

ncyclopedia for Developers 1.75 SR2537

AR Manager - a console file manager for the Windows family of operating systems used on Win32 API). The program provides a comfortable user interface for working with e systems (real and emulated) and files.

AR Manager is a command-line shell unfolding the abilities of the native Win32 nmand prompt. FAR Manager has a modular structure and can be easily extended with a mber of <u>plugins</u> expanding its capabilities far beyond the simple console based command-e shell.

Generally a FAR plugin is a DLL exporting specific functions and interacting with FAR inager in a certain way. By creating a custom plugin you can add a new command, tor's function, or emulate a file system in the file panel (i.e. list archives, network ources or the registry). Some of the standard and important FAR manager functions are plemented as plugin modules, so you can never think of using FAR without using igins. (See the <u>Overview of plugin capabilities</u>).

Programming FAR plugins - Encyclopedia for Developers» is the most nprehensive reference guide to the plugin API for FAR Manager. It is meant for all these pple creating or making their first steps in creating FAR plug-ins. The encyclopedia tains authored API reference along with comments from 3-rd party professionals perienced in this area (See the <u>articles and notes</u>).

Ve hope that this Encyclopedia will provide a very useful source of information in the cess of writing FAR Manager plugins.

Copyright © PlugRinG, 1999-2009.

Introduction W About the proje How to setup th License Authors Structures Service functio Exported functi **Dialog** API Viewer API Panel API Editor API Far Standard Fi **Localization Custom API** Macros Addons Overview of pl Professional eth **API History** Examples Articles FAQ

bout the Programming FAR plugins ncyclopedia for Developers project

<u>iin | License | How to setup the Encyclopedia</u>

Project's goal - to create a "full-functioning-plentiful" help file in russian and english, for the C/C++ and Pascal programming languages. This encyclopedia will be mostly of interest to developers of FAR Manager plugins.

The Encyclopedia exists in two variants - HTMLHelp (pluginsr.chm/pluginse.chm) and OnlineHelp-version:

pluginsr.chm/pluginse.chm

Applicable in conjunction with MSDN (a variant with CHM/CHI-files, "MSDN Library - July 2001").

pluginsr online

Online-version of the encyclopedia. Always available for browsing at the following address: <u>http://api.farmanager.com/</u>

The Encyclopedia contains a wide variety of topics, besides the original topics on Plugins API, that (we hope so) can help plugin developers to write their own "creations".

Valentin Skirdin vskirdin@mail.ru

Project coordinator

icense

in | authors

- 1. All rights to the "**Programming FAR plugins Encyclopedia for Developers**" are **exclusively owned** by the **<u>authors</u>**.
- 2. The Encyclopedia, may be distributed only by a permission from the authors, provided the distribution package is not modified and no commercial or other interest of the authors is violated. No person or company may charge a fee for the distribution of the Encyclopedia without a written permission from <u>all the authors</u>.
- 3. The Encyclopedia exists only in Russian and English languages and is viewable in two forms HTMLHelp (FarEncyclopedia.ru.chm/FarEncyclopedia.en.chm) and as Online Help: <u>http://api.farmanager.com/</u>. Other sources are considered as violating the given license agreement.
- 4. THE ENCYCLOPEDIA IS DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE AT YOUR OWN RISK. THE AUTHORS WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS PRODUCT.
- 5. You may not use, copy, emulate, clone, rent, lease, sell, modify, decompile, disassemble, otherwise reverse engineer, or transfer the licensed product, or any subset of the licensed product, except as provided for in this agreement. Any such unauthorized use shall result in immediate and automatic termination of this license and may result in criminal and/or civil prosecution.
- 6. Installing and using the Encyclopedia signifies acceptance of these terms and conditions of the license.
- 7. If you do not agree with the terms of this license you must remove the Encyclopedia files from your storage devices and cease to use the product.
- 8. All rights not expressly granted here are reserved by the authors. This also

implies that any issues concerning the Encyclopedia will be resolved by a majority vote of the authors.

9. The authors reserve the right to change these terms and conditions from time to time at their sole discretion.

We thank you for using the encyclopedia: "**Programming FAR plugins - Encyclopedia for Developers**".

coordinator - Valentin Skirdin

Copyright © PlugRinG, 1999-2009.

/hat's new in Far Manager Encyclopedia 1.75 R2555?

<u>in</u>

October 2009

• Macro: <u>checkhotkey</u> function refined.

June 2009

• Process of going out of limits of "just API description" has begun... New file names FarEncyclopedia.en.chm and FarEncyclopedia.ru.chm.

April 2009

- Added flags <u>LIF_HIDDEN</u>, <u>LIF_GRAYED</u>, <u>MIF_HIDDEN</u>, <u>MIF_GRAYED</u>.
- <u>Color palette</u> now contains 10 more colors.

March 2009

• Macro: <u>Eval</u> function now has a second parameter, allowing just to check macro sequence for correctness. Also, function return codes are described.

March 2009

• Macro: <u>Menu.Select</u> function returns -1, if not called from the menu.

March 2009

• Macro: macros can be assigned to mouse buttons. These constants can be used in macro sequences: MsX, MsY, MsButton, MsCtrlState.

1 February 2009

• Macro: new function <u>atoi</u> - converts number from string representation into numeric.

3 January 2009

- Macro: additions and adjustments in <u>editor.sel</u> function it now works in the editor, dialog edit lines, and command line.
- Fixed errors in <u>ECTL_SELECT</u>:
 - block highlight was always cleared, even if parameters were incorrect
 - block highlight was always cleared, even if specified string (BlockStartLine) was absent (BlockStartLine was greater than number of strings in editor)
 - if BlockHeight was greater than number of strings in editor (i.e. try to highlight a block by specifying the last line, with BlockHeight equal to 10), EditorControl returned FALSE, though the block was highlighted.
 - when BlockWidth was equal to -1, the block highlight transaction for BTYPE_COLUMN was incoplete that could cause various problems with blocks.

3 January 2009

- Macro: <u>waitkey</u> function has a second optional parameter type of returned value
- Macro: new function <u>editor.sel</u> handles blocks in text.
- Macro: new function key converts the value V into string equivalent of the key name.
- Macro: Named keys (i.e. CtrlK) can be present in any expressions; in this case they are treated as numbers.

3 November 2008

• A new color was added to the palette - <u>COL_EDITORSCROLLBAR</u>.

November 2008

• Dialogs with FDLG_SMALLDIALOG flag are drawn with shadow now. To disable shadow drawing, set FDLG_NODRAWSHADOW flag.

1 September 2008

• New flags for <u>MkLink</u> function: FLINK_SYMLINKFILE, FLINK_SYMLINKDIR allow Windows Vista/2008 symbolic links creation. FLINK_SYMLINK flag renamed into FLINK_JUNCTION.

L August 2008

• User screen color is added to <u>color palette</u>.

3 August 2008

- Macro: CtrlBreak combination breaks macro execution.
- Macro: Added function <u>S=trim(S[,Mode])</u> removes all whitespace symbols.
- Viewer API: <u>VCTL_GETINFO</u> command returned WindowSizeY value that was less by 1 than the real value.

) June 2008

• Macro: Now <u>Dlg.ItemType</u> returns the same value as Dlg.GetValue(XXX,1).

7 June 2008

• Viewer API: added two events <u>VE_GOTFOCUS</u> and <u>VE_KILLFOCUS</u>.

5 May 2008

• New constants in <u>FarDialogSettings</u> enum: FDIS_DELREMOVESBLOCKS ("Del removes blocks in edit controls") and FDIS_MOUSECLICKOUTSIDECLOSESDIALOG ("Mouse click outside a dialog closes it").

April 2008

• Macro: added functions to handle stack bookmarks <u>BM.XXX()</u>.

) March 2008

 6 new EditorControl commands for navigation positions ("stack bookmarks") control in editor:
 <u>ECTL_ADDSTACKBOOKMARK</u>,
 <u>ECTL_CLEARSTACKBOOKMARKS</u> and
 <u>ECTL_DELETESTACKBOOKMARK</u> - creating and deleting navigation positions,
 <u>ECTL_GETSTACKBOOKMARKS</u> - receiving navigation positions information,
 <u>ECTL_NEXTSTACKBOOKMARK</u> and
 <u>ECTL_PREVSTACKBOOKMARK</u> - navigation in editor

3 March 2008

• new panel flag: <u>PFLAGS_PANELLEFT</u>.

7 March 2008

• Macro: added "Consts" macro area to store <u>named constants</u>.

5 March 2008

- Colors for long string markers in menus, lists and combo lists were added to <u>color palette</u>.
- Macro: Added "<u>Editor.RealPos</u>" constant current cursor position in the string in editor (tab size independent).
- Macro: Some functions can have optional parameters.
- Macro: New <u>prompt</u> function allows to input one text string.

L March 2008

- •
- Macro: New <u>V=akey(N)</u> function returns name or code of the key that initiated the macro.

L December 2007

• API: New command <u>ACTL_REDRAWALL</u> - redraw all FAR windows.

3 December 2007

- Plugins menu can be opened from a dialog.
- Plugin can export <u>ProcessDialogEvent</u> function to process dialog events.

1 December 2007

• Macro: Added function <u>Result=replace(Str,Find,Replace,Cnt</u>).

- API: New <u>MCMD_GETSTATE</u> command for <u>ACTL_KEYMACRO</u>, returns macro engine work status.
- Macro: <u>Menu.Select</u> function now has two parameters.

December 2007

- 2 new events for panel plugins: <u>FE_GOTFOCUS</u> and <u>FE_KILLFOCUS</u>.
- 2 new events for editor plugins: <u>EE_GOTFOCUS</u> and <u>EE_KILLFOCUS</u>.
- In dialogs, immediately after <u>DN_INITDIALOG</u>, <u>DN_GOTFOCUS</u> is fired for the element with Focus=1.

December 2007

- Macro: Added function <u>n=mod(n1,n2)</u>.
- Macro: Added function <u>N=Menu.Select(S)</u>. Selects the first item that contains S.
- New flag <u>DIF_NOAUTOCOMPLETE</u> for edit strings disables autocomplete.

2 October 2007

- Macro: Added constant "<u>MacroArea</u>" returns the name of current Macro area.
- Macro: Bof/Eof/Empty/Selected in panels like QView/Info/Tree return values for these types of panels. For regular panels, values are returned for command line.

3 September 2007

 Now when DI_COMBOBOX is open, DN_KEY or DN_MOUSEEVENT events are sent to the dialog procedure. This behavior can be controlled using events <u>DM_SETCOMBOBOXEVENT</u> and <u>DM_GETCOMBOBOXEVENT</u>.

) August 2007

- Bug: Macro: If menu item does not contain hot key, <u>GetHotkey()</u> returns "0" instead of "".
- Macro: Added command <u>\$SelWord</u> selects a word.
- Bug: Macro: Some variables did not work in QuickView and Infopanel: ItemCount, CurPos, Selected, Bof, Eof were always equal to 0; Empty was always equal to 1.

August 2007

- Bug: If a plug-in disabled the mouse cursor tracking reaction (<u>LMRT_NEVER</u>) for a list, the dialog procedure did not get DN_MOUSECLICK event for mouse double click (Mantis#0000309).
- In the <u>PluginPanelItem</u> structure, <u>FAR_FIND_DATA</u> structure was used instead of <u>WIN32_FIND_DATA</u>. <u>FAR_USE_WIN32_FIND_DATA</u> macro should be used instead of _FAR_USE_FARFINDDATA.

3 July 2007

 Added commands <u>DM_GETEDITPOSITION</u> and <u>DM_SETEDITPOSITION</u> - line position control within edit strings and dialogs.

July 2007

• Macro: Added function <u>gethotkey</u>.

- Macro: Variables beginning with a number (such as %3DO) were not processed.
- Bug: <u>DM_LISTGETCURPOS</u> retrned wrong value after opening a ComboBox, moving through the list and cancelling (pressing Esc).

L May 2007

• Macro: Added function <u>Panel.SetPosIdx</u> - File panel positioning using index.

) May 2007

• Macro: new functions <u>asc()</u> and <u>chr</u>.

3 March 2007

- Macros in the Registry can have REG_MULTI_SZ type.
- Added flag <u>KSFLAGS_REG_MULTI_SZ</u>.

1 March 2007

• After calling <u>EditorControl(ECTL_SELECT</u>) with <u>BlockStartLine</u> = -1, next call to <u>ECTL_GETINFO</u> also returned -1 in <u>BlockStartLine</u>.

3 March 2007

• Macro: new functions <u>waitkey()</u> and <u>eval</u>.

) February 2007

• Added flag <u>DIF_NOTCVTUSERCONTROL</u> for <u>DI_USERCONTROL</u>.

February 2007

• Macro: new words <u>APanel.ColumnCount</u> and <u>PPanel.ColumnCount</u>.

) January 2007

• New command: <u>FCTL_GETUSERSCREEN</u>.

5 January 2007

- For <u>DI_VTEXT</u> element, added handling of flags: DIF_CENTERGROUP, DIF_SEPARATOR, DIF_SEPARATOR2, DIF_SHOWAMPERSAND.
- In <u>DefDlgProc</u> handler, <u>DIF_BTNNOCLOSE</u> flag was not handled for <u>DN_BTNCLICK</u> event.
- Incorrect <u>DI_TEXT</u> element drawing for conditions: DIF_CENTERTEXT + DIF_SEPARATOR + X1 not equal to "-1"

1 January 2007

• Rules regarding X2 and Y2 coordinates in the dialog elements are hardened. Y2 should be always set, do not use 0.

