WORKING WITH WINDOWS SHELL32 ................................................................. 3

MANAGING DISK QUOTAS ON THE NTFS FILE SYSTEM ................................. 4

MICROSOFT.DIDISKQUOTAOBJECT 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.Used Property ..................................................................... 7
  DIDiskQuotaUser.UsedText Property ............................................................... 7
  DIDiskQuotaUser.InvalidMethod .................................................................. 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.TranslateLogicalNameToSID 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. I. SHELL DISPATCH 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
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.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIDiskQuotaUser</td>
<td>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</td>
</tr>
<tr>
<td>DiskQuotaControl</td>
<td>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</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Description</th>
<th>The <strong>AccountContainerName</strong> property retrieves the name of the user's account container.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Type</strong></td>
<td>String value that is set to the user's account container name.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>- For Microsoft Windows NT 4.0 accounts or other accounts without directory services information, this property contains the domain name.</td>
</tr>
<tr>
<td></td>
<td>- For accounts with directory services information, this property contains a canonical name with the terminating object name removed.</td>
</tr>
</tbody>
</table>

**DIDiskQuotaUser.AccountStatus Property**

<table>
<thead>
<tr>
<th>Description</th>
<th>The <strong>AccountStatus</strong> property retrieves the status of the user's account.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Type</strong></td>
<td>Integer – on of the AccountStatus Values in Table 3 on page 61</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>- The property is read-only. The property has no default value.</td>
</tr>
</tbody>
</table>

**DIDiskQuotaUser.DisplayName Property**

<table>
<thead>
<tr>
<th>Description</th>
<th>The <strong>DisplayName</strong> property retrieves the status of the user's account.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Type</strong></td>
<td>String value that is set to the user's display name.</td>
</tr>
</tbody>
</table>
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.
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.

**DIDiskQuotaUser.Invalidate Method**

**Description**

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

**Syntax**

```csharp
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

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

DiskQuotaControl.DefaultQuotaLimitText Property

**Description**

The **DefaultQuotaLimitText** property returns 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

```vbscript
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
tsDQLimitText = 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sLogonName</td>
<td>Required. String value that contains the user's logon name. Use the <strong>UserNameResolution</strong> property to specify how the name is to be resolved.</td>
</tr>
</tbody>
</table>

**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 **AddUser** method deletes a user from the volume.
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sLogonName</code></td>
<td>Required. String value that contains the user's logon name.</td>
</tr>
</tbody>
</table>

**Note**

- This method returns a **DIDiskQuotaUser** 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```
```vba
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

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

Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oUser</td>
<td>Required. Object expression that evaluates to the user's DIDiskQuotaUser object.</td>
</tr>
</tbody>
</table>

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

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

Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sPath</td>
<td>Required. String value that contains the fully qualified path of the volume to be initialized.</td>
</tr>
<tr>
<td>bReadWrite</td>
<td>Required. Boolean value that specifies how the volume is to be opened.</td>
</tr>
</tbody>
</table>
Set `bReadWrite` 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

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

#### Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logonname</td>
<td>Required. String value that specifies the user's logon name.</td>
</tr>
</tbody>
</table>

#### 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**.

```vbscript
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(oDQC.TranslateLogonNameToSID(sName))
    End If
End Function
```

#### 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.

```vbscript
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").

```vba
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**

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

**Arguments**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vItem</td>
<td>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.</td>
</tr>
<tr>
<td>vOptions</td>
<td>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</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>

Dani Vainstein  Shell32  Page 19 of 68
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.

`iColumn` 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`.

```vba
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 = oFolderItem.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
```

![Screenshot of InfoTip Information dialog box]
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.

