and property ID (**PID**). An **FMTID** is a **GUID** that identifies the property set, and a **PID** is an integer that identifies a particular property within the property set.

- Specifying a property by its **FMTID/PID** values is usually more efficient than using its name. To use a property's **FMTID/PID** values with **ExtendedProperty**, they must be combined into an **SCID**. An **SCID** is a string that contains the **FMTID/PID** values in the form "FMTIDPID", where the **FMTID** is the string form of the property set's globally unique identifier (**GUID**).
- For a list of FMTIDs and PIDs that are currently supported by the Shell, see [SHCOLUMNID](#) values in Table 12 on page 68.

Example

The following example uses `Item` to retrieve the extended properties of a document file using the **ExtendedProperty** method

```
Option Explicit
Private Const CSIDL_WINDOWS = 36
Private Const SHCONTF_NONFOLDERS = 64
Dim oShell32, oFolder, oFolderItem, oFolderItems
Dim nCount
'--- Creating a shell application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.Namespace(CSIDL_WINDOWS)
'--- Checking Folder object validation
If (Not oFolder Is Nothing) then
  '--- Retrieve the FolderItems collection object
  Set oFolderItems = oFolder.Items()
  '--- Validation...
  If (Not oFolderItems Is Nothing) Then
    nCount = oFolderItems.Count
    MsgBox "Items before Filter = " & nCount, 0, "Filter Method"
    oFolderItems.Filter SHCONTF_NONFOLDERS, "*.exe"
    nCount = oFolderItems.Count
    MsgBox "Items after Filter = " & nCount, 0, "Filter Method"
  End If
  Set oFolderItem = Nothing : Set oFolderItems = Nothing
End If
'--- Cleaning objects
Set oFolder = Nothing : Set oShell32 = Nothing
```

Q&A

How to list items in the administrative tools folder?

```
Option Explicit
Const CSIDL_COMMON_ADMINTOOLS = &H2f
Dim oShell, oFolder, oItem
Dim nRow : nRow = 1
```

```
'--- Creating an entry in local datasheet
DataTable.LocalSheet.AddParameter "ITEM_NAME", ""
'--- Creating a new shell application object
Set oShell = CreateObject("Shell.Application")
Set oFolder = oShell.Namespace(CSIDL_COMMON_ADMINTOOLS)
'--- Retrieving collection of items
For Each oItem in oFolder.Items
    DataTable.LocalSheet.SetCurrentRow nRow
    DataTable("ITEM_NAME", dtLocalSheet) = oItem.Name
    nRow = nRow + 1
Next
Set oShell = Nothing
```

	ITEM_NAME
1	Component Services
2	Computer Management
3	Data Sources (ODBC)
4	Event Viewer
5	Local Security Policy
6	Microsoft .NET Framework 1.1 Configuration
7	Microsoft .NET Framework 1.1 Wizards
8	Performance
9	Services

How to list items in the "My Music" folder?

```
Option Explicit
Const CSIDL_MYMUSIC = 13
Dim oShell, oFolder, oItem
Dim nRow : nRow = 1
'--- Creating an entry in local datasheet
DataTable.LocalSheet.AddParameter "ITEM_NAME", ""
'--- Creating a new shell application object
Set oShell = CreateObject("Shell.Application")
Set oFolder = oShell.Namespace(CSIDL_MYMUSIC)
'--- Retrieving collection of items
For Each oItem in oFolder.Items
    DataTable.LocalSheet.SetCurrentRow nRow
    DataTable("ITEM_NAME", dtLocalSheet) = oItem.Name
    nRow = nRow + 1
Next
Set oShell = Nothing
```

How to retrieve the path to "Common Desktop" folder?

```
Option Explicit
Const CSIDL_COMMON_DESKTOPDIR = 25
Dim oShell, oFolder, oFolderItem
Dim nRow : nRow = 1
'--- Creating an entry in local datasheet
DataTable.LocalSheet.AddParameter "ITEM_NAME", ""
'--- Creating a new shell application object
Set oShell = CreateObject("Shell.Application")
```

```

Set oFolder = oShell.Namespace(CSIDL_COMMON_DESKTOPDIR)
Set oFolderItem = oFolder.Self
MsgBox oFolderItem.Path
Set oFolderItem = Nothing : Set oShell = Nothing