3 January 2007

- <u>EE_CLOSE</u> event was not fired for "?New File?".
- Incorrect color drawing for information dialog when FMSG_WARNING flag was used in a <u>Message</u> function without buttons.

7 December 2006

- Macro: New states <u>Help.FileName</u>, <u>Help.Topic</u> and <u>Help.SelTopic</u>.
- The command <u>ACTL_GETWINDOWINFO</u> now fills the Name field for help windows contains the full path to the HLF file.

L December 2006

- Viewer API: Structure member <u>ViewerMode</u>.TypeWrap renamed to ViewerMode.WordWrap
- Viewer API: Added command <u>VCTL_SETMODE</u>.

3 November 2006

• Macro: <u>\$AKey</u>.

L November 2006

- Bringing the API to 64bit compatibility:
 - DlgProc: long Param2 -> LONG_PTR Param2 and returns LONG_PTR instead of long
 - SendDlgMessage: long Param2 -> LONG_PTR Param2 and returns LONG_PTR instead of long
 - DefDlgProc: long Param2 -> LONG_PTR Param2 and returns LONG_PTR instead of long
 - DialogEx: long Param -> LONG_PTR Param
 - FarMenuItemEx: DWORD UserData -> DWORD_PTR UserData
 - PluginPanelItem: DWORD UserData -> DWORD_PTR UserData
 - AdvControl: returns INT_PTR instead of int
 - OpenPlugin: int Item -> INT_PTR Item

) September 2006

- Macro: <u>fexist()</u> understands file mask symbols '*' and '?'.
- Macro: If the name contains '*' or '?', then <u>fattr()</u> will return the attributes of the first found file.
- Macro: fexist() and fattr() do not longer work with the panels, new functions were added to work with the panels: panel.fexist() and panel.fattr().

5 August 2006

• Corrected the description of the <u>DN_BTNCLICK</u> event for <u>DI_RADIOBUTTON</u>.

) July 2006

• New function: <u>FSF.snprintf</u>.

) June 2006

- Macro function "<u>N=sleep(N)</u>".
- Macro constant "<u>Far.Height</u>".
- Macro constant "<u>Far.Title</u>".
- Macro constant "<u>Title</u>".
- Macro constant "<u>Drv.ShowPos</u>".
- Macro constant "Drv.ShowMode".

Long history, API history

xported functions - Common functions

Function	Description
<u>Configure</u>	plugin configuration
ExitFAR	before closing the FAR Manager
<u>GetMinFarVersion</u>	get mininum FAR Manager version
<u>GetPluginInfo</u>	get plugin information
<u>SetStartupInfo</u>	global settings

e also:

Service functions, Structures, Archive support, Addons

onfigure

in | exported functions

The **Configure** function allows the user to configure the plugin module. It is called when one of the items exported by this plugin to the "Plugin configuration" menu is selected.

```
int WINAPI Configure(
    int ItemNumber
);
```

arameters

ItemNumber

The number of selected item in the list of items exported by this plugin to the "Plugin configuration" menu.

eturn value

If the function succeeds, the return value must be TRUE - in this case FAR updates the panels. If the configuration is canceled by user, FALSE should be returned.

emarks

If your plugin exports only one menu item then you can ignore *ItemNumber*.

xample

```
int WINAPI _export Configure(int ItemNumber)
{
    switch(ItemNumber)
    {
        case 0:
            return(Config());
    }
    return(FALSE);
}
```

e also: <u>GetPluginInfo</u>

xitFAR

in | exported functions

The **ExitFAR** function is called before FAR exits. In this function plugins can release all used resources.

void WINAPI ExitFAR(void);

arameters

None.

eturn value

None.

e also: <u>ClosePlugin</u>

etMinFarVersion

in | exported functions

The **GetMinFarVersion** is called to get the minimum FAR version required for the plugin to work correctly.

int WINAPI GetMinFarVersion(void);

arameters

None.

eturn value

This function must return an integer in the form 0xZZZXXYY:

Component	Description
ZZZZ	build number (FAR 1.70.387 = 0x0183)
XX	major version (FAR 1.70 = 0x01)
ҮҮ	minor version (FAR $1.70 = 0x46$)

It is **recomended** to use the macro <u>MAKEFARVERSION</u>.

emarks

If the required version is greater than the current FAR version, an error message is displayed and the plugin is then unloaded.

xample

```
// For a plugin that requires FAR Manager 1.70 build
int WINAPI _export GetMinFarVersion(void)
{
   return MAKEFARVERSION(1,70,591);
}
// And this plugin will work in FAR Manager version 1
int WINAPI _export GetMinFarVersion(void)
{
   return MAKEFARVERSION(1,70,0);
}
```

e also: MAKEFARVERSION, FARMANAGERVERSION

etPluginInfo

in | exported functions

The **GetPluginInfo** function is called to get general plugin information.

```
void WINAPI GetPluginInfo(
   struct PluginInfo *Info
);
```

arameters

Info

Points to a <u>PluginInfo</u> structure that should be filled by this function. The plugin must fill this structure.

eturn value

None.

emarks

- 1. This function is called before the following actions:
 - before the plugins configuration menu is shown
 - before the plugin commands menu (F11) is shown
 - before the disks menu (Alt-F1/Alt-F2) is shown
 - when a command with a prefix is entered in the command line (for example, net:\\share).
- 2. Because this function gets called frequently, time consuming operations should not be done within it.
- 3. The <u>PluginInfo</u> structure passed to this function is already filled with zeros. The plugin is required to fill the StructSize field of the structure.
- 4. FAR calls GetPluginInfo function rather often, therefore it is necessary to minimize "computations" performed by this function when developing a plugin.

xample

```
void WINAPI _export GetPluginInfo(struct PluginInfo *
{
    Info->StructSize=sizeof(struct PluginInfo);
    Info->Flags=PF_EDITOR|PF_DISABLEPANELS;
```

```
static char *PluginMenuStrings[1];
PluginMenuStrings[0]=GetMsg(MBrackets);
Info->PluginMenuStrings=PluginMenuStrings;
Info->PluginMenuStringsNumber=sizeof(PluginMenuStri
sizeof(PluginMenuStrings[0]);
```

}

e also: <u>PluginInfo</u>

etStartupInfo

in | exported functions

The **SetStartupInfo** function is called once, after the DLL module is loaded to memory. This function gives the plugin information necessary for further operation.

```
void WINAPI SetStartupInfo(
   const struct PluginStartupInfo *Info
);
```

arameters

Info

Points to a <u>PluginStartupInfo</u> structure.

eturn value

None.

emarks

- 1. In FAR Manager 1.65 or older this function is called first just after the DLL module is loaded.
- 2. In FAR Manager 1.70 or later this function is called after <u>GetMinFarVersion</u>.
- 3. The **Info** pointer is valid only until return, so the structure must be copied to an internal variable for further usage:

```
static struct PluginStartupInfo Info;
...
void WINAPI _export SetStartupInfo(const struct P:
{
    ::Info=*Info;
    ...
}
```

4. If the plugin uses "standard functions" from the <u>FarStandardFunctions</u> structure then the <u>PluginStartupInfo.FSF</u> member must be copied to an internal variable for further usage:

```
static struct PluginStartupInfo Info;
```

```
static struct FarStandardFunctions FSF;
void _export SetStartupInfo(struct PluginStartup:
{
    Info=*psInfo;
    FSF=*psInfo->FSF;
    Info.FSF=&FSF; // adjust the address in the lo
    ....
}
```

xported functions - Panel specific functions

in | exported functions

Attention! All file names passed to FAR must be in OEM code page. FAR also passes file names in OEM code page. Before calling plugin functions FAR calls **SetFileApisToOEM**. If plugin uses anywhere SetFileApisToANSI, it must call SetFileApisToOEM again before returning control to FAR.

Function	Description
<u>ClosePlugin</u>	before closing an open plugin instance.
Compare	overrides sorting algorithm
<u>DeleteFiles</u>	delete files
<u>FreeFindData</u>	frees memory, allocated by GetFindData
<u>FreeVirtualFindData</u>	frees memory, allocated by GetVirtualFindData
<u>GetFiles</u>	get files
<u>GetFindData</u>	get file list
<u>GetOpenPluginInfo</u>	get information about an open plugin instance
<u>GetVirtualFindData</u>	get files
<u>MakeDirectory</u>	make a directory
<u>OpenFilePlugin</u>	open a file
<u>OpenPlugin</u>	create a new plugin instance
ProcessEvent	process events
ProcessHostFile	execute archive commands
<u>ProcessKey</u>	process keyboard events
PutFiles	put files to the emulated file system
<u>SetDirectory</u>	set current directory in the emulated file system
<u>SetFindList</u>	transfers found files from the "Find file" dialog to the emulated file system

e also:

Service functions, Structures, Archive support, Addons

losePlugin

in | exported functions

The **ClosePlugin** function closes an open plugin instance.

```
void WINAPI ClosePlugin(
   HANDLE hPlugin
);
```

arameters

hPlugin Open plugin handle

eturn value

None

e also:

<u>ExitFar</u>

ompare

```
in | exported functions
```

A plugin can export the function **Compare** to override the default file panel sorting algorithm.

```
int WINAPI Compare(
   HANDLE hPlugin,
   const struct PluginPanelItem *Item1,
   const struct PluginPanelItem *Item2,
   unsigned int Mode
);
```

arameters

hPlugin

Plugin handle, returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

Item1, Item2

Pointers to **PluginPanelItem** structures to compare.

Mode

See Sort modes

eturn value

This function returns an **int** value that is:

- **-1** if *Item1* **<** *Item2*
- **0** if *Item1* **==** *Item2*
- 1 if Item1 > Item2
- 2 if the default FAR compare function should be used for this sort mode.

emarks

The standard RTL **qsort** function that implements and unstable sorting algorithm is used by FAR for sorting needs. In other words if array elements are equal to the compare function then on the panels they will be shown in random order, that changes upon each redrawing of the panel.

eleteFiles

in | exported functions

The **DeleteFiles** function is called to delete files in the file system emulated by the plugin.

(FAR to plugin: "this file(s) from your panel need to be deleted").

```
int WINAPI DeleteFiles(
   HANDLE hPlugin,
   struct PluginPanelItem *PanelItem,
   int ItemsNumber,
   int OpMode
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

PanelItem

Points to an array of <u>PluginPanelItem</u> structures. Each structure describes a file to delete.

ItemsNumber

Number of elements in the *PanelItem* array.

OpMode

Combination of the <u>operation mode</u> flags. This function should process the flag <u>OPM_SILENT</u>.

eturn value

If the function succeeds, the return value must be TRUE. If the function fails, FALSE should be returned.

emarks

- 1. The plugin should process "its folders" by itself.
- 2. It is considered a good form of behaviour to inquire the user about his/her intensions (see also <u>ACTL_GETCONFIRMATIONS</u>, flags <u>FCS_DELETE</u> and <u>FCS_DELETENONEMPTYFOLDERS</u>)

xample

```
int WINAPI DeleteFiles(HANDLE hPlugin, PluginPanelIte
           int ItemsNumber, int OpMode)
{
  struct PluginPanelItem * curPI=&PanelItem[0];
  for(int CurItem=0;
      CurItem<ItemsNumber;CurItem++,</pre>
      curPI++)
  {
    char* aFile=curPI->FindData.cFileName;
    if(!RemoveFileFromFS(aFile))
    {
      char *MsgItems[]={"Delete failed","","OK"};
      MsgItems[1] = GetErrorStringFS();
      Message(MyNumber, 0, NULL, MsgItems,
               sizeof(MsgItems)/sizeof(MsgItems[0]),1)
      return(FALSE);
    }
  }
  return(TRUE);
}
```

reeFindData

in | exported functions

The **FreeFindData** function is called to release the data allocated by <u>GetFindData</u> (FAR to plugin: "the list I requested, well, I no longer need it, free the memory").

```
void WINAPI FreeFindData(
   HANDLE hPlugin,
   struct PluginPanelItem *PanelItem,
   int ItemsNumber
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

PanelItem

Points to an array of <u>PluginPanelItem</u> structures previously allocated by <u>GetFindData</u>.

ItemsNumber

Number of elements in the *PanelItem* array.

eturn value

None.

e also: <u>GetFindData</u>

reeVirtualFindData

in | exported functions

The **FreeVirtualFindData** function is called to release the data allocated by <u>GetVirtualFindData</u>.

```
void WINAPI FreeVirtualFindData(
    HANDLE hPlugin,
    struct PluginPanelItem *PanelItem,
    int ItemsNumber
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

PanelItem

Points to an array of <u>PluginPanelItem</u> structures previously allocated by <u>GetVirtualFindData</u>.

ItemsNumber

Number of elements in the *PanelItem* array.

eturn value

None.

e also:

GetVirualFindData

etFiles

in | exported functions

The **GetFiles** function is called to get files from the file system emulated by the plugin.

(FAR to plugin: "I want those files from your panel, destination is specified").

```
int WINAPI GetFiles(
   HANDLE hPlugin,
   struct PluginPanelItem *PanelItem,
   int ItemsNumber,
   int Move,
   char *DestPath,
   int OpMode
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

PanelItem

Points to an array of <u>PluginPanelItem</u> structures. Each structure describes a file to get.

ItemsNumber

Number of elements in the *PanelItem* array.

Move

If zero, files should be copied, if nonzero - moved.

DestPath

Destination path to put files. If *OpMode* flag **OPM_SILENT** is not set, you can allow the user to change it, but in that case the new path must be copied to *DestPath*.