```vbscript
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**

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

**Arguments**

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

**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 all the files in MyTemp from the root directory of the C:\ drive to the C:\MyTempFolder folder, with a progress bar.

```vbnet
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**

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

**Arguments**
### Shell32.Folder.ParseName Method

**Description**

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

**Syntax**

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

**Arguments**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sName</td>
<td>Required. A string that specifies the name of the item.</td>
</tr>
</tbody>
</table>

**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.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sName</td>
<td>Required. A string that specifies the name of the new folder.</td>
</tr>
<tr>
<td>vOptions</td>
<td>Optional. This value is not currently used.</td>
</tr>
</tbody>
</table>

**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
```
(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 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**

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

Example

The following example uses \texttt{GetFolder} to retrieve the \texttt{Folder} object for the \texttt{System32} folder.

```vbscript
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

\textbf{Description}

The \texttt{GetLink} property contains the item's \texttt{ShellLinkObject} object, if the item is a shortcut.

\textbf{Note}

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

Shell32.FolderItem.IsBrowsable Property

\textbf{Description}

The \texttt{IsBrowsable} property indicates if the item can be browsed.

\textbf{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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vVerb</td>
<td>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.</td>
</tr>
</tbody>
</table>

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".

Example

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
```
Example 2

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

```vbs
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**

```vbs
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.

```vbs
Option Explicit
Private Const CSIDL_WINDOWS = 36
Dim oShell132, oFolder, oFolderItem, oFolderItemVerbs, oFolderItemVerb
'--- Creating a shell application object
Set oShell132 = CreateObject("Shell.Application")
Set oFolder = oShell132.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 oShell132 = Nothing
```
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nIndex</code></td>
<td>Optional. Specifies the zero-based index of the item to retrieve. This value must be less than the value of the Count property.</td>
</tr>
</tbody>
</table>
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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grfFlags</td>
<td>Required. This parameter can be one of the flags listed in SHCONTF Enumerated Type Values.</td>
</tr>
<tr>
<td>bstrFilter</td>
<td>Required. Contains a filter string.</td>
</tr>
</tbody>
</table>

### 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:

```vbnet
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"
    End If
    nCount = oFolderItems.Count
    MsgBox "Items after Filter = " & nCount, 0, "Filter Method"
End If

'--- Cleaning objects
Set oFolderItem = Nothing : Set oFolderItems = Nothing
Set oFolder = Nothing : Set oShell32 = Nothing
```

---

**Filter Method**

- Items before Filter: 251
- Items after Filter: 23

---

Dani Vainstein  
Shell32  
Page 34 of 68
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](#).

---

**Shell32.FolderItemVerb.Name Property**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>Name</strong> property contains the verb's name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>String that receives the <strong>Name</strong> property.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The property is read-only. The property has no default value.</td>
</tr>
</tbody>
</table>

**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_PROGRAMS)
'--- 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 & vbCrLf
      Next
   MsgBox sVerbList, 0, "Verbs List"
```

---
'--- Resetting variable
sVerbsList = Empty
'--- This method implements the For i to Count loop
For i = 0 To oVerbs.Count - 1
    sVerbsList = sVerbsList & oVerbs.Item(i).Name & vbCrLf
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

```vbnet
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:

<table>
<thead>
<tr>
<th>Verb</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>edit</td>
<td>Launches an editor and opens the document for editing.</td>
</tr>
<tr>
<td>find</td>
<td>Initiates a search starting from the specified directory.</td>
</tr>
<tr>
<td>open</td>
<td>If this file is not an executable file, its associated application is launched.</td>
</tr>
<tr>
<td>print</td>
<td>Prints the document file.</td>
</tr>
<tr>
<td>properties</td>
<td>Displays the object's properties.</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nIndex</td>
<td>Required. Specifies the zero-based index of the item to retrieve. This value must be less than the value of the Count property</td>
</tr>
</tbody>
</table>

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**

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

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sServiceName</td>
<td>Required. String that contains the name of the service.</td>
</tr>
</tbody>
</table>

**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.

![Cascade Windows](image)

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sDir</td>
<td>Required. Specifies the Control Panel application's file name. All Control Panel applications have the .cpl extension.</td>
</tr>
</tbody>
</table>

**Return Value**

No return value.

**Tip**

---

**Example**

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vDir</td>
<td>Required. Specifies the folder to be displayed. This can be a string that specifies the path of the folder or one of the ShellSpecialFolderConstants</td>
</tr>
</tbody>
</table>
values.

Example

The following example shows Explore in use.

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

![Figure 4 - Explore Window](image)

**Shell32.ExplorerPolicy Method**

**Description**

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

**Syntax**

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

**Arguments**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrPolicyName</td>
<td>Required. String that specifies the name of the Internet Explorer policy.</td>
</tr>
</tbody>
</table>

**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**

```vbscript
object.FileRun()
```

**Return Value**

No return value.

