

# Manual of LuaFAR 3

---

## Brief description of LuaFAR package

This package's purpose is to facilitate writing plugins for FAR in Lua programming language. The package provides source code for **luafar3.dll** (main product) and a **luaplug.dll** ("generic" plugin).

**luafar3.dll** allows:

- To program functions exported by the plugin in Lua: e.g., the function [OpenW](#) should be named [export.Open](#) in the Lua program, the same goes for all functions exported by the plugin.
- To call FAR service functions from a Lua program. The service functions are placed in the `far`, `editor`, `viewer`, `panel` and `regex` tables, and their names aren't identical to the names of the respective FAR API functions. For instance, function [EditorControl](#) with parameter [ECTL\\_GETINFO](#) will be named [editor.GetInfo](#) in Lua programs.
- **Note:** the functions of the Unicode LuaFAR library assume string arguments and string return values to be in the **UTF-8** encoding. (There are a few exceptions from this rule, mostly when strings are used to exchange binary data).

Currently, the documentation on LuaFAR API is incomplete and may be somewhat inaccurate. When in doubt, consult the source code. The API may change in the future.

As **luafar3.dll** itself is not a plugin, it should not be placed in the `%FARHOME%\Plugins` subtree (instead, place it in any directory listed in `PATH`). The usual way of using **luafar3.dll** is to create a wrapper plugin that redirects FAR's calls of the exported functions to **luafar3.dll**, that in turn calls the functions written in Lua. This way, one instance of **luafar3.dll** can "serve" arbitrary number of plugins at the same time. An example of "wrapper plugin" is **luaplug.dll** (its source code is part of this package).

There are two Lua files that must be present in plugin's DLL directory:

1. **\_globalinfo.lua** (it should only contain the function [export.GetGlobalInfo](#)).

This script must not call functions from [far](#) library (e.g., far.Message), as they are not yet available to LuaFAR when Far calls [GetGlobalInfo](#).

2. The starting (or “default”) Lua script, that is searched in the following order (see [LF\\_RunDefaultScript](#) for details):

1. The plugin name with the extension .lua is looked for, e.g.: luaplug.dll -> luaplug.lua
2. If the plugin name contains a hyphen, then its part preceding the last hyphen is looked for, e.g.: luaplug-x64.dll -> luaplug.lua.

When the starting script is run it receives one boolean argument:

- true – on its first run and after Lua state was [recreated](#)
- false – on subsequent runs caused by setting [far.ReloadDefaultScript](#) = true.

# Flags

---

As Far API makes wide use of named integer constants, there is a necessity to represent those constants in Lua. LuaFAR keeps the names and values of the constants in tables. Most constants are embedded into LuaFAR, in the tables [far.Flags](#) and [far.Colors](#).

This manual uses the term **flags** for both a single constant contained in the table `far.Flags`, and for the [bitwise OR](#) of two or more such constants.

When some function parameter in this manual is denoted as *flags*, it can be supplied from Lua as either an integer, a string (when a combination consists of one flag), or a table, with keys corresponding to constants' names and having non-false values.

## Example 1 (single flag)

The following 2 calls are equivalent:

- (1) `editor.UndoRedo(nil, F.EUR_UNDO)` -- where `F=far.Flags`
- (2) `editor.UndoRedo(nil, "EUR_UNDO")`

## Example 2 (flags combination)

The 7-th parameter to function [editor.Editor](#) can be supplied from Lua in the following equivalent forms:

- (1) `bit64.bor(F.EF_NOMODAL, F.EF_IMMEDIATERETURN, ...)`
- (2) `{ EF_NOMODAL=1, EF_IMMEDIATERETURN=1, ... }`

# Unicode support

---

## Passing string arguments and returning string values

The functions of the Unicode LuaFAR library assume string arguments and string return values to be in the **UTF-8** encoding. (There are a few exceptions from this rule, mostly when strings are used to exchange binary data).

## Calling functions on strings with colon syntax

LuaFAR modifies the `__index` field of the strings metatable to point to `unicode.utf8` table of [Selene Unicode](#) library, rather than to `string` table. That means `("abc"):match("b")` is equivalent to `unicode.utf8.match("abc", "b")` rather than to `string.match("abc", "b")`. Operator `#` is not affected by this change.

## Unicode-capable Lua libraries and functions

The following standard Lua libraries and functions were modified by LuaFAR in order to work with file names and paths in UTF-8 encoding:

- `io`
- `dofile`
- `loadfile`
- `require`
- `package.loadlib`

## Unicode-capable functions using Windows API

- [`win.CopyFile`](#)
- [`win.CreateDirectory`](#)
- [`win.DeleteFile`](#)
- [`win.Getenv`](#)
- [`win.GetFileAttr`](#)
- [`win.GetFileInfo`](#)
- [`win.MoveFile`](#)

- [win.RemoveDir](#)
- [win.RenameFile](#)
- [win.SearchPath](#)
- [win.SetEnv](#)
- [win.SetFileAttr](#)
- [win.ShellExecute](#)
- [win.SHGetFolderPath](#)
- [win.system](#)

# Generic LuaFAR plugin

---

There are two files in the LuaFAR distribution (**luaplug.c** and **luaplug.mak**) that do not participate in the build of LuaFAR DLL, they comprise the “generic LuaFAR plugin”. It is called generic, since most of a LuaFAR plugin functionality is contained in its Lua part, and its C-part is more or less the same for most LuaFAR plugins.

- [Customizing the generic plugin](#)
- [Recreation of the Lua virtual machine](#)
- [Extra exported functions](#)
- [Other functions](#)

# Customizing the generic plugin

---

The easiest way to build a LuaFAR plugin DLL is via `luaplug.mak` makefile (MinGW is required). This makefile can be used either directly or included in another makefile. It accepts the following parameters:

- **TARGET** – name of the resulting DLL, e.g.: “`TARGET = helloworld.dll`”.
- **FAR\_EXPORTS** – list of functions that should be exported by the plugin (in uppercase, whitespace-separated, final letter `w` removed), e.g.: “`FAR_EXPORTS = OPEN PROCESSEDITORINPUT`”.  
The following functions are always exported, so there’s no need to include them in the list: [GetGlobalInfoW](#), [SetStartupInfoW](#), [GetPluginInfoW](#), [ExitFARW](#) and [ProcessSynchroEventW](#).
- **FARDIR** – the root directory of Far Manager source tree, e.g. “`FARDIR = C:\farmanager`”.
- **MYCFLAGS** – list of additional compilation flags
- **MYLDFLAGS** – list of additional linking-stage flags

## **luafar\_init.lua**

---

If the file %FARPROFILE%\luafar\_init.lua exists it is executed before the first run of the plugin's default script.

It is also executed when the Lua state is [recreated](#).

It is **not** executed when the default script is reloaded due to `far.ReloadDefaultScript == true`.

### **Note**

This feature is disabled by default. To enable it, add `-DRUN_LUAFAIR_INIT` to compilation flags.

See also:

[LF\\_LuafarInit](#)

[LF\\_RunDefaultScript](#)

[LF\\_Open](#)

## Recreation of the Lua virtual machine

---

It is possible to “recreate” the Lua virtual machine (`lua_State`), used by the plugin, i.e: to close the current one and create a new one.

- In order to do that the global variable `RecreateView` should be assigned a true value.
- The analysis of `RecreateView` and the virtual machine recreation are conducted in the `openw` plugin’s exported function, just before returning control to Far Manager.
- Prior to the virtual machine recreation the [`LF\_ExitFAR`](#) function is called.
- After the virtual machine is recreated all actions that normally occur when Far Manager loads a LuaFAR plugin are conducted, including running the plugin’s “starting” Lua script (which in this case receives `true` as an argument).
- After the virtual machine is recreated the global variable `IsLuaStateRecreated` is set to true.

## **Extra exported functions**

---

## **GetLuaState**

---

```
lua_State* GetLuaState()
```

Returns the current main `lua_State` of the plugin.

Do not store the returned value for further use, as it may be invalidated by [recreation](#). Instead, call this function again whenever needed.

## **luaopen\_luaplug**

---

Some libraries (e.g. `rings`) can create additional `lua_State` instances from Lua scripts. To initialize those instances with libraries included in LuaFAR (`far`, `bit64`, `win`, `unicode`), call `luaopen_luaplug()` passing it the `lua_State*` to be initialized.

This function is also accessible from Lua as the global variable `_luaplug`. It is NOT intended to be called from the running `lua_State` but rather to be passed to another `lua_State` for its initialization.

Alternatively, this function can be loaded via `package.loadlib` as in the following example.

### **Example:**

```
local rings = require "rings"
local r1 = rings.new()
r1:dostring([
  local modname = ...
  assert(package.loadlib(modname, "luaopen_luaplug"))()
  far.Message ("Hello world", "Called from a ring")
]], far.PluginStartupInfo().ModuleName)
```

**Note:** After replacing the standard Lua 5.1 DLL with one from LuaJIT 2.0, this example has stopped working. The issue is investigated.

## **Other functions**

---

## **GetPluginStartupInfo**

---

```
struct PluginStartupInfo* GetPluginStartupInfo()
```

This helper function is needed when the plugin consists of more than one C-file.  
It gives access to Far API functions to files other than `luaplug.c`.

(The same could be achieved by making the struct accessible, but it is not.)

## **What's new in this version**

---

# Version 2.8.0

---

## Added functions

1. [far.GetDriveType](#)
2. [far.GetLogicalDriveStrings](#)

## Extended functions

1. [far.AdvControl](#): ACTL\_KEYMACRO: field AKey added
2. [far.GetPluginInfo](#): field SysId added.
3. [far.OpenPlugin](#): handles [OPEN\\_FROMMACRO](#)
4. [far.PluginsControl](#): [PCTL\\_FORCELOADPLUGIN](#) added.
5. [far.SendDlgMessage](#): (a) [DM\\_LISTGETDATA](#) fixed, (b) [DM\\_LISTSETDATA](#) implemented.

## Bug fixes

1. [far.DialogRun](#): errors in the dialog procedure were ignored (did not prevent the script from normal run flow).
2. [far.SetRegKey](#): data corruption with (DataType=="string").

# Version 2.7.0

---

- The first LuaFAR version that compiles for both 32 and 64-bit DLLs.
- Generic plugin [customization](#): added a new parameter ENV\_PREFIX.
- Source code: all C++ files were rewritten in C.

## Added functions

1. [`far.CtrlSetCaseSensitiveSort`](#)
2. [`far.EnumSystemCodePages`](#)
3. [`far.GetACP`](#)
4. [`far.GetCPInfo`](#)
5. [`far.GetOEMCP`](#)

## Extended functions

1. [`far.SendDlgItemMessage`](#): `DM_GETCONSTTEXTPTR` added.
2. [`far.MultiByteToWideChar`](#): *flags* parameter added.
3. [`LF\_RunDefaultScript`](#): plugin DLL's name can have a suffix that is ignored in searching the default script.

## Modified functions

1. Implementation of [`far.OemToUtf8`](#) and [`far.Utf8ToOem`](#) changed.

## Bug fixes

1. Fixes related to 64-bit compilation.

# **Version 2.6.0**

---

## **Added functions**

1. [regex:gsub](#)
2. [LF\\_InitLuaState](#)
3. [luaopen\\_luaplug](#)

## **Modified functions**

1. [far.Editor](#) — default values for StartLine, StartChar made -1 (was: 0).
2. [far.GetPluginInfo](#) — field DiskMenuNumbers removed from the returned table, as it does not exist in Far builds >= 2.0.1692.

# Version 2.5.0

---

## Added functions

1. [`far.CtrlGetPanelFormat`](#)
2. [`far.CtrlGetPanelHostFile`](#)
3. [`far.DialogInit`](#)
4. [`far.DialogRun`](#)
5. [`far.DialogFree`](#)
6. [`regex:bracketscount`](#)

## Extended functions

1. [`far.AdvControl`](#) — added [`ACTL\_PROGRESSNOTIFY`](#).

## Modified functions

1. All [`regular expression functions`](#) renamed to lower case.
2. [`far.gsub`](#) – “invalid capture number” error in the replace pattern is detected before entering the search/replace loop.
3. [`LF\_RunDefaultScript`](#) – better error handling.

## Bug fixes

1. [`far.gsub`](#) – fix crash (poorly handled % at the end of the replace string).
2. [`far.GetDlgItem`](#), [`far.SendDlgMessage \(DM\_GETDLGITEM\)`](#) – fix crash with invalid item number.

## Misc

1. `LF_GetMinFarVersionW` function is removed.
2. New setting `MINFARVERSION` in the [`generic plugin's`](#) makefile.

# **Version 2.4.0**

---

## **Added functions**

1. [far.MultiByteToWideChar](#)
2. [far.CreateDir](#) (works with UTF-8 names)
3. [far.RemoveDir](#) (works with UTF-8 names)

## **Extended functions**

1. [far.EditorSetParam](#) – 2-nd parameter can be boolean.

## **Modified functions**

1. [far.ViewerSetPosition](#) – actual value of StartPos is returned (number; was: boolean).

## **Bug fixes**

1. [far.EditorSetParam](#) – [ESPT\\_GETWORDDIV](#) did not work.
2. [far.AdvControl](#) – [ACTL\\_GETSYSWORDDIV](#) did not work.

## **Misc**

1. Version information is embedded into the DLL (.rc file added).

# Version 2.3.0

---

## Added functions

1. [`far.GetConsoleScreenBufferInfo`](#)
2. [`far.Timer`](#)
3. [`far.GetCustomData`](#) (experimental)
4. [`far.CopyFile`](#) (works with UTF-8 names)
5. [`far.DeleteFile`](#) (works with UTF-8 names)
6. [`far.MoveFile`](#) (works with UTF-8 names)
7. [`far.RenameFile`](#) (works with UTF-8 names)

## Extended functions

1. [`far.Message`](#) – standard button combinations supported.
2. [`far.AdvControl`](#) – added [`ACTL\_GETFARRECT`](#), [`ACTL\_GETCURSORPOS`](#), [`ACTL\_SETCURSORPOS`](#).

## Modified functions

1. [`far.Editor`](#) – autodetect code page if not specified.
2. [`far.Viewer`](#) – autodetect code page if not specified.
3. [`far.EditorGetSelection`](#) – API (table fields) changed.

## Bug fixes

1. [`far.EditorGetString`](#) – work with strings containing nuls.
2. [`far.Gsub`](#) – fix crash (a missing `lua_pop` call).
3. [`far.Gsub`](#) – fix ignoring errors in replace function.
4. [`far.Message`](#) – fix to work with Far 2.0 builds 1564+.
5. [`far.SendDlgMessage \(DM\_LISTGETCURPOS\)`](#) – returned `nil` when `SendDlgMessage` returned 0.
6. [`far.ViewerSetPosition`](#) – fix case with extra arguments.

# Version 2.2.0

---

## Added functions

1. [`far.CtrlClearSelection`](#)
2. [`far.CtrlGetCmdLineSelection`](#)
3. [`far.CtrlIsActivePanel`](#)
4. [`far.CtrlSetCmdLineSelection`](#)
5. [`far.LoadFile`](#)
6. [`far.LoadLib`](#)
7. [`far.PluginsControl`](#)
8. [`far.ProcessSynchroEvent`](#)
9. [`far.Require`](#)
10. [`far.XLat`](#)
11. [`File Filter Control`](#) – a function and 4 methods
12. [`uio` library](#)

## Extended functions

1. [`far.AdvControl`](#) – ACTL\_KEYMACRO – added support for MCMD\_CHECKMACRO
2. [`far.AdvControl`](#) – ACTL\_SYNCHRO added
3. [`far.OpenFilePlugin`](#) – return value -2 is passed on to Far unchanged (see Far API)
4. [`far.OpenPlugin`](#) – return value 0 is passed on to Far unchanged (see Far API)
5. [`far.ProcessDialogEvent`](#) – second return added
6. [`far.SendDlgMessage`](#) – DM\_GETDIALOGINFO added

## Modified functions

1. [`far.CtrlGetCmdLinePos`](#) – position made 1-based
2. [`far.CtrlSetCmdLinePos`](#) – position made 1-based
3. [`far.EditorSetTitle`](#) – nil argument is allowed
4. [`far.ProcessKey`](#) – API changed: made closer to Far API
5. [`far.SendDlgMessage`](#): changed return values for DM\_RESIZEDIALOG and DM\_MOVEDIALOG.

6. [LF\\_LuaOpen](#) – API changed: type of the 3-rd parameter
7. [Regular expression](#) functions: the optimisation stage is skipped now, as it is too slow.

# Version 2.1.0

---

## Added functions

1. [`far.ConvertPath`](#)
2. [`far.EditorUndoRedo`](#)
3. [`far.GetVirtualKeys`](#)
4. [`far.MakeMenuItems`](#)
5. [`far.Show`](#)

## Extended functions

1. [`far.AdvControl`](#) — commands [`ACTL\_QUIT`](#), [`ACTL\_SETPROGRESSSTATE`](#) and [`ACTL\_SETPROGRESSVALUE`](#) added.
2. [`far.Dialog`](#) — more conforming processing of the following events:  
[`DN\_CTLCOLORDLGITEM`](#), [`DN\_DRAWDLGITEM`](#), [`DN\_EDITCHANGE`](#),  
[`DN\_GETDIALOGINFO`](#), [`DN\_HELP`](#), [`DN\_MOUSECLICK`](#), [`DN\_MOUSEEVENT`](#),  
[`DN\_RESIZECONSOLE`](#).
3. [`far.Menu`](#) — *checked* property of a menu item can be a string (only its first character is used).

## Modified functions

1. [`LF\_LuaOpen`](#)

## **Version 2.0.1**

---

1. **Selene Unicode** library: fixed a serious bug in the function `gmatch`.

## **Version 2.0.0**

---

This is the first release of the **Unicode LuaFAR** – intended for use by plugins working with Unicode Far (Far 2.0).

# What's new in Unicode version

---

## Passing string arguments and returning string values

The functions of the Unicode LuaFAR library assume string arguments and string return values to be in the **UTF-8** encoding. (There are a few exceptions from this rule, mostly when strings are used to exchange binary data).

## Removed functions

1. `far.ConvertNameToReal`
2. `far.CtrlGetPanelShortInfo`
3. `far.DetectCharTable`
4. `far.EditorEditorToOEM`
5. `far.EditorLower`
6. `far.EditorOEMToEditor`
7. `far.EditorStrCmp`
8. `far.EditorStrICmp`
9. `far.EditorUpper`
10. `far.ExpandEnvironmentStr`
11. `far.GetCharTable`

## Changed structures and functions

1. [FindData](#)
2. [PluginPanelItem](#)
3. [ViewerInfo.CurMode](#)
4. [far.AdvControl](#)
5. [far.CtrlGetPanelInfo](#)
6. [far.CtrlSetSelection](#)
7. [far.CtrlSetSortOrder](#)
8. [far.Editor](#)
9. [far.EditorGetInfo](#)
10. [far.OpenFilePlugin](#)
11. [far.Viewer](#)
12. [far.ViewerGetInfo](#)

13. [LF\\_LuaOpen](#)

## Added functions

1. [far.CompareString](#)
2. [far.CtrlGetColumnTypes](#)
3. [far.CtrlGetColumnWidths](#)
4. [far.CtrlGetCurrentPanelItem](#)
5. [far.CtrlGetPanelDir](#)
6. [far.CtrlGetPanelItem](#)
7. [far.CtrlGetSelectedPanelItem](#)
8. [far.EditorGetFileName](#)
9. [far.Find](#)
10. [far.GetCurrentDirectory](#)
11. [far.Gmatch](#)
12. [far.Gsub](#)
13. [far.Match](#)
14. [far.OemToUtf8](#)
15. [far.Regex](#)
16. [far.Utf8ToOem](#)
17. [far.Utf16ToUtf8](#)
18. [far.Utf8ToUtf16](#)
19. [far.wcscmp](#)

## Version 1.0.1

---

- [`far.CtrlGetPanelInfo`](#) and [`far.CtrlGetPanelShortInfo`](#): fields `ItemsNumber` and `SelectedItemsNumber` added to the return table.
- [`far.Menu`](#): fields `grayed` and `hidden` added to menu item.
- Several bug fixes.

# Version 1.0.0

---

## Important:

- This version of LuaFAR adds functions and constants from FAR 1.75 API, that are not present in FAR 1.70 API.
- LuaFAR plugins that make no use of this API will work fine under FAR 1.70. However, the library itself does no verification of whether some feature is supported by the running FAR version – it is the plugin's responsibility.

## Changes:

1. [Auxiliary files](#): *farkeys.lua* and *farcolor.lua* updated to FAR 1.75 build 2614.
2. [far.AdvControl](#): `ACTL_REDRAWALL` command supported.
3. [far.CtrlGetUserScreen](#): service function added.
4. [far.EditorAddStackBookmark](#): service function added.
5. [far.EditorClearStackBookmarks](#): service function added.
6. [far.EditorDeleteStackBookmark](#): service function added.
7. [far.EditorGetStackBookmarks](#): service function added.
8. [far.EditorNextStackBookmark](#): service function added.
9. [far.EditorPrevStackBookmark](#): service function added.
10. [far.GetFlags](#): added new flags, covering FAR API up to FAR 1.75 build 2614.
11. [far.Message](#)
  - new flag ‘o’ added.
  - removed hard limit of message box sizes.
12. [far.OpenPlugin](#): added support for opening plugin from dialogs.
13. [far.ProcessDialogEvent](#): exported function added.
14. [far.ProcessEditorEvent](#): `EE_GOTFOCUS` and `EE_KILLFOCUS` commands supported.
15. [far.ProcessViewerEvent](#): `VE_GOTFOCUS` and `VE_KILLFOCUS` commands supported.
16. [far.SendDlgItemMessage](#): `DM_GETCOMBOBOXEVENT`, `DM_GETEDITPOSITION`, `DM_SetComboBoxEvent` and `DM_SetEditPosition` commands supported.

17. [far.ViewerSetMode](#): service function added.
18. [far.AdvControl](#) (ACTL\_GETFARVERSION): result can be obtained either as a string, or as 3 numbers.
19. [far.LuafarVersion](#): result can be obtained either as a string, or as 3 numbers.

