cFileSelectFolder 1.6.1

_cFileSelectFolder() is intended as a replacement for native FileSelectFoldeer() for selecting folders: it implements the left pane of FileSelectFolder(), and goes beyond FileSelectFolder() in several ways.

Features:
- It is user friendly and designed to be easy to use;
- New folder, Delete folder and Rename folder controls are optional;
- Its GUI can be placed anywhere, including centred on another window;
- Neither needs nor uses CLSIDs;
- Checks calling parameters rigorously, with the user choosing either to show error messages in a dialog or to set @error and return them to the caller;
- For local and mapped drives, accepts file specifications with drives that can be drive letters, drive labels, or both,
- For local and mapped drives, always returns `*drive letter*:\ ...\` so its output is compatible with native AutoIt functions.
- For UNC paths, returns `\\*computer*\*share*\...\`
- For unmapped drives on computers on the network, offers a shortcut to shares[1].
- Selecting a treeview item to be highlighted initially is specified in a user-friendly way, including a diagnostic for when the user gets it wrong
- Because it can be called with many arguments, a user can show a list of the arguments / parameters and their values

The root of the treeview can be:
- The Desktop hierarchy with Desktop at the top
- Partial Desktop hierarchy with any other Desktop item at the top, with or without local/mapped drives and file folders, or remote shares and file folders (as appropriate),
- Local/mapped drives and file folders, or
- Remote shares and folders.

Calling _cFileSelectFolder

Before you call this function:
- If your script has neither a `WM_COMMAND` nor `WM_NOTIFY` handler, call `_FSF_RegMsg()`
- If it has a `WM_COMMAND` handler, call `_FSF_WM_COMMAND()` from it
- If it has a `WM_NOTIFY` handler, call `_FSF_WM_NOTIFY()` from it

You can then call `_cFileSelectFolder()` with no arguments. If you do, the treeview will represent the Desktop hierarchy; but to understand other capabilities of this function you need to read on.

If you code `_cFileSelectFolder('','?')` brief help is shown in a dialog box.

In the script, there is a UDF header to which you should refer. The syntax is:

```
_cFileSelectFolder([$sTitle = ''[, $sRoot = ''[, $sInitialTree = ''[, $iFolder = 7[,
$iWid = 400[,$iHt = 600[, $iLeft = -1[, $iTop = -1[, $hParent = 0[, $sExclFolders =
''[, $bShowHidden = False[,$bShowSystem = True]]]]]]]]]]]])
```

In more easily readable form, it look like this:

```
 _cFileSelectFolder($sTitle='', $sRoot='',$sInitialTree='', $iFolder=7, $iWid=400,
$iHt=600, $iLeft=-1, $iTop=-1, $hParent=0, $sExclFolders='', $bShowHidden=False,
$bShowSystem=True)
```

[1] If the shares end with `(*letter*)` where *letter* is usually the drive letter on the remote computer

All of the parameters are optional. Most of them are fully described there, but `$sRoot` and `$sInitialTree` need further explanation.

$sRoot

This parameter determines which of six types of display is used is determined by the first parameter, `$sRoot`. The root shows bolded as the top item in the treeview.

1. For **whole Desktop**, specify `''` or `'Desktop'` as `$sRoot`

2. To have the root be **below Desktop**, specify an item or a chain of items as the root:
   - If you specify `@UserName` as `$sRoot`, the top item will be your user name (In my case, Chris)
   - In Windows, Desktop appears twice in the hierarchy: at the very top, and further down. To specify the one further down, code `@UserName&'\Desktop'` as the root. Chaining like this is possible for any item, but it is only required for the user's Desktop.

3. To specify a **local or mapped folder path** as the root, begin `$sRoot` with the drive letter (with a colon). The top item appears as it does in Windows Explorer, that is with the label of the drive. On my computer, when I specify `'F:'`, the treeview shows `MProgs (F:)` as the top item. This also applies to mapped drives.

4. A local or mapped drive can also be specified as `Computer\` followed by the drive

5. To specify a path to a share on a remote computer to which your computer has access, specify the **UNC path** as `$sRoot`. For example, my computer has access to the share called `PDataH (G)` on `PETRA` so `$sRoot` begins `\\PETRA\PDataH (G)` . cFileSelectFolder tries to keep things as easy to use as possible. In this case, because the share on PETRA ends with `(G)` , I need only to specify `\\PETRA\G` (without a colon).

6, A path to a drive on a remote computer can also be specified as `Network\` followed by the UNC path without the leading double backslashes.

For local and mapped drives, rather than specifying a drive letter, you can specify a drive label followed by a colon. This is handy for USB sticks and the like, where the drive letter may change.

For types 3 and 4, the drive can be followed by a folder or folders. For example, on my computer, I can specify `H:\Comments\Readings`

For types 5 and 6, the share can be followed by a folder or folders.For example, on my computer, I can specify `\\PETRA\PDataH (G)\New palette` or `\\PETRA\G\New palette`

Some Desktop items have folder-path equivalents but others do not. For example, `My Documents` is `'C:\Users\'&@UserName&\Documents` but `Documents` has no equivalent. When the last Desktop item in `$sRoot` has no equivalent, _cFileSelectFolder expands it into folders. For example, if `$sRoot` is `'Documents'`, this script initially shows `My Documents` and `Public Documents` below `Documents`.