![Figure 5 - File Run Dialog](image)

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**

```vbscript
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**

```vbnet
object.FindFiles ()
```

**Return Value**

No return value.
The `FindPrinter` method displays the Find Printer dialog box to allow the user to find a printer.

### Syntax

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

### Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sName</code></td>
<td>Optional. String that contains the printer name.</td>
</tr>
<tr>
<td><code>sLocation</code></td>
<td>Optional. String that contains the printer location.</td>
</tr>
<tr>
<td><code>sModel</code></td>
<td>Optional. String that contains the printer model.</td>
</tr>
</tbody>
</table>

### 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.

![Find Printer dialog](image)

**Figure 8 - Find Printer dialog**

### Shell32.GetSetting Method

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>GetSetting</code> method retrieves a Shell global setting.</td>
</tr>
</tbody>
</table>

**Syntax**

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

**Arguments**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nSetting</code></td>
<td>Required. A value of type Integer that specifies the current Shell settings to retrieve. The <code>SHGetSetSettings Enumerated Type</code> described on Table 10 on page 66.</td>
</tr>
</tbody>
</table>

**Return Value**

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

### Shell32.GetSystemInformation Method

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>GetSystemInformation</code> method retrieves a Shell global setting.</td>
</tr>
</tbody>
</table>

**Syntax**

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

**Arguments**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sName</code></td>
<td></td>
</tr>
</tbody>
</table>
sName

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.

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

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

\texttt{object.Help ()}

Return Value

No return value.
Shell32.IsRestricted Method

**Description**

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

**Syntax**

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

**Arguments**

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

**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

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

Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sServiceName</td>
<td>Required. String that contains the service's name.</td>
</tr>
</tbody>
</table>

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:

```vba
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 **MinimizeAll** 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

```plaintext
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vDir</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.

```vb
Public Function fGetShellNameSpace(ByVal vDir, ByRef sTitle)
    Dim oShell32, oFolder
    Set oShell32 = CreateObject("Shell.Application")
    Set oFolder = oShell32.NameSpace(vDir)
    If (Not oFolder 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vDir</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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**

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

**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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFile</td>
<td>Required. String that contains the name of the file on which ShellExecute will perform the action specified by vOperation.</td>
</tr>
<tr>
<td>vArguments</td>
<td>Optional. Variant that contains the parameter values for the operation.</td>
</tr>
<tr>
<td>vDirectory</td>
<td>Optional. Variant that contains the fully qualified path of the directory that contains the file specified by sFile. If this parameter is not specified, the current working directory is used.</td>
</tr>
<tr>
<td>vOperation</td>
<td>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.</td>
</tr>
<tr>
<td>vShow</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Note**

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

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

**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](image)

**Shell32.ShowBrowserBar Method**

**Description** Not Supported By QuickTest Professional

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

**Syntax**

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

**Arguments**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sCLSID</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>vShow</code></td>
<td>Required. Variant that is set to true to show the browser bar, or false to hide it.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Explorer Bar</th>
<th>CLSID string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Favorites</td>
<td>{EFA24E61-B078-11d0-89E4-00C04FC9E26E}</td>
</tr>
<tr>
<td>Folders</td>
<td>{EFA24E64-B078-11d0-89E4-00C04FC9E26E}</td>
</tr>
<tr>
<td>History</td>
<td>{EFA24E62-B078-11d0-89E4-00C04FC9E26E}</td>
</tr>
<tr>
<td>Search</td>
<td>{30D02401-6A81-11d0-8274-00C04FD5AE38}</td>
</tr>
</tbody>
</table>

**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 – Shut Down Windows Dialog](i)

**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.
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.

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](image)

**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](image)
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

```plaintext
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