OpMode

Combination of the <u>operation mode</u> flags. This function should be ready to process OPM_SILENT, OPM_FIND, OPM_VIEW, OPM_QUICKVIEW and OPM_EDIT flags. Also it can process OPM_DESCR and OPM_TOPLEVEL to speed up operation if necessary.

eturn value

If the function succeeds, the return value must be 1. If the function fails, 0 should be returned. If the function was interrupted by the user, it should return -1.

emarks

- 1. The plugin should process "its folders" by itself.
- 2. If the operation has failed, but part of the files was successfully processed, the plugin can remove selection only from the processed files. To perform it, plugin should clear the <u>PPIF_SELECTED</u> flag for processed items in the *PluginPanelItem* list passed to function.
- 3. This function is called only for plugins that implement virtual file systems. For this it is necessary to remove the <u>OPIF_REALNAMES</u> flag when <u>GetOpenPluginInfo</u> is called, otherwise this function will never be called.

e also:

PutFiles, GetDirList, GetPluginDirList

etFindData

in | exported functions

The **GetFindData** function is called to get the list of files in the current directory of the file system emulated by the plugin.

(FAR to plugin: "let me look at your file list, allocate the memory yourself :-)").

```
int WINAPI GetFindData(
   HANDLE hPlugin,
   struct PluginPanelItem **pPanelItem,
   int *pItemsNumber,
   int OpMode
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

pPanelItem

Points to a variable that receives the address of a <u>PluginPanelItem</u> structures array.

pItemsNumber

Points to a variable that receives the number of **PluginPanelItem** structures.

OpMode

Combination of the <u>operation mode</u> flags. This function should be ready to process the <u>OPM_FIND</u> flag.

eturn value

If the function succeeds, the return value must be TRUE. If the function fails, FALSE should be returned.

emarks

- 1. The memory for the requested data should be allocated by the plugin. Because of that it is important to export the <u>FreeFindData</u> function so the allocated memory will be freed when no longer needed.
- 2. If this function returns FALSE, the plugin will be closed.
- 3. In some cases (for example when searching in archives Alt-F7) a panel in

not created physically, so it necessary to check the return value of the <u>Control</u> function, to prevent a crash in the most upappropriate moment while carying out work for a none existant panel.

4. If you want to prevent your plugin from participating in the "search in archive" ("[x] Search in archives" in the Find file dialog) then return FALSE when *OpMode* contains the OPM_FIND flag.

e also: <u>FreeFindData</u>
etOpenPluginInfo

in | exported functions

The **GetOpenPluginInfo** function is called to get the information about an open plugin instance.

```
void WINAPI GetOpenPluginInfo(
   HANDLE hPlugin,
   struct OpenPluginInfo *Info
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

Info

Points to an <u>OpenPluginInfo</u> structure that should be filled by this function.

eturn value

None.

emarks

- 1. The <u>OpenPluginInfo</u> structure passed to this function is already filled with zeroes. The plugin is required to fill the OpenPluginInfo.StructSize field.
- 2. FAR calls GetOpenPluginInfo function rather often, therefore it is necessary to minimize "computations" performed by this function when developing a plugin.

e also: <u>OpenPluginInfo</u>

etVirtualFindData

in | exported functions

The **GetVirtualFindData** function can be used to return a list of files to show in another file panel in addition to the real files.

```
int WINAPI GetVirtualFindData(
    HANDLE hPlugin,
    struct PluginPanelItem **pPanelItem,
    int *pItemsNumber,
    const char *Path
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

pPanelItem

Points to a variable that receives the address of an array of <u>PluginPanelItem</u> structures.

pItemsNumber

Points to a variable that receives the number of **PluginPanelItem** structures.

Path

Path for which the list of files is returned (the current directory on another panel). The path is terminated with a backslash.

eturn value

If the function succeeds, the return value must be TRUE. If the function fails, FALSE should be returned.

emarks

This function can be used to implement "delayed file copying". When delayed copying is used, the files copied from a plugin panel to a file panel are shown on the file panel immediately, but the physical copy operation is performed later, when the plugin is closed or a special command is executed. Delayed copying can be useful, for example, for plugins supporting Arvid.

e also: <u>FreeVirualFindData</u>

lakeDirectory

in | exported functions

The **MakeDirectory** function is called to create a new directory in the file system emulated by the plugin.

```
int WINAPI MakeDirectory (
   HANDLE hPlugin,
   char *Name,
   int OpMode
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

Name

Name of the directory. If *OpMode* flag <u>OPM_SILENT</u> is not set, you can allow the user to change it, but in that case the new name must be copied to **Name** (max. <u>NM</u> bytes).

OpMode

Combination of the <u>operation mode</u> flags. This function should be ready to process **OPM_SILENT** flag.

eturn value

If the function succeeds, the return value must be 1. If the function fails, 0 should be returned. If the function was interrupted by the user, it should return -1.

e also <u>OpenPlugin, OpenFilePlugin</u>

penFilePlugin

in | exported functions

The **OpenFilePlugin** function is called to open a plugin which emulates a file system based on a file passed to this function (an archive for example).

```
HANDLE WINAPI OpenFilePlugin(
    char *Name,
    const unsigned char *Data,
    int DataSize
);
```

arameters

Name

Points to the full name of the file (including the path). This pointer is valid only until return, so if the plugin will process this file, it should copy this name to an internal variable.

The **OpenFilePlugin** function is also called when the user is going to create a new file (when Shift-F1 is pressed). In that case Name is NULL and other parameters are undefined. If a plugin does not support creating new files, it must return INVALID_HANDLE_VALUE, otherwise it must return the handle of a new plugin instance that must be ready to process <u>GetOpenPluginInfo</u> and <u>PutFiles</u> functions. If *Name* is NULL, the plugin needs to request *Name* from the user in the <u>PutFiles</u> function.

Data

Points to data from the beginning of the file. It can be used to detect file type. The plugin must not change this data.

DataSize

Size of the passed file data. Currently it can be from 0 to 128Kb, depending on file size, but you should be ready to process any other value.

eturn value

- If the plugin will process the passed file, the return value must be new plugin handle.
- If this file type is not supported, the return value must be INVALID_HANDLE_VALUE.

• If operation is interrupted by the user, the value -2 (cast to the **HANDLE** type) should be returned.

emarks

- When <Enter> is pressed on a selected file, FAR queries all plugins that export this function. The plugins are queried in alphabetic order (sorted by the DLL name). When a plugin returns a value different from INVALID_HANDLE_VALUE, FAR stops querying other plugins.
- 2. The size of data read from the file can be configured TechInfo #63:

```
[HKEY_CURRENT_USER\Software\Far\System]
"PluginMaxReadData":REG_DWORD
The key "System/PluginMaxReadData" of DWORD type a
maximum size of the data read from a file after an
it from the panels (Enter or Ctrl-PgDn) was made.
be passed to plugins to determine which plugin sup
type.
Can be any value in the range of 0x1000 to 0x8000(
The default value is 0x20000.
```

penPlugin

in | exported functions

The **OpenPlugin** is called to create a new plugin instance.

```
HANDLE WINAPI OpenPlugin(
    int OpenFrom,
    INT_PTR Item
);
```

arameters

OpenFrom

Identifies how the plugin is invoked. Can be one of the following values (OPENPLUGIN_OPENFROM enum):

Constant	Description	
OPEN_DISKMENU	Opened from the disks menu	
OPEN_PLUGINSMENU	Opened from the plugins menu (F11)	
OPEN_FINDLIST	Opened from the "Find File" dialog. The plugin will be called with this identifier only if it exports the <u>SetFindList</u> function, and <u>SetFindList</u> will be called only if OpenPlugin returns a valid handle.	
OPEN_SHORTCUT	Opened using a folder shortcut command.	
OPEN_COMMANDLINE	Opened from the command line This type is used if the plugin has defined a command prefix in the <u>GetPluginInfo</u> function, and this prefix, followed by a colon, is found in the command line.	
OPEN_EDITOR	Opened from internal editor	
OPEN_VIEWER	Opened from internal viewer.	
OPEN_DIALOG	Opened from dialog	

Item

Its meaning depends on the value of **OpenFrom**:

• For OPEN_DISKMENU, OPEN_PLUGINSMENU, OPEN_EDITOR and OPEN_VIEWER *Item* is a position of the activated plugin item in the exported items list in disks or plugins menu. If a plugin exports only one item, this field is always zero.

- For OPEN_FINDLIST *Item* is always zero.
- For OPEN_SHORTCUT *Item* contains the address of a string that was passed in the ShortcutData member of the <u>OpenPluginInfo</u> structure, when saving the shortcut. The plugin can use it to store any additional information about its current state. It is not necessary to save the information about the current directory, because it is restored by FAR when using folder shortcuts.
- For OPEN_DIALOG *Item* contains adress of an <u>OpenDlgPluginData</u> structure.
- For OPEN_COMMANDLINE *Item* contains address of a string containing the command line entered by the user. Plugin command prefix is not included in this string, unless the <u>PF_FULLCMDLINE</u> flag is set. For example, if a plugin defined the prefix *ftp* and the user entered ftp://ftp.abc.com, *Item* will point to //ftp.abc.com. However, if PF_FULLCMDLINE is set, *Item* will point to ftp://ftp.abc.com.

eturn value

If the function succeeds, the return value is a plugin handle. This handle will be passed later to other plugin functions to allow them to distinguish different plugin instances. Handle format is not important for FAR, it can be the address of a new plugin class object, or the address of a structure with plugin data, or an array index, or any other value but zero.

If the function fails, the return value must be INVALID_HANDLE_VALUE.

emarks

- 1. Note that you can use this function to implement FAR commands that work without creating new panels. Just perform all necessary actions here and return INVALID_HANDLE_VALUE.
- 2. If this functions returns zero, the plugin will be unloaded.

rocessEvent

in | exported functions

The **ProcessEvent** function informs plugin about different FAR events and allows to process some of them.

```
int WINAPI ProcessEvent(
   HANDLE hPlugin,
   int Event,
   void *Param
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

Event

Event type. Can be one of the following values (FAR_EVENTS enum):

Event	Description
FE_CHANGEVIEWMODE	Panel view mode is changed. <i>Param</i> points to a null-terminated string specifying <u>column types</u> , for example N,S,D,T . Return value must be FALSE.
FE_REDRAW	The panel is about to redraw. <i>Param</i> is equal to NULL. Return FALSE to use the FAR redraw routine or TRUE to disable it. In the latter case the plugin must redraw the panel itself.
FE_IDLE	Sent every few seconds. A plugin can use this event to request panel updating and redrawing, if necessary. <i>Param</i> is equal to NULL. Return value must be FALSE.
FE_CLOSE	The panel is about to close. <i>Param</i> is equal to NULL. Return FALSE to close the panel or TRUE to cancel it.
FE_BREAK	Ctrl-Break is pressed. <i>Param</i> currently can be only (int)CTRL_BREAK_EVENT Return value must be FALSE.

	Processing of this event is performed in separate thread, so the plugin must be careful when performing console input or output and must not use FAR service functions.
FE_COMMAND	About to execute a command from the FAR command line. <i>Param</i> points to the command text. he plugin should return FALSE to allow standard command execution or TRUE if it is going to process the command internally.
FE_KILLFOCUS	Panel has lost keyboard focus. <i>Param</i> = NULL. Return value must be FALSE.
FE_GOTFOCUS	Panel received keyboard focus. The active panel receives the FE_GOTFOCUS event immediately after its creation. <i>Param</i> = NULL. Return value must be FALSE.

Param

Points to data dependent on event type. Read events description for concrete information.

eturn value

Return value depends on event type. Read events description for concrete information.

Return FALSE for unknown event types

rocessHostFile

in | exported functions | archive support

The **ProcessHostFile** function is called to perform FAR <u>archive commands</u>. It is recommended to use this function to perform additional operations on the file that is handled by a file processing plugin.

```
int WINAPI ProcessHostFile(
   HANDLE hPlugin,
   struct PluginPanelItem *PanelItem,
   int ItemsNumber,
   int OpMode
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

PanelItem

Points to an array of <u>PluginPanelItem</u> structures. Each structure corresponds to a selected file in the plugin panel.

ItemsNumber

Number of elements in the *PanelItem* array.

OpMode

Combination of the <u>operation mode</u> flags. For this function it is either 0 or OPM_TOPLEVEL.

eturn value

If the function succeeds, the return value must be TRUE. If the function fails, FALSE should be returned.

emarks

If the operation failed, but part of files was successfully processed, plugin can remove selection only from the processed files. To perform it, the plugin should clear <u>PPIF_SELECTED</u> flag in processed items in the PluginPanelItem list passed to the function.

rocessKey

in | exported functions

The **ProcessKey** function allows to override standard control keys processing in a plugin panel.

```
int WINAPI ProcessKey(
   HANDLE hPlugin,
   int Key,
   unsigned int ControlState
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

Key

Virtual key code.

Key may have the <u>PKF_PREPROCESS</u> flag set.

ControlState

Indicates control keys state. One or more flags from the <u>FAR_PKF_FLAGS</u> enum (except <u>PKF_PREPROCESS</u>).

For example, when Shift-F7 is pressed, *Key* is equal to VK_F7 and *ControlState* is equal to PKF_SHIFT.

eturn value

Return FALSE to use standard FAR key processing. If the plugin processes the key combination by itself, it should return TRUE

emarks

• FAR 1.70 build 2051 and earlier versions:

Because of FAR kernel implementation specifics this function does not receive the following keys: Tab, Ctrl-F1, Ctrl-F2, Ctrl-B, Cltr-L, Ctrl-Q, Ctrl-T, Ctrl-O, Ctrl-P, Ctrl-I, Ctrl-U, Alt-F1, Alt-F2, Alt-F7, Ctrl-Down, Ctrl-Up, Ctrl-Left, Ctrl-Right, Ctrl-Num5, F9, Shift-F10, Ctrl-0..9, Alt-Ins, Ctrl-W, F11, Alt-F9, F12, Ctrl-Tab and Ctrl-Shift-Tab.

Since 1.70 build 2052 those restrictions no longer apply (refer to the remarks on the <u>PKF_PREPROCESS</u> flag)

utFiles

in | exported functions

The **PutFiles** function is called to put files to the file system emulated by the plugin. (FAR to plugin: *"those files are for you, you should place then on your panel"*).