# **Version 0.9.0**

---

## **Service Functions added:**

1. [far.DefDlgProc](#)

## **FAR Standard Functions added:**

1. [far.ConvertNameToReal](#)
2. [far.FarRecursiveSearch](#)
3. [far.GetPathRoot](#)
4. [far.GetReparsePointInfo](#)
5. [far.LIsAlpha](#)
6. [far.LIsAlphanum](#)
7. [far.LIsLower](#)
8. [far.LIsUpper](#)
9. [far.LLowerBuf](#)
10. [far.LUpperBuf](#)
11. [far.MkLink](#)
12. [far.MkTemp](#)
13. [far.TruncPathStr](#)
14. [far.TruncStr](#)

## **Other functions added:**

1. [far.FileTimeToSystemTime](#)
2. [far.GetFileInfo](#)
3. [far.GetSystemTime](#)
4. [far.GetTimeZoneInformation](#)
5. [far.SystemTimeToFileTime](#)

## **Changes:**

1. [FindData](#) structure: some fields changed.
2. [PluginPanelItem](#) structure: one field changed.

3. [`far.Dialog`](#) function: changed meaning of [`DlgProc`](#) return values.
4. The [generic LuaFAR plugin](#) will export only functions listed explicitly in the makefile.

# Version 0.8.0

---

The following new functions were added:

1. [far.FarInputRecordToKey](#)
2. [far.LStricmp](#)
3. [far.LStrnicmp](#)
4. [far.LuafarVersion](#)
5. [far.ProcessName](#)
6. [far.SetEnv](#)

The following functions were enhanced:

1. [far.GetRegKey](#), [far.SetRegKey](#): added support of REG\_EXPAND\_SZ and REG\_MULTI\_SZ data types.

## **Version 0.7.2**

---

1. [far.CPluginStartupInfo](#) – new function added

## Version 0.7

---

1. [far.SendDlgMessage](#) – fixed and extended (practically all message types are supported).
2. [far.Dialog](#) – added support for the following controls: [DI\\_LISTBOX](#) and [DI\\_COMBOBOX](#)

## Version 0.6

---

1. [far.AdvControl](#) – added
2. [far.GetEnv](#) – added
3. [far.ProcessEditorInput](#) – extended
4. [farcolor.lua](#) – module added
5. [Lua BitOp](#) – library updated to version 1.0.1

## **Version 0.5.1**

---

1. Added virtual keys that formerly were not available (keys with punctuation marks).

# Version 0.5

---

## What's added in LuaFAR 0.5:

1. [Lua BitOp library](#)
2. [far.CtrlCheckPanelsExist](#)
3. [far.CtrlSetNumericSort](#)
4. [far.CtrlSetSortMode](#)
5. [far.CtrlSetSortOrder](#)
6. [far.EditorGetBookmarks](#)
7. [far.EditorProcessKey](#)
8. [far.EditorSetKeyBar](#)
9. [far.EditorSetParam](#)
10. [far.EditorTurnOffMarkingBlock](#)
11. [far.OnError](#)
12. [far.ViewerGetInfo](#)
13. [far.ViewerQuit](#)
14. [far.ViewerRedraw](#)
15. [far.ViewerSelect](#)
16. [far.ViewerSetKeyBar](#)
17. [far.ViewerSetPosition](#)

## **Incompatibilities with the Previous Version**

---

LuaFAR 3.0 is incompatible with versions 2.x in numerous aspects, due to the move from Far 2 API to Far 3 API.

## Version 2.6.0

---

1. [`far.GetPluginInfo`](#) — field `DiskMenuNumbers` removed from the returned table, as it does not exist in Far builds  $\geq 2.0.1692$ .

## **Version 2.5.0**

---

1. `far.Find` — renamed to [`far.find`](#)
2. `far.Gmatch` — renamed to [`far.gmatch`](#)
3. `far.Gsub` — renamed to [`far.gsub`](#)
4. `far.Match` — renamed to [`far.match`](#)
5. `far.Regex` — renamed to [`far.regex`](#)
6. `LF_GetMinFarVersionW` — removed.

## Version 2.4.0

---

1. [`far.ViewerSetPosition`](#) – real start position number is returned (was: boolean).

## Version 2.3.0

---

1. [`far.EditorGetSelection`](#) – API (table fields) changed.
2. [`far.Message`](#) – flags 'd' and 'o' are deprecated and no more supported.

## Version 2.2.0

---

1. [LF\\_LuaOpen](#) – API changed (type of the 3-rd parameter). **All LuaFAR plugins (except ANSI ones) must be rebuilt to work with this version.**
2. [far.CtrlGetCmdLinePos](#) – position is 1-based
3. [far.CtrlSetCmdLinePos](#) – position is 1-based
4. [far.ProcessKey](#) – API changed: made closer to Far API
5. [far.SendDlgMessage](#) – [DM\\_RESIZEDIALOG](#) – return value is a table
6. [far.SendDlgMessage](#) – [DM\\_MOVEDIALOG](#) – return value is a table

# Version 2.1.0

---

## Dialog Messages:

1. [\*\*DM\\_LISTADDSTR\*\*](#) – return value is 1-based
2. [\*\*DM\\_LISTDELETE\*\*](#) – StartIndex is 1-based
3. [\*\*DM\\_LISTFINDSTRING\*\*](#) – StartIndex and return value are 1-based; a *nil* is returned if no match found
4. [\*\*DM\\_LISTGETCURPOS\*\*](#) – SelectPos and TopPos are 1-based
5. [\*\*DM\\_LISTgetitem\*\*](#) – ItemIndex is 1-based
6. [\*\*DM\\_LISTINFO\*\*](#) – SelectPos and TopPos are 1-based
7. [\*\*DM\\_LISTINSERT\*\*](#) – Index is 1-based; a *nil* is returned on failure
8. [\*\*DM\\_LISTUPDATE\*\*](#) – Index is 1-based; a *boolean* is returned
9. [\*\*DM\\_LISTSETCURPOS\*\*](#) – SelectPos and TopPos are 1-based

## Dialog Events:

1. [\*\*DN\\_LISTCHANGE\*\*](#) – Param2 is 1-based
2. [\*\*DN\\_LISTHOTKEY\*\*](#) – Param2 is 1-based

# Version 2.0.0

---

1. When some *flags* value passed to LuaFAR API is of table type, it should be a dictionary (was: an array).

**NOTE:** This change breaks many existing scripts, which must be fixed in order to work with the new flags format.

**Example:**

Was: {"DIF\_BOXCOLOR", "DIF\_SEPARATOR"}

Now: {DIF\_BOXCOLOR=1, DIF\_SEPARATOR=1}

2. There are many incompatibilities with the non-Unicode LuaFAR versions.

See [What's new in Unicode version](#).

## Version 1.0.0

---

1. [ViewerInfo.CurMode](#): member TypeWrap renamed to WordWrap.

# Version 0.9.0

---

1. [FindData](#) structure:
  - Field **dwFileAttributes** (integer) replaced by field **FileAttributes** (string).
  - Fields **nFileSizeLow** and **nFileSizeHigh** replaced by field **FileSize**.
  - Field **cFileName** renamed to **FileName**
  - Field **cAlternateFileName** renamed to **AlternateFileName**
  - Fields **LastWriteTimeL** and **LastWriteTimeH** replaced by field **LastWriteTime**
  - Fields **LastAccessTimeL** and **LastAccessTimeH** replaced by field **LastAccessTime**
  - Fields **CreationTimeL** and **creationTimeH** replaced by field **CreationTime**
2. [PluginPanelItem](#) structure:
  - Fields **PackSize** and **PackSizeHigh** replaced by field **PackSize**.
3. [far.Dialog](#) function:
  - Changed meaning of [DlgProc](#) return values.

## **Version 0.8.0**

---

No new incompatibilities introduced.

## **Version 0.7.2**

---

This is a bug-fix release; no new incompatibilities introduced.

## Version 0.7

---

1. [`far.DeleteRegKey`](#), [`far.GetRegKey`](#), [`far.SetRegKey`](#): the Key argument is treated as relative to the FAR registry key (which is usually `HKEY_CURRENT_USER\Software\Far`).
2. Function `far.GetDlgItems` is removed. Use [`far.GetDlgItem`](#) in cycle instead.

## Version 0.6

---

1. [far.Menu](#): 1-st argument must be a table.
2. [far.Menu](#): field *text* in break key tables renamed to *BreakKey*.

## Version 0.5.1

---

1. Changed string representations of “virtual key codes” (Win32): removed “VK\_” prefix, e.g., “VK\_TAB” -> “TAB”. This change affects the following functions:
  - o [far.ExtractKey](#)
  - o [far.Menu](#)
  - o [far.ProcessEditorInput](#)
  - o [far.ProcessKey](#)

# Version 0.5

---

1. Function [LF\\_LuaOpen](#):

- Name of an environment variable to process made a parameter (was: LUA\_INIT)
- Function prepends <plugin directory>\?.lua; to package.path
- File %FARHOME%\luafar\_init.lua is not run (and is not treated as a special file anymore)

2. Functions far.band, far.bnot, far.bor and far.bxor were removed. Instead, [Lua BitOp library](#) is included.

# Version 0.4

---

1. far.EditorGetSelection:  
**Info.BlockType**: was string, now integer
2. far.GetPluginInfo:  
**PluginInfo.Flags**: was table, now integer
3. far.GetFindData, far.GetFiles, far.DeleteFiles, far.CreateDirectory  
far.ProcessHostFile, far.PutFiles, far.SetDirectory:  
**OpMode**: was table, now integer
4. far.Menu: every menu item must be table type  
(was: if it's not a table then it's assumed to be a menu separator)
5. far.Menu: the ["text"] field of an item is always treated as a string  
(was: if it's a number, then [GetMsg\(n\)](#) was called).

## **LuaFAR plugin-exported functions**

---

## Common functions

---

## **export.Configure**

---

```
result = export.Configure (Guid)
```

**Parameters:**

    Guid:   string

**Returns:**

    result: boolean

**Far API used:**

[ConfigureW](#), [ConfigureInfo](#)

## **export.ExitFAR**

---

`export.ExitFAR()`

**Parameters:**

none

**Returns:**

nothing

**Far API used:**

[ExitFARW](#), [ExitInfo](#)

## **export.GetGlobalInfo**

---

```
GInfo = export.GetGlobalInfo ()
```

**Parameters:**

none

**Returns:**

GInfo: table	
fields of GInfo:	
MinFarVersion:	table (array of 5 integers)
Version:	table (array of 5 integers)
Guid:	string (binary GUID)
Title:	string
Description:	string
Author:	string

**Far API used:**

[GetGlobalInfow](#), [GlobalInfo](#)

# **export.GetPluginInfo**

---

```
PInfo = export.GetPluginInfo ()
```

**Parameters:**

none

**Returns:**

PInfo: table

fields of PInfo:

Flags: flags

DiskMenuStrings: array of strings

DiskMenuGuids: string (concatenated GUIDs)

PluginMenuStrings: array of strings

PluginMenuGuids: string (concatenated GUIDs)

PluginConfigStrings: array of strings

PluginConfigGuids: string (concatenated GUIDs)

CommandPrefix: string

**Far API used:**

[GetPluginInfoW](#), [PluginInfo](#)

# export.Open

---

```
ret = export.Open (OpenFrom, Guid, Item)
```

## Definitions:

**"int64"**

Type of userdata, produced by [bit64](#) library.

**"binary"**

By convention, it is a string enclosed in a table as its first element.  
The string is intended to be passed to and from without conversion.

**"panel"**

By convention, it is a panel handle (any non-false Lua value) enclosed in a table as its first element. The table must contain the field ["type"] =

## Parameters:

OpenFrom: integer

Guid: string

Item:

```
if OpenFrom == OPEN FROMMACRO:
```

```
    table (array of arguments):
```

- each argument can be either of:

- string, nil, boolean, number, "int64", or "binary"

- field *n* of the table holds the length of the array

```
if OpenFrom == OPEN COMMANDLINE:
```

```
    string
```

```
if OpenFrom == OPEN SHORTCUT:
```

```
    table:
```

```
        HostFile: string
```

```
        ShortcutData: string
```

```
        Flags: flags
```

```
if OpenFrom == OPEN DIALOG:
```

```
    table:
```

```
        hDlg: userdata
```

```
if OpenFrom == OPEN ANALYSE:
```

```
    table tAnalyseInfo
```

```
if OpenFrom == something else:
```

```
    integer
```

## Returns:

```
if OpenFrom == OPEN FROMMACRO:
```

0 or more Lua values.

Currently, values of the following types are supported:

- string, nil, boolean, number, "int64", "binary", or "panel".

To open a panel, return a "panel" value as the first return value.

```
if OpenFrom == something else:  
    ret: any Lua value; it is processed by LuaFAR as follows:  
    - Nil and false: NULL is returned to Far.  
    - Number -1: PANEL_STOP is returned to Far.  
    - Otherwise: ret is considered a panel object, stored in the Lua  
        registry, its registry reference is returned to Far.  
        The object is passed to other exported functions when they are  
        called by Far.
```

**Note:**

Return value of -1 can be used with *OpenFrom* == [OPEN ANALYSE](#),  
when the plugin does actions other than opening a panel, to tell  
Far that the file has already been processed.

**Far API used:**

[OpenW](#), [OpenInfo](#)

# export.ProcessConsoleInput

---

```
result = export.ProcessConsoleInput (rec, flags)
```

**Parameters:**

  rec:   table [tInputRecord](#)  
  flags: flags

**Returns:**

  result: 0, 1, or table [tInputRecord](#)

**Details:**

- 0/false/nil/nothing result in returning 0 to Far Manager
- table results in returning 2 to Far Manager
- any other value results in returning 1 to Far Manager

**Far API used:**

[ProcessConsoleInputW](#), [ProcessConsoleInputInfo](#)

## **export.ProcessSynchroEvent**

---

```
ret = export.ProcessSynchroEvent (event, param)
```

**Parameters:**

  event: integer  
  param: integer

**Returns:**

  ret: integer

**Far API used:**

[ProcessSynchroEventW](#), [ProcessSynchroEventInfo](#)

## **export.SetStartupInfo**

---

**This function is not called by LuaFAR**

The C-language part of a plugin should implement and export function [SetStartupInfo](#). Within this function, it should call:

1. [LF\\_InitLuaState2](#)
2. [LF\\_RunDefaultScript](#)

## Panel functions

---

# **export.Analyse**

---

```
Handle = export.Analyse (Data)
```

**Parameters:**

  Data:   table [tAnalyseInfo](#)

**Returns:**

  Handle: any Lua value, or nothing

**Description:**

The returned value of false/nil/nothing means the plugin will not write to the file. Otherwise, the value is stored in the registry, may be passed to the plugin in [export.Open](#) and will be removed from the registry either [export.Open](#) or [LF.CloseAnalyse](#).

**Far API used:**

[AnalyseW](#), [AnalyseInfo](#)

## **export.CloseAnalyse**

---

**This function is not called by LuaFAR**

See Far API: [CloseAnalyseW](#)

## **export.ClosePanel**

---

`export.ClosePanel (object, handle)`

**Parameters:**

object: plugin object  
handle: panel handle

**Returns:**

nothing

**Far API used:**

[ClosePanelW](#), [ClosePanelInfo](#)

## **export.Compare**

---

```
result = export.Compare (object, handle, PanelItem1, PanelItem2, Mod
```

**Parameters:**

object: plugin object  
handle: panel handle  
PanelItem1: table [tPluginPanelItem](#)  
PanelItem2: table [tPluginPanelItem](#)  
Mode: integer

**Returns:**

result: integer

**Far API used:**

[CompareW](#), [CompareInfo](#)

## **export.DeleteFiles**

---

```
result = export.DeleteFiles (object, handle, PanelItems, OpMode)
```

**Parameters:**

object: plugin object  
handle: panel handle  
PanelItems: table (array of [tPluginPanelItem](#))  
OpMode: flags

**Returns:**

result: boolean

**Far API used:**

[DeleteFilesW](#), [DeleteFileInfo](#)

## **export.FreeFindData**

---

**This function is not called by LuaFAR**

See Far API: [FreeFindDataW](#)

## **export.GetFiles**

---

```
result [, newDestPath] = export.GetFiles (object, handle, PanelItems  
                                         Move, DestPath, OpMode)
```

**Parameters:**

object: plugin object  
handle: panel handle  
PanelItems: table (array of [tPluginPanelItem](#))  
Move: boolean  
DestPath: string  
OpMode: flags

**Returns:**

result: integer  
newDestPath: string (optional)

**Far API used:**

[GetFilesW](#), [GetFileInfo](#)

## **export.GetFindData**

---

```
Items = export.GetFindData (object, handle, OpMode)
```

**Parameters:**

object: plugin object  
handle: panel handle  
OpMode: flags

**Returns:**

Items: table (array of [tPluginPanelItem](#))

**Far API used:**

[GetFindDataW](#), [GetFindDataInfo](#)

# export.GetOpenPanelInfo

---

```
OPInfo = export.GetOpenPanelInfo (object, handle)
```

**Parameters:**

object: plugin object  
handle: panel handle

**Returns:**

OPInfo: table  
fields of OPInfo:  
Flags: flags  
HostFile: string  
CurDir: string  
Format: string  
PanelTitle: string  
InfoLines: table (array of [tInfoLine](#) tables)  
InfoLinesNumber: integer  
DescrFiles: table (array of strings)  
PanelModesArray: table (array of [tPanelMode](#) tables)  
PanelModesNumber: integer  
StartPanelMode: integer  
StartSortMode: flag  
StartSortOrder: integer  
KeyBar: table (array of [tKeyBarLabel](#) tables)  
ShortcutData: string  
FreeSize: number

**Far API used:**

[GetOpenPanelInfoW](#), [OpenPanelInfo](#)

## **export.MakeDirectory**

---

```
Status [, NewName] = export.MakeDirectory (object, handle, Name, OpM
```

**Parameters:**

object: plugin object  
handle: panel handle  
Name: string  
OpMode: flags

**Returns:**

Status: integer (in accordance with FAR API)  
NewName: string

**Far API used:**

[MakeDirectoryW](#), [MakeDirectoryInfo](#)

## **export.ProcessHostFile**

---

```
result = export.ProcessHostFile (object, handle, Items, OpMode)
```

**Parameters:**

object: plugin object  
handle: panel handle  
Items: table (array of [tPluginPanelItem](#))  
OpMode: flags

**Returns:**

result: boolean

**Far API used:**

[ProcessHostFileW](#), [ProcessHostFileInfo](#)

## **export.ProcessPanelEvent**

---

```
result = export.ProcessPanelEvent (object, handle, Event, Param)
```

**Parameters:**

object: plugin object  
handle: panel handle  
Event: integer  
Param: string (for [FE\\_CHANGEVIEWMODE](#) and [FE\\_COMMAND](#)), or nil

**Returns:**

result: boolean

**Far API used:**

[ProcessPanelEventW](#), [ProcessPanelEventInfo](#)

## **export.ProcessPanelInput**

---

```
result = export.ProcessPanelInput (object, handle, rec)
```

**Parameters:**

object: plugin object  
handle: panel handle  
rec: table [tInputRecord](#)

**Returns:**

result: boolean

**Far API used:**

[ProcessPanelInputW](#), [ProcessPanelInputInfo](#)

## **export.PutFiles**

---

```
result = export.PutFiles (object, handle, Items, Move, SrcPath, OpMo
```

**Parameters:**

object: plugin object  
handle: panel handle  
Items: table (array of [tPluginPanelItem](#))  
Move: boolean  
SrcPath: string  
OpMode: flags

**Returns:**

result: integer

**Far API used:**

[PutFilesW](#), [PutFileInfo](#)

## **export.SetDirectory**

---

```
result = export.CreateDirectory (object, handle, Dir, OpMode, UserData)
```

**Parameters:**

object: plugin object  
handle: panel handle  
Dir: string  
OpMode: flags  
UserData: any Lua value

**Returns:**

result: boolean

**Far API used:**

[SetDirectoryW](#), [SetDirectoryInfo](#)

## **export.SetFindList**

---

```
result = export.SetFindList (object, handle, Items)
```

**Parameters:**

object: plugin object  
handle: panel handle  
Items: table (array of [tPluginPanelItem](#))

**Returns:**

result: boolean

**Far API used:**

[SetFindListW](#), [SetFindListInfo](#)

## **Editor functions**

---

## **export.ProcessEditorEvent**

---

```
result = export.ProcessEditorEvent (EditorID, Event, Param)
```

**Parameters:**

EditorID: integer  
Event: integer  
Param: - table (for Event == [EE\\_CHANGE](#)):  
          Type:       integer  
          StringNumber: integer (1-based)  
- table (for Event == [EE\\_SAVE](#))  
          FileName:   string  
          FileEOL:     string  
          CodePage:    integer  
- integer (for other events)

**Returns:**

result: integer

**Far API used:**

[ProcessEditorEventW](#), [ProcessEditorEventInfo](#)

## **export.ProcessEditorInput**

---

```
result = export.ProcessEditorInput (Rec)
```

**Parameters:**

  Rec:     table [tInputRecord](#)

**Returns:**

  result: boolean

**Far API used:**

[ProcessEditorInputW](#), [ProcessEditorInputInfo](#)

## **Viewer functions**

---

## **export.ProcessViewerEvent**

---

```
result = export.ProcessViewerEvent (ViewerID, Event, Param)
```

**Parameters:**

ViewerID: integer  
Event: integer  
Param: nil

**Returns:**

result: integer

**Far API used:**

[ProcessViewerEventW](#), [ProcessViewerEventInfo](#)

## **Dialog functions**

---

# **export.ProcessDialogEvent**

---

```
ret = export.ProcessDialogEvent (Event, Param)
```

**Parameters:**

Event: integer  
Param: table *tFarDialogEvent*  
    hDlg : userdata  
    Msg : integer  
    Param1 : integer  
    Param2 : depends on *Msg*

**Returns:**

ret: integer, or nil

**Description:**

If *ret* is *nil* or *false*, then the event processing will continue.  
Otherwise, it is assumed that the plugin has processed the event,  
will then be used as the return value of the dialog event handler.

**Note:**

There is a group of drawing and coloring events that by default do  
to this function. See [far.SubscribeDialogDrawEvents](#) for the detail

**Far API used:**

[ProcessDialogEventW](#), [ProcessDialogEventInfo](#), [FarDialogEvent](#)

## **Content functions**

---

## **export.GetContentFields**

---

```
Ret = export.GetContentFields (Names)
```

**Parameters:**

Names: table (an array containing names of column content types)

**Returns:**

Ret: boolean

**Description:**

Return true if the plugin supports at least one of the content typ

**Far API used:**

[GetContentFieldsW](#), [GetContentFieldsInfo](#)

# **export.GetContentData**

---

```
Ret = export.GetContentData (FilePath, Names)
```

**Parameters:**

FilePath: full name of the file

Names: table (an array containing names of column content types

**Returns:**

Ret: table, or nil

**Description:**

The returned table should contain string values at the indexes corresponding to indexes of supported content types in the *Names* array.