The values of `$sExclFolders`, `$bShowHidden` and `$bShowSystem` affect what is shown initially (and when your user expands folder items in the treeview). All three affect what folders are shown as treeview items.

If _cFileSlectFolder finds an error, it tells you. It detects some 40 errors. You never see such an error when `$sRoot` is valid, so your users will never see such an error! To me, this is preferable to setting @error to 40 values, magic numbers. (See Notitication of Errors below.)

When the user clicks on Select, if there is a folder path for the selected item, it is returned. For example, selecting My Documents on my computer returns 'C:\Users\Chris\Documents' without the drive label.

12 examples are included in the distribution. Most of them demonstrate what the types of $sRoot do.

$sInitialTree

The default is not to select any item in the treeview, but like FileSelectFolder, _cFileSelectFolder can initially highlight an item, but there is a restriction, described below.

$sInitialTree is relative to $sRoot. It can be a single treeview item, or a tree with items separated by backslashes. The first (or only) item can be any descendant of $sRoot; each subsequent item must be a child of the preceding item.

The restriction is that the first (or only) item must be one that shows initially. To see what items can be the initial or only item, call _cFileSelectFolder() with $sInitialTree set to '' (the default).

A few examples:

$sRoot	**$InitialTree**	**Selects**
'' or 'Desktop'	'Libraries'	Libraries
	'Libraries\Documents'	Documents
	'Libraries\Documents \My Documents'	My Documents
	My Documents	My Documents
'Computer'	'C:'	C:
	'C:\Program Files'	Program Files
	'C:\Program Files\AutoIt3'	Autoit3
'	'C:\Program Files\AutoIt3\Examples'	Examples
'C:\Program Files\AutoIt3'	'Examples	Examples
	'SciTE\api'	api

Declaring global variables in the calling script

There are two variables you can define globally to affect what _cFileSelectFolder does:

- If you declare Global **$g_FSF_ReturnErrorMessage** before you call _cFileSelectFolder(), if there is an error in your call, it sets @error to 1 and returns the error message rather than showing it in a dialog box.

- To aid in diagnosing problems in your call, if you declare **Global g_bFSF_ShowParameters** the function displays them, then exits.

Notification of Errors

This subject is treated in the UDF header; I provide a longer deception here for clarity.

Out of the box, _cFileSelectFolder() reports errors in a dialog box. Most of these errors pertain to $sRoot. Having reported an error, my script causes yours to exit. But this may not be what you want in a few cases. For example, if $sRoot is 'BUC180610:' and this is the label of a USB stick, but the user inserted the wrong stick before running your script, the message 'There is no drive labelled ...' will show.

But there is another way. You code `Global $g_sFSF_ReturnUserErrorMessage` before your call to `_cFileSeleectFolder()` . Now you won't see error messages; instead the script sets `@error` to 1 and returns them.

Limitations

- `Homegroup` is in the Desktop hierarchy but has not been implemented. This is because, not using `Homegroup`, I have no way of knowing how to code it. If the reader has this item in his hierarchy, perhaps he would like to add it to cFileSelectFolder, or tell me what is required, so I can add code to _cFileSelectFolder..
- cFileSelectFolder does not do libraries as FileSelectFolder() does, for example, it will not return `::{031E4825-7B94-4DC3-B131-E946B44C8DD5}\Documents.library-ms` when Documents is selected (as `FileSelectFolder does`). Rather, it shows, in this case, the two items below Documents: My Documents and Public Documents. The user can then choose. (It is debatable as to whether this is a limitation or a feature.)
- It is possible that the computer on which this script was developed, which runs Windows 7 SP1, is non-standard in some way (although I try to avoid customizing Windows). If there are problems with it on your computer, perhaps we can work together to resolve them.
- As mentioned above, `if $initialTree` is not `''`, its first or only element must be one that shows when _cFileSelectFolder is called with `$sInitialTree` set to `''`. This is for performance reasons: If the first or only element could be any folder at all, searching for it across your whole computer or network could take a long time.

Technical

I repeat: the types are 1. Whole Desktop, 2. Partial Desktop, 3. Local/mapped folder path, 4. `Computer\` plus local/mapped path, 5. UNC and 6. `Network\` plus UNC path.

There are four main functions. Which is run depends on what is in `$sRoot` :

Type	Main function	Called for `$sRoot` contains:	Data?
1, 2	`_FSF_InitializePerDesktop()`	Only Desktop hierarchy item(s)	yes
4, 6	`_FSF_InitializePerHybrid()`	Desktop items, drive, folders	yes
3	`_FSF_InitializePerFileSpec()`	Drives, folders	no
5	`_FSF_InitializePerUNC()`	UNC path	no

In the table above, Data? Indicates whether the Desktop hierarchy is used – as defined by `CreateDesktopHierarchyData()`. In this function, each item in the treeview is defined in a vector:

Index	Contents
0	text of treeview item
1	index of icon in image list
2	`'b'` for ⊞ expansion symbol to show beside item, else `''`
>= 3	names of child-item vectors

Even if element 2 is `'b'` the expansion symbol will only show if child items (Desktop items, computers, drives or folders) will show when the item is expanded. For example: for `'My Music'`, if there are subfolders of `'C:\Users\'&@UserName&'\Music'`.