```
int WINAPI PutFiles(
   HANDLE hPlugin,
   struct PluginPanelItem *PanelItem,
   int ItemsNumber,
   int Move,
   int OpMode
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

PanelItem

Points to an array of <u>PluginPanelItem</u> structures. Each structure describes a file to put.

ItemsNumber

Number of elements in the *PanelItem* array.

Move

If zero, files should be copied, if nonzergo - moved.

OpMode

Combination of the <u>operation mode</u> flags. This function should be ready to process OPM_SILENT flag. Also it can process OPM_DESCR.

If OPM_SILENT is not set, you can ask the user for confirmation and allow to edit destination path.

eturn value

If the function succeeds, the return value must be 1 or 2. If the return value is 1, FAR tries to position the cursor to the most recently created file on the active panel. If the plugin returns 2, FAR does not perform any positioning operations.

If the function fails, 0 should be returned. If the function was interrupted by the user, it should return -1.

emarks

1. If the operation has failed, but part of the files was successfully processed, the plugin can remove selection only from the processed files. To perform it, plugin should clear the <u>PPIF_SELECTED</u> flag for processed items in the *PluginPanelItem* list passed to function.

e also:

GetFiles, GetDirList, GetPluginDirList

etDirectory

in | exported functions

The **SetDirectory** function is called to set the current directory in the file system emulated by the plugin.

```
int WINAPI SetDirectory(
   HANDLE hPlugin,
   const char *Dir,
   int OpMode
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

Dir

Directory name. Usually contains only the name, without full path. To provide basic functionality the plugin should also process the names '...' and '\'. For correct restoring of current directory after using "Search from the root folder" mode in the Find file dialog, the plugin should be able to process full directory name returned in the <u>GetOpenPluginInfo</u> function. It is not necessary when "Search from the current folder" mode is set in the Find file dialog.

OpMode

Combination of the <u>operation mode</u> flags. This function should be ready to process the <u>OPM_FIND</u> flag. If the <u>OPM_FIND</u> flag is set, the function is called from Find file or another directory scanning command, and the plugin must not perform any actions except changing directory and returning TRUE if successful or FALSE if it is impossible to change the directory. (The plugin should not try to close or update the panels, ask the user for confirmations, show messages and so on.)

eturn value

If the function succeeds, the return value must be TRUE. If the function fails, FALSE should be returned.

emarks

If the <u>OPM_FIND</u> flag is set in *OpMode* then:

- 1. It means that the **SetDirectory** function is called from the Find file dialog (Alt-F7) or any other command that scans directory listings. Because if that the plugin must not perform any other actions except changing directory and returning TRUE if successful or FALSE if it is impossible to change the directory.
- 2. The plugin should carefully process ".." and if changing to the parent directory is not possible it should return FALSE. Otherwise the search can enter an endless loop on panel of that plugin.

etFindList

in | exported functions

The **SetFindList** function is called to put the file names found by the Find file command to the file system emulated by the plugin. The files should not be physically copied or changed.

```
int WINAPI SetFindList(
   HANDLE hPlugin,
   const struct PluginPanelItem *PanelItem,
   int ItemsNumber
);
```

arameters

hPlugin

Plugin handle returned by <u>OpenPlugin</u> or <u>OpenFilePlugin</u>.

PanelItem

Points to an array of <u>PluginPanelItem</u> structures. Each structure describes a file to put.

ItemsNumber

Number of elements in the *PanelItem* array.

eturn value

If the function succeeds, the return value must be TRUE. If the function fails, FALSE should be returned.

emarks

- This function is typically used by the Temporary panel plugin (TMPCLASS.CPP, function TmpPanel::SetFindList)
- Before calling this function, FAR calls the <u>OpenPlugin</u>(OPEN_FINDLIST,0) function. The **SetFindList** function is called only after a successful return of <u>OpenPlugin</u>.

xported functions - Editor specific functions

in exported functions

Function	Description
ProcessEditorInput	process keyboard events
ProcessEditorEvent	process editor events

e also:

Service functions, Structures, Archive support, Addons

rocessEditorInput

in | exported functions

The **ProcessEditorInput** function is called from the internal editor every time there is user input (mouse or keyboard) to process.

```
int WINAPI ProcessEditorInput(
   const INPUT_RECORD *Rec
);
```

arameters

Rec

Points to the <u>INPUT_RECORD</u> structure. This structure is defined in Win32 API and contains information about the last input (keyboard or mouse) event.

eturn value

If the plugin returns 0, the input event is processed by the FAR editor. The plugin should return 1 if it has completely processed the event or if the event should be discarded.

emarks

- 1. A plugin should correctly process incoming events by analysing INPUT_RECORD. EventType fields and in the case of an unhandled even to return control back to FAR.
- 2. While in macro playback keyboard events (KEY_EVENT) have a new type the INPUT_RECORD.EventType field equals 0x8001.
- 3. <u>EditorControl</u> commands can be called from this function, but be careful when calling <u>ECTL_PROCESSINPUT</u> from here, because this command calls the **ProcessEditorInput** function again. So the plugin should take steps to prevent recursion at this place.
- 4. A plugin does not receive the following key combinations: Ctrl-W, F11, Alt-F9, F12, Ctrl-Tab, Ctrl-Shift-Tab, Alt-Ins and Ctrl-Alt-Shift.
- 5. A plugin receives the following key combinations with the following restrictions:
 - Alt-F5 if the PrintMan plugin is not installed
 - Alt-F11 if the editor is modal

• **F6** - if switching to viewer is disabled (when calling <u>Editor</u> with the <u>EF_ENABLE_F6</u> flag omitted).

e also: <u>INPUT RECORD</u>

rocessEditorEvent

in | exported functions

The **ProcessEditorEvent** function informs plugins about different internal editor events.

```
int WINAPI ProcessEditorEvent(
    int Event,
    void *Param
);
```

arameters

Event

Event type.

Can be one of the following values (EDITOR_EVENTS enum):

Event	Description
EE_CLOSE	One of the internal editors is closing. Plugins can use this event to free internal data structures. Note that several editors can be active at the same time. Param points to an integer variable containing the EditorID parameter of the editor instance beeing closed. The EditorID of the current editor can be obtained earlier using the <u>ECTL_GETINFO</u> <u>EditorControl</u> command. But the plugin should not call the EditorControl function when processing this event, because the editor is already closed. Return value must be 0. Remark. Starting with FAR 1.70 build 1989 only the following commands can be used from <u>EE_CLOSE ECTL_GETINFO</u> and <u>ECTL_GETBOOKMARKS</u> .
EE_READ	A new file has just been read. The plugin can use <u>EditorControl</u> commands to modify the read data. <i>Param</i> equals NULL. Return value must be 0.
EE_SAVE	The file being edited is about to be saved. The plugin can use <u>EditorControl</u> commands to modify data before saving. <i>Param</i> equals NULL.

	Return value must be 0.		
EE_REDRAW	The editor screen is about to redraw. Plugin can useEditorControl ECTL ADDCOLORLine colors.Param can be one of the following vslue:ConstansDescription		
	EEREDRAW_ALL	The whole screen will be redrawn	
	EEREDRAW_LINE	Only the current line will be redrawn	
	EEREDRAW_CHANGE	Redrawing caused by text change	
		In the case of EEREDRAW_CHANGE the current line or the whole screen might be redrawn. So if changes were made to the highlighting outside the current line, it is recomended to call <u>ECTL_REDRAW</u> when you finished highlighting. Otherwise those changes will be seen only after cursor movement or other actions that cause screen redraw. Most important is not to enter recursion upon doing so.	
	Return value must be 0.		
EE_KILLFOCUS	Editor has lost keyboard focus. <i>Param</i> points to a variable containing the EditorID of the editor that looses focus. Return value must be 0.		
EE_GOTFOCUS	Editor received keyboard focus. <i>Param</i> points to a variable containing the EditorID of the editor that receives focus. Return value must be 0.		

Attention!

When processing EE_REDRAW it is <u>HIGLY</u> <u>UNDESIRABLE</u> TO CALL <u>Info.Message</u>, <u>Info.Menu</u>, <u>Info.Dialog</u> and <u>Info.DialogEx</u>. Calling those function leads to recursive calling of EE_REDRAW.

Param

Points to data dependent on the event type.

eturn value

Return value depends on the event type. Return 0 for unknown event types.

emarks

- EE_READ is called only once for each file.
- EE_SAVE is called every time F2 or Shift-F2 is pressed.
- EE_REDRAW is called every time the screen is redrawn (for example, after moving the cursor).

xported functions - Viewer specific functions

in | exported functions

Function	Description
ProcessViewerEvent	process viewer events

e also:

Service functions, Structures, Archive support, Addons

rocessViewerEvent

in | exported functions

The **ProcessViewerEvent** function informs plugins about different internal viewer events.

```
int WINAPI ProcessViewerEvent(
    int Event,
    void *Param
);
```

arameters

Event

Event type.

Can be one of the following values (VIEWER_EVENTS enum):

Event	Description
VE_CLOSE	One of the internal viewers is closing. Plugins can use this event to free internal data structures. Note that several viewers can be active at the same time. Param points to an integer variable containing the ViewerID parameter of the viewer instance being closed. The ViewerID of the current viewer can be obtained earlier using the <u>VCTL_GETINFO</u> <u>ViewerControl</u> command. But the plugin should not call the ViewerControl function when processing this event, because the viewer is already closed. Return value must be 0.
VE_READ	A new file has just been loaded. <i>Param</i> = NULL. Return value must be 0.
VE_KILLFOCUS	Viewer has lost input focus. <i>Param</i> points to a variable containing the ViewerID value of the viewer instance that has lost focus. Return value must be 0.
VE_GOTFOCUS	Viewer has got input focus. <i>Param</i> points to a variable containing the ViewerID value of the viewer instance that has got focus. Return value must be 0.

Param

Points to data dependent on the event type.

eturn value

Return value depends on the event type. Return 0 for unknown event types.

emarks

• VE_READ is called only once for each file.

e also: <u>ViewerControl</u>

xported functions - Dialog

Function	Description
<u>ProcessDialogEvent</u>	process dialog events

e also:

Service functions, Structures, Archive support, Addons

rocessDialogEvent

in | exported functions

The **ProcessDialogEvent** function informs plugins about different dialog events.

```
int WINAPI ProcessDialogEvent(
    int Event,
    void *Param
);
```

arameters

Event

Event type.

Can be one of the following values (DIALOG_EVENTS enum):

Event	Description
DE_DLGPROCINIT	Event was sent to the dialog handler. <i>Param</i> - pointer to the <u>FarDialogEvent</u> structure.
DE_DEFDLGPROCINIT	Event was sent to the <u>internal</u> dialog handler. <i>Param</i> - pointer to the <u>FarDialogEvent</u> structure.
DE_DLGPROCEND	Dialog handler processed the event. FarDialogEvent.Result contains the dialog handler return value. <i>Param -</i> pointer to the <u>FarDialogEvent</u> structure.

Param

Pointer to the <u>FarDialogEvent</u> structure.

eturn value

TRUE - the event was processed internally. FarDialogEvent.Result will be used as dialog handler return value.

FALSE - the event should be processed by the internal handler of the Dialog API kernel.

emarks

e also: FarDialogEvent, events, Dialog API

ervice functions - Common

in

Function	Description
<u>AdvControl</u>	advanced control functions; can be called from anywhere: panels, viewer or editor.
<u>CharTable</u>	allows to get information about installed character tables.
<u>CmpName</u>	function compares a text string (for example, a file name) with a pattern (mask).
<u>GetMsg</u>	returns a message from the <u>language file</u> .
<u>Menu</u>	shows a menu.
RestoreScreen	restores a screen area previously saved by SaveScreen.
<u>SaveScreen</u>	saves a screen area.
<u>ShowHelp</u>	shows the specified FAR help topic for the specified hlf file.
Text	writes a text string to the screen.

e also:

Exported functions, <u>Structures</u>, <u>Archive support</u>, <u>Addons</u>, <u>Win32</u> <u>structures and function</u>

dvControl

in | service functions

The **AdvControl** function provides access to FAR services and allows to query information. It can be called from anywhere: panels, viewer or editor.