**Far API used:**

[GetContentDataW](#), [GetContentDataInfo](#)

## **Other functions**

---

## **export.OnError**

---

```
export.OnError (ErrorMsg)
```

**Parameters:**

ErrorMsg: any type, usually a string

**Returns:**

nothing

**Description:**

This is a user-defined hook into LuaFAR.

If this function is defined by a user application, then, instead of displaying an error message box, LuaFAR calls this function, passing it the error message.

The return values of export.OnError (if any) are ignored.

## **LuaFAR Far libraries**

---

## Common functions

---

# far.AdvControl

---

```
Result = far.AdvControl (Command, Param1, Param2)
```

**Parameters:**

Command: flags  
Param1: (depends on Command); default=0  
Param2: (depends on Command); default=0

**Returns:**

Result: (*depends on Command*)

**Details:**

[ACTL\\_COMMIT](#),  
[ACTL\\_GETWINDOWCOUNT](#),  
[ACTL\\_PROGRESSNOTIFY](#),  
[ACTL\\_QUIT](#),  
[ACTL\\_REDRAWALL](#):

Result: integer

[ACTL\\_GETFARWND](#):

Result: light userdata

[ACTL\\_SETCURRENTWINDOW](#),

[ACTL\\_SYNCHRO](#):

Param1: integer  
Result: integer

[ACTL\\_WAITKEY](#):

Param2: table [tInputRecord](#), or string (e.g. "CtrlF12"), or nil  
Result: integer

[ACTL\\_GETCOLOR](#):

Param1: integer  
Result: table [tFarColor](#), or nil

[ACTL\\_GETARRAYCOLOR](#):

Result: table (array of [tFarColor](#) tables)

[ACTL\\_GETFARMANAGERVERSION](#):

Param1: boolean (optional)  
Result: string (e.g. "3.0.0.2040.0"), if Param==false;  
5 numbers (e.g. 3,0,0,2040,0), if Param==true;

ACTL\_GETWINDOWINFO:

Param1: window number; 1-based integer (optional; default=0)  
Result: table (see its fields below), or nil.  
Type: integer  
Id: userdata (if Type is [WTYPE\\_DIALOG](#) or [WTYPE\\_VMENU](#))  
Pos: integer (1-based)  
Flags: flags  
TypeName: string  
Name: string

ACTL\_SETARRAYCOLOR:

Param2: table (see its fields below;  
its array part contains colors  
(each color can be supplied as either a [tFarColor](#) table  
Flags: flags  
StartIndex: integer  
Result: integer

ACTL\_SETPROGRESSSTATE:

Param1: flags  
Result: integer

ACTL\_SETPROGRESSVALUE:

Param2: table (see its fields below)  
Completed, Total: numbers  
Result: integer

ACTL\_GETHARRECT:

Result: table (see its fields below), or nil  
Left, Top, Right, Bottom: integers

ACTL\_GETCURSORPOS:

Result: table (see its fields below), or nil  
X, Y: integers

ACTL\_SETCURSORPOS:

Param2: table (see its fields below)  
X, Y: integers  
Result: integer

ACTL\_GETWINDOWTYPE:

Result: table (see its fields below), or nil  
Type: integer

**Far API used:**

[AdvControl](#)

## **far.GetDirList**

---

```
list = far.GetDirList (dir)
```

**Parameters:**

dir: string

**Returns:**

list: array of [tPluginPanelItem](#) tables, or nil

**Far API used:**

[GetDirList](#), [FreeDirList](#)

## **far.GetMsg**

---

```
msg = far.GetMsg (MsgId)
```

**Parameters:**

  MsgId: integer

**Returns:**

  msg: string

**Far API used:**

[GetMsg](#)

## **far.GetPluginDirList**

---

```
list = far.GetPluginDirList (handle, Dir)
```

**Parameters:**

handle: panel handle, or nil  
Dir: string

**Returns:**

list: array of [tPluginPanelItem](#) tables, or nil

**Far API used:**

[GetPluginDirList](#), [FreePluginDirList](#)

# far.Menu

---

Item, Position = far.Menu (Properties, Items [, BreakKeys])

## Parameters:

Properties: table

fields of Properties (every field is optional):

X: integer  
Y: integer  
MaxHeight: integer  
Flags: flags (default = [FMENU\\_WRAPMODE](#))  
Title: string  
Bottom: string  
HelpTopic: string  
SelectIndex: integer (1-based)  
Id: string (GUID)

Items: table (array of menu items)

a menu item is a table;

fields of a menu item (every field is optional):

text: string  
checked: boolean, or string  
separator: boolean  
disable: boolean  
grayed: boolean  
hidden: boolean  
selected: boolean  
AccelKey: table [tFarKey](#) (or string - key name used in Fa

BreakKeys:

Either **table** (array of break key items)

a break key item is a table;

fields of a break key item:

BreakKey: string (see **NOTE 2**)

Or **string** (space delimited sequence of keys)

## Returns:

Item: table Either the selected element of *Items* array (if *E* or an element of *BreakKeys* array (if a corresponding key was pressed). If a break key was pressed and *BreakKey* is nil then the returned table is created by *far.Menu* function.  
nil The menu was cancelled by the user.

Position: integer Position of the **selected** item at the moment of cancellation.

`nil` The menu was cancelled by the user.

**NOTE 1**

`Properties.SelectedIndex=N` is equivalent to `Items[N].selected=true` but it has a higher priority.

**NOTE 2**

Example of constructing a `BreakKey` string:

`BreakKey="AS+DELETE"`, where:

`AS` - stands for AltShift

`+` - separates key modifiers from key

`DELETE` - [virtual key](#) `VK_DELETE` (prefix `VK_` is removed).

Key names used in Far macros are also acceptable, e.g. `BreakKey="A`

**NOTE 3**

The first return value (`Item`) can be used by the program for performing the item-specific action. To do so, the program can put some specific info into each element of `Items` and `BreakKeys` tables. For example, such info can be "action", "filename", etc.

**Far API used:**

[Menu](#)

## **far.RestoreScreen**

---

`far.RestoreScreen (handle)`

**Parameters:**

handle: userdata, or nil. (When it is nil, NULL is passed to Far M

**Returns:**

nothing

**Far API used:**

[RestoreScreen](#)

## **far.SaveScreen**

---

```
handle = far.SaveScreen (X1, Y1, X2, Y2)
```

**Parameters:**

X1: integer, or nil  
Y1: integer, or nil  
X2: integer, or nil  
Y2: integer, or nil

**Returns:**

handle: userdata

**Far API used:**

[SaveScreen](#)

## **far.ShowHelp**

---

```
result = far.ShowHelp (ModuleName, HelpTopic, Flags)
```

**Parameters:**

  ModuleName: string  
  HelpTopic: string, or nil  
  Flags:       flags

**Returns:**

  result: boolean

**Far API used:**

[ShowHelp](#)

## **far.Text**

---

```
far.Text (X, Y, Color, Str)
```

**Parameters:**

X: integer  
Y: integer  
Color: table [tFarColor](#), or integer  
Str: string, or nil (if nil, NULL is passed to Far)

**Returns:**

nothing

**Note:**

All parameters are optional.  
`far.Text()` is equivalent to `far.Text(0,0,0x0F,nil)`.

**Far API used:**

[Text](#)

## **Dialog functions**

---

# **far.ColorDialog**

---

```
OutColor = far.ColorDialog (InColor, Flags)
```

**Parameters:**

InColor: table [tFarColor](#), or integer (optional; default=0x0F)  
Flags: flags (optional)

**Returns:**

OutColor: table [tFarColor](#), integer, or nil

**Note:**

Type of *OutColor* is the same as that of *InColor*.

**Far API used:**

[ColorDialog](#)

## far.CreateUserControl

---

Buffer support for [DI\\_USERCONTROL](#). Use it like this:

```
local buffer=far.CreateUserControl(10,10)
local textel={Char="a",Attributes={Flags=bit64.bor(F.FCF_FG_4BIT,F.F
for ii=1,#buffer do
    buffer[ii]=textel
end
local items={
    {"DI_DOUBLEBOX",0,0,11,11,0,0,0,0,""},
    {"DI_USERCONTROL",1,1,10,10,buffer,0,0,0,""}
}
```

# far.Dialog

---

```
result = far.Dialog (Guid,X1,Y1,X2,Y2,HelpTopic,Items,Flags,fDlgProc
```

**Parameters:**

Same parameters as in [far.DialogInit](#)

**Returns:**

result: 1-based integer (or nil in case of failure)

**Note:**

The function updates the *Items* array to the values the dialog items have at the moment of dialog closing.

**Far API used:**

[DialogInit](#), [DialogRun](#), [DialogFree](#), [DlgProc](#)

## **far.DialogFree**

---

`far.DialogFree (handle)`

**Parameters:**

`handle`: userdata (obtained from a [far.DialogInit](#) call)

**Returns:**

    nothing

**Note:**

If *far.DialogFree* is never called by the program, the dialog handle will be eventually freed by the garbage collector.

**Far API used:**

[DialogFree](#)

# far.DialogInit

---

```
hDlg = far.DialogInit (Guid,X1,Y1,X2,Y2,HelpTopic,Items,Flags,fDlgPr
```

## Parameters:

Guid:	string
X1:	integer
Y1:	integer
X2:	integer
Y2:	integer
HelpTopic:	string, or nil
Items:	table (array of <a href="#">tFarDialogItem</a> )
Flags:	flags
fDlgProc:	function ( <a href="#">dialog event handler</a> ), or nil

## Returns:

hDlg: userdata (or nil in case of failure)

## Return value

The dialog handle returned by this function has the following methods  
*\_\_tostring*, *\_\_gc*, *rawhandle* and *send*.

### **hDlg:rawhandle()**

returns the original value of handle received from Far Manager  
that is suitable for using with LuaJIT FFI and as a table key.

### **hDlg:send(...)**

is equivalent to [far.SendDlgMessage](#)(hDlg, ...)

## Far API used:

[DialogInit](#), [DlgProc](#)

## **far.DialogRun**

---

```
result = far.DialogRun (handle)
```

**Parameters:**

    handle: userdata (obtained from a [far.DialogInit](#) call)

**Returns:**

    result: integer (1-based)

**Far API used:**

[DialogRun](#)

## **far.GetDlgItem**

---

```
result = far.GetDlgItem (hDlg, numitem [,item])
```

**Parameters:**

hDlg: userdata  
numitem: integer (1-based)  
item: table, or nil

**Returns:**

result: table, or nil

**Note:**

This function is equivalent to

[far.SendDlgItemMessage](#)(hDlg, "DM\_GETDLGITEM", numitem, item)

**Far API used:**

[SendDlgItemMessage](#), [DM\\_GETDLGITEM](#)

# **far.InputBox**

---

```
result = far.InputBox (Id, Title, Prompt, HistoryName, SrcText,  
                      DestLength, HelpTopic, Flags)
```

**Parameters:**

Id: string (dialog GUID), or nil (default=Plugin Id)  
Title: string, or nil  
Prompt: string, or nil  
HistoryName: string, or nil  
SrcText: string, or nil  
DestLength: integer, or nil  
HelpTopic: string, or nil  
Flags: flags, or nil  
(default = [FIB\\_ENABLEEMPTY](#)|[FIB\\_BUTTONS](#)|[FIB\\_NOAMPERSAND](#))

**Returns:**

result: string, or nil

**Far API used:**

[InputBox](#)

# far.Message

---

```
result = far.Message (Msg, [Title], [Buttons], [Flags], [HelpTopic],
```

**Parameters:**

*Msg*: any type (usually: string)

*Title*: string

*Buttons*: string

- button captions are separated by semicolons
- default value is one button "OK"
- empty string means no buttons
- if *Buttons* starts with a semicolon (;), then it must be one of the following predefined strings (case insensitive):  
";Ok", ";OkCancel", ";AbortRetryIgnore", ";YesNo",  
";YesNoCancel", ";RetryCancel".

*Flags*: string

it is checked if some predefined letters are present:

- 'w' stands for [FMSG WARNING](#)
- 'e' stands for [FMSG ERRORTYPE](#)
- 'k' stands for [FMSG KEEPBACKGROUND](#)
- 'l' stands for [FMSG LEFTALIGN](#)
- 'n' stands for "no wrapping of long lines"

*HelpTopic*: string

*Id*: string (GUID); default=plugin Id

**Returns:**

*result*: integer (1-based)

**Note 1:**

Before calling FAR function *Message*, the function:

- a) wraps long lines (unless the flag 'n' is present), and
- b) limits the maximal number of lines and buttons.

**Note 2:**

As in the original [Message](#) function, every line in **Msg** parameter that starts with \1 is treated as a single delimiter line; a line starting with \2 as a double delimiter line.

**Far API used:**

## Message

# far.SendDlgMessage

---

```
Result = far.SendDlgMessage (hDlg, Msg, Param1, Param2)
      or
Result = hDlg:send (Msg, Param1, Param2)
```

**Parameters:**

hDlg: userdata  
Msg: flags  
Param1: integer (1-based where it is element index, 0-based otherwise)  
Param2: (*depends on Msg*)

**Returns:**

Result: (*depends on Msg*)

**Details:**

[DM\\_CLOSE](#),  
[DM\\_EDITUNCHANGEDFLAG](#),  
[DM\\_ENABLE](#),  
[DM\\_ENABLEREDRAW](#),  
[DM\\_GETCHECK](#),  
[DM\\_GETCURSORSIZE](#),  
[DM\\_GETDLGDATA](#),  
[DM\\_GETDROPDOWNOPENED](#),  
[DM\\_GETFOCUS](#),  
[DM\\_GETITEMDATA](#),  
[DM\\_LISTGETDATASIZE](#),  
[DM\\_LISTSORT](#),  
[DM\\_REDRAW](#),  
[DM\\_SET3STATE](#),  
[DM\\_SETCURSORSIZE](#),  
[DM\\_SETDLGDATA](#),  
[DM\\_SETDROPDOWNOPENED](#),  
[DM\\_SETFOCUS](#),  
[DM\\_SETINPUTNOTIFY](#),  
[DM\\_SETITEMDATA](#),  
[DM\\_SETMAXTEXTLENGTH](#),  
[DM\\_SETMOUSEEVENTNOTIFY](#),  
[DM\\_SHOWDIALOG](#),  
[DM\\_SHOWITEM](#),  
[DM\\_USER](#):

Param2: integer  
Result: integer

DM\_ADDHISTORY,  
DM\_LISTADDSTR,  
DM\_SETHISTORY,  
DM\_SETTEXT,  
DM\_SETTEXTPTR:

Param2: string  
Result: integer

DM\_GETCURSORPOS,  
DM\_GETDIALOGINFO,  
DM\_GETDLGRECT,  
DM\_GETEDITPOSITION,  
DM\_GETITEMPOSITION,  
DM\_GETSELECTION,  
DM\_LISTGETCURPOS,  
DM\_LISTGETTITLES,  
DM\_LISTINFO:

Param2: none  
Result: table, or nil

DM\_GETDIALOGTITLE,  
DM\_GETTEXT,  
DM\_GETCONSTTEXTPTR:

Param2: none  
Result: string

DM\_KEY:

Param1: ignored, specify *nil* for clarity.  
Param2: table (array), or string

- If it is an array then each array element should be a
- If it is a string, then it is treated as a sequence c by whitespaces, e.g., "a b CtrlC ShiftHome D". The sy the same as in Far Manager macros.

Result: integer

DM\_LISTADD,  
DM\_LISTSET:

Param2: table (array of tables)  
Result: integer

*Example:*

```
res = far.SendDlgItemMessage(hDlg, "DM_LISTADD", id,  
    { { Text="added line 1" },  
      { Text="added line 2", Flags="LIF_CHECKED" } })
```

DM\_LISTDELETE:

Param2: table, or nil

Result: integer  
*Example:*  
res = far.SendDlgItemMessage(hDlg, "DM\_LISTDELETE", id,  
{ StartIndex=1, Count=2 })

#### DM\_LISTFINDSTRING:

Param2: table  
Result: integer, or nil  
*Example:*  
res = far.SendDlgItemMessage(hDlg, "DM\_FINDSTRING", id,  
{ StartIndex=1, Pattern="File???.txt" })

#### DM\_LISTGETITEM:

Param2: integer (ItemIndex)  
Result: table, or nil

#### DM\_LISTINSERT,

Param2: table  
Result: integer, or nil  
*Example:*  
res = far.SendDlgItemMessage(hDlg, "DM\_LISTINSERT", id,  
{ Index=2, Text="Hello World", Flags="LIF\_CHECKED" })

#### DM\_LISTUPDATE:

Param2: table  
Result: boolean  
*Example:*  
res = far.SendDlgItemMessage(hDlg, "DM\_LISTUPDATE", id,  
{ Index=2, Text="Hello World", Flags="LIF\_CHECKED" })

#### DM\_LISTSETCURPOS:

Param2: table  
Result: integer  
*Example:*  
res = far.SendDlgItemMessage(hDlg, "DM\_LISTSETCURPOS", id,  
{ SelectPos=5, TopPos=2 })  
*Note:*

Elements numbering is 1-based.

#### DM\_LISTSETTITLES:

Param2: table  
Result: integer  
*Example:*  
res = far.SendDlgItemMessage(hDlg, "DM\_LISTSETTITLES", id,  
{ Title="hello", Bottom="goodbye" })

#### DM\_LISTGETDATA:

Param2: integer (1-based list index)  
Result: either of  
    - nil - if Far returned 0  
    - light userdata (raw data pointer) - if alien data d  
    - value passed by the last [DM\\_LISTSETDATA](#) call

#### [DM\\_LISTSETDATA](#):

Param2: table  
    ["Index"] - integer (1-based list index)  
    ["Data"] - any Lua value except nil  
Result: integer (returned from Far)

#### [DM\\_GETDLGITEM](#):

Param2: table, or nil  
Result: table, or nil  
*Note:*  
    res = far.SendDlgItem(hDlg, "DM\_GETDLGITEM", numitem, item)  
        is equivalent to  
    res = [far.GetDlgItem](#)(hDlg, numitem, item)

#### [DM\\_SETDLGITEM](#):

Param2: table  
Result: boolean  
*Note:*  
    res = far.SendDlgItem(hDlg, "DM\_SETDLGITEM", numitem, item)  
        is equivalent to  
    res = [far.SetDlgItem](#)(hDlg, numitem, item)

#### [DM\\_MOVEDIALOG](#):

Param1: integer (0=relative, otherwise absolute coordinates)  
Param2: table  
Result: table  
*Example:*  
    res = far.SendDlgItem(hDlg, "DM\_MOVEDIALOG", 0, { X=5, Y=-2 }

#### [DM\\_RESIZEDIALOG](#):

Param2: table  
Result: table

#### [DM\\_SETCHECK](#):

Param2: flag  
Result: integer

#### [DM\\_SETCURSORPOS](#):

Param2: table  
Result: integer

[DM\\_SETITEMPOSITION](#):

Param2: table  
Result: integer  
*Example:*

```
res = far.SendDlgItemMessage(hDlg, "DM_SETITEMPOSITION", id,  
{ Left=3, Top=3, Right=20, Bottom=5 })
```

[DM\\_SETSELECTION](#):

Param2: table  
Result: integer  
*Note:* this function has much in common with [editor.Select](#)

[DM\\_SetComboBoxEvent](#):

Param2: flags  
Result: integer

[DM\\_SETEDITPOSITION](#):

Param2: table  
Result: integer  
*Example:*

```
res = far.SendDlgItemMessage(hDlg, "DM_SETEDITPOSITION", id,  
{ CurPos=3, Overtype=0 })
```

[DN\\_BTNCLICK](#),

[DN\\_DRAGGED](#),

[DN\\_DRAWDIALOG](#),

[DN\\_DRAWDIALOGDONE](#),

[DN\\_DROPDOWNOPENED](#),

[DN\\_ENTERIDLE](#):

Param2: integer  
Result: integer

[DN\\_CONTROLINPUT](#):

Param2: [tInputRecord](#) or string  
Result: integer

**Far API used:**

[SendDlgItemMessage](#)

## **far.SetDlgItem**

---

```
result = far.SetDlgItem (hDlg, numitem, item)
```

**Parameters:**

hDlg: userdata  
numitem: integer (1-based)  
item: table

**Returns:**

result: boolean

**Note:**

This function is equivalent to

[far.SendDlgMessage](#)(hDlg, "DM\_SETDLGITEM", numitem, item)

**Far API used:**

[SendDlgMessage](#), [DM\\_SETDLGITEM](#)

# far.SubscribeDialogDrawEvents

---

far.SubscribeDialogDrawEvents ()

**Parameters:**

none

**Returns:**

nothing

**Description:**

This function affects the action of [export.ProcessDialogEvent](#).

There is a group of frequently called events that slow down the pr especially in menus containing 10000+ items. This occurs due to co of arguments between C and Lua.

For each new dialog, [export.ProcessDialogEvent](#) will **not** be called values of *Param.Msg*:

[DN\\_CTLCOLORDIALOG](#)  
[DN\\_CTLCOLORDLGITEM](#)  
[DN\\_CTLCOLORDLGLIST](#)  
[DN\\_DRAWDIALOG](#)  
[DN\\_DRAWDIALOGDONE](#)  
[DN\\_DRAWDLGITEM](#)  
[DN\\_DRAWDLGITEMDONE](#)

In order to start receiving this group of messages, call far.Subsc after the dialog creation, e.g. when [DN\\_INITDIALOG](#) comes.

**Far API used:**

none

# Dialog Events

---

## Common information about dialog event handler

When the 9-th argument in the [far.DialogInit](#) or [far.Dialog](#) call is function (*event handler*), that function will be called by Far whenever any of the numerous dialog events occur.

```
ret = fDlgProc (hDlg, Msg, Param1, Param2)
```

### Parameters:

```
hDlg    : userdata
Msg     : integer
Param1  : integer
Param2  : depends on Msg
```

### Returns:

```
ret      : depends on Msg
```

## Parameters passed to event handler

1. hDlg: dialog handle. It is the same userdata kind that [far.DialogInit](#) returns.  
See description of its methods on [far.DialogInit](#) page.
2. Msg: integer, any DN\_\* value, e.g. [DN\\_BTNCLICK](#), [DN\\_CLOSE](#), etc.
3. Param1: integer (1-based if it is ID of a dialog element, 0-based otherwise).
4. Param2: integer for most event types, in accordance to Far API.  
For some events, Param2 is non-integer, see the articles describing those events.