```plaintext
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

```plaintext
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**
  
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sPropName</td>
<td>Required. String value that specifies the property. See the Remarks section for details.</td>
</tr>
</tbody>
</table>

- **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 \textit{sPropName}. However, each property is a member of a \textbf{Component Object Model (COM)} property set and can also be identified by specifying its format ID (\textit{FMTID}) and property ID (\textit{PID}). An \textit{FMTID} is a \textbf{GUID} that identifies the property set, and a \textit{PID} is an integer that identifies a particular property within the property set.

- Specifying a property by its \textit{FMTID/PID} values is usually more efficient than using its name. To use a property's \textit{FMTID/PID} values with \textbf{ExtendedProperty}, they must be combined into an \textbf{SCID}. An \textbf{SCID} is a string that contains the \textit{FMTID/PID} values in the form "FMTID\text{PID}", where the \textit{FMTID} is the string form of the property set's globally unique identifier (\textbf{GUID}).

- For a list of FMTIDs and PIDs that are currently supported by the Shell, see \textbf{SHCOLUMNID} values in Table 12 on page 68.

\textbf{Example}

The following example uses \textit{Item} to retrieve the extended properties of a document file using the \textbf{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 oFolder Is Nothing Then
    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
```

\textbf{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

--- 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 list items in the "My Music" folder?

--- 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?

--- 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)
--- Retrieving collection of items
For Each oFolderItem in oFolder.Items
    DataTable.LocalSheet.SetCurrentRow nRow
    DataTable("ITEM_NAME", dtLocalSheet) = oFolderItem.Name
    nRow = nRow + 1
Next
Set oShell = Nothing

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

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 oShell32, oShell, oFolder, oFolderItem, oURLShortcut
Dim sDesktopFld
'--- Creating a application object
Set oShell32 = CreateObject("Shell.Application")
Set oFolder = oShell32.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

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dqStateDisable</td>
<td>0</td>
<td>Disk quotas are disabled.</td>
</tr>
<tr>
<td>dqStateTrack</td>
<td>1</td>
<td>Disk quotas are disabled.</td>
</tr>
<tr>
<td>dqStateEnforce</td>
<td>2</td>
<td>Enforce quota limit.</td>
</tr>
</tbody>
</table>

Table 1 – QuotaStateConstants Values

UserNameResolutionConstants Values

<table>
<thead>
<tr>
<th>Constant</th>
<th>Val</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dqResolveNone</td>
<td>0</td>
<td>Do not resolve user name information.</td>
</tr>
<tr>
<td>dqResolveSync</td>
<td>1</td>
<td>Wait while resolving name information.</td>
</tr>
<tr>
<td>dqResolveAsync</td>
<td>2</td>
<td>Do not wait while resolving name information. The OnUserNameChanged event fires when the name is resolved.</td>
</tr>
</tbody>
</table>

Table 2 – UserNameResolutionConstants Values

QuotaStateConstants Values

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dqAcctResolved</td>
<td>0</td>
<td>Account information is resolved.</td>
</tr>
<tr>
<td>dqAcctUnavailable</td>
<td>1</td>
<td>Account information is unavailable.</td>
</tr>
<tr>
<td>dqAcctDeleted</td>
<td>2</td>
<td>Account has been deleted.</td>
</tr>
<tr>
<td>dqAcctInvalid</td>
<td>3</td>
<td>Account is invalid.</td>
</tr>
<tr>
<td>dqAcctUnknown</td>
<td>4</td>
<td>Account cannot be found.</td>
</tr>
</tbody>
</table>

Table 3 – AccountStatus Values

Shell32.OfflineFolderStatus Values

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS_DIRTYCACHE</td>
<td>3</td>
<td>Server is online with unsynchronized changes.</td>
</tr>
<tr>
<td>OFS_INACTIVE</td>
<td>-1</td>
<td>Offline caching is not enabled for this folder.</td>
</tr>
<tr>
<td>OFS_OFFLINE</td>
<td>1</td>
<td>Server is offline.</td>
</tr>
<tr>
<td>OFS_ONLINE</td>
<td>0</td>
<td>Server is online.</td>
</tr>
<tr>
<td>OFS_SERVERBACK</td>
<td>2</td>
<td>Server is offline but can be reached.</td>
</tr>
</tbody>
</table>