```
INT_PTR WINAPI AdvControl(
    int ModuleNumber,
    int Command,
    void *Param
);
```

arameters

ModuleNumber

Number of the plugin module. It is passed to the plugin in the <u>SetStartupInfo</u> function.

Command

Control command type. Can be one of the following values (ADVANCED_CONTROL_COMMANDS enum):

Command	Description	
ACTL_COMMIT	"Commits" the results of the last operation with FAR windo ACTL_SETCURRENTWINDOW). <i>Param</i> is ignored (set to zero). The function returns TRUE on success or FALSE in case of	
ACTL_CONSOLEMODE	Retrieves or sets console mode (FullScreen <-> Windowed <i>Param</i> can be one of the following constants:	
	Constant	Description
	FAR_CONSOLE_GET_MODE	returns current console mode
	FAR_CONSOLE_SET_FULLSCREE	N sets the console to fullscreen mode
	FAR_CONSOLE_SET_WINDOWED	sets the console to windowed mod
	FAR_CONSOLE_TRIGGER	toggles console mode
	The returned value is the curre constants:	ent console mode. Can be one
	Constant	Description
	FAR_CONSOLE_WINDOWED	windowed mode
	FAR_CONSOLE_FULLSCREEN	fullscreen mode

ACTL_EJECTMEDIA	 Allows to programmatically eject media from removable dr ROM/USB/SUBST). <i>Param</i> points to an <u>ActlEjectMedia</u> structure. This command returns TRUE if the media was succesfully e FALSE is returned.
ACTL_KEYMACRO	Various actions with macro commands. <i>Param</i> points to an <u>ActlKeyMacro</u> structure. The return value is TRUE if the command was executed suc the execution failed (not enough memory, a macro comman executed or played).
ACTL_GETARRAYCOLOR	 Allows to get an array of all FAR colors. <i>Param</i> points to an array of bytes to receive the current FAI color is stored in one byte - high 4 bits is the background cc foreground color. If <i>Param</i> is equal to NULL, the size of the FAR color array i indexes defined in farcolor.hpp can be used for accessing th This command always returns the size of the FAR color array
ACTL_GETCOLOR	Allows to get the FAR color with the specified index. <i>Param</i> must contain one of the <u>COL</u> * color indexes define farcolor.hpp. The return value is the color if a correct index was specified
ACTL_GETFARVERSION	Gets the FAR version. <i>Param</i> either points to a variable of type DWORD, or it car Version format:
	HIWORD: = build number (FAR 1.7 LOWORD: HIBYTE = major version (FAR 1.7 LOBYTE = minor version (FAR 1.7
	<i>Param</i> can be set to NULL. This command returns the FAR version.
ACTL_GETFARHWND	Returns the window handle (HWND) of the current FAR Mai <i>Param</i> is ignored (set to 0).
ACTL_GETCONFIRMATIONS	Returns information about the confirmation settings. Return FarConfirmationsSettings flags set according to a "Confirmations" dialog. Param is ignored (set to 0).
ACTL_GETDESCSETTINGS	Returns information about the file description settings. Retu <u>FarDescriptionSettings</u> flags set according to opti descriptions" dialog. <i>Param</i> is ignored (set to 0).
ACTL_GETDIALOGSETTINGS	Returns information about the dialog settings. Returns a DW FarDialogSettings flags set according to options in the dialog. <i>Param</i> is ignored (set to 0).

ACTL_GETINTERFACESETTINGS	Returns information about the interface settings. Returns a I <u>FarInterfaceSettings</u> flags set according to options settings" dialog. <i>Param</i> is ingoner (set to 0).
ACTL_GETPANELSETTINGS	Returns information about the panel settings. Returns a DW <u>FarPanelSettings</u> flags set according to options in the dialog. <i>Param</i> is ignored (set to 0).
ACTL_GETPLUGINMAXREADDATA	Returns the maximum data size that can be passed to <u>OpenI</u> DWORD of any value from 0x1000 to 0x80000 (4KB to 51 be ready to receive any other value. <i>Param</i> is ignored (set to
ACTL_GETSYSTEMSETTINGS	Returns information about the system settings. Returns a DV <u>FarSystemSettings</u> set according to options in the "Sy <i>Param</i> is ignored (set to 0).
ACTL_GETSYSWORDDIV	Retrieves a string containing the word delimiter characters. <i>Param</i> points to a string buffer in which the word delimiters. Set <i>Param</i> to NULL to get string length (without the termina The maximum length of the buffer is 260 characters, includ null.
ACTL_GETWCHARMODE	Returns the FAR console working mode <i>Param</i> is ingnored (set to 0). Returns TRUE, is the FAR console mode is set to Unicode. set to 8-bit mode.
ACTL_GETWINDOWCOUNT	Returns the count of open windows in FAR Manager. <i>Param</i> is ignored (set to 0). There is always at least 1 open window (file panels, or an equivalent open window started with a command line parameter -e or -v)
ACTL_GETWINDOWINFO	Retrieve information about a FAR Manager window. <i>Param</i> - Param points to a <u>WindowInfo</u> structure. You must initialize the member WindowInfo.Pos before cal <i>WindowInfo.Pos</i> is equal to -1, information about the curren The return value is TRUE if the window with the index Wir and FALSE if there is no such window (in the latter case, th structure is not filled).
ACTL_GETSHORTWINDOWINFO	Retrieve information about a FAR Manager window. <i>Param</i> - Param points to a <u>WindowInfo</u> structure. You must initialize the member WindowInfo.Pos before cal <i>WindowInfo.Pos</i> is equal to -1, information about the curren The return value is TRUE if the window with the index Wir and FALSE if there is no such window (in the latter case, th structure is not filled). In oppose to the ACTL_GETWINDOWINFO command the <i>WindowInfo.TypeName</i> and <i>WindowInfo.Name</i> members are command can be called from any thread.

ACTL_POSTKEYSEQUENCE	Post a sequence of <u>internal</u> key codes to the FAR keyboard <i>Param</i> Param points to a <u>KeySequence</u> structure. The return value is TRUE if the keys have been posted succ case of an error. The keys will be interpreted as soon as the plugin returns cc
ACTL_REDRAWALL	Redraw all FAR windows. <i>Param</i> is ignored (set to 0).
ACTL_SETARRAYCOLOR	Allows to change a specified range of the FAR color schem <i>Param</i> points to a <u>FarSetColors</u> structure. The return value is TRUE if the range was successfully char parameters in the <u>FarSetColors</u> structure were specified ince
ACTL_SETCURRENTWINDOW	 Allows to switch to a specific FAR Manager window. <i>Param</i> is an integer specifying the index of the window to s numbering starts at 0). The function returns TRUE if the switch was successful or F failure (the window to switch to does not exist). Xttention! Attention! The switching will not occur untill ACTL_COM Manager receives control.
ACTL_WAITKEY	Allows to wait for a keystroke. If <i>Param</i> is set to -1 or NULL - waits for any key If <i>Param</i> is set to the internal key code - waits for that key. Returns value is always zero.

Param

Points to data dependent on the command type. See the command descriptions for specific information.

eturn value

Return value depends on the command type. See the command descriptions for specific information.

e also: <u>Control</u>, <u>EditorControl</u>

harTable

<u>in | service functions</u>

The **CharTable** function allows to get information about installed character tables.

```
int WINAPI CharTable(
    int Command,
    char *Buffer,
    int BufferSize
);
```

arameters

Command

Either the number of the requested character table or one of the following commands (the FARCHARTABLE_COMMAND enum):

Command	Description
FCT_DETECT	Autodetect the character table for given text

Buffer

If *Command* is equal to **FCT_DETECT**, specifies the address of a buffer with text data. Otherwise, specifies the address of a <u>CharTableSet</u> structure that receives information about the requested character table.

BufferSize

If *Command* is **FCT_DETECT**, *BufferSize* should contain the size of the buffer with text data to analyze. Otherwise it is the size of the <u>CharTableSet</u> structure.

eturn value

-1, if the requested table is not present or autodetection failed.

If successful, the function returns the number of the requested table and fills the structure pointed by **Buffer**. In **FCT_DETECT** mode it returns the number of the detected table and does not change **Buffer** data.

emarks

1. To enumerate all FAR character tables, start with **Command** equal to 0 and

increment it until the return value will be -1.

2. The <u>CharTableSet</u> structure is filled with OEM data if there where problems while reading settigs of some table (when *Command* does not equals FCT_DETECT).
mpName

in | service functions | FSF.ProcessName

The **CmpName** function compares a null-terminated text string (for example, a file name) with a pattern (mask).

```
int WINAPI CmpName(
   const char *Pattern,
   const char *String,
   int SkipPath
);
```

arameters

Pattern

Address of the pattern string.

String

Address of the null-terminated text string.

SkipPath

If TRUE, the file path in **String** is ignored and only the file name is used in comparison.

eturn value

TRUE if the string matches the pattern, otherwise FALSE.

emarks

The **Pattern** parameter can contain any characters allowed in file names and the following special characters (wildcards):

Wildcard	Description
*	any number of characters
?	any single character
[c,x-z]	any character from the range specified in square brackets; both individual characters and character ranges can be specified.

For example, files ftp.exe, fc.exe and f.ext will meet the following mask f*.ex?, the mask *co* corresponds to color.ini and edit.com, the mask [c-

ft]*.txt corresponds to config.txt, demo.txt, faq.txt and tips.txt.

ontrol

in | service functions

The **Control** function allows to request misc information and perform various control actions for the panels and the command line.

```
int WINAPI Control(
   HANDLE hPlugin,
   int Command,
   void *Param
);
```

arameters

hPlugin

Current plugin instance handle. To request information about the active panel set this parameter to INVALID_HANDLE_VALUE. This allows to use this function in plugin commands that work without creating new panel. The INVALID_HANDLE_VALUE is also used with none plugin panels.

Command

Control command type. Can be one of the following values (FILE_CONTROL_COMMANDS enum):

Command	Description
Panel	
FCTL_CHECKPANELSEXIST	Checks if the file panels exist. Param must be equal to 0 (unused). The function returns FALSE if FAR was started with th command line arguments (as an external viewer or edit this mode the panels are not created. Attention! When FAR is started with the /e or /v comma arguments, this function processes only one - FCTL_CHECKPANELSEXIST.
FCTL_CLOSEPLUGIN	Closes the current plugin. <i>Param</i> points to the name of the directory that will be s panel after closing the plugin.
FCTL_GETPANELINFO FCTL_GETANOTHERPANELINFO	Gets information about a plugin active/passive panel. <i>Param</i> points to a <u>PanelInfo</u> structure that will receive information.

	If no items are selected in panel, PanelInfo.SelectedItemsNumber is equal to 1 and PanelInfo.SelectedItems contains data for the item und In order to verify whether the file is actually selected, of the <u>PPIF_SELECTED</u> flag is set for that item.
	While processing the following request
	<pre>Info.Control(INVALID_HANDLE_VALUE,</pre>
	FAR call the <u>GetOpenPluginInfo</u> exported function of to which the panel belongs. FAR contains a protection endless recursion in the case when the plugin, from ins GetOpenPluginInfo() function, also calls Info.Control(, FCTL_GETPANELINFO), a secondary call of GetOpenPluginInfo() will not In some cases (e.g. searching in archives by Alt-F7) plugin panel is not really created, for that reason your
	the return value of the Control function, as to not crass most unfitting moment by working on an none existing Z Attention!
	The PanelItems and SelectedItems fields of <u>PanelInfo</u> structure will have different addre each new call of FCTL_GETPANELINFO FCTL_GETANOTHERPANELINFO.
FCTL_GETPANELSHORTINFO FCTL_GETANOTHERPANELSHORTINFO	Similar to FCTL_GETPANELINFO/FCTL_GETANOTHERPAN but the PanelItems and the SelectedItems fields of the structure are not filled and are set to NULL. This comm intended to be used when only the common information the active/passive panel is needed, without any concre- information on elements in that panel.
FCTL_REDRAWPANEL FCTL_REDRAWANOTHERPANEL	Redraws the plugin active/passive panel. <i>Param</i> can be either NULL or the address of a <u>PanelRe</u> structure, so you can set a new cursor position and the element for that panel. If <i>Param</i> is set to NULL, the cu position and the top element will not be changed. If <i>hPlugin</i> equals INVALID_HANDLE_VALUE, then panel will be redrawn no matter what command was u plugin. The panel will be redrawn only if that panel is visible moment.
FCTL_SETNUMERICSORT FCTL_SETANOTHERNUMERICSORT	Sets <u>numeric sort</u> mode for the active/passive panel. <i>Param</i> points to an integer value: 0 (turn numeric sort

	(turn numeric sort on). Setting <i>Param</i> to NULL is equivalent to setting the nur off.
FCTL_SETPANELDIR FCTL_SETANOTHERPANELDIR	Sets the current directory of a plugin active/passive par <i>Param</i> points to the directory name. If the plugin support own panel, it will be closed after execution of this com
	Note that this function resets the file selection in a direct makes it impossible to restore by pressing Ctrl-M, ev directory passed to this function is the same as the curr directory.
FCTL_SETSELECTION FCTL_SETANOTHERSELECTION	Sets active/passive panel items selection. <i>Param</i> points to the <u>PanelInfo</u> structure filled by a prev FCTL_GETPANELINFO or FCTL_GETANOTHERPANELINFO call. You must no other Control functions between FCTL_GETPANELII FCTL_SETSELECTION.
	To change selection, set or clear PPIF_SELECTED flag items of the array pointed to by the PanelItems member <u>PanelInfo</u> structure. Note that FCTL_GETPANELINFO FCTL_GETANOTHERPANELINFO return PPIF_SEL in this array set to its real state.
	You need to call FCTL_REDRAWPANEL to show the
FCTL_SETSORTMODE FCTL_SETANOTHERSORTMODE	Sets the active/passive panel sort mode. <i>Param</i> points to an integer containing the new sort mode " <u>Sort modes</u> ").
FCTL_SETSORTORDER FCTL_SETANOTHERSORTORDER	Sets the active/passive panel sort order. <i>Param</i> points to an integer value representing the sort of for normal order or 1 for reverse order. Setting <i>Param</i> to NULL is equivalent to setting the nor order (0).
FCTL_SETVIEWMODE FCTL_SETANOTHERVIEWMODE	Sets active/passive panel view mode. <i>Param</i> points to an integer containing the new view monumber, from 0 to 9. Setting <i>Param</i> to NULL is equivalent to setting mode 0
FCTL_UPDATEPANEL FCTL_UPDATEANOTHERPANEL	Updates plugin active/passive panel contents. If <i>Param</i> is NULL, the file selection will be cleared, oth selection is not changed.
Command line	
FCTL_GETCMDLINE	Gets the command line contents. <i>Param</i> points to the buffer to receive data (the buffer sl be smaller than 1 Kb).
FCTL_GETCMDLINEPOS	Gets the cursor position in the command line.

	<i>Param</i> points to a variable of type int that receives the position.
FCTL_GETCMDLINESELECTEDTEXT	Retrieves the selected text in the command line. <i>Param</i> points to the buffer to receive data (the buffer sl be smaller than 1 Kb).
FCTL_GETCMDLINESELECTION	Returns the parameters of the text selection in the com- line. <i>Param</i> points to a <u>CmdLineSelect</u> structure.
FCTL_INSERTCMDLINE	Inserts text into the command line beginning from the cursor position. <i>Param</i> points to a zero terminated string to insert to the command line.
FCTL_SETCMDLINE	Sets the command line contents. <i>Param</i> points to a zero terminated string to copy to the command line.
FCTL_SETCMDLINEPOS	Sets the cursor position in the command line. <i>Param</i> points to a variable of type int that contains the cursor position.
FCTL_SETCMDLINESELECTION	Selects a text fragment in the command line. <i>Param</i> points to a <u>CmdLineSelect</u> structure.
Other	
FCTL_SETUSERSCREEN	Copies the current screen contents to the FAR user scree (which is displayed when the panels are switched off). <i>Param</i> must be NULL.
FCTL_GETUSERSCREEN	Outputs the FAR user screen buffer (which is displayed the panels are switched off) to the screen. <i>Param</i> must be NULL.

Param

Points to control command parameters. Read the description of the **Command** parameter for concrete information.

eturn value

If the function succeeds, the return value is TRUE. If the function fails, FALSE is returned.

emarks

Usually you do not need to update or redraw panel and close plugin directly. FAR does this itself, when performing standard operations. These functions can become necessary to implement some non-standard functionality.

e also: <u>AdvControl</u>, <u>EditorControl</u>

reeDirList

in | service functions

The **FreeDirList** function releases the memory allocated for files list by <u>GetDirList</u> and <u>GetPluginDirList</u> functions.