## Treating the return value of dialog event handler (in the order of preference)

1. Return value of **nil** (or no value) causes [DefDlgProc](#) to be called. Its return value is returned to Far.

2. For some events ([DN\\_CTLCOLORDLGLIST](#), [DN\\_HELP](#), etc.), see the return values and their meaning in the article describing that event.
3. If the return value is **number**, then that number is returned to Far.
4. Otherwise, the boolean equivalent of the return value (that is, either 0 or 1) is returned to Far.

# **DN\_BTNCLICK**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_BTNCLICK](#)

# **DN\_CLOSE**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_CLOSE](#)

# **DN\_CONTROLINPUT**

---

**Param1**

integer; according to Far API

**Param2**

[tInputRecord](#)

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_CONTROLINPUT](#)

# **DN\_CTLCOLORDIALOG**

---

**Param1**

integer; according to Far API

**Param2**

table [tFarColor](#)

**Return**

table [tFarColor](#), or integer, or nothing

For default processing of the return value, see the [description](#).

**Far API used:**

[DlgProc](#), [DN\\_CTLCOLORDIALOG](#)

# **DN\_CTLCOLORDLGITEM**

---

**Param1**

integer; according to Far API

**Param2**

table ([tFarDialogItemColors](#))

**Return**

If the function returned a table (an array of new color attributes to be set), 1 is returned to Far, otherwise 0.

**Far API used:**

[DlgProc](#), [DN\\_CTLCOLORDLGITEM](#)

# **DN\_CTLCOLORDLGLIST**

---

**Param1**

integer; according to Far API

**Param2**

table ([tFarDialogItemColors](#))

**Return**

If the function returned a table (an array of new color attributes to be set), 1 is returned to Far, otherwise 0.

**Far API used:**

[DlgProc](#), [DN\\_CTLCOLORDLGLIST](#)

# **DN\_DRAGGED**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_DRAGGED](#)

# **DN\_DRAWDIALOG**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_DRAWDIALOG](#)

## **DN\_DRAWDIALOGDONE**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_DRAWDIALOGDONE](#)

## **DN\_DRAWDLGITEM**

---

**Param1**

integer; according to Far API

**Param2**

table ([tFarDialogItem](#))

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_DRAWDLGITEM](#)

## **DN\_DRAWDLGITEMDONE**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_DRAWDLGITEMDONE](#)

# **DN\_DROPDOWNOPENED**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_DROPDOWNOPENED](#)

# **DN\_EDITCHANGE**

---

**Param1**

integer; according to Far API

**Param2**

table ([tFarDialogItem](#))

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_EDITCHANGE](#)

# **DN\_ENTERIDLE**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_ENTERIDLE](#)

# **DN\_GETVALUE**

---

**Param1**

integer; according to Far API

**Param2**

table

GetType: integer

ValType: integer (enum [FARMACROVARTYPE](#))

Value: nil, flags, string, or number (depends on ValType)

**Return**

table

ValType: integer (enum [FARMACROVARTYPE](#))

Value: flags, string, or number (depends on ValType)

**Note**

If the returned value is not a table containing valid values, then 0 is returned to Far.

**Far API used:**

[DlgProc](#), [DN\\_GETVALUE](#)

# **DN\_GOTFOCUS**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_GOTFOCUS](#)

# **DN\_HELP**

---

**Param1**

integer; according to Far API

**Param2**

string; according to Far API

**Return**

If the function returned a string, its address is returned to Far, otherwise 0 is returned.

**Far API used:**

[DlgProc](#), [DN\\_HELP](#)

# **DN\_HOTKEY**

---

**Param1**

integer; according to Far API

**Param2**

[tInputRecord](#)

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_HOTKEY](#)

# **DN\_INITDIALOG**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_INITDIALOG](#)

# **DN\_INPUT**

---

**Param1**

integer; according to Far API

**Param2**

[tInputRecord](#)

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_INPUT](#)

# **DN\_KILLFOCUS**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_KILLFOCUS](#)

# **DN\_LISTCHANGE**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_LISTCHANGE](#)

## **DN\_LISTHOTKEY**

---

**Param1**

integer; according to Far API

**Param2**

integer; according to Far API

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_LISTHOTKEY](#)

# **DN\_RESIZECONSOLE**

---

**Param1**

integer; according to Far API

**Param2**

table ([tCoord](#))

**Return**

default processing of the return value, see [description](#).

**Far API used:**

[DlgProc](#), [DN\\_RESIZECONSOLE](#)

## **Editor functions**

---

## editor.AddColor

---

```
result = editor.AddColor (EditorId, StringNumber, StartPos, EndPos,  
                         Color, Priority, Owner)
```

**Parameters:**

EditorId: integer, or nil  
StringNumber: integer, or nil  
StartPos: integer (1-based)  
EndPos: integer (1-based)  
Flags: flags  
Color: table [tFarColor](#) or integer  
Priority: integer (optional; default=EDITOR\_COLOR\_NORMAL\_PRIORITY)  
Owner: string, or nil (defaults to the calling plugin's GU

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_ADDCOLOR](#)

## **editor.AddSessionBookmark**

---

```
result = editor.AddSessionBookmark (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_ADDSESSIONBOOKMARK](#)

## **editor.ClearSessionBookmarks**

---

```
result = editor.ClearSessionBookmarks (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_CLEARSESSIONBOOKMARKS](#)

## **editor.DelColor**

---

```
result = editor.DelColor (EditorId, StringNumber, StartPos, Owner)
```

**Parameters:**

EditorId: integer, or nil (defaults to CURRENT\_EDITOR)  
StringNumber: integer, or nil  
StartPos: integer, or nil  
Owner: string, or nil (defaults to the calling plugin's GU

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_DELCOLOR](#)

## **editor.DeleteBlock**

---

```
result = editor.DeleteBlock (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_DELETEBLOCK](#)

## **editor.DeleteChar**

---

```
result = editor.DeleteChar (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_DELETECHAR](#)

## **editor.DeleteSessionBookmark**

---

```
result = editor.DeleteSessionBookmark (EditorId [,position])
```

**Parameters:**

EditorId: integer, or nil

position: integer (1-based), or nil (nil = current position)

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_DELETESESSIONBOOKMARK](#)

## **editor.DeleteString**

---

```
result = editor.DeleteString (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_DELETESTRING](#)

## **editor.Editor**

---

```
result = editor.Editor (FileName, Title, X1, Y1, X2, Y2,  
                      Flags, StartLine, StartChar, CodePage)
```

**Parameters:**

  FileName: string  
  Title: string, or nil (default = NULL)  
  X1: integer, or nil (default = 0)  
  Y1: integer, or nil (default = 0)  
  X2: integer, or nil (default = -1)  
  Y2: integer, or nil (default = -1)  
  Flags: flags, or nil (default = 0)  
  StartLine: integer, or nil (1-based; default = -1)  
  StartChar: integer, or nil (1-based; default = -1)  
  CodePage: integer, or nil (default = CP\_AUTODETECT)

**Returns:**

  result: integer

**Far API used:**

[Editor](#)

## **editor.ExpandTabs**

---

```
result = editor.ExpandTabs (EditorId, StringNumber)
```

**Parameters:**

EditorId: integer, or nil  
StringNumber: integer (1-based), or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_EXPANDTABS](#)

## editor.GetBookmarks

---

```
bookmarks = editor.GetBookmarks (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

bookmarks: table (array of "bookmark tables"), or nil

fields of a "bookmark table":

Line: integer

Cursor: integer

ScreenLine: integer

LeftPos: integer

**Note:**

This function groups data differently from how Far API's [EditorControl\(ECTL\\_GETBOOKMARKS\)](#) does. It places data of each bookmark into a separate table.

**Far API used:**

[EditorControl](#), [ECTL\\_GETBOOKMARKS](#), [ECTL\\_GETINFO](#)

## **editor.GetColor**

---

```
colordata = editor.GetColor(EditorId, StringNumber, ColorItem)
```

**Parameters:**

EditorId: integer, or nil  
StringNumber: integer (1-based), or nil  
ColorItem: integer (0-based)

**Returns:**

colordata: table (or nil)  
            StartPos:     integer  
            EndPos:     integer  
            Priority:    integer  
            Flags:       flags  
            Color:       table [tFarColor](#)  
            Owner:       string (GUID)

**Far API used:**

[EditorControl](#), [ECTL\\_GETCOLOR](#)

## **editor.GetFileName**

---

```
FileName = editor.GetFileName (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

FileName: string

**Far API used:**

[EditorControl](#), [ECTL\\_GETFILENAME](#)

## editor.GetInfo

---

```
Info = editor.GetInfo (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

Info: table, or nil

Info fields:

EditorID:	integer
FileName:	string
WindowSizeX:	integer
WindowSizeY:	integer
TotalLines:	integer
CurLine:	integer, 1-based
CurPos:	integer, 1-based
CurTabPos:	integer, 1-based
TopScreenLine:	integer, 1-based
LeftPos:	integer, 1-based
Overtype:	integer
BlockType:	integer
BlockStartLine:	integer, 1-based
Options:	integer
TabSize:	integer
BookmarkCount:	integer
SessionBookmarkCount:	integer
CurState:	integer
CodePage:	integer

**Far API used:**

[EditorControl](#), [ECTL\\_GETINFO](#)

## **editor.GetSelection**

---

```
Sel = editor.GetSelection (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

Sel: table, or nil

Sel fields:

BlockType: integer

StartLine: integer, 1-based (Far API + 1)

StartPos: integer, 1-based (Far API + 1)

EndLine: integer, 1-based (Far API + 1)

EndPos: integer, 1-based (Far API + 0)

**Note:**

This function does not have a prototype in Far API.

**Far API used:**

[EditorControl](#), [ECTL\\_GETINFO](#), [ECTL\\_GETSTRING](#), [ECTL\\_SETPOSITION](#)

## editor.GetSessionBookmarks

---

```
bookmarks = editor.GetSessionBookmarks (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

bookmarks: table (array of "bookmark tables"), or nil

fields of a "bookmark table":

Line: integer

Cursor: integer

ScreenLine: integer

LeftPos: integer

**Note:**

This function groups data differently from how Far API's [EditorControl\(ECTL\\_GETSESSIONBOOKMARKS\)](#) does. It places data of each bookmark into a separate table.

**Far API used:**

[EditorControl](#), [ECTL\\_GETSESSIONBOOKMARKS](#)

# editor.GetString

---

**Note:**

This function is an extension over its prototype in Far API.

```
LineInfo = editor.GetString (EditorId, StringNumber, Mode)
or
StringText, StringEOL = editor.GetString (EditorId, StringNumber, Mo
```

**Parameters:**

EditorId: integer (default=CURRENT\_EDITOR)  
StringNumber: integer, 1-based (default=0)  
Mode: integer (0,1,2,3; default=0)

**Returns:**

LineInfo: table (Mode<2), or 2 strings (Mode>=2), or nil  
fields of LineInfo:  
    StringNumber: integer, 1-based  
    StringText: string  
    StringEOL: string  
    StringLength: integer  
    SelStart: integer, 1-based  
    SelEnd: integer, 1-based

**Description:**

The function's behavior depends on its *Mode* argument:

Mode: 0 = returns: table LineInfo;	changes current position
1 = returns: table LineInfo;	changes current position
2 = returns: StringText, StringEOL;	changes current position
3 = returns: StringText, StringEOL;	changes current position

Modes 1 and 2 have a side effect: the current line position in the  
is set to *StringNumber*.

**Far API used:**

[EditorControl](#), [ECTL\\_GETINFO](#), [ECTL\\_SETPOSITION](#), [ECTL\\_GETSTRING](#)

## **editor.GetStringW**

---

Same as [editor.GetString](#) but *StringText* and *StringEOL* are encoded in

## **editor.GetTitle**

---

```
Title = editor.GetTitle (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

Title: string, or nil

**Far API used:**

[EditorControl](#), [ECTL\\_GETTITLE](#)

## **editor.InsertString**

---

```
result = editor.InsertString ([EditorId [, indent]])
```

**Parameters:**

EditorId: integer, or nil  
indent: boolean, defaults to false

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_INSERTSTRING](#)

## **editor.InsertText**

---

```
result = editor.InsertText (EditorId, text)
```

**Parameters:**

EditorId: integer, or nil  
text: string

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_INSERTTEXT](#)

## **editor.InsertTextW**

---

Same as [editor.InsertText](#) but *text* should be encoded in UTF-16LE.

If *text* argument does not have two zero bytes at the end, the function so the caller does not have to.

## **editor.NextSessionBookmark**

---

```
result = editor.NextSessionBookmark (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_NEXTSESSIONBOOKMARK](#)

## **editor.PrevSessionBookmark**

---

```
result = editor.PrevSessionBookmark (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_PREVSESSIONBOOKMARK](#)

## editor.ProcessInput

---

```
result = editor.ProcessInput (EditorId, Input)
```

**Parameters:**

EditorId: integer, or nil  
Input: table [tInputRecord](#).  
(Field *EventType* is mandatory, other fields  
are optional. If some field is missing, the default  
value of 0/false is used, except for *wRepeatCount*  
that defaults to 1).

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_PROCESSINPUT](#)

## **editor.Quit**

---

```
result = editor.Quit (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_QUIT](#)

## **editor.ReadInput**

---

```
Info = editor.ReadInput (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

Info: table [tInputRecord](#), or nil

**Far API used:**

[EditorControl](#), [ECTL\\_READINPUT](#)

## **editor.RealToTab**

---

```
result = editor.RealToTab (EditorId, StringNumber, SrcPos)
```

**Parameters:**

EditorId: integer, or nil  
StringNumber: integer (1-based), or nil  
SrcPos: integer (1-based)

**Returns:**

result: integer (1-based), or nil

**Far API used:**

[EditorControl](#), [ECTL REALTOTAB](#)

## **editor.Redraw**

---

```
result = editor.Redraw (EditorId)
```

**Parameters:**

EditorId: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_REDRAW](#)

## **editor.SaveFile**

---

```
result = editor.SaveFile (EditorId, FileName, FileEOL, CodePage)
```

**Parameters:**

EditorId: integer, or nil  
FileName: string, or nil  
FileEOL: string, or nil  
CodePage: integer, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_SAVEFILE](#)

## editor.Select

---

```
result = editor.Select (EditorId, SelectData)
or
result = editor.Select (EditorId, BlockType, BlockStartLine, BlockSt
                      BlockWidth, BlockHeight)
```

### Parameters:

EditorId: integer, or nil

SelectData: table

fields of SelectData:

BlockType: flags

BlockStartLine: integer, 1-based

BlockStartPos: integer, 1-based

BlockWidth: integer

BlockHeight: integer

Parameters of the 2-nd function variant correspond to fields of the *SelectData* table.

### Returns:

result: boolean

### Far API used:

[EditorControl](#), [ECTL\\_SELECT](#)

## **editor.SetKeyBar**

---

```
result = editor.SetKeyBar (EditorId, param)
```

**Parameters:**

EditorId: integer, or nil  
param, either of:  
- string "redraw": redraw the key bar titles;  
- string "restore": restore the original titles;  
- table (array of [tKeyBarLabel](#) tables)

**Returns:**

result: boolean

**Far API used:**

[EditorControl1](#), [ECTL\\_SETKEYBAR](#)

## **editor.SetParam**

---

```
res1 [, res2] = editor.SetParam (EditorId, Type, Param [, Flags])
```

**Parameters:**

EditorId: integer, or nil  
Type: flags  
Param: integer, boolean, or string  
Flags: flags, or nil

**Returns:**

res1: boolean  
res2: string (if Type==[ESPT\\_GETWORDDIV](#)), or nil

**Far API used:**

[EditorControl](#), [ECTL\\_SETPARAM](#)

## editor.SetPosition

---

```
result = editor.SetPosition (EditorId, PosData)
      or
result = editor.SetPosition (EditorId, CurLine, CurPos,
                           CurTabPos, TopScreenLine, LeftPos, Overtype)
```

### Parameters:

EditorId: integer, or nil

PosData: table

fields of PosData:

CurLine: integer, 1-based

CurPos: integer, 1-based

CurTabPos: integer, 1-based

TopScreenLine: integer, 1-based

LeftPos: integer, 1-based

Overtype: integer

Parameters of the 2-nd function variant correspond to fields of the *PosData* table.

### Returns:

result: boolean

### Far API used:

[EditorControl1](#), [ECTL\\_SETPOSITION](#)

## **editor.SetString**

---

```
result = editor.SetString (EditorId, StringNumber, StringText, StringEOL)
```

**Parameters:**

EditorId: integer (default=-1)  
StringNumber: integer, 1-based (default=0)  
StringText: string  
StringEOL: string, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_SETSTRING](#)

## **editor.SetStringW**

---

Same as [editor.SetString](#) but *StringText* and *StringEOL* should be encoded in UTF-16LE.

## **editorSetTitle**

---

```
result = editor.SetTitle ([EditorId [, Title]])
```

**Parameters:**

EditorId: integer, or nil  
Title: string, or nil

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_SETTITLE](#)

## **editor.SubscribeChangeEvent**

---

```
result = editor.SubscribeChangeEvent (EditorId, Subscribe)
```

**Parameters:**

EditorId: integer, or nil  
Subscribe: boolean (true means "subscribe", false means "unsubscr

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_SUBSCRIBECHANGEVENT](#), [ECTL\\_UNSUBSCRIBECHANGEV](#)

## **editor.TabToReal**

---

```
result = editor.TabToReal (EditorId, StringNumber, SrcPos)
```

**Parameters:**

EditorId: integer, or nil  
StringNumber: integer (1-based), or nil  
SrcPos: integer (1-based)

**Returns:**

result: integer (1-based), or nil

**Far API used:**

[EditorControl](#), [ECTL\\_TABTOREAL](#)

## **editor.UndoRedo**

---

```
result = editor.UndoRedo (EditorId, Command)
```

**Parameters:**

EditorId: integer, or nil  
Command: flags

**Returns:**

result: boolean

**Far API used:**

[EditorControl](#), [ECTL\\_UNDOREDO](#)

# Panel functions

---

## Note about parameters in some panel functions

Many panel functions have the 1-st parameter *handle* and the 2-nd parameter *whatpanel*. Only one of them is actually used:

- If *handle* parameter is supplied as a non-nil value then *whatpanel* parameter is ignored
- If *handle* parameter is nil then *whatpanel* parameter is mandatory

## panel.CheckPanelsExist

---

```
result = panel.CheckPanelsExist ([handle])
```

**Parameters:**

    handle: panel handle

**Returns:**

    result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_CHECKPANELSEXIST](#)

## panel.ClearSelection

---

```
result = panel.ClearSelection (handle, whatpanel, items)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active\_panel, 0=inactive\_panel  
items: integer (1-based item index),  
or table (array of item indexes)

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_GETPANELINFO](#), [FCTL\\_BEGINSELECTION](#),  
[FCTL\\_CLEARSELECTION](#), [FCTL\\_ENDSELECTION](#)

## panel.ClosePanel

---

```
result = panel.ClosePanel (handle [, dir])
```

**Parameters:**

handle: panel handle  
dir: string (optional);  
nil/nothing pass NULL to Far.

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL CLOSEPANEL](#)

## panel.GetCmdLine

---

```
result = panel.GetCmdLine ([handle])
```

**Parameters:**

handle: panel handle

**Returns:**

result: string (or nil)

**Far API used:**

[PanelControl](#), [FCTL\\_GETCMDLINE](#)

## panel.GetCmdLinePos

---

```
pos = panel.GetCmdLinePos ([handle])
```

**Parameters:**

handle: panel handle

**Returns:**

pos: 1-based position (or nil)

**Far API used:**

[PanelControl](#), [FCTL\\_GETCMDLINEPOS](#)

## panel.GetCmdLineSelection

---

```
SelStart, SelEnd = panel.GetCmdLineSelection ([handle])
```

**Parameters:**

handle: panel handle

**Returns:**

SelStart: integer (1-based)

SelEnd: integer

**Far API used:**

[PanelControl](#), [FCTL\\_GETCMDLINESELECTION](#)

## **panel.GetColumnTypes**

---

```
ColumnTypes = panel.GetColumnTypes (handle, whatpanel)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;

**Returns:**

ColumnTypes: string

**Far API used:**

[PanelControl](#), [FCTL\\_GETCOLUMN TYPES](#)

## panel.GetColumnWidths

---

```
ColumnWidths = panel.GetColumnWidths (handle, whatpanel)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;

**Returns:**

ColumnWidths: string

**Far API used:**

[PanelControl](#), [FCTL\\_GETCOLUMNWIDTHS](#)

## panel.GetCurrentPanelItem

---

```
item = panel.GetCurrentPanelItem (handle, whatpanel)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;

**Returns:**

item: [tPluginPanelItem](#)

**Far API used:**

[PanelControl](#), [FCTL\\_GETCURRENTPANELITEM](#)

## panel.GetPanelDirectory

---

PanelDir = panel.GetPanelDirectory (handle, whatpanel)

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;

**Returns:**

PanelDir: table [tFarPanelDirectory](#)

**Far API used:**

[PanelControl](#), [FCTL\\_GETPANELDIRECTORY](#)

## panel.GetPanelFormat

---

```
PanelFormat = panel.GetPanelFormat (handle, whatpanel)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;

**Returns:**

PanelFormat: string

**Far API used:**

[PanelControl](#), [FCTL\\_GETPANELFORMAT](#)

## **panel.GetPanelHostFile**

---

```
HostFile = panel.GetPanelHostFile (handle, whatpanel)
```

**Parameters:**

    handle:     panel handle, or nil  
    whatpanel: 1=active panel; 0=passive panel;

**Returns:**

    HostFile: string

**Far API used:**

[PanelControl](#), [FCTL\\_GETPANELHOSTFILE](#)

# panel.GetPanelInfo

---

```
tPanelInfo = panel.GetPanelInfo (handle, whatpanel)
```

## Parameters:

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;

## Returns:

tPanelInfo: table, or nil

fields of tPanelInfo:

OwnerGuid:	string
PluginHandle:	userdata, or nil
PanelType:	integer
PanelRect:	table (fields: left, top, right, bottom)
ItemsNumber:	integer
SelectedItemsNumber:	integer
CurrentItem:	1-based integer
TopPanelItem:	1-based integer
ViewMode:	integer
SortMode:	integer
Flags:	flags
PluginObject:	non-false Lua value

## Note 1:

The *PluginHandle* field has a method:

*PluginHandle:rawhandle()*

returns the original value of handle received from Far Manager  
that is suitable for using with LuAJIT FFI and as a table key.

## Note 2:

The *PluginObject* field is LuaFAR-specific. It is only present if C i.e. when *panel.GetPanelInfo* was called from the same plugin that The intended use of this field is when a panel plugin processes ma on an already opened panel.