```

How to list the local computer information?

```

Option Explicit
Dim oComp
Dim sInfo
Set oComp = CreateObject("Shell.LocalMachine")
sInfo = "Computer name: " & oComp.MachineName & vbCrLf
sInfo = sInfo & "Shutdown allowed: " & oComp.IsShutdownAllowed & vbCrLf
sInfo = sInfo & "Friendly UI enabled: " & oComp.IsFriendlyUIEnabled & vbCrLf
sInfo = sInfo & "Guest access mode: " & oComp.IsGuestAccessMode & vbCrLf
sInfo = sInfo & "Guest account enabled: " & oComp.IsGuestEnabled(0) & vbCrLf
sInfo = sInfo & "Multiple users enabled: " & oComp.IsMultipleUsersEnabled & vbCrLf
sInfo = sInfo & "Offline files enabled: " & oComp.IsOfflineFilesEnabled & vbCrLf
sInfo = sInfo & "Remote conn. enabled: " & oComp.IsRemoteConnectionsEnabled & vbCrLf
sInfo = sInfo & "Undock enabled: " & oComp.IsUndockEnabled & vbCrLf
MsgBox sInfo, 0, "Local Computer Information"
Set oComp = Nothing

```



How to add a web site to the favorites menu?

```

Option Explicit
Const CSIDL_FAVORITES = 6
Set oShell132, oShell, oFolder, oFolderItem, oURLShortcut
Dim sDesktopFld
'--- Creating a application object
Set oShell132 = CreateObject("Shell.Application")
Set oFolder = oShell132.Namespace(CSIDL_FAVORITES)
'--- Using the favorites folder for the path.
Set oFolderItem = oFolder.Self
sDesktopFld = oFolderItem.Path
'--- Creating a shell folder to use the URLShortcut object
Set oShell = CreateObject("WScript.Shell")
Set oURLShortcut = oShell.CreateShortcut(sDesktopFld & "\MSDN Site.url")
oURLShortcut.TargetPath = "http://msdn.microsoft.com"
oURLShortcut.Save

```

```
'--- Cleaning
Set oURLShortcut = Nothing : Set oShell = Nothing : Set oComp = Nothing
```

Appendix 14.A – Shell32 Constants

QuotaStateConstants Values

Constant	Value	Description
dqStateDisable	0	Disk quotas are disabled.
dqStateTrack	1	Disk quotas are disabled.
dqStateEnforce	2	Enforce quota limit.

Table 1 – QuotaStateConstants Values

UserNameResolutionConstants Values

Constant	Val	Description
dqResolveNone	0	Do not resolve user name information.
dqResolveSync	1	Wait while resolving name information.
dqResolveAsync	2	Do not wait while resolving name information. The OnUserNameChanged event fires when the name is resolved.

Table 2 – UserNameResolutionConstants Values

QuotaStateConstants Values

Constant	Value	Description
dqAcctResolved	0	Account information is resolved.
dqAcctUnavailable	1	Account information is unavailable.
dqAcctDeleted	2	Account has been deleted.
dqAcctInvalid	3	Account is invalid.
dqAcctUnknown	4	Account cannot be found.

Table 3 – AccountStatus Values

Shell32.OfflineFolderStatus Values

Constant	Value	Description
OFS_DIRTYCACHE	3	Server is online with unsynchronized changes.
OFS_INACTIVE	-1	Offline caching is not enabled for this folder.
OFS_OFFLINE	1	Server is offline.
OFS_ONLINE	0	Server is online.
OFS_SERVERBACK	2	Server is offline but can be reached.

Table 4 – Shell32.OfflineFolderStatus Values

SHFILEOPSTRUCT Values

Constant	Value	Description
FOF_SILENT	4	Do not display a progress dialog box.
FOF_RENAMEONCOLLISION	8	Give the file being operated on a new name in a move, copy, or rename operation if a file with the target name already exists.
FOF_NOCONFIRMATION	16	Respond with "Yes to All" for any dialog box that is displayed.
FOF_ALLOWUNDO	64	Preserve undo information, if possible.
FOF_FILESONLY	128	Perform the operation on files only if a wildcard file name (*.*) is specified.
FOF_SIMPLEPROGRESS	256	Display a progress dialog box but do not show the file names.
FOF_NOCONFIRMMKDIR	512	Do not confirm the creation of a new directory if the operation requires one to be created.
FOF_NOERRORUI	1024	Do not display a user interface if an error occurs.
FOF_NOCOPYSECURITYATTRIBS	2048	Do not copy the security attributes of the file.
FOF_NORECURSION	4096	Only operate in the local directory. Don't operate recursively into subdirectories.
FOF_NO_CONNECTED_ELEMENTS	9182	Version 5.0. Do not copy connected files as a group. Only copy the specified files.