Table 4 – Shell32.OfflineFolderStatus Values
**SHFILEOPSTRUCT Values**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOF_SILENT</td>
<td>4</td>
<td>Do not display a progress dialog box.</td>
</tr>
<tr>
<td>FOF_RENAMEONCOLLISION</td>
<td>8</td>
<td>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.</td>
</tr>
<tr>
<td>FOF_NOCONFIRMATION</td>
<td>16</td>
<td>Respond with &quot;Yes to All&quot; for any dialog box that is displayed.</td>
</tr>
<tr>
<td>FOF_ALLOWWUNDO</td>
<td>64</td>
<td>Preserve undo information, if possible.</td>
</tr>
<tr>
<td>FOF_FILESONLY</td>
<td>128</td>
<td>Perform the operation on files only if a wildcard file name (<em>.</em>) is specified.</td>
</tr>
<tr>
<td>FOF_SIMPLEPROGRESS</td>
<td>256</td>
<td>Display a progress dialog box but do not show the file names.</td>
</tr>
<tr>
<td>FOF_NOCONFIRMMDIR</td>
<td>512</td>
<td>Do not confirm the creation of a new directory if the operation requires one to be created.</td>
</tr>
<tr>
<td>FOF_NOERRORUI</td>
<td>1024</td>
<td>Do not display a user interface if an error occurs.</td>
</tr>
<tr>
<td>FOF_NOCOPYSECURITYATTRIBS</td>
<td>2048</td>
<td>Do not copy the security attributes of the file.</td>
</tr>
<tr>
<td>FOF_NORECURSION</td>
<td>4096</td>
<td>Only operate in the local directory. Don’t operate recursively into subdirectories.</td>
</tr>
<tr>
<td>FOF_NO_CONNECTED_ELEMENTS</td>
<td>9182</td>
<td>Version 5.0. Do not copy connected files as a group. Only copy the specified files.</td>
</tr>
</tbody>
</table>

**Table 5 – SHFILEOPSTRUCT Values**

**Details Options Values**

<table>
<thead>
<tr>
<th>Index</th>
<th>Property</th>
<th>Index</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Name</td>
<td>18</td>
<td>Year</td>
</tr>
<tr>
<td>1</td>
<td>Size</td>
<td>19</td>
<td>Track Number</td>
</tr>
<tr>
<td>2</td>
<td>Type</td>
<td>20</td>
<td>Genre</td>
</tr>
<tr>
<td>3</td>
<td>Date Modified</td>
<td>21</td>
<td>Duration</td>
</tr>
<tr>
<td>4</td>
<td>Date Created</td>
<td>22</td>
<td>Bit Rate</td>
</tr>
<tr>
<td>5</td>
<td>Date Accessed</td>
<td>23</td>
<td>Protected</td>
</tr>
<tr>
<td>6</td>
<td>Attributes</td>
<td>24</td>
<td>Camera Model</td>
</tr>
<tr>
<td>7</td>
<td>Status</td>
<td>25</td>
<td>Date Picture Taken</td>
</tr>
<tr>
<td>8</td>
<td>Owner</td>
<td>26</td>
<td>Dimensions</td>
</tr>
<tr>
<td>9</td>
<td>Author</td>
<td>27</td>
<td>Not used</td>
</tr>
<tr>
<td>10</td>
<td>Title</td>
<td>28</td>
<td>Not used</td>
</tr>
<tr>
<td>11</td>
<td>Subject</td>
<td>29</td>
<td>Not used</td>
</tr>
<tr>
<td>12</td>
<td>Category</td>
<td>30</td>
<td>Company</td>
</tr>
<tr>
<td>13</td>
<td>Pages</td>
<td>31</td>
<td>Description</td>
</tr>
<tr>
<td>14</td>
<td>Comments</td>
<td>32</td>
<td>File Version</td>
</tr>
<tr>
<td>15</td>
<td>Copyright</td>
<td>33</td>
<td>Product Name</td>
</tr>
<tr>
<td>Constant</td>
<td>Val</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CSIDL_ADMINTOOLS</td>
<td>48</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_ALTSTARTUP</td>
<td>29</td>
<td>File system directory that corresponds to the user's nonlocalized Startup program group.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_APPDATA</td>
<td>26</td>
<td>File system directory that serves as a common repository for application-specific data. A typical path is C:\Documents and Settings\username\Application Data.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_BITBUCKET</td>
<td>10</td>
<td>Virtual folder containing the objects in the user's Recycle Bin.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_CDBURN_AREA</td>
<td>59</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_ADMINTOOLS</td>
<td>47</td>
<td>The file system directory containing administrative tools for all users of the computer.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_ALTSTARTUP</td>
<td>30</td>
<td>File system directory that corresponds to the nonlocalized Startup program group for all users. Valid only for Microsoft Windows NT systems.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_APPDATA</td>
<td>35</td>
<td>Application data for all users. A typical path is C:\Documents and Settings\All Users\Application Data.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_DESKTOPDIR</td>
<td>25</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_DOCUMENTS</td>
<td>46</td>
<td>The file system directory that contains documents that are common to all users. A typical path is C:\Documents and Settings\All Users\Documents.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_FAVORITES</td>
<td>31</td>
<td>File system directory that serves as a common repository for all users' favorite items. Valid only for Windows NT systems.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_MUSIC</td>
<td>53</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_PICTURES</td>
<td>54</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_PROGRAMS</td>
<td>23</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_COMMON_STARTMENU</td>
<td>22</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6 – Details Options Values