```
void WINAPI FreeDirList(
   const struct PluginPanelItem *PanelItem
);
```

arameters

PanelItem

Address of an array of <u>PluginPanelItem</u> structures.

eturn value

None.

etDirList

in | service functions

The **GetDirList** function returns the list of files in the specified directory including subdirectories.

```
int WINAPI GetDirList(
   const char *Dir,
   struct PluginPanelItem **pPanelItem,
   int *pItemsNumber
);
```

arameters

Dir

Name of the directory to scan. It can be a name only or a full pathname.

pPanelItem

Points to the variable that will receive the address of an array of <u>PluginPanelItem</u> structures.

When this array is no longer needed, it must be passed to the <u>FreeDirList</u> function.

pItemsNumber

Points to the variable that will receive the number of PluginPanelItem <u>PluginPanelItem</u> structures.

eturn value

If the function succeeds, the return value is TRUE. If the function fails or directory scanning is cancelled by the user, FALSE is returned.

emarks

- 1. The function returns file names relative to the specified directory. For example, if **Dir** is D:\DIR1\DIR2, file names will be in DIR2\file.ext format.
- 2. The user can interrupt the directory scanning process by pressing Esc. In this case the function will return FALSE.

e also:

<u>GetPluginDirList</u>

etMsg

in | service functions

The **GetMsg** function returns a message from the <u>language file</u>. It is strongly recommended to use this function instead of hard-coding text constants directly in the program, because it allows to localize your plugin and switch the language of FAR and plugins simultaneously.

```
const char* WINAPI GetMsg(
    int PluginNumber,
    int MsgId
);
```

arameters

PluginNumber

Number of the plugin module. It is passed to the plugin in the <u>SetStartupInfo</u> function.

MsgId

Index of the message in the message file.

eturn value

The function returns the address of the requested message.

emarks

All * . lng files in plugin directory are considered as language files. FAR selects the necessary * . lng file depending on the current language. Before using **GetMsg** first time, all messages are loaded into the memory, so they can be accessed later much faster and you don't need to store the messages in an additional buffers.

xample

In all the <u>examples</u>, as you can see, the following function is used:

C/C++:

```
const char *GetMsg(int MsgId)
{
   return(Info.GetMsg(Info.ModuleNumber,MsgId));
}
```

Delphi:

```
function GetMsg(MsgId: TMessageStrings): PChar;
begin
  result:= Info.GetMsg(Info.ModuleNumber,integer(MsgId)
end;
```

Info is declared as a global variable:

```
struct PluginStartupInfo Info;
```

...and initialized in the <u>SetStartupInfo</u> function:

```
void WINAPI _export SetStartupInfo(struct PluginStart
{
    ...
    ::Info=*Info;
    ...
}
```

e also: Language and help files | LocMsg

etPluginDirList

<u>in | service functions</u>

The **GetPluginDirList** function returns list of files in the specified directory (including subdirectories) in the file system emulated by a plugin.

```
int WINAPI GetPluginDirList(
    int PluginNumber,
    HANDLE hPlugin,
    const char *Dir,
    struct PluginPanelItem **pPanelItem,
    int *pItemsNumber
);
```

arameters

PluginNumber

Number of the plugin module. It is passed to the plugin in the <u>SetStartupInfo</u> function.

hPlugin

Current plugin instance handle. If the plugin handle is equal to INVALID_HANDLE_VALUE, the value of PluginNumber is ignored and the function works with the plugin that controls the active panel.

Dir

Name of the plugin directory to scan. It must be in the form acceptable by the <u>SetDirectory</u> function of the plugin.

To keep the current plugin directory unchanged after the **GetPluginDirList** call, either *Dir* has to be a subdirectory of the current plugin directory, or the <u>SetDirectory</u> function of the plugin must be able to process a directory name returned by <u>GetOpenPluginInfo</u>. Otherwise the current directory will be changed and you should be ready to restore it.

pPanelItem

oints to the variable that will receive the address of an array of <u>PluginPanelItem</u> structures.

When this array is no longer needed, it must be passed to the <u>FreeDirList</u> function.

pItemsNumber

Points to the variable that will receive the count of <u>PluginPanelItem</u> structures.

eturn value

If the function succeeds, the return value is TRUE. If the function fails or directory scanning is cancelled by the user, FALSE is returned.

emarks

- 1. Returned file names are relative to the specified directory. For example, if **Dir** is D:\DIR1\DIR2, file names will be in DIR2\file.ext format.
- 2. The user can interrupt the directory scanning process by pressing Esc, which will cause the function to return FALSE.

e also: <u>GetDirList</u>

lenu

in | service functions

The **Menu** function shows a menu.

```
int WINAPI Menu(
    int PluginNumber,
    int X,
    int Y,
    int MaxHeight,
    DWORD Flags,
    const char *Title,
    const char *Bottom,
    const char *HelpTopic,
    const int *BreakKeys,
    int *BreakCode,
    const struct FarMenuItem *Item,
    int ItemsNumber
);
```

arameters

PluginNumber

Number of the plugin module. It is passed to the plugin in the <u>SetStartupInfo</u> function.

X, Y

Top left menu corner coordinates.

To assign coordinates automatically set them to **-1**

MaxHeight

Maximum count of visible menu items. If it is less than the items number, items will be scrolled.

To use maximum possible height set this parameter to 0.

Flags

Can be a combination of the following values (FARMENUFLAGS enum):

Constant	Description
FMENU_AUTOHIGHLIGHT	If specified, item hot keys will be assigned automatically, beginning from the first item.

FMENU_CHANGECONSOLETITLE	If specified, the the title of the console window will be set to Title (if Title is not
	empty).
FMENU_SHOWAMPERSAND	Shows ampersands in menu item texts. Without this flags ampersands are used to specify item hot keys.
FMENU_REVERSEAUTOHIGHLIGHT	If specified, item hot keys will be assigned automatically, beginning from the last item.
FMENU_USEEXT	Instead of <u>FarMenuItem</u> the <u>FarMenuItemEx</u> structure is used.
FMENU_WRAPMODE	If specified, attempts to move the cursor above the first item or below the last will move the cursor to the last or the first item, respectively.
	It is recommended to always set this flag, unless you have specific reasons not to do so.

If the **FMENU_USEEXT** flag is set then it is necessary to perform a type cast:

```
struct FarMenuItemEx FooEx[]={
    ...
```

```
};
Info.Menu(...,FMENU_USEEXT|...,(const struct FarMen
```

Title

Menu title. Set to NULL if menu title is not needed.

Bottom

Menu bottom title. Set to NULL if menu bottom title is not needed.

НеlpTopic

The <u>help topic</u> associated with the menu. Set to NULL if help is not needed.

BreakKeys

Address of an array with virtual key codes (VK_*), that will close the menu. The last array item must be 0. If you do not need to define such keys in addition to the standard keys (<Enter>, <Esc> and <F10>), set this parameter to NULL. The high word of an array item can be either 0 or a combination of **PKF_CONTROL**, **PKF_ALT** and **PKF_SHIFT** flags to describe corresponding key combinations.

For example in the MultiArc plugin in the "Archive commands" menu (Shift-F3 on archive) the F4 keystroke is processed in the following way:

```
int BreakCode;
int BreakKeys[2]={VK_F4,0};
ExitCode=Info.Menu(Info.ModuleNumber, -1, -1, 0, FMENU_
     GetMsg(MArcCmdTitle),GetMsg(MSelectF4),"ArcCmd
     (struct FarMenuItem *)MenuItems,Count);
if(ExitCode>=0)
{
  if(BreakCode == 0) // F4 pressed
  {
    GetFormatName(MenuItems[0].Text.Text);
    ConfigCommands(MenuItems[0].Text.Text,2+MenuDat
    continue;
  }
}
else
  return FALSE;
```

BreakCode

Address of a variable that will receive the index in the *BreakKeys* array of the key used to close the menu, or -1 if the menu was closed using one of the standard keys. This parameter can be NULL.

Item

Address of an array of <u>FarMenuItem</u> structures or if the **FMENU_USEEXT** flag is specified, address of an array of <u>FarMenuItemEx</u> structures. Each structure describes one menu item.

ItemsNumber

Number of <u>FarMenuItem</u> structures.

eturn value

This function returns either -1, if the user cancelled the menu, or the selected menu item number.

xample

This example is taken from the EditCase plugin:

```
struct FarMenuItem MenuItems[2];
memset(MenuItems,0,sizeof(MenuItems));
strcpy(MenuItems[0].Text,GetMsg(MCaseLower));
strcpy(MenuItems[1].Text,GetMsg(MCaseUpper));
MenuItems[0].Selected=TRUE;
int MenuCode=Info.Menu(Info.ModuleNumber,-1,-
        1,0,FMENU_AUTOHIGHLIGHT|FMENU_WRAPMODE,
        GetMsg(MCaseConversion),NULL,
        "Contents",NULL,NULL,
        MenuItems,
        sizeof(MenuItems)/sizeof(MenuItems[0]));
if (MenuCode<0)
    return(INVALID_HANDLE_VALUE);
. . . .
```

Info is defined as a global variable:

```
struct PluginStartupInfo Info;
```

...and is initialized in the <u>SetStartupInfo</u> function:

```
void WINAPI _export SetStartupInfo(struct PluginStart
{
    ...
    ::Info=*Info;
    ...
}
```

e also: <u>FarMenuItem</u>

estoreScreen

<u>in | service functions</u>

The **RestoreScreen** function restores a screen area previously saved by <u>SaveScreen</u>.

```
void WINAPI RestoreScreen(
   HANDLE hScreen,
);
```

arameters

hScreen

A handle received from <u>SaveScreen</u>. This handle is no longer usable after calling **RestoreScreen**.

eturn value

None.

emarks

To improve speed **RestoreScreen** redraws only the modified screen area. But if there was screen output produced by non-FAR functions (for example, if an external program was executed from a plugin), **RestoreScreen** cannot correctly calculate this area. In that case you need first to call **RestoreScreen** with **hScreen** set to NULL to inform FAR that the screen was changed and then call **RestoreScreen** as usual with <u>SaveScreen</u> handle.

e also: <u>SaveScreen</u>

aveScreen

in | service functions

The **SaveScreen** function saves a screen area. To restore it use the <u>RestoreScreen</u> function.

```
HANDLE WINAPI SaveScreen(
    int X1,
    int Y1,
    int X2,
    int Y2
);
```

arameters

X1,Y1,X2,Y2

Screen area coordinates. If *X2* or *Y2* is equal to -1, they are replaced with screen right or screen bottom coordinate correspondingly. So **SaveScreen(0,0,-1,-1)** will save the entire screen.

eturn value

The return value is a handle that can be passed to <u>RestoreScreen</u>. All handles allocated by **SaveScreen** must be passed to <u>RestoreScreen</u> to avoid memory leaks.

e also: RestoreScreen

howHelp

in | service functions

The **ShowHelp** function shows the specified topic from a given hlf-file.