When this symbol is clicked, `Expand_Item()` is called. If it finds a blank child item:

- If the parent item is `Computer`, it calls `_FSF_GetComputerDrives_TextAndIcon()`, adds the drives to the treeview, then deletes the blank item
- If the parent item is the CD/DVD drive, if the drive is not ready, it asks the user to insert a disk. If he does, it changes the text of the item to the label of the CD/DVD, adds the folders on the CD/DVD to the treeview, then deletes the blank item
- If the parent item is `Network`, it finds the names of the computers on the network and adds them to the treeview, each with a blank item after it so a computer can be expanded.
- If the grandparent item is `Network`, the parent is the name of a remote computer. The script finds the shares on that computer and adds them to the treeview, each with a blank item when the share can be expanded.
- In all other cases, it adds (sub)folders to the treeview, with blank items as appropriate[2].

Internally, all *paths*, independent of which of the 6 types of visible root, start with `Desktop` . Before the treeview is shown, $g_sPreTree is set to this partial *path* . A few examples:

Root begins with (or is)	`$g_sPreTree`
`'Desktop'`	`''`
`'Computer'` or `'Network'` or `@UserName`	'Desktop\'
'My Documents'	'Desktop\Libraries\Documents\'
@Username&'\Desktop'	`'Desktop\'`
`'C:'`	`'Desktop\Computer\'`
`'\\Petra\PDataH (G)\Tech'`	`'Desktop\Network\'`

This global variable is used by `_FSF_Expand_Item()` and `_FSF_GetPathForItem()` .

`GetPathforItem()` is the the function that converts Desktop-based *paths* to folder paths, e.g. in returning a value from `_cFileSelectFolder()`.

Operations involving the network (mapped drives and UNCs) are slower than operations only involving the local computer. For finding the names of computers on the network, calling WinNet resource functions is faster than running `net view` *`computer`* . Ping() is called to determine whther a remote computer is online.

Edit history

1.0.0	First release to AutoIt forum
1.0.1	Fixed bug in InitializePerDesktop() $g_sFSF_PreTree that caused 'Network' to fail
1.0.2	Added in-script documentation
1.0.3	InitializePerHybrid(): Shows folder icon for $sRoot ending with folder
1.0.4	InitializePerHybrid() > Fixed share name bug

2 I thank Melba23 for this method.

1.1.0	Now always returns selection path ending with \ : a script-breaking change Checks $sExclFolders parameter, removing spaces around `|` separators Faster checking of $sRoot parameter with removal of spaces around \ separators When $sRoot parameter ends with folder that contains no folders, shows this in treeview
1.2.0	Shows no-folders when only subfolder is deleted Corrects upper/lower-casing of drive/folder names Much faster in seeing network Sees non-persistently mapped drives Network > @ComputerName shows shares which are resolved to local paths
1.3.0	Added `$sInitialTree` to specify a treeview item to be selected initially, including diagnostic feature. A SCRIPT-BREAKING CHANGE Non-persistent mapped drives are handled Improved error reporting where a short-cut letter for a share does not exist Checks for access to remote computer by pinging it No longer depends on readiness of remote drive as an indication of accessibility Improved error reporting for shares Enumerating Resources now uses globally allocated buffer per Microsoft When `Global $g_bFSF_ShowArguments` is declared in user code, arguments/parameters are shown: a diagnostic aid
1.4.0	Added `$iFolder` parameter to make New folder, Delete folder and Rename folder buttons optional. A SCRIPT BREAKING CHANGE Added detection of errors in the types of parameter values
1.4.1	On entry, disables parent dialog and re-enables/activates it when function returns
1.4.2	When $sInitialTree is specified, collapses all other leaves at this level
1.5.0	Handles non-standard items under Desktop > @UserName
1.5.1	If $sRoot=`''` or `'Desktop'`, now, if connected to router shows Desktop once else not at all
1.6.0	_FSF_FillFolder(): To avoid erratic runtime errors, rewritten to avoid more than one file search at a time; now sorts folder names
1.6.1	_FSF_FillFolder(): To correct display of expand + signs, rewrote file attribute logic

Acknowledgements

This project began with examining, and experimenting with, Melba23's `ChooseFileFolder()`. I learnt much from his approach and code, some of which is in `cFileSelectFolder.au3`. I also thank him for `ExtMsgBox.au3` and `StringSize.au3`; My `VarSizeDialog()` is based on his `ExtMsgBox.au3`; my script incorporates `StringSize.au3`

I also thank many who have contributed to the Forum over the years, and those who have answered my questions there. They will see their contributions in parts of the code.

c.haslam