Shell32.ShellSpecialFolderConstants Values
<table>
<thead>
<tr>
<th>CSIDL_Common_Startup</th>
<th>24</th>
<th>File system directory that contains the programs that appear in the Startup folder for all users. A typical path is <code>C:\Documents and Settings\All Users\Start Menu\Programs\Startup</code>. Valid only for Windows NT systems.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSIDL_Common_Templates</td>
<td>45</td>
<td>The file system directory that contains the templates that are available to all users. A typical path is <code>C:\Documents and Settings\All Users\Templates</code>.</td>
</tr>
<tr>
<td>CSIDL_Common_Video</td>
<td>55</td>
<td>The file system directory that serves as a repository for video files common to all users. A typical path is <code>C:\Documents and Settings\All Users\Videos</code>.</td>
</tr>
<tr>
<td>CSIDL_ComputersNearMe</td>
<td>61</td>
<td>The folder representing other machines in your workgroup.</td>
</tr>
<tr>
<td>CSIDL_Connections</td>
<td>49</td>
<td>The virtual folder representing Network Connections, containing network and dial-up connections.</td>
</tr>
<tr>
<td>CSIDL_Co控ols</td>
<td>3</td>
<td>Virtual folder containing icons for the Control Panel applications.</td>
</tr>
<tr>
<td>CSIDL_Cookies</td>
<td>33</td>
<td>File system directory that serves as a common repository for Internet cookies. A typical path is <code>C:\Documents and Settings\username\Cookies</code>.</td>
</tr>
<tr>
<td>CSIDL_Desktop</td>
<td>0</td>
<td>Microsoft Windows Desktop virtual folder that is the root of the namespace.</td>
</tr>
<tr>
<td>CSIDL_DesktopDirectory</td>
<td>16</td>
<td>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 <code>C:\Documents and Settings\username\Desktop</code>.</td>
</tr>
<tr>
<td>CSIDL_Drives</td>
<td>17</td>
<td>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.</td>
</tr>
<tr>
<td>CSIDL_Favorites</td>
<td>6</td>
<td>File system directory that serves as a common repository for the user’s favorite items. A typical path is <code>C:\Documents and Settings\username\Favorites</code>.</td>
</tr>
<tr>
<td>CSIDL_Fonts</td>
<td>20</td>
<td>Virtual folder containing installed fonts. A typical path is <code>C:\WINNT\Fonts</code>.</td>
</tr>
<tr>
<td>CSIDL_History</td>
<td>34</td>
<td>File system directory that serves as a common repository for Internet history items.</td>
</tr>
<tr>
<td>CSIDL_Internet</td>
<td>1</td>
<td>A virtual folder for Internet Explorer (icon on desktop).</td>
</tr>
<tr>
<td>CSIDL_Internet_Cache</td>
<td>32</td>
<td>File system directory that serves as a common repository for temporary Internet files. A typical path is <code>C:\Documents and Settings\username\Temporary Internet Files</code>.</td>
</tr>
<tr>
<td>CSIDL_Local_APPData</td>
<td>28</td>
<td>File system directory that serves as a data repository for local (non-roaming) applications. A typical path is <code>C:\Documents and Settings\username\Local Settings\Application Data</code>.</td>
</tr>
<tr>
<td>CSIDL_MyDocuments</td>
<td>12</td>
<td>The virtual folder representing the My Documents desktop item.</td>
</tr>
<tr>
<td>CSIDL_MyMusic</td>
<td>13</td>
<td>The file system directory that serves as a common repository for music files. A typical path is <code>C:\Documents and Settings\User\My Documents\My Music</code>.</td>
</tr>
<tr>
<td>CSIDL_MyPictures</td>
<td>39</td>
<td>My Pictures folder. A typical path is <code>C:\Documents and Settings\username\My Documents\My Pictures</code>.</td>
</tr>
<tr>
<td>CSIDL_MyVideo</td>
<td>14</td>
<td>The file system directory that serves as a common repository for video files. A typical path is <code>C:\Documents and Settings\username\My Documents\My Videos</code>.</td>
</tr>
<tr>
<td>Special Folder Constant</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CSIDL_NETHOOD</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_NETWORK</td>
<td>Network Neighborhood virtual folder representing the root of the network namespace hierarchy.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PERSONAL</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PHOTOALBUMS</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PLAYLISTS</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PRINTERS</td>
<td>Virtual folder containing installed printers.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PRINTHOOD</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PROFILE</td>
<td>User's profile folder.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PROGRAM_FILES</td>
<td>Program Files folder. A typical path is C:\Program Files.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PROGRAM_FILES_COMM</td>
<td>A folder for components that are shared across applications. A typical path is C:\Program Files\Common.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_PROGRAMS</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_RECENT</td>
<td>File system directory that contains the user's most recently used documents. A typical path is C:\Documents and Settings\username\Recent.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_SENDTO</td>
<td>File system directory that contains Send To menu items. A typical path is C:\Documents and Settings\username\SendTo.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_STARTMENU</td>
<td>File system directory containing Start menu items. A typical path is C:\Documents and Settings\username\Start Menu.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_STARTUP</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_SYSTEM</td>
<td>System folder. A typical path is C:\WINNT\SYSTEM32.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_TEMPLATES</td>
<td>File system directory that serves as a common repository for document templates.</td>
<td></td>
</tr>
<tr>
<td>CSIDL_WINDOWS</td>
<td>Windows directory or SYSROOT. This corresponds to the %windir% or %SYSTEMROOT% environment variables. A typical path is C:\WINNT.</td>
<td></td>
</tr>
</tbody>
</table>