Table 5 – SHFILEOPSTRUCT Values

Details Options Values

Index	Property	Index	Property
0	Name	18	Year
1	Size	19	Track Number
2	Type	20	Genre
3	Date Modified	21	Duration
4	Date Created	22	Bit Rate
5	Date Accessed	23	Protected
6	Attributes	24	Camera Model
7	Status	25	Date Picture Taken
8	Owner	26	Dimensions
9	Author	27	Not used
10	Title	28	Not used
11	Subject	29	Not used
12	Category	30	Company
13	Pages	31	Description
14	Comments	32	File Version
15	Copyright	33	Product Name

16	Artist	34	Product Version
17	Album Title	-1	Retrieves the info tip information for the item.

Table 6 – Details Options Values**Shell32.ShellSpecialFolderConstants Values**

Constant	Val	Description
CSIDL_ADMINTOOLS	48	The file system directory that is used to store administrative tools for an individual user. The Microsoft Management Console (MMC) will save customized consoles to this directory, and it will roam with the user.
CSIDL_ALTSTARTUP	29	File system directory that corresponds to the user's nonlocalized Startup program group.
CSIDL_APPDATA	26	File system directory that serves as a common repository for application-specific data. A typical path is C:\Documents and Settings\username\Application Data.
CSIDL_BITBUCKET	10	Virtual folder containing the objects in the user's Recycle Bin.
CSIDL_CDBURN_AREA	59	The file system directory acting as a staging area for files waiting to be written to CD. A typical path is C:\Documents and Settings\username\Local Settings\Application Data\Microsoft\CD Burning.
CSIDL_COMMON_ADMINTOOLS	47	The file system directory containing administrative tools for all users of the computer.
CSIDL_COMMON_ALTSTARTUP	30	File system directory that corresponds to the nonlocalized Startup program group for all users. Valid only for Microsoft Windows NT systems.
CSIDL_COMMON_APPDATA	35	Application data for all users. A typical path is C:\Documents and Settings\All Users\Application Data.
CSIDL_COMMON_DESKTOPDIR	25	File system directory that contains files and folders that appear on the desktop for all users. A typical path is C:\Documents and Settings\All Users\Desktop. Valid only for Windows NT systems.
CSIDL_COMMON_DOCUMENTS	46	The file system directory that contains documents that are common to all users. A typical paths is C:\Documents and Settings\All Users\Documents.
CSIDL_COMMON_FAVORITES	31	File system directory that serves as a common repository for all users' favorite items. Valid only for Windows NT systems.
CSIDL_COMMON_MUSIC	53	The file system directory that serves as a repository for music files common to all users. A typical path is C:\Documents and Settings\All Users\Documents\My Music.
CSIDL_COMMON_PICTURES	54	The file system directory that serves as a repository for image files common to all users. A typical path is C:\Documents and Settings\All Users\Documents\My Pictures.
CSIDL_COMMON_PROGRAMS	23	File system directory that contains the directories for the common program groups that appear on the Start menu for all users. A typical path is C:\Documents and Settings\All Users\Start Menu\Programs. Valid only for Windows NT systems.
CSIDL_COMMON_STARTMENU	22	File system directory that contains the programs and folders that appear on the Start menu for all users. A typical path is C:\Documents and Settings\All Users\Start Menu. Valid only