```
BOOL WINAPI ShowHelp(
   const char *ModuleName,
   const char *HelpTopic,
   int Flags
);
```

arameters

ModuleName

Name of the plugin module. It is passed to the plugin in the <u>SetStartupInfo</u> function.

HelpTopic

<u>Help topic</u>. If this parameter is NULL, then the topic "<u>Contents</u>" will be used.

Flags

Can be one of the following values (FarHelpFlags enum):

Constant	Description
FHELP_SELFHELP	Assume <i>ModuleName</i> is Info.ModuleName and show the topic from the help file of the calling plugin. If <i>HelpTopic</i> begins with a coulomb ':', the topic from the main FAR help file will be shown (in that case <i>ModuleName</i> is ignored).
FHELP_FARHELP	<i>ModuleName</i> is ignored and the topic from the main FAR help file will be shown. In this case you do not need to start the <i>HelpTopic</i> with a coulomb ':'.
FHELP_CUSTOMFILE	Assume <i>ModuleName</i> specifies full path to a hlf-file (c:\path\filename).
FHELP_CUSTOMPATH	Assume <i>ModuleName</i> specifies full path to a folder (c:\path) from which a help file will be selected according to current language settings.
FHELP_USECONTENTS	If the specified <i>HelpTopic</i> is not found, will try to show the "Contents" topic. This flag can be combined with other flags.
FHELP_NOSHOWERROR	Disable file or topic not found error messages for this

1	function call. This flag can be combined with other flags.
---	--

eturn value

TRUE - parameters were successfully transferred to the Help Manager. FALSE - one of the following errors occurred:

- *Flags* contains an illegal value.
- *ModuleName* = NULL and **FHELP_FARHELP** flag is not set.
- Specified help file or topic were not found by the Help Manager.
- Help file or topic were not found while browsing the help file.

xample

For convience when frequently used the following function can be used in your plugin:

```
void ShowHelp(const char *HelpTopic)
{
    Info.ShowHelp(Info.ModuleName,HelpTopic,0);
}
```

Info is defined as a global variable...

struct PluginStartupInfo Info;

...ans is initialized in the <u>SetStartupInfo</u> function:

```
void WINAPI _export SetStartupInfo(const struct Plugi
{
. . .
. ::Info=*Info;
. . .
}
```

e also: <u>Help files</u>

ext

in | service functions

The **Text** function writes a text string to the screen. FAR uses internal screen buffering to improve performance so for compatibility reasons plugins must not write text directly to screen, but should use the **Text** function instead.

```
void WINAPI Text(
    int X,
    int Y,
    int Color,
    const char *Str
);
```

arameters

X,*Y*

Text coordinates. The origin of the coordinate system (0,0) - is at the top left cell of the screen.

Color

Text color attributes.

Str

Null-terminated text string. To display changes immediately call **Text** with *Str* set to NULL just after writing the string, for FAR to flush its screen buffer. But do not overuse it, because frequent buffer flushing decreases overall performance.

eturn value

None.

ervice functions - Editor

in

Function	Description
Editor	allows to invoke the FAR internal editor.
<u>EditorControl</u>	provides access to low level internal editor API.

e also:

Exported functions, Structures, Archive support, Addons, Win32 structures and function

ditor

in | service functions

The **Editor** function allows to invoke the FAR internal editor.

```
int WINAPI Editor(
  const char *FileName,
  const char *Title,
  int X1,
  int Y1,
  int X2,
  int Y2,
  DWORD Flags,
  int StartLine,
  int StartChar
);
```

arameters

FileName

Name of the file to edit. Unless **EF_CREATENEW** is set in the Flags parameter, must specify an existing file.

Title

Null-terminated text string that will be shown in the top line of the editor window. If this parameter is NULL, the file name will be used.

X1, Y1, X2, Y2

Editor window coordinates. If *X*² or *Y*² is -1, they will be replaced with the screen width or height. If *X*¹ or *Y*¹ are less than zero, then their are taken as zero.

Flags

Editor flags. Can be a combination of the following values (EDITOR_FLAGS enum):

Flag	Description
EF_NONMODAL	Creates a non-modal editor window. If this flag is present, the user will be able to switch to other FAR windows.
	The plugin will regain control only after the editor

	is closed, or after the user switches to a different window (e.g. by pressing Ctrl+Tab). If you need to regain control immediately after the editor has been opened, use the EF_IMMEDIATERETURN flag.
EF_IMMEDIATERETURN	If this flag is set, the Editor function returns immediately after the editor has been opened. The newly opened editor becomes the active window. This flag can be used <u>only</u> with EF_NONMODAL .
EF_DELETEONCLOSE	Instructs FAR to delete the file being edited after the editor is closed. If the directory with the edited file contains no other files, it will also be deleted. If only the file needs to be deleted use the EF_DELETEONLYFILEONCLOSE flag. The file will not be deleted:
	 if the user switched to the viewer by pressing F6, or if the file is open in other viewer or editor windows. if the user has saved the file.
EF_DELETEONLYFILEONCLOSE	Similar to EF_DELETEONCLOSE , but only the file will be deleted. The directory will not be deleted even if it is empty. This flag has a lower priority than EF_DELETEONCLOSE .
EF_CREATENEW	Opens a new (non-existing) file in the editor, similar to pressing Shift-F4 in FAR.
EF_ENABLE_F6	Enables switching from the editor to the viewer by pressing F6.
EF_DISABLEHISTORY	Disables adding the file name to the view/edit history (Alt-F11). If this flag is not specified, the name is added to the history.

StartLine

Number of the line to which the cursor is positioned (0-based).

StartChar

Initial cursor position in the line (1-based).

eturn value

This function can return one of the following values (EDITOR_EXITCODE enum):

Returned value	Description
EEC_OPEN_ERROR	 File open error, occurs in the following cases: FAR could not allocate enough memory while creating the editor (as an object); if <i>FileName</i> is an empty line; if <i>FileName</i> is a path to an existing folder; if the file to be opend is read-only and the user refused to continue editing this file in the corresponding dialog.
EEC_MODIFIED	Successful return. File was modified. This value is also returned if the EF_NONMODAL flag was used.
EEC_NOT_MODIFIED	Successful return. File was not modified.
EEC_LOADING_INTERRUPTED	File loading was stopped by user.

emarks

If StartLine and StartChar are both equal to -1 and the option "Save file position" is enabled, the previously saved file position will be restored.

e also:

<u>Viewer</u>

ditorControl

in | service functions

The **EditorControl** function provides access to the low level API of the internal editor.

```
int WINAPI EditorControl(
    int Command,
    void* Param
);
```

arameters

Command

Control command type. Can be one of the following (EDITOR_CONTROL_COMMANDS enum):

Command	Description
ECTL_ADDCOLOR	Specifies color for a line area. This command can be applied to specify several color areas. <i>Param</i> points to an EditorColo line does not exist, this command will return FALSE, otherw
ECTL_ADDSTACKBOOKMARK	Create navigation position ("stack bookmark") at current edit positions with index greater then current one will be deleted. command is processed successfully, this command returns TI
ECTL_CLEARSTACKBOOKMARKS	Deletes all navigation positions. <i>Param</i> must be NULL.
ECTL_DELETEBLOCK	Deletes the block currently selected in the editor. Returns TR deleted successfully or FALSE in case the editor is locked (th no block is selected. <i>Param</i> must be NULL.
ECTL_DELETECHAR	Deletes the character under cursor. <i>Param</i> must be NULL.
ECTL_DELETESTACKBOOKMARK	Deletes specified navigation position. <i>Param</i> contains index deleted (0 and greater) or -1 for deleting current navigation p positions can be recieved after executing <u>ECTL GETSTACE</u> with <i>Param</i> containing NULL. If command is processed successfully, this command returns
ECTL_DELETESTRING	Deletes the current line. <i>Param</i> must be NULL.
ECTL_EDITORTOOEM	Converts text from the editor codepage to the OEM codepage <u>EditorConvertText</u> structure.
ECTL_EXPANDTABS	Expands all tabulation characters in a line to spaces. <i>Param</i> p that contains the number of the line to expand or -1 to proces
ECTL_GETBOOKMARKS	Returns information about bookmarks for the current editor.

	EditorBookMarks structure. This command returns FALSE in case:
	 the file is not yet open; <i>Param</i> is NULL;
	If the command succeeds TRUE is returned.
ECTL_GETSTACKBOOKMARKS	Returns information about navigation positions ("stack book <i>Param</i> points to an <u>EditorBookMarks</u> structure or contains N count of navigation positions were successfully set (or summ count in case if <i>Param</i> was NULL).
ECTL_GETCOLOR	Gets the color of a line area. <i>Param</i> points to an <u>EditorColor</u> string or the specified color area does not exist, this comman otherwise TRUE.
ECTL_GETINFO	Gets editor information. <i>Param</i> points to an EditorInfo struct
ECTL_GETSTRING	Gets information about a line. <i>Param</i> points to an <u>EditorGetS</u> string will be in the editor codepage.
	<pre>// get the first line of the edited fil struct EditorGetString egs; egs.StringNumber=0; Info.EditorControl(ECTL_GETSTRING,⪖)</pre>
ECTL_INSERTSTRING	Inserts a new line at the current cursor position and moves the the new line or to the indented position. If <i>Param</i> oints to an the value 1, indent will be used when executing this comman <i>Param</i> to NULL or pass 0 in the variable pointed to by <i>Param</i> same as if the user presses <enter> in the editor; for example inserted into the new line if it does not contain any characters position.</enter>
	<pre>// insert an empty string without inder Info.EditorControl(ECTL_INSERTSTRING,0)</pre>
ECTL_INSERTTEXT	Inserts text at the current cursor position. <i>Param</i> points to a r the OEM codepage. The command correctly processes newli is processed in the same way as it it had been entered from the
	<pre>// insert the string "Text" at the curr Info.EditorControl(ECTL_INSERTTEXT,"Te></pre>
ECTL_NEXTSTACKBOOKMARK	Go to next navigation position. <i>Param</i> must be NULL. If command is processed successfully TRUE, otherwise FALSE.
ECTL_OEMTOEDITOR	Converts text from the OEM codepage to the editor codepage <u>EditorConvertText</u> structure.
ECTL_PREVSTACKBOOKMARK	Go to previous navigation position. If there were no navigatio

	ECTL_ADDSTACKBOOKMARKcommand, current editornew navigation position before executing this command.Param must be NULL. If command is processed successfully.TRUE, otherwise FALSE.
ECTL_PROCESSINPUT	Passes an <u>INPUT_RECORD</u> structure to the internal editor for to an <u>INPUT_RECORD</u> structure. Note: if your plugin exports the <u>ProcessEditorInput</u> function, immediately passed to that function. The scheme is simple:
	<pre>case ECTL_PROCESSINPUT: if (ProcessEditorInput(Param)) return(TRUE); </pre>
	So if you use EditorControl(ECTL_PROCESSINPUT) <u>ProcessEditorInput</u> function, you should take care to avoid in
ECTL_PROCESSKEY	 This command allows to send keystrokes to the internal editor keystrokes are passed in <i>Param</i>. The <u>internal key codes</u> are used (see farkeys.hpp). This command always returns TRUE.
	<pre>// go to the end of the file Info.EditorControl(ECTL_PROCESSKEY,(voi</pre>
ECTL_QUIT	Closes the editor. Any unsaved information will be lost. <i>Parc</i> command always returns TRUE.
ECTL_READINPUT	Fills the INPUT_RECORD structure with data recieved from <i>Param</i> points to an INPUT_RECORD (<i>this structure is defin the <u>ReadConsoleInput</u> function</i>).
ECTL_REALTOTAB	Converts real string position to screen position. If string does characters, source and result positions will be equal. <i>Param</i> p structure.
ECTL_REDRAW	Redraws the editor window. <i>Param</i> must be NULL.
ECTL_SAVEFILE	Saves the file currently being edited. <i>Param</i> points to an <u>Edite</u> <i>Param</i> is NULL, the default file name and format (DOS-forn "\r\n", Unix-format - "\n"). If the file is saved successfully, th otherwise FALSE.
ECTL_SELECT	Selects or deselects a block. <i>Param</i> points to an EditorSelect
ECTL_SETKEYBAR	Allows to control key bar titles in the editor: <i>Param</i> = NULL - restores the previous value <i>Param</i> = -1 - redraws the key bar <i>Param</i> = pointer to a <u>KeyBarTitles</u> structure. This command cannot be used in the code that processes the
	because when this event is processed, the key bar titles objecThis command returns TRUE on success or FALSE if it wastitles (if the key bar titles object does not yet exist).

ECTL_SETPARAM	Changes the settings of the current editor. <i>Param</i> points to an structure. This function returns TRUE if the settings have been success otherwise.
ECTL_SETPOSITION	Sets the cursor position. <i>Param</i> points to an EditorSetPositio
ECTL_SETSTRING	Sets the text of a line. <i>Param</i> points to an <u>EditorSetString</u> structure be in the editor codepage.
ECTL_SETTITLE	Sets the editor window title (top status line). The standard title restored after the plugin has finished processing. <i>Param</i> point text string that will be used as the title.
	<pre>// DrawLine\DrawLine.cpp: SetTitle func Info.EditorControl(ECTL_SETTITLE,(char</pre>
ECTL_TABTOREAL	Converts screen cursor position to a real string position. If sti tabulation characters, source and result positions will be equa EditorConvertPos structure.
ECTL_TURNOFFMARKINGBLOCK	Resets the editor flag that is set while the user is marking a binternal to FAR Manager and is not used by plugins. Howeve defects may appear if the user starts marking a block, then landunched automatically) and the plugin modifies, for example, Therefore, you should use this command before returning complugin modifies the text in the editor, block selection or curse <i>Param</i> must be NULL.

Param

Points to control command parameters. Read the description of the **Command** parameter for concrete information.

eturn value

If the function succeeds, the return value is TRUE. If the function fails, FALSE is returned.

emarks

The editor window contents is updated upon any active user operation. Call the **ECTL_REDRAW** command to force an update after any changes to the contents.

e also: <u>AdvControl</u>, <u>Control</u>

ervice functions - Viewer

in

Function	Description
<u>Viewer</u>	allows to invoke the internal viewer.
<u>ViewerControl</u>	allows to query and control the state of the internal viewer

e also:

Exported functions, <u>Structures</u>, <u>Archive support</u>, <u>Addons</u>, <u>Win32</u> <u>structures and function</u>

iewer

in | service functions

The Viewer function allows to invoke the FAR internal viewer.