Table 7 – Shell32.ShellSpecialFolderConstants Values
Shell32.ShellFolderViewOptions Values

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

Table 8 – Shell32.ShellFolderViewOptions Values

SHCONTF Enumerated Type Values

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

Table 9 – SHCONTF Enumerated Type Values

SHGetSetSettings Enumerated Type Values

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSF_DESKTOPHTML</td>
<td></td>
<td>The state of the Active Desktop.</td>
</tr>
<tr>
<td>SSF_DONTPRETTYPATH</td>
<td></td>
<td>The state of the Allow all uppercase names option.</td>
</tr>
<tr>
<td>SSF_DOUBLECLICKINWEBVIEW</td>
<td></td>
<td>The state of the Double-click to open an item option.</td>
</tr>
<tr>
<td>SSF_FILTER</td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>SSF_HIDEICONS</td>
<td></td>
<td>The state of Desktop icons.</td>
</tr>
<tr>
<td>SSF_MAPNETDRVBUTTON</td>
<td></td>
<td>The state of the Show map network drive button in toolbar option.</td>
</tr>
<tr>
<td>SSF_NOCONFIRMRECYCLE</td>
<td></td>
<td>The state of the Recycle Bin’s Display delete confirmation dialog option.</td>
</tr>
<tr>
<td>SSF_NONETCRAWLING</td>
<td></td>
<td>The state of the Automatically search for network folders and printers option.</td>
</tr>
<tr>
<td>SSF_SEPPROCESS</td>
<td></td>
<td>The state of the Launch folder windows in a separate process option.</td>
</tr>
<tr>
<td>SSF_SERVERADMINUI</td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>SSF_SHOWALLOBJECTS</td>
<td></td>
<td>The state of the Show hidden files and folders option.</td>
</tr>
<tr>
<td>SSF_SHOWATTRIBCOL</td>
<td></td>
<td>The state of the Show File Attributes in Detail View.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>SSF_SHOWCOMPCOLOR</td>
<td>The state of the Show encrypted or compressed NTFS files in color option.</td>
<td></td>
</tr>
<tr>
<td>SSF_SHOWEXTENSIONS</td>
<td>The state of the Hide extensions for known file types option.</td>
<td></td>
</tr>
<tr>
<td>SSF_SHOWINFOTIP</td>
<td>The state of the Show pop-up description for folder and desktop items option.</td>
<td></td>
</tr>
<tr>
<td>SSF_SHOWSTARTPAGE</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>SSF_SHOWSYSFILES</td>
<td>The state of the Do not show hidden files and folders option.</td>
<td></td>
</tr>
<tr>
<td>SSF_SHOWSUPERHIDDEN</td>
<td>The state of the Hide protected operating system files option.</td>
<td></td>
</tr>
<tr>
<td>SSF_WEBVIEW</td>
<td>Display as a Web View</td>
<td></td>
</tr>
<tr>
<td>SSF_WIN95CLASSIC</td>
<td>The state of the Classic Style option.</td>
<td></td>
</tr>
</tbody>
</table>