## Far API used:

[PanelControl](#), [FCTL\\_GETPANELINFO](#), [PanelInfo](#)

## panel.GetPanelItem

---

```
item = panel.GetPanelItem (handle, whatpanel, itemindex)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
itemindex: integer (1-based)

**Returns:**

item: [tPluginPanelItem](#)

**Far API used:**

[PanelControl](#), [FCTL\\_GETPANELITEM](#)

## panel.GetPanelPrefix

---

```
Prefix = panel.GetPanelPrefix (handle, whatpanel)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;

**Returns:**

Prefix: string, or nil

**Far API used:**

[PanelControl](#), [FCTL\\_GETPANELPREFIX](#)

## panel.GetSelectedPanelItem

---

```
item = panel.GetSelectedPanelItem (handle, whatpanel, itemindex)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
itemindex: integer (1-based)

**Returns:**

item: [tPluginPanelItem](#)

**Far API used:**

[PanelControl](#), [FCTL\\_GETSELECTEDPANELITEM](#)

# panel.GetUserScreen

---

```
result = panel.GetUserScreen ([handle], [scrolltype])
```

**Parameters:**

    handle:     panel handle  
    scrolltype: integer (defaults to 0)

**Returns:**

    result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_GETUSERSCREEN](#)

**Example:**

```
function printToFARConsole(...)  
    panel.GetUserScreen()  
    io.write(...)  
    io.write("\n")  
    panel.SetUserScreen()  
end
```

## panel.InsertCmdLine

---

```
result = panel.InsertCmdLine (handle, str)
```

**Parameters:**

handle: panel handle, or nil  
str: string

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_INSERTCMDLINE](#)

## **panel.IsActivePanel**

---

```
result = panel.IsActivePanel ([handle])
```

**Parameters:**

    handle: panel handle

**Returns:**

    result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_ISACTIVEPANEL](#)

## panel.RedrawPanel

---

```
result = panel.RedrawPanel (handle, whatpanel [, redrawinfo])
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
redrawinfo: table with the following fields:  
                  CurrentItem : integer (1-based)  
                  TopPanelItem : integer (1-based)

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_REDRAWPANEL](#)

## **panel.SetActivePanel**

---

```
result = panel.SetActivePanel (handle, whatpanel)
```

**Parameters:**

    handle:     panel handle, or nil  
    whatpanel: 1=active panel; 0=passive panel;

**Returns:**

    result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETACTIVEPANEL](#)

## panel.SetCmdLine

---

```
result = panel.SetCmdLine (handle, str)
```

**Parameters:**

handle: panel handle, or nil  
str: string

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETCMDLINE](#)

## panel.SetCmdLinePos

---

```
result = panel.SetCmdLinePos (handle, pos)
```

**Parameters:**

handle: panel handle, or nil  
pos: 1-based integer

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETCMDLINEPOS](#)

## panel.SetCmdLineSelection

---

```
result = panel.SetCmdLineSelection (handle, selStart, selEnd)
```

**Parameters:**

handle: panel handle, or nil  
selStart: integer (1-based)  
selEnd: integer

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETCMDLINESELECTION](#)

## panel.SetDirectoriesFirst

---

```
result = panel.SetDirectoriesFirst (handle, whatpanel [, setfirst])
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
setfirst: boolean (false=don't set directories first; true=set fi  
          defaults to false)

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETDIRECTORIESFIRST](#)

## panel.SetPanelDirectory

---

```
result = panel.SetPanelDirectory (handle, whatpanel, dir)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
dir: string, or table [tFarPanelDirectory](#)  
(if dir.PluginId is not set, it defaults to Far ID)

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETPANELDIRECTORY](#)

## panel.SetSelection

---

```
result = panel.SetSelection (handle, whatpanel, items, selection)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active\_panel, 0=inactive\_panel  
items: integer (1-based item index),  
 or table (array of item indexes)  
selection: boolean

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_GETPANELINFO](#), [FCTL\\_BEGINSELECTION](#),  
[FCTL\\_SETSELECTION](#), [FCTL\\_ENDSELECTION](#)

## panel.SetSortMode

---

```
result = panel.SetSortMode (handle, whatpanel, mode)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
mode: flags

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETSORTMODE](#)

## panel.SetSortOrder

---

```
result = panel.SetSortOrder (handle, whatpanel [, order])
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
order: boolean (false=direct order; true=reverse order;  
          defaults to false)

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETSORTORDER](#)

## panel.SetUserScreen

---

```
result = panel.SetUserScreen ([handle], [scrolltype])
```

**Parameters:**

handle: panel handle  
scrolltype: integer (defaults to 0)

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETUSERSCREEN](#)

**Example:**

```
function printToFARConsole(...)  
    panel.GetUserScreen()  
    io.write(...)  
    io.write("\n")  
    panel.SetUserScreen()  
end
```

## panel.SetViewMode

---

```
result = panel.SetViewMode (handle, whatpanel, mode)
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
mode: integer (0...9)

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_SETVIEWMODE](#)

## panel.UpdatePanel

---

```
result = panel.UpdatePanel (handle, whatpanel [, keepselection])
```

**Parameters:**

handle: panel handle, or nil  
whatpanel: 1=active panel; 0=passive panel;  
keepselection: boolean, defaults to false

**Returns:**

result: boolean

**Far API used:**

[PanelControl](#), [FCTL\\_UPDATEPANEL](#)

## **Viewer functions**

---

## **viewer.GetFileName**

---

```
FileName = viewer.GetFileName (ViewerId)
```

**Parameters:**

ViewerId: integer, or nil

**Returns:**

FileName: string

**Far API used:**

[ViewerControl](#), [VCTL\\_GETFILENAME](#)

## viewer.GetInfo

---

```
Info = viewer.GetInfo (ViewerID)
```

**Parameters:**

ViewerID: integer, or nil

**Returns:**

Info: table, or nil

Info fields:

ViewerID:	integer
FileName:	string
FileSize:	double
FilePos:	double, 0-based
WindowSizeX:	integer
WindowSizeY:	integer
Options:	integer
TabSize:	integer
LeftPos:	integer, 1-based
<u>CurMode</u> :	table

**Far API used:**

[ViewerControl](#), [VCTL\\_GETINFO](#)

## viewer.Quit

---

viewer.Quit (ViewerID)

**Parameters:**

ViewerID: integer, or nil

**Returns:**

nothing

**Far API used:**

[ViewerControl](#), [VCTL\\_QUIT](#)

## **viewer.Redraw**

---

`viewer.Redraw (ViewerID)`

**Parameters:**

ViewerID: integer, or nil

**Returns:**

nothing

**Far API used:**

[ViewerControl](#), [VCTL\\_REDRAW](#)

## viewer.Select

---

```
result = viewer.Select (ViewerID, BlockStartPos, BlockLen)
```

**Parameters:**

ViewerID: integer, or nil  
BlockStartPos: number  
BlockLen: integer

**Returns:**

result: boolean

**Far API used:**

[ViewerControl](#), [VCTL SELECT](#)

## viewer.SetKeyBar

---

```
result = viewer.SetKeyBar (ViewerID, param)
```

**Parameters:**

ViewerID: integer, or nil

param, either of:

- string "redraw": redraw the key bar titles;
- string "restore": restore the original titles;
- table (array of [tKeyBarLabel](#) tables)

**Returns:**

result: boolean

**Far API used:**

[ViewerControl](#), [VCTL\\_SETKEYBAR](#)

## viewer.SetMode

---

```
result = viewer.SetMode (ViewerID, ModeData)
```

**Parameters:**

ViewerID: integer, or nil  
ModeData: table  
fields of ModeData:  
Type: flags  
iParam: integer  
Flags: flags

**Returns:**

result: boolean

**Far API used:**

[ViewerControl](#), [VCTL\\_SETMODE](#)

## viewer.SetPosition

---

```
pos = viewer.SetPosition (ViewerID, PosData)
or
pos = viewer.SetPosition (ViewerID, StartPos, LeftPos, Flags)
```

**Parameters:**

ViewerID: integer, or nil  
PosData: table  
    fields of PosData:  
        StartPos: number, 0-based  
        LeftPos: number, 1-based  
        Flags: flags

Parameters of the 2-nd function variant correspond to fields of the *PosData* table.

**Returns:**

pos: resulting value of *StartPos* (number), or nil

**Far API used:**

[ViewerControl](#), [VCTL\\_SETPOSITION](#)

## viewer.Viewer

---

```
result = viewer.Viewer (FileName, Title, X1, Y1, X2, Y2, Flags, Code
```

**Parameters:**

  FileName: string  
  Title:    string (defaults to NULL)  
  X1:       integer (defaults to 0)  
  Y1:       integer (defaults to 0)  
  X2:       integer (defaults to -1)  
  Y2:       integer (defaults to -1)  
  Flags:    flags (defaults to 0)  
  CodePage: integer (defaults to CP\_AUTODETECT)

**Returns:**

  result: boolean

**Far API used:**

[Viewer](#)

# File Filter Control

---

File filter control is implemented as bindings of Far API function [FileFilterControl](#). LuaFAR API is chosen to be different from Far API (in fact, it is much simpler).

The API consists of 1 function ([far.CreateFileFilter](#)) that creates an instance of file filter object, the latter has 4 methods: [FreeFileFilter](#), [OpenFiltersMenu](#), [StartingToFilter](#) and [IsFileInFilter](#).

## Example

```
-- create a filter for the "active panel" area
local AFilter = far.CreateFileFilter(1, "FFT_PANEL")
if AFilter then
    -- update the "current time" parameter
    AFilter:StartingToFilter()

    -- manipulate panel elements
    local pInfo = panel.GetPanelInfo(nil, 1)
    for i=1, pInfo.ItemsNumber do
        -- apply the filter...
        local item = panel.GetPanelItem(nil, 1, i)
        if AFilter:IsFileInFilter(item) then
            -- the element matches filter conditions;
            -- process the element...
        end
    end

    -- free memory
    AFilter:FreeFileFilter()
end
```

## **far.CreateFileFilter**

---

```
f = far.CreateFileFilter (whatPanel, filterType)
```

**Parameters:**

whatPanel: integer (0=passive panel, 1=active panel)  
filterType: flags

**Returns:**

f: userdata (file filter object), or nil

**Description:**

The created object has the following methods: [FreeFileFilter](#), [OpenFiltersMenu](#), [StartingToFilter](#) and [IsFileInFilter](#).

**Far API used:**

[FileFilterControl](#), [FFCTL\\_CREATEFILEFILTER](#)

## f:FreeFileFilter

---

```
result = f:FreeFileFilter ()
```

**Parameters:**

f: file filter object

**Returns:**

result: boolean

**Note:**

File filter objects are subject to garbage collection,  
thus calling this method is not mandatory.

**Far API used:**

[FileFilterControl](#), [FFCTL\\_FREEFILEFILTER](#)

## f:OpenFiltersMenu

---

```
result = f:OpenFiltersMenu ()
```

**Parameters:**

f: file filter object

**Returns:**

result: boolean

**Far API used:**

[FileFilterControl](#), [FFCTL\\_OPENFILTERSMENU](#)

## **f:StartingToFilter**

---

```
result = f:StartingToFilter ()
```

**Parameters:**

f: file filter object

**Returns:**

result: boolean

**Far API used:**

[FileFilterControl](#), [FFCTL\\_STARTINGTOFILTER](#)

## f:IsFileInFilter

---

```
result = f:IsFileInFilter (fileData)
```

**Parameters:**

f: file filter object  
fileData: table [tPluginPanelItem](#)

**Returns:**

result: boolean

**Far API used:**

[FileFilterControl](#), [FFCTL\\_ISFILEINFILTER](#)

# **Macro Control**

---

# far.MacroAdd

---

```
Id = far.MacroAdd (Area, Flags, AKey, SequenceText, Description, Cal
```

**Parameters:**

Area:	flags (defaults to <a href="#">MACROAREA_COMMON</a> )
Flags:	flags (defaults to 0)
AKey:	table <a href="#">tInputRecord</a> , or string (e.g. "CtrlShiftF12")
SequenceText:	string
Description:	string (defaults to "")
Callback:	function, or nil
Priority:	integer (defaults to 50)

**Returns:**

Id:	userdata (success), or nil (failure) used in Callback and <a href="#">far.MacroDelete</a>
-----	--

**Far API used:**

[MacroControl](#)([MCTL\\_ADDMACRO](#))

**Example:**

```
local F = far.Flags

local function Callback (Id, Flags)
    return some_condition and another_condition
end

local Id = far.MacroAdd(
    F.MACROAREA_SHELL,
    0,
    "CtrlShiftF12",
    'msgbox("test","test")',
    "cool macro",
    Callback)
```

## **far.MacroCheck**

---

```
result = far.MacroCheck (SequenceText, Flags)
```

**Parameters:**

SequenceText: string  
Flags: flags (defaults to 0)

**Returns:**

result: boolean

**Far API used:**

[MacroControl](#)([MCTL\\_SENDSTRING](#), [MSSC\\_CHECK](#))

## **far.MacroDelete**

---

```
result = far.MacroDelete (Id)
```

**Parameters:**

  Id:       userdata (value returned by [far.MacroAdd](#))

**Returns:**

  result: boolean

**Far API used:**

[MacroControl](#)([MCTL\\_DELMACRO](#))

# **far.MacroExecute**

---

```
result = far.MacroExecute (SequenceText, Flags, ...)
```

**Parameters:**

SequenceText: string  
Flags: flags (optional)  
...: arguments for passing to script (optional)

**Returns:**

result: table (array of elements returned by the script), or nil

**Description:**

Execute a script in the context of LuaMacro plugin.

**Notes:**

1. Arguments being passed to the script may be of the following Lu string, boolean, number, nil, "int64", or "binary".
2. The returned table contains field "n" - number of elements in t Use it instead of #result, since there can be elements with nil

To properly unpack the table, do:

```
unpack(result, 1, result.n)
```

**Definitions:**

**"int64"**

Type of userdata, produced by [bit64](#) library.

**"binary"**

By convention, it is a string enclosed in a table as its first e The string is intended to be passed to and from without conversi

**Far API used:**

[MacroControl\(MCTL\\_EXECSTRING\)](#)

## **far.MacroGetArea**

---

```
result = far.MacroGetArea ()
```

**Parameters:**

none

**Returns:**

result: integer

**Far API used:**

[MacroControl\(MCTL\\_GETAREA\)](#)

## **far.MacroGetLastError**

---

```
result = far.MacroGetLastError()
```

**Parameters:**

none

**Returns:**

```
result: table
    ErrCode: integer
    ErrPosX: integer
    ErrPosY: integer
    ErrSrc: string
```

**Far API used:**

[MacroControl\(MCTL\\_GETLASTERROR\)](#)

## **far.MacroGetState**

---

```
result = far.MacroGetState ()
```

**Parameters:**

none

**Returns:**

result: integer

**Far API used:**

[MacroControl\(MCTL\\_GETSTATE\)](#)

## **far.MacroLoadAll**

---

```
result = far.MacroLoadAll ([path [,flags]])
```

**Parameters:**

path: string, or nil  
flags: flags

**Returns:**

result: boolean

**Far API used:**

[MacroControl\(MCTL\\_LOADALL\)](#)

## far.MacroPost

---

```
result = far.MacroPost (SequenceText, Flags, AKey)
```

**Parameters:**

SequenceText: string  
Flags: flags (defaults to 0)  
AKey: table [tInputRecord](#), or string (e.g. "CtrlShiftF12");

**Returns:**

result: boolean

**Far API used:**

[MacroControl](#)([MCTL\\_SENDSTRING](#), [MSSC\\_POST](#))

## **far.MacroSaveAll**

---

```
result = far.MacroSaveAll ()
```

**Parameters:**

none

**Returns:**

result: boolean

**Far API used:**

[MacroControl\(MCTL\\_SAVEALL\)](#)

## **Plugins Control**

---

## **far.FindPlugin**

---

```
plugin_handle = far.FindPlugin (Param1, Param2)
```

**Parameters:**

Param1: flags

Param2: string (either ModuleName or GUID, depending on Param1)

**Returns:**

plugin\_handle: userdata, or nil

**Far API used:**

[PluginsControl](#)

## **far.ForcedLoadPlugin**

---

```
plugin_handle = far.ForcedLoadPlugin (Param1, Param2)
```

**Parameters:**

Param1: flags  
Param2: string

**Returns:**

plugin\_handle: userdata, or nil

**Far API used:**

[PluginsControl](#)

# far.GetPluginInformation

---

```
Info = far.GetPluginInformation (Handle)
```

**Parameters:**

Handle: userdata

**Returns:**

```
Info:    table, or nil
        ModuleName: string
        Flags:      flags
        PInfo:     table
                    Flags:      flags
                    DiskMenu:   table tPluginMenuItem
                    PluginMenu: table tPluginMenuItem
                    PluginConfig: table tPluginMenuItem
                    CommandPrefix: string, or nil
        GInfo:     table
                    MinFarVersion: table (array of 5 integers)
                    Version:       table (array of 5 integers)
                    Guid:          string (binary GUID)
                    Title:         string
                    Description:  string
                    Author:        string
```

**Far API used:**

[PluginsControl](#)

## **far.GetPlugins**

---

```
plugin_handles = far.GetPlugins ()
```

**Parameters:**

none

**Returns:**

plugin\_handles: table (array of plugin handles)

**Far API used:**

[PluginsControl](#)

## **far.IsPluginLoaded**

---

```
result = far.IsPluginLoaded (Guid)
```

**Parameters:**

    Guid:     string (either binary or text-form plugin's GUID)

**Returns:**

    result:   boolean

**Description:**

    Returns *true* if the plugin is loaded in memory.

**Far API used:**

[PluginsControl](#): [PCTL\\_FINDPLUGIN](#), [PCTL\\_GETPLUGININFORMATION](#)

# **far.LoadPlugin**

---

```
plugin_handle = far.LoadPlugin (Param1, Param2)
```

**Parameters:**

Param1: flags  
Param2: string

**Returns:**

plugin\_handle: userdata, or nil

**Far API used:**

[PluginsControl](#)

## **far.UnloadPlugin**

---

```
result = far.UnloadPlugin (Handle)
```

**Parameters:**

Handle: userdata

**Returns:**

result: boolean

**Far API used:**

[PluginsControl](#)

# Settings Control

---

**Far Settings Control** contains one function [`far.CreateSettings`](#). When called, this function returns an object (userdata) that has a few methods:

- [`CreateSubkey`](#)
- [`Delete`](#)
- [`Enum`](#)
- [`Free`](#)
- [`Get`](#)
- [`OpenSubkey`](#)
- [`Set`](#)

## far.CreateSettings

---

```
obj = far.CreateSettings (Guid, Location)
```

**Parameters:**

    Guid:     string (optional; defaults to calling plugin's ID)  
    Location: flags (optional; defaults to [PSL\\_ROAMING](#))

**Returns:**

    obj:     userdata

**Note:**

    For creating an object for accessing Far settings, pass Guid == "f"

**Far API used:**

[SettingsControl](#)

# **far.FreeSettings**

---

`far.FreeSettings ()`

**Parameters:**

none

**Returns:**

nothing

**Description:**

Frees all settings handles that might be open by previous calls to [far.CreateSettings](#) and not freed.

**Far API used:**

[SettingsControl](#)

**WARNING:**

For debug purposes only!

Normally you should use method [Free](#) to free each settings handle in

## **obj>CreateSubkey**

---

```
subkey = obj>CreateSubkey (Root, Name [, Description])
```

**Parameters:**

Root: flag  
Name: string  
Description: string, or nil

**Returns:**

subkey: integer, or nil

**Notes:**

1. If the specified subkey does not exist, it is created.
2. If *Description* is nil, the description of the subkey is neither created nor modified.

**Far API used:**

[SettingsControl](#)

## **obj:Delete**

---

```
result = obj:Delete (Root, Name)
```

**Parameters:**

Root: flag  
Name: string, or nil

**Returns:**

result: boolean

**Note:**

When *Name* is nil, *Root* is deleted.

**Far API used:**

[SettingsControl](#)

## obj:Enum

---

```
items = obj:Enum (Root[, from[, to]])
```

**Parameters:**

Root: flag  
from: integer; 1-based; negative values count from the end; defa  
to: integer; 1-based; negative values count from the end; defa

**Returns:**

items: table (array of tables), or nil  
items.Count: total number of items  
#items: returned number of items  
Each table in the array has the following fields:  
if *obj* was created with plugin's GUID:  
    Name: string  
    Type: integer  
if *obj* was created with Far's GUID:  
    Name: string  
    Param: string  
    PluginId: string (GUID)  
    File: string  
    Time: number (number of milliseconds elapsed since Janua  
    Lock: boolean

**Far API used:**

[SettingsControl](#)

## **obj:Free**

---

**obj:Free ()**

**Parameters:**

none

**Returns:**

nothing

**Note:**

If Free() is never called, an object will eventually be freed by the garbage collector. Still, it is recommended to call Free() when the object is no more needed.

**Far API used:**

[SettingsControl](#)

# **obj:Get**

---

```
result = obj:Get (Root, Name, Type)
```

**Parameters:**

Root: flag  
Name: string  
Type: flags

**Returns:**

result: flags, for Type == [FST QWORD](#)  
string, for Type == [FST STRING](#) (in UTF-8 encoding)  
string, for Type == [FST DATA](#)  
nil ([SettingsControl](#) failed, or Type is unknown)

**Far API used:**

[SettingsControl](#)

## **obj:OpenSubkey**

---

subkey = obj:OpenSubkey (Root, Name)

**Parameters:**

Root: flag  
Name: string

**Returns:**

subkey: integer, or nil

**Note:**

If the specified subkey does not exist, nil is returned.