		for Windows NT systems.
CSIDL_COMMON_STARTUP	24	File system directory that contains the programs that appear in the Startup folder for all users. A typical path is C:\Documents and Settings\All Users\Start Menu\Programs\Startup. Valid only for Windows NT systems.
CSIDL_COMMON_TEMPLATES	45	The file system directory that contains the templates that are available to all users. A typical path is C:\Documents and Settings\All Users\Templates.
CSIDL_COMMON_VIDEO	55	The file system directory that serves as a repository for video files common to all users. A typical path is C:\Documents and Settings\All Users\Documents\My Videos.
CSIDL_COMPUTERSNEARME	61	The folder representing other machines in your workgroup.
CSIDL_CONNECTIONS	49	The virtual folder representing Network Connections, containing network and dial-up connections.
CSIDL_CONTROLS	3	Virtual folder containing icons for the Control Panel applications.
CSIDL_COOKIES	33	File system directory that serves as a common repository for Internet cookies. A typical path is C:\Documents and Settings\username\Cookies.
CSIDL_DESKTOP	0	Microsoft Windows Desktop virtual folder that is the root of the namespace.
CSIDL_DESKTOPDIRECTORY	16	The file system directory used to physically store file objects on the desktop (not to be confused with the desktop folder itself). A typical path is C:\Documents and Settings\username\Desktop.
CSIDL_DRIVES	17	The virtual folder representing My Computer, containing everything on the local computer: storage devices, printers, and Control Panel. The folder may also contain mapped network drives.
CSIDL_FAVORITES	6	File system directory that serves as a common repository for the user's favorite items. A typical path is C:\Documents and Settings\username\Favorites.
CSIDL_FONTS	20	Virtual folder containing installed fonts. A typical path is C:\WINNT\Fonts.
CSIDL_HISTORY	34	File system directory that serves as a common repository for Internet history items.
CSIDL_INTERNET	1	A virtual folder for Internet Explorer (icon on desktop).
CSIDL_INTERNET_CACHE	32	File system directory that serves as a common repository for temporary Internet files. A typical path is C:\Documents and Settings\username\Temporary Internet Files.
CSIDL_LOCAL_APPDATA	28	File system directory that serves as a data repository for local (non-roaming) applications. A typical path is C:\Documents and Settings\username\Local Settings\Application Data.
CSIDL_MYDOCUMENTS	12	The virtual folder representing the My Documents desktop item.
CSIDL_MYMUSIC	13	The file system directory that serves as a common repository for music files. A typical path is C:\Documents and Settings\User\My Documents\My Music.
CSIDL_MYPICTURES	39	My Pictures folder. A typical path is C:\Documents and Settings\username\My Documents\My Pictures.
CSIDL_MYVIDEO	14	The file system directory that serves as a common repository for video files. A typical path is C:\Documents and

		Settings\username\My Documents\My Videos.
CSIDL_NETHOOD	19	A file system folder containing the link objects that may exist in the My Network Places virtual folder. It is not the same as ssfNETWORK, which represents the network namespace root. A typical path is C:\Documents and Settings\username\NetHood.
CSIDL_NETWORK	18	Network Neighborhood virtual folder representing the root of the network namespace hierarchy.
CSIDL_PERSONAL	5	File system directory that serves as a common repository for a user's documents. A typical path is C:\Documents and Settings\username\My Documents.
CSIDL_PHOTOALBUMS	69	This documentation is preliminary and is subject to change. Windows Vista and later. The virtual folder used to store photo albums, typically username\My Pictures\Photo Albums.
CSIDL_PLAYLISTS	63	This documentation is preliminary and is subject to change. Windows Vista and later. The virtual folder used to store play albums, typically username\My Music\Playlists.
CSIDL_PRINTERS	4	Virtual folder containing installed printers.
CSIDL_PRINTHOOD	27	File system directory that contains the link objects that may exist in the Printers virtual folder. A typical path is C:\Documents and Settings\username\PrintHood.
CSIDL_PROFILE	40	User's profile folder.
CSIDL_PROGRAM_FILES	38	Program Files folder. A typical path is C:\Program Files.
CSIDL_PROGRAM_FILES_COMMON	43	A folder for components that are shared across applications. A typical path is C:\Program Files\Common.
CSIDL_PROGRAMS	2	File system directory that contains the user's program groups (which are also file system directories). A typical path is C:\Documents and Settings\username\Start Menu\Programs.
CSIDL_RECENT	8	File system directory that contains the user's most recently used documents. A typical path is C:\Documents and Settings\username\Recent.
CSIDL_SENDTO	9	File system directory that contains Send To menu items. A typical path is C:\Documents and Settings\username\SendTo.
CSIDL_STARTMENU	12	File system directory containing Start menu items. A typical path is C:\Documents and Settings\username\Start Menu.
CSIDL_STARTUP	7	File system directory that corresponds to the user's Startup program group. The system starts these programs whenever any user logs onto Windows NT or starts Windows 95. A typical path is C:\Documents and Settings\username\Start Menu\Programs\Startup.
CSIDL_SYSTEM	37	System folder. A typical path is C:\WINNT\SYSTEM32.
CSIDL_TEMPLATES	21	File system directory that serves as a common repository for document templates.
CSIDL_WINDOWS	36	Windows directory or SYSROOT. This corresponds to the %windir% or %SYSTEMROOT% environment variables. A typical path is C:\WINNT.