**Table 10 – SHGetSetSettings Enumerated Type Values**

**wProcessorArchitectureType SYSTEM_INFO Values**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESSOR_ARCHITECTURE_AMD64</td>
<td>9</td>
<td>x64 (AMD or Intel)</td>
</tr>
<tr>
<td>PROCESSOR_ARCHITECTURE_IA32_ON_WIN64</td>
<td>10</td>
<td>WOW64</td>
</tr>
<tr>
<td>PROCESSOR_ARCHITECTURE_IA64</td>
<td>6</td>
<td>Intel Itanium Processor Family (IPF)</td>
</tr>
<tr>
<td>PROCESSOR_ARCHITECTURE_INTEL</td>
<td>0</td>
<td>x86</td>
</tr>
<tr>
<td>PROCESSOR_ARCHITECTURE_UNKNOWN</td>
<td>&amp;HFFFF</td>
<td>Unknown processor.</td>
</tr>
</tbody>
</table>

**Table 11 – wProcessorArchitectureType SYSTEM_INFO Values**

**SHCOLUMNID Values**

<table>
<thead>
<tr>
<th>PID</th>
<th>Property Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Title</td>
<td>String</td>
</tr>
<tr>
<td>3</td>
<td>Subject</td>
<td>String</td>
</tr>
<tr>
<td>4</td>
<td>Author</td>
<td>String</td>
</tr>
<tr>
<td>5</td>
<td>Keywords</td>
<td>String</td>
</tr>
<tr>
<td>6</td>
<td>Comments</td>
<td>String</td>
</tr>
<tr>
<td>7</td>
<td>Template</td>
<td>String</td>
</tr>
<tr>
<td>8</td>
<td>Last Saved By</td>
<td>String</td>
</tr>
<tr>
<td>9</td>
<td>Revision Number</td>
<td>String</td>
</tr>
<tr>
<td>10</td>
<td>Total Editing Time</td>
<td>Date</td>
</tr>
<tr>
<td>11</td>
<td>Last Printed</td>
<td>Date</td>
</tr>
<tr>
<td>12</td>
<td>Create Time/Date</td>
<td>Date</td>
</tr>
<tr>
<td>13</td>
<td>Last Saved Time/Date</td>
<td>Date</td>
</tr>
<tr>
<td>14</td>
<td>Number of Pages</td>
<td>Integer</td>
</tr>
<tr>
<td>15</td>
<td>Number of Words</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>16</td>
<td>Number of Characters</td>
<td>Integer</td>
</tr>
<tr>
<td>17</td>
<td>Thumbnail</td>
<td>VT_CF</td>
</tr>
<tr>
<td>18</td>
<td>Name of Creating Application</td>
<td>String</td>
</tr>
<tr>
<td>19</td>
<td>Security</td>
<td>Integer</td>
</tr>
</tbody>
</table>

**Table 12 – SHCOLUMNID Values**