**Far API used:**

[SettingsControl](#)

## **obj:Set**

---

```
result = obj:Set (Root, Name, Type, Data)
```

**Parameters:**

Root: flag  
Name: string, or nil  
Type: flags  
Data: flags, for Type == [FST\\_QWORD](#)  
string, for Type == [FST\\_STRING](#) (in UTF-8 encoding)  
string, for Type == [FST\\_DATA](#)

**Returns:**

result: boolean

**Note:**

To set description for a key, pass *Name* as nil and *Type* as [FST\\_STR](#)

**Far API used:**

[SettingsControl](#)

## **Far Standard Functions**

---

## **far.ConvertPath**

---

```
Dest = far.ConvertPath (Src [, Mode])
```

**Parameters:**

Src: string

Mode: flags, or nil (default="CPM\_FULL")

**Returns:**

Dest: string

**Far API used:**

[ConvertPath](#)

## **far.CopyToClipboard**

---

```
result = far.CopyToClipboard (str [,type])
```

**Parameters:**

str: string, or nil  
type: flag (defaults to [FCT\\_STREAM](#))

**Returns:**

result: boolean

**Note:**

To clear the clipboard, pass *str* as nil

**Far API used:**

[CopyToClipboard](#)

## **far.FarClock**

---

```
result = far.FarClock ()
```

**Parameters:**

none

**Returns:**

result: number

**Far API used:**

[FarClock](#)

## **far.FormatFileSize**

---

```
result = far.FormatFileSize (Size, Width, Flags [, MinSizeIndex])
```

**Parameters:**

Size: number  
Width: integer  
Flags: flags

MinSizeIndex: integer (only takes effect when *Flags* contain [FFFS\\_M](#))

**Returns:**

result: string

**Far API used:**

[FormatFileSize](#)

## **far.GetCurrentDirectory**

---

```
CurDir = far.GetCurrentDirectory ()
```

**Parameters:**

none

**Returns:**

CurDir: string

**Far API used:**

[GetCurrentDirectory](#)

## **far.GetFileOwner**

---

```
result = far.GetFileOwner (Computer, Name)
```

**Parameters:**

Computer: string, or nil  
Name: string

**Returns:**

result: string, or nil

**Far API used:**

[GetFileOwner](#)

## **far.GetNumberOfLinks**

---

```
result = far.GetNumberOfLinks (Name)
```

**Parameters:**

Name: string

**Returns:**

result: integer

**Far API used:**

[GetNumberOfLinks](#)

## **far.GetPathRoot**

---

```
Root = far.GetPathRoot (Path)
```

**Parameters:**

Path: string

**Returns:**

Root: string

**Far API used:**

[GetPathRoot](#)

## **far.GetReparsePointInfo**

---

```
Dest = far.GetReparsePointInfo (Src)
```

**Parameters:**

Src: string

**Returns:**

Dest: string, or nil

**Far API used:**

[GetReparsePointInfo](#)

## **far.InputRecordToName**

---

```
KeyComb = far.InputRecordToName (Input)
Ctrl,Alt,Shift,Key = far.InputRecordToName (Input, true)
```

**Parameters:**

Input: table [tInputRecord](#)  
(Field *EventType* is mandatory, other fields  
are optional. If some field is missing, the default  
value of 0/false is used, except for *RepeatCount*  
that defaults to 1).

**Returns:**

KeyComb: string  
  
Ctrl: "Ctrl", "RCtrl", or *false*  
Alt: "Alt", "RAlt", or *false*  
Shift: "Shift", or *false*  
Key: key without key-modifiers, or *false*

In the case of failure *nil* is returned.

**Far API used:**

[FarInputRecordToName](#)

## **far.LIsAlpha**

---

```
result = far.LIsAlpha (str)
```

**Parameters:**

str: string (only its first character is tested)

**Returns:**

result: boolean

**Far API used:**

[LIsAlpha](#)

## **far.LIsAlphanum**

---

```
result = far.LIsAlphanum (str)
```

**Parameters:**

str: string (only its first character is tested)

**Returns:**

result: boolean

**Far API used:**

[LIsAlphanum](#)

## **far.LIsLower**

---

```
result = far.LIsLower (str)
```

**Parameters:**

str: string (only its first character is tested)

**Returns:**

result: boolean

**Far API used:**

[LIsLower](#)

## **far.LIsUpper**

---

```
result = far.LIsUpper (str)
```

**Parameters:**

str: string (only its first character is tested)

**Returns:**

result: boolean

**Far API used:**

[LIsUpper](#)

## **far.LLowerBuf**

---

```
result = far.LLowerBuf (str)
```

**Parameters:**

  str:     string

**Returns:**

  result: string

**Far API used:**

[LLowerBuf](#)

## **far.LStricmp**

---

```
result = far.LStricmp (str1, str2)
```

**Parameters:**

    str1: string  
    str2: string

**Returns:**

    result: integer

**Far API used:**

[LStricmp](#)

## **far.LStrnicmp**

---

```
result = far.LStrnicmp (str1, str2, num)
```

**Parameters:**

str1: string  
str2: string  
num : integer

**Returns:**

result: integer

**Far API used:**

[LStrnicmp](#)

## **far.LUpperBuf**

---

```
result = far.LUpperBuf (str)
```

**Parameters:**

  str:     string

**Returns:**

  result: string

**Far API used:**

[LUpperBuf](#)

## **far.MkLink**

---

```
result = far.MkLink (Src, Dest, Type, Flags)
```

**Parameters:**

Src: string  
Dest: string  
Type: flags  
Flags: flags

**Returns:**

result: boolean

**Far API used:**

[MkLink](#)

## **far.MkTemp**

---

```
result = far.MkTemp ([prefix])
```

**Parameters:**

prefix: string, or nil

**Returns:**

result: string

**Far API used:**

[MkTemp](#)

## **far.NameToInputRecord**

---

```
Record = far.NameToInputRecord (Name)
```

**Parameters:**

Name: string

**Returns:**

Record: table [tInputRecord](#), or nil  
er

**Far API used:**

[FarNameToInputRecord](#)

## **far.PasteFromClipboard**

---

```
result = far.PasteFromClipboard ([type])
```

**Parameters:**

type: flag (defaults to [FCT\\_ANY](#))

**Returns:**

result: string, or nil

**Far API used:**

[PasteFromClipboard](#)

## **far.ProcessName**

---

```
Result = far.ProcessName (Op, Mask, Name, Flags, Size)
```

**Parameters:**

Op: flag (either of [PN\\_CMPNAME](#), [PN\\_CMPNAMELIST](#), [PN\\_GENERATENAME](#))  
Mask: string  
Name: string (ignored when Op==[PN\\_CHECKMASK](#))  
Flags: flags (any combination of [PN\\_SKIPPATH](#), [PN\\_SHOWERRORMESSAGE](#);  
Size: integer 0...0xFFFF (only used when Op==[PN\\_GENERATENAME](#); def

**Returns:**

Result: boolean, or string.  
String is returned on success if Op==[PN\\_GENERATENAME](#)

**Far API used:**

[ProcessName](#)

# far.RecursiveSearch

---

far.RecursiveSearch (InitDir, Mask, UserFunc [, Flags [, ...]])

**Parameters:**

InitDir: string  
Mask: string  
UserFunc: function  
Flags: flags  
...: additional arguments, any Lua values

**Returns:**

In case the search was stopped by *UserFunc*: all values returned by *UserFunc*.  
In case no more files left: nothing.  
In case of error in *UserFunc*: nothing.

**Description:**

On every file found, *UserFunc* is called with the following arguments:  
(1) a [tPluginPanelItem](#) table  
(2) the full pathname of the found file  
(3...) additional arguments (if any) that were passed to far.R

If the 1-st return value of *UserFunc* is neither *false* nor *nil* then the search is stopped.

**Far API used:**

[FarRecursiveSearch](#)

## **far.TruncPathStr**

---

```
result = far.TruncPathStr (Str, MaxLength)
```

**Parameters:**

  Str:       string  
  MaxLength: integer

**Returns:**

  result:     string

**Far API used:**

[TruncPathStr](#)

## **far.TruncStr**

---

```
result = far.TruncStr (Str, MaxLength)
```

**Parameters:**

Str: string  
MaxLength: integer

**Returns:**

result: string

**Far API used:**

[TruncStr](#)

## **far.XLat**

---

```
result = far.XLat (Line, StartPos, EndPos, Flags)
```

**Parameters:**

Line: string  
StartPos: integer (optional, default=1)  
EndPos: integer (optional, default=Line:len())  
Flags: flags

**Returns:**

result: string, or nil

**Far API used:**

[XLat](#)

## **Regular expression functions**

---

## **regex.exec, regex.execW**

---

```
from, to, offsets = regex.exec (s, pattern, [init], [cflags])
```

**Parameters:**

```
s      : string
pattern : string
init    : integer (default=1; can be negative)
cflags  : string (any combination of 'i','m','o','s' and 'x')
```

**Returns:**

```
from    : integer
to     : integer
offsets : table - array of captures' offsets (there can be false
                  in place of captures not participated in the mat
```

**Example:**

If the whole match is at offsets 10,20 and substring matches are at offsets 12,14 and 16,19 then the function returns the following  
10, 20, { 12,14,16,19 }.

**Note:**

*regex.execW* receives its 1-st argument in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## **regex.find, regex.findW**

---

```
from, to [,c1 [,c2]...] = regex.find (s, pattern, [init], [cflags])
```

**Parameters:**

```
s      : string
pattern : string
init    : integer (default=1; can be negative)
cflags  : string (any combination of 'i','m','o','s' and 'x')
```

**Returns:**

```
from   : integer
to     : integer
c1,... : strings (there can be false in place of captures
                  not participated in the match)
```

**Description:**

The function is similar to *string.find*, but differs from it by the meaning of its optional 4-th parameter:  
-- *cflags* : compilation flags;

**Note:**

*regex.findW* receives its 1-st argument and returns captures in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## regex.gmatch, regex.gmatchW

---

```
for c1 [,c2 ...] in regex.gmatch (s, pattern [, cflags]) do
    .....
end
```

**Parameters:**

s : string  
pattern : string  
cflags : string (any combination of 'i','m','o','s' and 'x')

**Returns:**

c1,... : strings (there can be *false* in place of captures  
not participated in the match)

**Description:**

The function is API-compatible with *string.gmatch*, with the  
following extensions:

-- *cflags* : optional 3-rd parameter: compilation flags;

**Note:**

*regex.gmatchw* receives its 1-st argument and returns captures in  
UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## **regex.gsub, regex.gsubW**

---

```
res, nmatch, nrep = regex.gsub (s, pattern, repl, [n], [cflags])
```

**Parameters:**

s: string  
pattern: string  
repl: string, or table, or function  
n: integer  
cflags: string (any combination of 'i', 'm', 'o', 's' and 'x')

**Returns:**

res: string  
nmatch: integer  
nrep: integer

**Description:**

The function is API-compatible with *string.gsub*, with the following extensions:  
-- *cflags* : optional 5-th parameter: compilation flags;  
-- *nrep* : 3-rd return value: number of replacements made;  
-- *repl* can specify more captures than in *string.gsub*: %0-%9, %A-%

**Note:**

*regex.gsubW* receives its 1-st argument and returns captures in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## **regex.match, regex.matchW**

---

```
c1 [,c2]... = regex.match (s, pattern, [init], [cflags])
```

**Parameters:**

```
s      : string
pattern : string
init    : integer (default=1; can be negative)
cflags  : string (any combination of 'i','m','o','s' and 'x')
```

**Returns:**

```
c1,... : strings (there can be false in place of captures
                  not participated in the match)
```

**Description:**

The function is API-compatible with *string.match*, with the following extensions:  
-- *cflags* : optional 4-th parameter: compilation flags;

**Note:**

*regex.matchW* receives its 1-st argument and returns captures in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## **regex.new**

---

```
regex_object = regex.new (pattern [, cflags])
```

**Parameters:**

pattern: string  
cflags: string (any combination of 'i','m','o','s' and 'x')

**Returns:**

regex\_object: userdata

**Description:**

Compiles pattern into the internal form.

Compilation flags can be specified in either of two ways:

- If the pattern is enclosed in '/' (forward slashes)  
then the flags can follow the trailing slash.  
In this case, the *cflags* parameter is ignored.
- Via the *cflags* parameter.

**Object methods:**

[bracketscount](#), [find](#), [gsub](#), [match](#)

**Far API used:**

[RegExpControl](#)

## method bracketscount

---

```
count = regex_object:bracketscount ()
```

**Parameters:**

none

**Returns:**

count : integer

**Far API used:**

[RegExpControl](#)

## method exec, execW

---

```
from, to, offsets = regex_object:exec (subj [, init])
```

**Parameters:**

```
subj    : string  
init    : integer (default=1; can be negative)
```

**Returns:**

```
from    : integer  
to     : integer  
offsets : table - array of captures' offsets (there can be false  
                  in place of captures not participated in the mat
```

**Example:**

If the whole match is at offsets 10,20 and substring matches are at offsets 12,14 and 16,19 then the method returns the following:  
10, 20, { 12,14,16,19 }.

**Note:**

*regex\_object:execW* receives its 1-st argument in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## method **find**, **findW**

---

```
from, to [,c1 [,c2]...] = regex_object:find (subj [, init])
```

**Parameters:**

```
subj    : string  
init    : integer (default=1; can be negative)
```

**Returns:**

```
from   : integer  
to     : integer  
c1,... : strings (there can be false in place of captures  
                  not participated in the match)
```

**Note:**

*regex\_object:findW* receives its 1-st argument and returns captures  
in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## method gsub, gsubW

---

```
res, nmatch, nrep = regex_object:gsub (s, repl, [n])
```

**Parameters:**

s: string  
repl: string, or table, or function  
n: integer

**Returns:**

res: string  
nmatch: integer  
nrep: integer

**Note:**

*regex\_object:gsubW* receives its 1-st argument and returns captures in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## method match, matchW

---

```
c1 [,c2 ...] = regex_object:match (subj [, init])
```

**Parameters:**

subj : string  
init : integer (default=1; can be negative)

**Returns:**

c1,... : strings (there can be *false* in place of captures  
not participated in the match)

**Note:**

*regex\_object:matchW* receives its 1-st argument and returns captures  
in UTF-16LE encoding.

**Far API used:**

[RegExpControl](#)

## **Other functions**

---

## **far.CPluginStartupInfo**

---

```
Info = far.CPluginStartupInfo()
```

**Parameters:**

none

**Returns:**

Info: light userdata (a pointer to struct [PluginStartupInfo](#))

**Description:**

The function is used to give access to Far functions to DLL's other than the plugin DLL.

# **far.MakeMenuItems**

---

```
Items = far.MakeMenuItems (...)
```

**Parameters:**

Arbitrary number of arguments.  
Each argument may be of any Lua type.

**Returns:**

Items: table (array of tables)  
- each element is a table in the form  
  { text=<string>, arg=<argument value> }  
- number of elements >= number of arguments, as one  
  argument can have a multiple line representation.

**Description:**

This function returns representation of its arguments, suitable  
for passing to function [far.Menu](#) as its *Items* parameter.

## far.PluginStartupInfo

---

```
Info = far.PluginStartupInfo ()
```

**Parameters:**

none

**Returns:**

Info: table  
fields of Info:  
  ModuleName: string  
  ModuleDir: string (has a backslash at the end)  
  PluginGuid: string

**Description:**

Returns a table containing the data elements of the [\*PluginStartupInfo\*](#) struct that plugin obtains when FAR calls its function [\*SetStartupInfoW\*](#).

# **far.RunDefaultScript**

---

```
Result = far.RunDefaultScript ()
```

**Parameters:**

none

**Returns:**

Result: boolean (true = success)

**Description:**

Runs the default script of the plugin.

This usually means that all plugin's code is **immediately** reloaded.

This is different from setting *far.ReloadDefaultScript = true* which a deferred reload (that occurs when [export.Open](#) is called the next

**See also:**

[LF\\_RunDefaultScript](#)

[far.ReloadDefaultScript](#)

[RecreateLuaState](#)

## **far.Show**

---

Item, Position = far.Show (...)

**Parameters:**

Arbitrary number of arguments.  
Each argument may be of any Lua type.

**Returns:**

Item, Position - same as in [far.Menu](#).

**Description:**

This function shows representation of its arguments in a menu window. It is handy for displaying multiple values at once, or long multi-line strings.

The function calls [far.MakeMenuItems](#) internally, then [far.Menu](#).

Due to using [far.MakeMenuItems](#) the returned *Item* table contains the value of the corresponding original argument under the field '

# far.Timer

---

```
timer = far.Timer (interval, handler [, ...])
```

**Parameters:**

interval : integer (milliseconds)  
handler : function  
... : additional arguments (any Lua types)

**Returns:**

timer : userdata (timer object), or nil

**Description:**

The function creates a periodic timer. Every time the next *interval* milliseconds are elapsed, the *handler* is called with the timer object as its first argument followed by all the additional arguments (if any) passed to far.Timer.

The returned "timer object" can be used to query or modify the timer parameters. The object has an API similar to Delphi classes: properties, methods, events.

**Notes:**

1. *handler* can be called only when the plugin returns control to Far. Thus the real interval between *handler* calls is not exact and can significantly differ from the specified value.
2. This function requires that the plugin DLL has [ProcessSynchroEventW](#) exported. (There is not such a requirement for [export.ProcessSynchroEvent](#) though, as it is not called by the Timer function and methods).

**Far API used:**

[AdvControl](#) (ACTL\_SYNCHRO), [ProcessSynchroEventW](#).

**Windows API used:**

CreateTimerQueueTimer, CreateTimerQueue, ChangeTimerQueueTimer, De

# **Properties**

---

## **Closed**

---

```
result = timer.Closed
```

This is a read-only boolean property.

## Enabled

---

```
value = timer.Enabled  
or  
timer.Enabled = value
```

**Description:**

This property is about calling the handler function. The timer continues to run even when not enabled, but the handler is not called. The initial value of this property is *true*.  
This is a read/write boolean property.

## Interval

---

```
value = timer.Interval  
or  
timer.Interval = value
```

**Description:**

Query or set the interval (in milliseconds) of the timer.  
This is a read/write integer property.

When the interval is set, the time already accumulated by the time is discarded, and the new accumulation is started from 0.

## **Methods**

---

# Close

---

timer:Close ()

**Parameters:**

none

**Returns:**

nothing

**Description:**

Closes the timer.

**Note:**

A timer object is garbage-collected only when its lua\_State closes thus it is the application's responsibility to call the *Close* method when the timer is not needed anymore.

## **Events**

---

## OnTimer

---

```
handler = timer.OnTimer  
or  
timer.OnTimer = handler
```

**Description:**

Query or set the timer handler function.

The handler function is called with the timer object as its first followed by all the additional arguments (if any) passed to [far.Ti](#)

## **Tables**

---

## **far.Colors**

---

`far.Colors`

**Description:**

A table containing values of all constants (32 bit integers) from *farcolor.hpp* indexed by their string names.

**Example of use:**

```
local FC = far.Colors
far.AdvControl ("ACTL_SETARRAYCOLOR", nil,
    { Flags="FCLR_REDRAW", StartIndex=FC.COL_EDITORTEXT,
        0x4E, 0xE4
    })
```

## **far.Flags**

---

`far.Flags`

**Description:**

A table containing values of various FAR and Windows console constants keyed by their string names.

Keys of the table are always strings. Values are 64-bit integers represented either as numbers, or strings in the format acceptable by the [bit64 library](#). Therefore, these values should be tested or manipulated only by functions of that library.

# **far.Guids**

---

far.Guids

**Description:**

A table containing GUIDs of various Far Manager dialogs and menus.  
This table is auto-generated from file *DlgGuid.hpp* at the build ti

GUIDs are given in upper-case text representation, e.g.

FindFileDialog = '8C9EAD29-910F-4B24-A669-EDAFBA6ED964'

# **"Structures"**

---

# **tAnalyseInfo**

---

**tAnalyseInfo** is a table with the following predefined fields:

```
FileName: string (UTF-8)
Buffer:   string
OpMode:   flags
Handle:   any Lua type
```

## **Note:**

In [export.Analyse](#) Handle is nil.

In [export.Open](#) Handle is the value returned by [export.Analyse](#).

## **tCoord**

---

**The table has the following fields:**

```
X    : integer  
Y    : integer
```

## **tFarColor**

---

**tFarColor** is a table with the following predefined fields:

Flags:            flags  
ForegroundColor: integer  
BackgroundColor: integer

# **tFarDialogItem**

---

**tFarDialogItem** is a table (array)

Fields of a dialog item:

```
[1] Type:           integer, or string
[2] X1:             integer
[3] Y1:             integer
[4] X2:             integer
[5] Y2:             integer
[6] Selected/ListItems: integer/table
    When table, its array part is a table per item:
        ListItems[i]:
            Flags : flags
            Text  : string
    Its hash part has a field:
        SelectIndex : integer
[7] History:         string
[8] Mask:            string
[9] Flags:           flags
[10] Data:            string
[11] MaxLength:       integer
[12] UserData:        integer
```

## **tFarDialogItemColors**

---

**tFarDialogItemColors** is a table with the following predefined fields:

Flags: flags

The array part of the table keeps colors, represented by [tFarColor](#) tables. Color count can be obtained via the operator #.

## **tFarKey**

---

**tFarKey** is a table with the following predefined fields:

VirtualKeyCode:	integer
ControlKeyState:	integer

## **tFarPanelDirectory**

---

**tFarPanelDirectory** is a table with the following predefined fields:

Name:	string
Param:	string
PluginId:	string (GUID)
File:	string

## **tInfoLine**

---

**tInfoLine** is a table with the following predefined fields:

```
Text:    string
Data:    string
Flags:   flags
```

# tInputRecord

---

The table has the following fields:

```
EventType:           integer  
  
--if EventType is KEY_EVENT:  
KeyDown:            boolean  
RepeatCount:        integer  
VirtualKeyCode:     integer  
VirtualScanCode:    integer  
UnicodeChar:        string  
ControlKeyState:    integer  
  
--if EventType is MOUSE_EVENT:  
see tMouseEventRecord  
  
--if EventType is WINDOW_BUFFER_SIZE_EVENT:  
SizeX:              integer  
SizeY:              integer  
  
--if EventType is MENU_EVENT:  
CommandId:          integer  
  
--if EventType is FOCUS_EVENT:  
SetFocus:            boolean
```

## **tKeyBarLabel**

---

**tKeyBarLabel** is a table with the following predefined fields:

```
VirtualKeyCode: integer  
ControlKeyState: flags  
Text:           string  
LongText:       string
```

## **tMouseEventRecord**

---

**The table has the following fields:**

```
MousePositionX    : integer  
MousePositionY    : integer  
ButtonState       : integer  
ControlKeyState   : integer  
EventFlags        : integer
```

## **tPanelMode**

---

**tPanelMode** is a table with the following predefined fields:

```
ColumnTypes:      string
ColumnWidths:     string
StatusColumnTypes: string
StatusColumnWidths: string
ColumnTitles:     table (array of strings)
Flags:            flags
```

## **tPluginMenuItem**

---

**tPluginMenuItem** is a table with the following predefined fields:

Count: integer  
Guids: array of strings (binary GUID's)  
Strings: array of strings

# tPluginPanelItem

---

**tPluginPanelItem** is a table with the following predefined fields:

LastWriteTime:	number
LastAccessTime:	number
CreationTime:	number
ChangeTime:	number
FileSize:	number
AllocationSize:	number
FileName:	string
AlternateFileName:	string
FileAttributes:	string
Flags:	flags
NumberOfLinks:	integer
CRC32:	integer
Description:	string, or nil
Owner:	string, or nil
CustomColumnData:	table, or nil
UserData:	any type

## Notes

1. **FileAttributes** is a string of concatenated attribute letters (see below). If a letter representing some attribute is present in the string, that means the corresponding attribute is set, otherwise it is not.

a	- archive
c	- compressed
d	- directory
e	- reparse point
h	- hidden
i	- not content indexed
n	- encrypted
o	- offline
p	- sparse file
r	- read only
s	- system file
t	- temporary

u - no scrub data

v - virtual

2. **LastWriteTime**, **LastAccessTime**, **CreationTime** and **ChangeTime** are values measured in number of milliseconds elapsed since January 1, 1601.