Table 7 – Shell32.ShellSpecialFolderConstants Values

Shell32.ShellFolderViewOptions Values

Constant	Val	Description
SFVVO_SHOWALLOBJECTS	1	The Show All Files option is enabled.
SFVVO_SHOWEXTENSIONS	2	The Hide File Extensions for Known File Types option is disabled.
SFVVO_SHOWCOMPCOLOR	8	The Display Compressed Files and Folders with Alternate Color option is enabled.
SFVVO_SHOWSYSFILES	32	The Do Not Show Hidden Files option is enabled.
SFVVO_WIN95CLASSIC	64	The Classic Style option is enabled.
SFVVO_DOUBLECLICKINWEBVIEW	128	The Double-Click to Open an Item option is enabled.
SFVVO_DESKTOPHTML	512	The Active Desktop View as Web Page option is enabled.

Table 8 – Shell32.ShellFolderViewOptions Values

SHCONTF Enumerated Type Values

Constant	Value	Description
SHCONTF_FOLDERS	32	Include items that are folders in the enumeration.
SHCONTF_NONFOLDERS	64	Include items that are not folders in the enumeration.
SHCONTF_INCLUDEHIDDEN	128	Include hidden items in the enumeration.
SHCONTF_NETPRINTERSRCH	512	The caller is looking for printer objects.
SHCONTF_SHAREABLE	1024	The caller is looking for remote shares.
SHCONTF_STORAGE	2048	Include items with accessible storage and their ancestors.

Table 9 – SHCONTF Enumerated Type Values

SHGetSetSettings Enumerated Type Values

Constant	Value	Description
SSF_DESKTOPHTML		The state of the Active Desktop.
SSF_DONTPRETTYPATH		The state of the Allow all uppercase names option.
SSF_DOUBLECLICKINWEBVIEW		The state of the Double-click to open an item option.
SSF_FILTER		Not used.
SSF_HIDEICONS		The state of Desktop icons.
SSF_MAPNETDRVBUTTON		The state of the Show map network drive button in toolbar option.
SSF_NOCONFIRMRECYCLE		The state of the Recycle Bin's Display delete confirmation dialog option.
SSF_NONETCRAWLING		The state of the Automatically search for network folders and printers option.
SSF_SEPPROCESS		The state of the Launch folder windows in a separate process option.
SSF_SERVERADMINUI		Not used.
SSF_SHOWALLOBJECTS		The state of the Show hidden files and folders option.
SSF_SHOWATTRIBCOL		The state of the Show File Attributes in Detail View

		option.
SSF_SHOWCOMPColor		The state of the Show encrypted or compressed NTFS files in color option.
SSF_SHOWEXTENSIONS		The state of the Hide extensions for known file types option.
SSF_SHOWINFOTIP		The state of the Show pop-up description for folder and desktop items option.
SSF_SHOWSTARTPAGE		Not used.
SSF_SHOWSYSFILES		The state of the Do not show hidden files and folders option.
SSF_SHOWSUPERHIDDEN		The state of the Hide protected operating system files option.
SSF_WEBVIEW		Display as a Web View
SSF_WIN95CLASSIC		The state of the Classic Style option.

Table 10 – SHGetSetSettings Enumerated Type Values**wProcessorArchitectureType SYSTEM_INFO Values**

Constant	Value	Description
PROCESSOR_ARCHITECTURE_AMD64	9	x64 (AMD or Intel)
PROCESSOR_ARCHITECTURE_IA32_ON_WIN64	10	WOW64
PROCESSOR_ARCHITECTURE_IA64	6	Intel Itanium Processor Family (IPF)
PROCESSOR_ARCHITECTURE_INTEL	0	x86
PROCESSOR_ARCHITECTURE_UNKNOWN	&HFFFF	Unknown processor.

Table 11 – wProcessorArchitectureType SYSTEM_INFO Values

SHCOLUMNID

SHCOLUMNID Values

PID	Property Name	Data Type
2	Title	String
3	Subject	String
4	Author	String
5	Keywords	String
6	Comments	String
7	Template	String
8	Last Saved By	String
9	Revision Number	String
10	Total Editing Time	Date
11	Last Printed	Date
12	Create Time/Date	Date
13	Last Saved Time/Date	Date
14	Number of Pages	Integer
15	Number of Words	Integer

16	Number of Characters	Integer
17	Thumbnail	VT_CF
18	Name of Creating Application	String
19	Security	Integer

Table 12 – SHCOLUMNID Values