## **tSystemTime**

---

**tSystemTime** is a table with the following predefined fields:

wYear:	number
wMonth:	number
wDayOfWeek:	number
wDay:	number
wHour:	number
wMinute:	number
wSecond:	number
wMilliseconds:	number

## **ViewerInfo.CurMode**

---

**ViewerInfo.CurMode** is a table with the following predefined fields:

CodePage:	integer
Flags:	flags
ViewMode:	flag

## **Additional libraries**

---

# bit64

---

As Far 3.0 started to use 64-bit integer values (“flags”) in its API, arose the need to represent and handle them from Lua. This is the role of **bit64** library.

The library represents and accepts flags as either numbers or “64 bit integer” userdata. Values that fit in 53 bits can be represented in both ways. Greater values are always represented as userdata.

Userdata values created by the library support the following operations:  
+, -, \*, /, %, ==, <, <=, unary minus.

`__tostring` metamethod of “64-bit type” userdata returns an acceptable string representation that when passed to [new](#) will produce a value equal to the original one.

The library has the following functions:

- `bnot` – gets a flag, returns a flag
- `band` – gets multiple flags, returns a flag
- `bor` – gets multiple flags, returns a flag
- `bxor` – gets multiple flags, returns a flag
- `lshift` – gets a flag and an integer, returns a flag
- `rshift` – gets a flag and an integer, returns a flag
- `arshift` – gets a flag and an integer, returns a flag
- `add` – gets multiple flags, returns a flag
- `mul` – gets multiple flags, returns a flag
- `sub` – gets 2 flags, returns a flag
- `div` – gets 2 flags, returns a flag
- `mod` – gets 2 flags, returns a flag

The functions above always return a number if the result fits into 53 bits.

- `eq` – gets 2 flags, returns a boolean
- `lt` – gets 2 flags, returns a boolean
- `le` – gets 2 flags, returns a boolean

The functions above are useful for comparisons where each operand may be either a “64 bit integer” type userdata or a number. They compare underlying 64-bit integers.

- *new* – gets a string, number or flag representation, returns a flag
- *type* – gets a value, returns a string (or nil)

## **bit64.type**

---

```
result = bit64.type (val)
```

**Parameters:**

  val:     any type

**Returns:**

  result: string, or nil

**Description:**

The function returns string "*64 bit integer*" if argument *val* is a userdata of the type created by this library, otherwise nil is returned.

## bit64.new

---

```
result = bit64.new (arg)
```

**Parameters:**

arg: string, number or userdata

**Returns:**

result: userdata, or nil

**Description:**

This function creates a new "64-bit int" type userdata from the input.

If *arg* is a string, it must be an acceptable string representation of a 64-bit integer. The function returns *nil* if the input string can not be converted.

The string representation of an integer acceptable by the function must have either the hexadecimal form

-?0[xX][0-9A-Fa-f]{1,16}, e.g. "0x56DF5"

or the decimal form

-?[0-9]+, e.g. "-234567".

If *arg* is a number, it must fit in 53 bits, otherwise *nil* is returned.

If *arg* is a "64-bit int" type userdata then a new userdata with the same value is created and returned.

## **win**

---

**win** library contains Lua bindings for some Windows API and other functions.

## **Files and directories**

---

## **win.CopyFile**

---

```
result = win.CopyFile (source, target [, fail_if_exists])
```

**Parameters:**

source:	string
target:	string
fail_if_exists:	boolean (defaults to false)

**Returns:**

result:	true (or: nil, errormessage)
---------	------------------------------

**Windows API used:**

CopyFileW

## win.CreateDir

---

```
result = win.CreateDirectory (path [, flags])
```

**Parameters:**

```
path:      string
flags:     string (set of characters; optional)
           't' - stands for 'tolerant'
           'o' - stands for 'original'
           default: ''
```

**Returns:**

```
result:    true (or: nil, errormessage)
```

**Description:**

This function creates a directory specified by the *path* parameter.  
Nested directories can be created by a single call.

Flag 't': If the target directory already exists, then the function  
Flag 'o': The function will not attempt to create nested directories.  
The *path* argument is passed to *CreateDirectoryW* as is.

**Note:**

For backwards compatibility, a non-string true value passed as the  
is equivalent to flags=='t'.

**Windows API used:**

```
CreateDirectoryW
```

## **win.DeleteFile**

---

```
result = win.DeleteFile (filename)
```

**Parameters:**

filename: string

**Returns:**

result: true (or: nil, errormessage)

**Windows API used:**

DeleteFileW

## **win.GetCurrentDir**

---

```
dir = win.GetCurrentDir ()
```

**Parameters:**

none

**Returns:**

dir : string (or: nil, errormessage)

**Windows API used:**

GetCurrentDirectoryW

## **win.GetDriveType**

---

```
type = win.GetDriveType ([root])
```

**Parameters:**

root : string

**Returns:**

type : one of the following strings:  
    "unknown type"  
    "no root directory"  
    "removable"  
    "fixed"  
    "remote"  
    "cdrom"  
    "ramdisk"

**Windows API used:**

GetDriveTypeW

## **win.GetFileAttr**

---

```
attr = win.GetFileAttr (FileName)
```

**Parameters:**

  FileName: string

**Returns:**

  attr:     string (see [tPluginPanelItem](#) page), or nil

**Windows API used:**

  GetFileAttributesW

# **win.GetFileInfo**

---

```
Info = win.GetFileInfo (FileName)
```

## **Parameters**

**FileName:** string

A valid directory or path and filename, which can contain wildcard characters (\*) and (?)

## **Returns**

On success: [tPluginPanelItem](#) table

On failure: nil + error message

## **Description**

This function provides information about the specified file. The same information can be obtained with [far.RecursiveSearch](#) function, but this function has simpler API and is much faster when single file data are needed.

## **Windows API used**

FindFirstFile, FindClose

## **win.GetLogicalDriveStrings**

---

```
drives = win.GetLogicalDriveStrings()
```

**Parameters:**

none

**Returns:**

drives: table (array of strings), or nil + error message

**Windows API used:**

GetLogicalDriveStringsW

## win.MoveFile

---

```
result = win.MoveFile (source, target [, flags])
```

**Parameters:**

```
source: string
target: string
flags: string (default = '')
      'c' = MOVEFILE_COPY_ALLOWED
      'd' = MOVEFILE_DELAY_UNTIL_REBOOT
      'r' = MOVEFILE_REPLACE_EXISTING
      'w' = MOVEFILE_WRITE_THROUGH
```

**Returns:**

```
result: true (or: nil, errormessage)
```

**Windows API used:**

```
MoveFileExW
```

## **win.RemoveDir**

---

```
result = win.RemoveDir (path)
```

**Parameters:**

path: string

**Returns:**

result: true (or: nil, errormessage)

**Windows API used:**

RemoveDirectoryW

## **win.RenameFile**

---

This function is an alias to [win.MoveFile](#).

## **win.SearchPath**

---

FullName, FilePart = win.SearchPath (Path, FileName, Extension)

**Parameters:**

Path : string, or nil  
FileName: string  
Extension: string, or nil

**Returns:**

FullName: string, or nil  
FilePart: string, or nil

**Windows API used:**

SearchPathW

## **win.SetCurrentDir**

---

```
result = win.SetCurrentDir (dir)
```

**Parameters:**

dir : string

**Returns:**

result : boolean

**Windows API used:**

SetCurrentDirectoryW

## **win.SetFileAttr**

---

```
result = win.SetFileAttr (FileName, Attr)
```

**Parameters:**

  FileName: string  
  Attr:      string (see [tPluginPanelItem](#) page)

**Returns:**

  result:    boolean

**Windows API used:**

  SetFileAttributesW

## **Registry access**

---

# win.DeleteRegKey

---

```
result = win.DeleteRegKey (RootKey, Key, [AccessMask])
```

**Parameters:**

RootKey: string (one of: "HKLM", "HKCC", "HKCR", "HKCU", "HKU")  
Key: string  
AccessMask: flag ("KEY\_WOW64\_32KEY" or "KEY\_WOW64\_64KEY"; the def

**Returns:**

result: boolean

**Description:**

This function deletes a registry key.

**Example of use:**

```
-- Recursively delete registry keys
function win.DeleteRegKey_Recurse(RootKey, Key, AccessMask)
    while true do
        local subkey = win.EnumRegKey(RootKey, Key, 0, AccessMask)
        if not (subkey and win.DeleteRegKey_Recurse(RootKey, Key.."\\"..
            break
        end
    end
    return win.DeleteRegKey(RootKey, Key, AccessMask)
end
```

## **win.DeleteRegValue**

---

```
result = win.DeleteRegValue (RootKey, Key, [ValueName], [AccessMask]
```

**Parameters:**

RootKey: string (one of: "HKLM", "HKCC", "HKCR", "HKCU", "HKU"  
Key: string  
ValueName: string (optional)  
AccessMask: flag ("KEY\_WOW64\_32KEY" or "KEY\_WOW64\_64KEY"; the def

**Returns:**

result: boolean

**Description:**

This function deletes a registry value.

# win.EnumRegKey

---

```
Subkey = win.EnumRegKey (RootKey, Key, Index, [AccessMask])
```

**Parameters:**

RootKey: string (one of: "HKLM", "HKCC", "HKCR", "HKCU", "HKU")  
Key: string (specify "" if RootKey itself is enumerated)  
Index: integer (0-based)  
AccessMask: flag ("KEY\_WOW64\_32KEY" or "KEY\_WOW64\_64KEY"; the def

**Returns:**

Subkey: string, or nil

**Description:**

This function enumerates subkeys of the given registry key.

**Example of use:**

```
-- Recursively delete registry keys
function win.DeleteRegKey_Recurse(RootKey, Key, AccessMask)
    while true do
        local subkey = win.EnumRegKey(RootKey, Key, 0, AccessMask)
        if not (subkey and win.DeleteRegKey_Recurse(RootKey, Key.."\\"..subkey))
            break
        end
    end
    return win.DeleteRegKey(RootKey, Key, AccessMask)
end
```

## **win.EnumRegValue**

---

```
ValueName = win.EnumRegValue (RootKey, Key, Index, [AccessMask])
```

**Parameters:**

RootKey: string (one of: "HKLM", "HKCC", "HKCR", "HKCU", "HKU")  
Key: string (specify "" if RootKey itself is enumerated)  
Index: integer (0-based)  
AccessMask: flag ("KEY\_WOW64\_32KEY" or "KEY\_WOW64\_64KEY"; the def

**Returns:**

ValueName: string, or nil

**Description:**

This function enumerates values of the given registry key.

# **win.GetRegKey**

---

```
ValueData, DataType = win.GetRegKey (RootKey, Key, ValueName, [AccessMask])
```

**Parameters:**

```
RootKey:      string (one of: "HKLM", "HKCC", "HKCR", "HKCU", "HKU")
Key:         string
ValueName:    string
AccessMask:   flag ("KEY_WOW64_32KEY" or "KEY_WOW64_64KEY"; the default is KEY_WOW64_32KEY)
```

**Returns:**

```
ValueData: string, or number
DataType:  string (one of: "string", "expandstring", "multistring",
                  "dword" and "binary")
```

**Returns on failure:**

```
nil + error message
```

**Description:**

This function gets a value from a registry key.

# **win.SetRegKey**

---

```
result = win.SetRegKey (RootKey, Key, ValueName, DataType, ValueData)
```

**Parameters:**

RootKey: string (one of: "HKLM", "HKCC", "HKCR", "HKCU", "HKU")  
Key: string  
ValueName: string  
DataType: string (one of: "string", "expandstring", "multistring")  
ValueData: string or number  
AccessMask: flag ("KEY\_WOW64\_32KEY" or "KEY\_WOW64\_64KEY"; the def

**Returns:**

result: boolean (true on success)

**Description:**

This function sets a value in a registry key.

## **Text related functions**

---

# win.CompareString

---

```
result = win.CompareString(s1, s2, [locale], [flags])
```

**Parameters:**

```
s1      : string
s2      : string
locale : string, or nil (default="u")
        either of:
          "s" (system default)
          "u" (user default)
          "n" (neutral)
flags   : string, or nil (default="")
        any combination of characters:
          'c' = NORM_IGNORECASE;
          'k' = NORM_IGNOREKANATYPE;
          'n' = NORM_IGNORENONSPACE;
          's' = NORM_IGNORESYMBOLS;
          'w' = NORM_IGNOREWIDTH;
          'S' = SORT_STRINGSORT;
```

**Returns:**

```
result : integer (either of -1, 0, 1), or nil
```

**Note:**

The underlying function *CompareStringW* seems to behave as if flag NORM\_IGNORECASE is always present, even if it is not.

**Windows API used:**

```
CompareStringW
```

## **win.EnumSystemCodePages**

---

```
pages = win.EnumSystemCodePages ([supported])
```

**Parameters:**

    supported : boolean  
        false means CP\_INSTALLED  
        true means CP\_SUPPORTED

**Returns:**

    pages : table (array of strings), or nil + error message

**Windows API used:**

EnumSystemCodePagesW

## **win.GetACP**

---

```
codepage = win.GetACP()
```

**Parameters:**

none

**Returns:**

codepage: integer

**Windows API used:**

GetACP

## **win.GetConsoleCP**

---

```
codepage = win.GetConsoleCP()
```

**Parameters:**

none

**Returns:**

codepage: integer

**Windows API used:**

GetConsoleCP

## **win.GetConsoleOutputCP**

---

```
codepage = win.GetConsoleOutputCP()
```

**Parameters:**

none

**Returns:**

codepage: integer

**Windows API used:**

GetConsoleOutputCP

## **win.GetOEMCP**

---

```
codepage = win.GetOEMCP()
```

**Parameters:**

none

**Returns:**

codepage: integer

**Windows API used:**

GetOEMCP

## **win.lenW**

---

```
length = win.lenW (str)
```

**Parameters:**

str: string

**Returns:**

length: integer

**Description:**

This function is similar to *string.len* except that it works on UTF and returns their lengths as number of characters rather than byt

## **win.MultiByteToWideChar**

---

```
str_out = win.MultiByteToWideChar (str_in, codepage [, flags])
```

**Parameters:**

str\_in : string in multi-byte encoding  
codepage : integer  
flags : string, a combination of 0 or more of the following characters:  
    'p' -> MB\_PRECOMPOSED  
    'c' -> MB\_COMPOSITE  
    'e' -> MB\_ERR\_INVALID\_CHARS  
    'u' -> MB\_USEGLYPHCHARS

**Returns:**

str\_out : string in wide character encoding, or nil+error message

**Windows API used:**

MultiByteToWideChar

## **win.OemToUtf8**

---

```
str_out = win.OemToUtf8 (str_in)
```

**Parameters:**

  str\_in : string in OEM encoding

**Returns:**

  str\_out : string in UTF-8 encoding

## **win.SetConsoleCP**

---

```
result = win.SetConsoleCP(codepage)
```

**Parameters:**

codepage: integer

**Returns:**

result: true (on success) or nil + error message

**Windows API used:**

SetConsoleCP

## **win.SetConsoleOutputCP**

---

```
result = win.SetConsoleOutputCP(codepage)
```

**Parameters:**

codepage: integer

**Returns:**

result: true (on success) or nil + error message

**Windows API used:**

SetConsoleOutputCP

## **win.subW**

---

```
strOut = win.subW (str, from, to)
```

**Parameters:**

str: string  
from: integer (default = 1)  
to: integer (default = -1)

**Returns:**

strOut: string

**Description:**

This function is similar to *string.sub* except that it works on UTF and considers arguments *from* and *to* expressed in characters rather

## **win.Utf16ToUtf8**

---

```
str_out = win.Utf16ToUtf8 (str_in)
```

**Parameters:**

  str\_in : string in UTF-16 (little endian) encoding

**Returns:**

  str\_out : string in UTF-8 encoding

## **win.Utf8ToOem**

---

```
str_out = win.Utf8ToOem (str_in)
```

**Parameters:**

  str\_in : string in UTF-8 encoding

**Returns:**

  str\_out : string in OEM encoding

## **win.Utf8ToUtf16**

---

```
str_out = win.Utf8ToUtf16 (str_in)
```

**Parameters:**

  str\_in : string in UTF-8 encoding

**Returns:**

  str\_out : string in UTF-16 (little endian) encoding

## **win.wcscmp**

---

```
result = win.wcscmp(s1, s2 [, insens])
```

**Parameters:**

s1 : string  
s2 : string  
insens : boolean (defaults to *false*)

**Returns:**

result : integer (either of -1, 0, 1)

**C API used:**

wcscmp, wcsicmp

## **win.WideCharToMultiByte**

---

```
str_out = win.WideCharToMultiByte (str_in, codepage [, flags])
```

**Parameters:**

str\_in : string in UTF-16LE encoding  
codepage : integer  
flags : string, a combination of 0 or more of the following  
characters:  
    'c' -> WC\_COMPOSITECHECK  
    'd' -> WC\_DISCARDNS  
    's' -> WC\_SEPCHARS  
    'f' -> WC\_DEFAULTCHAR

**Returns:**

str\_out : string in *codepage* encoding.

**Windows API used:**

WideCharToMultiByte

## Time related functions

---

## **win.FileTimeToLocalFileTime**

---

```
local_ftime = win.FileTimeToLocalFileTime (ftime)
```

**Parameters:**

ftime : number (number of milliseconds since January 1, 1601)

**Returns:**

local\_ftime : number (number of milliseconds since January 1, 1601)

**Windows API used:**

FileTimeToLocalFileTime

## **win.FileTimeToSystemTime**

---

```
result = win.FileTimeToSystemTime (ftime)
```

**Parameters:**

ftime : number (number of milliseconds since January 1, 1601)

**Returns:**

result : table [tSystemTime](#), or nil

**Windows API used:**

FileTimeToSystemTime

## **win.GetSystemTimeAsFileTime**

---

```
result = win.GetSystemTimeAsFileTime ()
```

**Parameters:**

none

**Returns:**

result: number

**Description:**

The system time is expressed as number of milliseconds since January 1, 1970.

**Windows API used:**

GetSystemTimeAsFileTime

## win.SystemTimeToFileTime

---

```
result = win.SystemTimeToFileTime (systime)
```

**Parameters:**

systime : table [tSystemTime](#) (missing fields default to 0)

**Returns:**

result : number, or nil  
(number of milliseconds since January 1, 1601)

**Windows API used:**

SystemTimeToFileTime

## **Various functions**

---

## win.ExtractKey

---

```
result = win.ExtractKey ()
```

**Parameters:**

none

**Returns:**

result: string, or nil

**Description:**

If a key was pressed, it is extracted, converted to a string and returned. Otherwise, a nil is returned.

**Windows API used:**

[GetStdHandle](#), [PeekConsoleInput](#), [ReadConsoleInput](#)

## win.ExtractKeyEx

---

```
result = win.ExtractKeyEx ()
```

**Parameters:**

none

**Returns:**

result: table [tInputRecord](#), or nil

**Description:**

If a key was pressed or released, the input record is returned.  
Otherwise, a nil is returned.

**Windows API used:**

[GetStdHandle](#), [PeekConsoleInput](#), [ReadConsoleInput](#)

## **win.GetConsoleScreenBufferInfo**

---

```
info = win.GetConsoleScreenBufferInfo ()
```

**Parameters:**

none

**Returns (on success):**

```
info : table
  fields of info:
    SizeX:          integer
    SizeY:          integer
    CursorPositionX: integer
    CursorPositionY: integer
    Attributes:     integer
    WindowLeft:     integer
    WindowTop:      integer
    WindowRight:    integer
    WindowBottom:   integer
    MaximumWindowSizeX: integer
    MaximumWindowSizeY: integer
```

**Returns (on failure):**

nil + error message

**Windows API used:**

GetStdHandle(STD\_OUTPUT\_HANDLE), GetConsoleScreenBufferInfo

## **win.GetCPIInfo**

---

```
info = win.GetCPIInfo (codepage)
```

**Parameters:**

codepage : integer

**Returns:**

info : table, or nil + error message  
fields of *info*:

MaxCharSize	: integer
DefaultChar	: string
LeadByte	: string
UnicodeDefaultChar	: string
CodePage	: integer
CodePageName	: string

**Windows API used:**

GetCPIInfoExW

## win.GetEnv

---

```
result = win.Getenv (var)
```

**Parameters:**

  var: string

**Returns:**

  result: string, or nil

**Description:**

Use this instead of os.getenv, since the latter does not always work correctly.

## **win.GetKeyState**

---

```
keyDown, keyToggled = win.GetKeyState (nVirtKey)
```

**Parameters:**

nVirtKey: integer

**Returns:**

keyDown: boolean

keyToggled: boolean

**Windows API used:**

GetKeyState

## win.GetVirtualKeys

---

```
vkeys = win.GetVirtualKeys ()
```

**Parameters:**

none

**Returns:**

vkeys: table

**Description:**

Returns a table containing values of virtual key codes keyed by their string names (and vice versa). The prefix VK\_ is stripped from the names, so VK\_LBUTTON is represented as "LBUTTON", etc.

**Note:**

If some key code N (where  $0 \leq N < 256$ ) does not have a corresponding name, then vkeys[N] is "" (empty string).

**Example:**

```
local vk = win.GetVirtualKeys()
far.Message(vk.ESCAPE) --> 27
far.Message(vk[27])    --> ESCAPE
```

## win.GlobalMemoryStatus

---

```
status = win.GlobalMemoryStatus ()
```

**Parameters:**

none

**Returns:**

status: table (see its fields below)  
    MemoryLoad: number  
    TotalPhys: number  
    AvailPhys: number  
    TotalPageFile: number  
    AvailPageFile: number  
    TotalVirtual: number  
    AvailVirtual: number  
    AvailExtendedVirtual: number

**Returns on failure:**

nil, error message

**Windows API used:**

GlobalMemoryStatusEx

## **win.IsProcess64bit**

---

```
result = win.IsProcess64bit ()
```

**Parameters:**

none

**Returns:**

result: boolean

**Description:**

Returns **true** if called from FAR x64, **false** otherwise.

# **win.OutputDebugString**

---

`win.OutputDebugString (param)`

**Parameters:**

`param` : string in UTF-8 encoding, or any other type.  
For non-string types, the global function **toString**  
will be called internally.

**Returns:**

`nothing`

**Windows API used:**

`OutputDebugStringW`

## **win.SetEnv**

---

```
result = win.SetEnv (var [,value])
```

**Parameters:**

var : string  
value: string, or nil

**Returns:**

result: boolean

**Description:**

This function sets or deletes a given environment variable.

## **win.ShellExecute**

---

```
result = win.ShellExecute (hwnd, Operation, File, Parameters, Direct
```

**Parameters:**

hwnd: userdata, or nil  
Operation: string, or nil  
File: string  
Parameters: string, or nil  
Directory: string, or nil  
ShowCmd: integer (defaults to SW\_SHOWNORMAL)

**Returns:**

result: integer

**Windows API used:**

ShellExecuteW

## **win.SHGetFolderPath**

---

```
path = win.SHGetFolderPath (nFolder, dwFlags)
```

**Parameters:**

nFolder: integer

dwFlags: integer (optional; default to 0)

**Returns:**

path: string, or nil

**Windows API used:**

SHGetFolderPathW

## **win.Sleep**

---

`win.Sleep (milliseconds)`

**Parameters:**

milliseconds: integer

**Returns:**

nothing

**Windows API used:**

Sleep

## **win.system**

---

```
result = win.system ([command])
```

**Parameters:**

    command: string, or nil  
        (when command==nil or no arguments are passed, the under  
        function *\_wsystem* is called with a NULL argument)

**Returns:**

    result: integer

**Description:**

Executes a command.

**API used:**

*\_wsystem*

## **win.Uuid**

---

There are 3 actions available, depending on arguments.  
All arguments and return values are strings.

**ret = win.Uuid ()**  
Generate a new (binary) UUID

**ret = win.Uuid (binUUID)**  
Convert a UUID to string representation

**ret = win.Uuid (strUUID)**  
Convert a string UUID representation to binary

**Note:**

If the function is given a string argument, it checks its length.  
When the length is 16, the function makes conversion from binary  
to string GUID representation, otherwise it tries to convert from  
string GUID representation to binary one.

# Selene Unicode

---

There are four string-like ctype closures: **unicode.ascii**, **latin1**, **utf8** and **grapheme**.

**ascii** and **latin1** are single-byte like `string`, but use the unicode table for upper/lower and character classes. `ascii` does not touch bytes > 127 on upper/lower.

**ascii** or **latin1** can be used as locale-independent string replacement. (There is a compile switch to do this automatically for `ascii`).

**utf8** operates on UTF-8 sequences as of RFC 3629: 1 byte 0-7F, 2 byte 80-7FF, 3 byte 800-FFFF, 4 byte 1000-10FFFF (not excluding UTF-16 surrogate characters). Any byte not part of such a sequence is treated as it's (Latin-1) value.

**grapheme** takes care of grapheme clusters, which are characters followed by “grapheme extension” characters (`Mn+Me`) like combining diacritical marks.

Calls are:

```
len(str)
sub(str, start [,end=-1])
byte(str, start [,end=-1])
lower(str)
upper(str)
char(i [,j...])
reverse(str)
```

Same as in `string`: `rep`, `format`, `dump`

TODO: use char count with %s in `format`? (`sub` does the job)

TODO: `grapheme.byte`: only first code of any cluster?

- `find`, `gfind`, `gsub`: done, but need thorough testing ...: `ascii` does not match them on any `%class` (but on `.`, literals and ranges).
- Behaviour of `%class` with class not ASCII is undefined.

- Frontier %f currently disabled — should we?

## Character classes are:

```
%a L* (Lu+Ll+Lt+Lm+Lo)
%c Cc
%d 0-9
%l Ll
%n N* (Nd+Nl+No, new)
%p P* (Pc+Pd+Ps+Pe+Pi+Pf+Po)
%s Z* (Zs+Zl+Zp) plus the controls 9-13 (HT,LF,VT,FF,CR)
%u Lu (also Lt ?)
%w %a+%n+Pc (e.g. '_')
%x 0-9A-Za-z
%z the 0 byte
```

c.f.

[http://www.unicode.org/reports/tr44/tr44-6.html#Property\\_Values](http://www.unicode.org/reports/tr44/tr44-6.html#Property_Values)

<http://unicode.org/Public/UNIDATA/UnicodeData.txt>

## NOTE:

- find positions are in bytes for all ctypes!
- use ascii.sub to cut found ranges! This is a) faster, b) more reliable

## utf8 behaviour:

- match is by codes, code ranges are supported

## grapheme behaviour:

- any %class, ‘.’ and range match includes any following grapheme extensions.
- Ranges apply to single code points only.
- If a [ ] enumeration contains a grapheme cluster, this matches only the exact same cluster.
- However, a literal single ‘o’ standalone or in an [ ] enumeration will match just that ‘o’, even if it has an extension in the string. Consequently, grapheme match positions are not always cluster positions.

## **unicode.utf8.utf8valid**

---

```
result, len = unicode.utf8.utf8valid (str)
or
result, len = str:utf8valid ()
```

**Parameters:**

str: string

**Returns:**

result: boolean  
len: integer

**Description:**

This function tests whether the entire input string is valid UTF-8. If it is, the function returns *true* followed by the string length. Otherwise, the function returns *false* followed by the number of valid characters preceded the first invalid one.

**Note:**

This is added function, not present in the original **Selene Unicode**.

## **Extensions**

---

## file:rawhandle

---

```
rh = file:rawhandle()
```

**Parameters:**

none

**Returns:**

rh: light userdata

**Description:**

This is an additional method of a file handle returned by *io.open*. It returns the value of FILE\* that is suitable for using with LuAJ.

**Note:**

*file:rawhandle()* throws an error in non-GCC compilations.

This is to prevent a crash caused by incompatible runtime libraries.

# **LuaFAR C-interface**

---

# Mediator functions

---

Functions in this sections act as mediators between the plugin's exported functions and their counterparts written in Lua.

When FAR calls an exported function, the latter in turn calls a mediator passing it a Lua State pointer as the first argument, followed by all the arguments it received from FAR.

The mediator then calls a corresponding Lua function, passing it the arguments converted to their Lua equivalents, receives return value(s) from Lua, converts them to their C equivalents and returns them to its caller (the exported function).

## Example:

- 1) FAR calls the exported function [OpenW](#):

```
HANDLE OpenW(const struct OpenInfo *Info)
```

- 2) [OpenW](#) calls the mediator function [LF\\_Open](#):

```
HANDLE LF_Open (lua_State* L, const struct OpenInfo *Info)
```

- 3) [LF\\_Open](#) calls the Lua function [export.Open](#):

```
export.Open (aFrom, aGuid, aItem)
```

## **LF\_Analyse**

---

```
intptr_t LF_Analyse(lua_State* L, const struct AnalyseInfo *Info)
```

### **Parameters:**

L:       Lua state  
Info:     Data about the file to analyse

### **Returns:**

TRUE or FALSE

### **Description:**

If Lua function [export.Analyse](#) exists, it is called.

## **LF\_CloseAnalyse**

---

```
void LF_CloseAnalyse (lua_State* L, const struct CloseAnalyseInfo *I
```

### **Parameters:**

L:           Lua state  
....

### **Returns:**

nothing

### **Description:**

This function removes the value returned by [export.Analyse](#) from the Lua registry.

## **LF\_ClosePanel**

---

```
void LF_ClosePanel(lua_State* L, const struct ClosePanelInfo *Info)
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

nothing

### **Description:**

If Lua function [export.ClosePanel](#) exists, it is called. Then, the resources associated with the plugin are freed.

## **LF\_Compare**

---

```
intptr_t LF_Compare(lua_State* L, const struct CompareInfo *Info)
```

### **Parameters:**

L:           Lua state  
....

### **Returns:**

comparison result

### **Description:**

If Lua function [export.Compare](#) exists, it is called, else -2 is returned (directing to use the default FAR compare function).

## **LF\_Configure**

---

```
intptr_t LF_Configure(lua_State* L, const struct ConfigureInfo *Info
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

a boolean (success or failure)

### **Description:**

If Lua function [export.Configure](#) exists, it is called, else FALSE is returned.

## **LF\_DeleteFiles**

---

```
intptr_t LF_DeleteFiles(lua_State* L, const struct DeleteFileInfo *
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

a boolean (success or failure)

### **Description:**

If Lua function [export.DeleteFiles](#) exists, it is called, else FALSE is returned.

## **LF\_DlgProc**

---

```
intptr_t LF_DlgProc (lua_State *L, HANDLE hDlg, intptr_t Msg, intptr
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

This function is called only if the 9-th argument ([DlgProc](#)) passed to [far.DialogInit](#) was a function.

## **LF\_ExitFAR**

---

```
void LF_ExitFAR(lua_State* L, const struct ExitInfo *Info)
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

nothing

### **Description:**

First, the acquired resources are freed. Then, if Lua function [export.ExitFAR](#) exists, it is called.

## **LF\_FreeCustomData**

---

```
void LF_FreeCustomData (lua_State* L, wchar_t* CustomData)
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

nothing

### **Description:**

No Lua function is called. LuAFAR frees the “custom data” internally.

## **LF\_FreeFindData**

---

```
void LF_FreeFindData(lua_State* L, const struct FreeFindDataInfo *In
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

nothing

### **Description:**

No Lua function is called. LuaFAR frees the “find data” internally.

## **LF\_GetCustomData**

---

```
intptr_t LF_GetCustomData(  
    lua_State* L, const wchar_t *FilePath, wchar_t **CustomData)
```

### **Parameters:**

L:           Lua state  
....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.GetCustomData](#) exists, it is called, else 0 is returned.

## **LF\_GetFiles**

---

```
intptr_t LF_GetFiles (lua_State* L, struct GetFileInfo *Info)
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.GetFiles](#) exists, it is called, else 0 is returned.

## **LF\_GetFindData**

---

```
intptr_t LF_GetFindData(lua_State* L, struct GetFindDataInfo *Info)
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.GetFindData](#) exists, it is called, else FALSE is returned.

# **LF\_GetGlobalInfo**

---

```
int LF_GetGlobalInfo (lua_State* L, struct GlobalInfo *Info, const w
```

## **Parameters:**

L:           Lua state  
Info:        Pointer to a [GlobalInfo](#) struct  
PluginDir:   Plugin directory

## **Returns:**

TRUE on success, FALSE otherwise

## **Description:**

First, the function tries to require a module with the predefined name **<\_globalinfo**. If package.preload["<\_globalinfo"] contains a function, then it is run.

Else, the file **\_globalinfo.lua** is searched in the plugin's directory and run if found.

Then, if Lua function [export.GetGlobalInfo](#) exists, it is called.

## **LF\_GetOpenPanelInfo**

---

```
void LF_GetOpenPanelInfo(lua_State* L, struct OpenPanelInfo *aInfo)
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

nothing

### **Description:**

If Lua function [export.GetOpenPanelInfo](#) exists, it is called.

# **LF\_GetPluginInfo**

---

```
void LF_GetPluginInfo(lua_State* L, struct PluginInfo *PI)
```

## **Parameters:**

L:       Lua state  
PI:      Pointer to a [PluginInfo](#) struct

## **Returns:**

nothing

## **Description:**

If Lua function [export.GetPluginInfo](#) exists, it is called.

## **LF\_MakeDirectory**

---

```
intptr_t LF_MakeDirectory (lua_State* L, struct MakeDirectoryInfo *I
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.MakeDirectory](#) exists, it is called.

# **LF\_Open**

---

```
HANDLE LF_Open (lua_State* L, const struct OpenInfo *Info)
```

## **Parameters:**

L:           Lua state

....

## **Returns:**

Value, according to FAR Plugins API.

## **Description:**

If far.ReloadDefaultScript is true, then the function runs the default script (see [LF\\_RunDefaultScript](#)). Then, if Lua function [export.Open](#) exists, it is called. Else, NULL is returned to FAR.

## **LF\_ProcessConsoleInput**

---

```
intptr_t LF_ProcessConsoleInput(lua_State* L, struct ProcessConsoleI
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessConsoleInput](#) exists, it is called. Else, 0 is returned.

## **LF\_ProcessDialogEvent**

---

```
intptr_t LF_ProcessDialogEvent (lua_State* L, const struct ProcessDi
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessDialogEvent](#) exists, it is called. Else, 0 is returned.

## **LF\_ProcessEditorEvent**

---

```
intptr_t LF_ProcessEditorEvent (lua_State* L, const struct ProcessEd
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessEditorEvent](#) exists, it is called. Else, 0 is returned.

## **LF\_ProcessEditorInput**

---

```
intptr_t LF_ProcessEditorInput (lua_State* L, const struct ProcessEd
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessEditorInput](#) exists, it is called. Else, 0 is returned.

## **LF\_ProcessPanelEvent**

---

```
intptr_t LF_ProcessPanelEvent(lua_State* L, const struct ProcessPane
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessPanelEvent](#) exists, it is called. Else, 0 is returned.

## **LF\_ProcessHostFile**

---

```
intptr_t LF_ProcessHostFile(lua_State* L, const struct ProcessHostFi
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessHostFile](#) exists, it is called. Else, FALSE is returned.

## **LF\_ProcessPanelInput**

---

```
intptr_t LF_ProcessPanelInput(lua_State* L, const struct ProcessPane
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessPanelInput](#) exists, it is called. Else, FALSE is returned.

## **LF\_ProcessSynchroEvent**

---

```
intptr_t LF_ProcessSynchroEvent (lua_State* L, const struct ProcessS
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessSynchroEvent](#) exists, it is called. Else, 0 is returned.

## **LF\_ProcessViewerEvent**

---

```
intptr_t LF_ProcessViewerEvent (lua_State* L, const struct ProcessVi
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.ProcessViewerEvent](#) exists, it is called. Else, 0 is returned.

## **LF\_PutFiles**

---

```
intptr_t LF_PutFiles(lua_State* L, const struct PutFileInfo *Info)
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.PutFiles](#) exists, it is called. Else, 0 is returned.

## **LF\_SetDirectory**

---

```
intptr_t LF_SetDirectory(lua_State* L, const struct SetDirectoryInfo
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.SetDirectory](#) exists, it is called. Else, 0 is returned.

## **LF\_SetFindList**

---

```
intptr_t LF_SetFindList(lua_State* L, const struct SetFindListInfo *
```

### **Parameters:**

L:           Lua state

....

### **Returns:**

Value, according to FAR Plugins API.

### **Description:**

If Lua function [export.SetFindList](#) exists, it is called. Else, FALSE is returned.

## **Service functions**

---

## **LF\_DoFile**

---

```
int LF_DoFile (lua_State *L, const wchar_t *filename, int argc, wchar_t *argv[])
```

### **Description:**

Runs Lua script *filename* passing it *argc* string arguments from *argv*. The script receives the arguments in UTF-8 encoding. The function returns 0 if no error occurred. Otherwise, it prints error message on *stderr* and returns an error code.

This function can be used for testing some LuaFAR libraries (bit64, win, unicode) and functions (require, package.loadlib, etc.) from an application that is not Far plugin, for example:

```
#include "luafar.h"
int wmain (int argc, wchar_t* argv[])
{
    int status = 0;
    if (argc >= 2) {
        lua_State *L = lua_open();
        LF_InitLuaState1(L, NULL);
        status = LF_DoFile(L, argv[1], argc-2, argv+2);
        lua_close(L);
    }
    return status;
}
```

## **LF\_Gsub**

---

```
const wchar_t *LF_Gsub (lua_State *L, const wchar_t *s,
                        const wchar_t *p, const wchar_t *r)
```

### **Description:**

The function can be used as a replacement for `luaL_gsub`, working with wide char parameters, otherwise identical.

## **LF\_LoadFile**

---

```
int LF_LoadFile (lua_State *L, const wchar_t *filename)
```

### **Description:**

The function can be used as a replacement for `luaL_loadfile`, working with wide char *filename* parameter, otherwise identical.

# LF\_Message

---

```
int LF_Message(  
    PluginStartupInfo *Info,  
    const wchar_t *Msg,  
    const wchar_t *Title,  
    const wchar_t *Buttons,  
    const char *Flags,  
    const wchar_t *HelpTopic)
```

## Parameters:

Info: Pointer to a static struct that must be already initialized during SetStartupInfo call.

Msg: Message text; if multiline, then the lines must be separated by '\n'

Title: Message box title

Buttons: Button captions; if multiple, then the captions must be separated by ';'

Flags: A combination of the following characters:  
'w' stands for [FMSG WARNING](#)  
'e' stands for [FMSG ERRORTYPE](#)  
'k' stands for [FMSG KEEPBACKGROUND](#)  
'l' stands for [FMSG LEFTALIGN](#)  
'n' stands for "no wrapping of long lines"

HelpTopic: Help Topic (can be NULL)

## Returns:

-1 if escape pressed, else - button number chosen (0 based).

## Description:

This is an alternative interface to the FAR function `Message`.

## **LF\_InitLuaState1**

---

```
void LF_InitLuaState1 (lua_State* L, lua_CFunction OpenLibs)
```

### **Parameters:**

L:           Lua state.

OpenLibs:     Pointer to a function that can open additional  
libraries. (NULL is allowed.)

### **Description:**

This function is called by the plugin during [GetGlobalInfoW](#) call. It opens the standard Lua libraries, [win](#), [bit64](#) and [unicode](#) libraries and calls OpenLibs function.

## **LF\_InitLuaState2**

---

```
void LF_InitLuaState2 (lua_State *L, TPluginData *Data)
```

### **Parameters:**

L:      Lua state.

Data: Pointer to the struct containing specific plugin data.

### **Description:**

This function is called by the plugin during [SetStartupInfo](#) call. It stores pointer to the plugin's data within the Lua state, making them available to all LuaFAR functions. Then it loads [far](#) library.

## **LF\_RunLuafarInit**

---

```
void LF_LuafarInit (lua_State *L)
```

### **Parameters:**

L:      Lua state.

### **Description:**

This function executes the file %FARPROFILE%\luafar\_init.lua if such a file exists. It should be called **before** the first run of the plugin's default script.

# **LF\_RunDefaultScript**

---

```
BOOL LF_RunDefaultScript(lua_State* L)
```

## **Parameters:**

L:           Lua state

## **Returns:**

TRUE if successful, FALSE otherwise

## **Description:**

First, the function tries to require a module with the predefined name <boot>. If package.preload["<boot>"] contains a function, then it is run.

Else, the starting (or “default”) Lua script is searched in the plugin’s directory, in the following order:

1. The plugin name with the extension .lua, e.g.: luaplug.dll -> luaplug.lua
2. If the plugin name contains a hyphen, then its part preceding the last hyphen is looked for, e.g.: luaplug-x64.dll -> luaplug.lua.

This mechanism allows the default script to be either embedded into the application or external.

## **See also:**

[far.RunDefaultScript](#)

## **LF\_GetLuafarAPI**

---

```
void LF_GetLuafarAPI (LuafarAPI *target);
```

### **Parameters:**

target: pointer to a LuafarAPI struct.

The field 'StructSize' must be initialized by the caller.  
On function return, this field is set by the function.

### **Description:**

LF\_GetLuafarAPI allows to get many service and convenience functions of  
LuaFAR for use in the C-part of the plugin.

## **Third party software used**

---

The following third party software is used as part of LuaFAR:

1. [FAR 3.0 Plugin SDK](#)
2. [Lua 5.1](#)
3. [Selene Unicode](#)

Many thanks to the authors for making their great software available!

# **Lua 5.1**

---

Lua is licensed under the terms of the MIT license reproduced below. This means that Lua is free software and can be used for both academic and commercial purposes at absolutely no cost.

For details and rationale, see <http://www.lua.org/license.html>.

---

Copyright © 1994-2008 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

# Selene Unicode

---

```
/*
 Porting to Lua 5.2 and bug fixes: (C) Shmuel Zeigerman, 2010-2014.
 MIT license.
 */

/*
 *   Selene Unicode/UTF-8
 *   This additions
 *   Copyright (c) 2005 Malete Partner, Berlin, partner@malete.org
 *   Available under "Lua 5.0 license", see http://www.lua.org/licenses
 *   $Id: slnunico.c,v 1.5 2006/07/26 17:20:04 paul Exp $
 *
 *   contains code from
 ** lstrlib.c,v 1.109 2004/12/01 15:46:06 roberto Exp
 ** Standard library for string operations and pattern-matching
 ** See Copyright Notice in lua.h
 *
 *   uses the udata table and a couple of expressions from Tcl 8.4.x
 * which comes with the following license.terms:
```

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, ActiveState Corporation and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file to which they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

(end of Tcl license terms)  
\*/

/\*

According to <http://ietf.org/rfc/rfc3629.txt> we support up to 4-byte (21 bit) sequences encoding the UTF-16 reachable 0-0x10FFFF. Any byte not part of a 2-4 byte sequence in that range decodes to it. Ill formed (non-shortest) "C0 80" will be decoded as two code points not code point 0; see security considerations in the RFC. However, UTF-16 surrogates (D800-DFFF) are accepted.

See [http://www.unicode.org/reports/tr29/#Grapheme\\_Cluster\\_Boundaries](http://www.unicode.org/reports/tr29/#Grapheme_Cluster_Boundaries) for default grapheme clusters.

Lazy westerners we are (and lacking the Hangul\_Syllable\_Type data), we care for base char + Grapheme\_Extend, but not for Hangul syllable

For [http://unicode.org/Public/UNIDATA/UCD.html#Grapheme\\_Extend](http://unicode.org/Public/UNIDATA/UCD.html#Grapheme_Extend) we use Mn (NON\_SPACING\_MARK) + Me (ENCLOSING\_MARK), ignoring the 18 mostly south asian Other\_Grapheme\_Extend (16 Mc, 2 C  
<http://www.unicode.org/Public/UNIDATA/PropList.txt>  
\*/

/\* Contains code from:  
Quylthulg Copyright (C) 2009 Kein-Hong Man <keinhong@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software") in the Software without restriction, including without limitation the

to use, copy, modify, merge, publish, distribute, sublicense, and/or copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE SOFTWARE.

\*/