```
int WINAPI Viewer (
   const char *FileName,
   const char *Title,
   int X1,
   int Y1,
   int X2,
   int Y2,
   DWORD Flags
);
```

arameters

FileName

Name of the file to view.

Title

Text string that will be shown in the top line of the viewer window. If set to NULL, the file name will be used as the title.

X1, Y1, X2, Y2

Viewer window coordinates. If *X2* or *Y2* equals -1, they will be replaced with screen width or height. If *X1* or Y1 is less than zero it is considered as zero.

Flags

Can be a combination of the following values (VIEWER_FLAGS enum):

Flag	Desciption
VF_DELETEONCLOSE	Instructs FAR to delete the viewed file after closing the viewer. If the directory containing the viewed file contains no other files, it will also be deleted. If only the file needs to be deleted use VF_DELETEONLYFILEONCLOSE . The file will not be deleted if the user pressed F6 to switch between viewing and editing, or if the same file is open in a different editor or viewer instance.
VF_DELETEONLYFILEONCLOSE	Similar to VF_DELETEONCLOSE , but only the file will be deleted. This flag has a lower priority

	than VF_DELETEONCLOSE.
VF_DISABLEHISTORY	Disables adding the file name to the viewer history (Alt - F11). By default, the file name is added to the history.
VF_ENABLE_F6	Enables switching from viewer to editor by pressing F6.
VF_IMMEDIATERETURN	Allows the plugin to receive control immediately after the viewer is opened. The newly opened viewer becomes the active window. This flag makes sense <u>only</u> if combined with the VF_NONMODAL flag.
VF_NONMODAL	Creates a non-modal viewer window. If this flag is present, the user will be able to switch to other FAR windows. The plugin will regain control only after the viewer is closed, or after the user switches to a different window (by pressing Ctrl-Tab). If you need to regain control immediately after the viewer has been opened, use the VF IMMEDIATERETURN flag.
	viewer has been opened, use the VF_IMMEDIATERETURN flag.

eturn value

If the **VF_NONMODAL** flag is not specified, the function returns **TRUE** if successful or **FALSE** if the file cannot be opened. If the flag is specified, the function always returns **TRUE**.

e also:

<u>Editor</u>

iewerControl

in | service functions

The ViewerControl function allows to query and control the state of the internal viewer.

```
int WINAPI ViewerControl(
    int Command,
    void *Param
);
```

arameters

Command

Control command type. Can be one of the following (VIEWER_CONTROL_COMMANDS enum):

Command	Description
VCTL_GETINFO	Gets viewer information. Param points to a <u>ViewerInfo</u> structure. This command always returns TRUE.
VCTL_QUIT	Close the viewer. <i>Param</i> must be NULL. This command always returns TRUE.
VCTL_REDRAW	Redraws the viewer window. <i>Param</i> must be NULL. This command always returns TRUE.
VCTL_SETKEYBAR	Allows to control key bar titles in the viewer: <i>Param</i> = NULL - restores the previous value <i>Param</i> = -1 - redraws the key bar <i>Param</i> = pointer to a KeyBarTitles structure. This command always returns TRUE.
VCTL_SELECT	Controls selection. <i>Param</i> points to a <u>ViewerSelect</u> structure. If <i>Param</i> = NULL, selection will be reset. If the command succeeds TRUE is returned.
VCTL_SETMODE	Change viewer mode. <i>Param</i> points to a <u>ViewerSetMode</u> structure. If the command succeeds TRUE is returned.
VCTL_SETPOSITION	Sets position in file. <i>Param</i> points to an <u>ViewerSetPosition</u> structure. If the command succeeds TRUE is returned.

Param

Read the description of the **Command** parameter for concrete information.

eturn Value

Read the description of the **Command** parameter for concrete information.

emark

In FAR 1.70 build 1579 and newer VCTL_QUIT when send from an information or a qiuck view panel does not close the viewer.

e also:

Service functions, ViewerInfo, ViewerSetPosition, ViewerSelect
ervice functions - Dialog API

in

Function	Description
<u>DefDlgProc</u>	allows to call the internal dialog callback function.
Dialog	shows a dialog.
<u>DialogEx</u>	shows a dialog that allows to assign for it a callback function.
InputBox	a simple dialog box allowing to enter one line of text.
Message	shows a message.
<u>SendDlgMessage</u>	used to send a message to the dialog callback function.

e also:

Exported functions, <u>Structures</u>, <u>Archive support</u>, <u>Addons</u>, <u>Win32</u> <u>structures and function</u>

efDlgProc

in | Dialog API | Events and Messages

The **DefDlgProc** function allows to call the internal dialog callback function.

```
LONG_PTR WINAPI DefDlgProc(
   HANDLE hDlg,
   int Msg,
   int Param1,
   LONG_PTR Param2
);
```

arameters

hDlg

Dialog handle

Msg

One of the Dialog API messages or events.

Param1

The 1st parameter.

Param2

The 2nd parameter.

eturn value

The return value depends on the **Msg** parameter.

xample

A fragment from the Reversi game dialog callback proc:

```
LONG_PTR WINAPI ReversiDialogProc(HANDLE hDlg, int Ms
{
    ...
    return Info.DefDlgProc(hDlg,Msg,Param1,Param2);
}
```

e also: <u>DialogEx</u> <u>SendDlgMessage</u>

ialog

in | <u>Dialog API</u>

The **Dialog** function shows a dialog.

```
int WINAPI Dialog(
    int PluginNumber,
    int X1,
    int Y1,
    int X2,
    int Y2,
    const char *HelpTopic,
    struct FarDialogItem *Item,
    int ItemsNumber
);
```

arameters

PluginNumber

Number of the plugin module. It is passed to the plugin in the <u>SetStartupInfo</u> function.

X1, Y1, X2, Y2

Dialog coordinates. You can specify them explicitly or use "Width x Height" formula - in this case both *X1* and *Y1* must be set to -1, while *X2* and *Y2* define dialog width and height respectively. In the latter case the dialog will be automatically centered on the screen. *X2* and *Y2* parameters can't be less than zero.

HelpTopic

Help topic associated with the dialog. It can be NULL if help is not required.

Item

Address of an array of <u>FarDialogItem</u> structures. Each structure describes one dialog item.

ItemsNumber

Number of FarDialogItem structures.

eturn value

This function returns either -1, if the user cancelled the dialog, or the index of

the selected dialog item in the Item array.

emarks

FAR transforms Item elements to its own internal structure before creating a dialog. After dialog processing is over, Item elements array is adjusted according to changes made in the progress of user work with the dialog.

xample

Example from the configuration dialog of TempPanel plugin:

```
int Config()
{
  struct InitDialogItem InitItems []={
    DI_DOUBLEBOX, 3, 1, 72, 8, 0, 0, 0, 0, (char *)MConfigTitl
    DI_CHECKBOX, 5, 2, 0, 2, 0, 0, 0, 0, 0, (char *)MConfigAddToL
    DI_FIXEDIT,7,3,7,3,1,0,0,0,"",
    DI_TEXT, 9, 3, 0, 3, 0, 0, 0, 0, (char *)MConfigDisksMenuL
    DI_TEXT, 5, 4, 0, 4, 0, 0, DIF_BOXCOLOR | DIF_SEPARATOR, 0,
    DI_CHECKBOX, 5, 5, 0, 5, 0, 0, 0, 0, 0, (char *)MConfigCommor
    DI_TEXT, 5, 6, 0, 6, 0, 0, DIF_BOXCOLOR | DIF_SEPARATOR, 0,
    DI_BUTTON,0,7,0,7,0,0,DIF_CENTERGROUP,1,(char *)
    DI_BUTTON, 0, 7, 0, 7, 0, 0, DIF_CENTERGROUP, 0, (char *)M
  };
  struct FarDialogItem DialogItems[sizeof(InitItems)/
  InitDialogItems(InitItems, DialogItems,
                     sizeof(InitItems)/sizeof(InitItems
  int ExitCode=Info.Dialog(Info.ModuleNumber,
                    -1, -1, 76, 10,
                    "TempCfg", DialogItems,
                    sizeof(DialogItems)/sizeof(DialogIt
  if (ExitCode != 7)
    return(FALSE);
}
```

ialogEx

in | Dialog API | Events and Messages

The DialogEx function shows a dialog with the possibility to assign a callback function for it.

```
int WINAPI DialogEx(
    int PluginNumber,
    int X1,
    int Y1,
    int X2,
    int Y2,
    const char *HelpTopic,
    struct FarDialogItem *Item,
    int ItemsNumber,
    DWORD Reserved,
    DWORD Flags,
    FARWINDOWPROC DlgProc,
    LONG_PTR Param
);
```

arameters

PluginNumber

Plugin module number. It is passed to the plugin in the <u>SetStartupInfo</u> function

X1, Y1, X2, Y2

Dialog coordinates. You can specify them explicitly or use "Width x Height" formula - in this case both *X1* and *Y1* must be set to -1, while *X2* and *Y2* define dialog width and height respectively. In the latter case the dialog will be automatically centered on the screen. *X2* and *Y2* parameters can't be less than zero.

HelpTopic

<u>Help topic</u> for the dialog. If help is not needed, set this parameter to NULL.

Item

Address of an array of <u>FarDialogItem</u> structures. Each structure describes one dialog item.

ItemsNumber

Number of *FarDialogItem* array elements.

Reserved

Reserved for future use. Must be 0.

Flags

Set of flags, specifying additional dialog parameters. It can be a combination of the following values (FARDIALOGFLAGS enumeration):

Flag	Description
FDLG_WARNING	Sets "Warning" color scheme for the dialog.
FDLG_SMALLDIALOG	Allows to create dialogs with reduced border size. When drawing separators (DIF_SEPARATOR) for these dialogs it's assumed there's no space between dialog border and dialog double-line frame.
FDLG_NODRAWSHADOW	Don't draw shadow under the dialog.
FDLG_NODRAWPANEL	Don't draw dialog panel.

DlgProc

Pointer to the <u>FARWINDOWPROC</u> dialog callback function.

Param

Data that will be sent to the dialog callback function with the <u>DN_INITDIALOG</u> event.

eturn value

The function returns either -1 when user cancels the dialog or the selected dialog item index (index of *Item* array element, it's emphasized in the example below).

emarks

- 1. FAR transforms Item elements to its own internal structure before creating a dialog. After dialog processing is over, Item elements array is adjusted according to the changes made in the progress of user work with the dialog.
- Starting from version 1.71 build 2451 "small" dialogs (FDLG_SMALLDIALOG flag) are drawn with a shadow. Set the FDLG_NODRAWSHADOW flag to suppress drawing a shadow for these dialogs.

e also:

Dialog, DefDlgProc, SendDlgMessage, Service functions, Message, InitDialogItems

lgProc

in | Dialog API | Events and Messages

In a plugin there must be a dialog callback function, which is responsible for processing events and messages sent to the dialog. The function has four parameters: dialog handle, message and two additional parameters.

Dialog handler function syntax is presented here, as it should appear in a plugin.

```
LONG_PTR WINAPI DlgProc(
HANDLE hDlg,
int Msg,
int Param1,
LONG_PTR Param2
);
```

arameters

hDlg Dialog handle

Msg

One of events or messages.

Param1

Parameter 1

Param2

Parameter 2

eturn value

The **DlgProc** function return value depends on the **Msg** parameter.

emarks

Sometimes information contained in **Param1** and **Param2** consists of two parts, which are placed in two 16-bit words, composing each parameter. There're two macros defined in Windows to provide access to each part of **Param1** and

Param2 - LOWORD

and **<u>HIWORD</u>**

They return high-order and low-order words respectively from long int 32-bit value.

xample

Dialog handler code fragment for Reversi game:

```
LONG_PTR WINAPI ReversiDialogProc(HANDLE hDlg, int Ms
{
 struct FarDialogItem DialogItem;
 struct FarListItem *ListItems;
  int i;
 switch(Msg)
  {
    case DN INITDIALOG:
      // Get information about the element
      Info.SendDlgMessage(hDlg,DM_GETDLGITEM,75,(LONC
      ListItems=DialogItem.ListItems->Items;
      . . .
      NewGame(hDlg);
      return FALSE;
    case DN_HELP:
    {
      // Show different help topics depending on game
      static char *Help[3]={"Contents", "Rule", "Recomm
```

```
if(NumPl1==2 && NumPl2 == 2)
        i=0;
      else if(NumPl1+NumPl2 > 16)
        i=2;
      else
        i=1;
      return (LONG_PTR)(Help[i]);
    }
    . . .
    case DM CLOSE:
      // Check the element with which the user tries
      if(Param1 != 10 && Param1 > 0)
        return FALSE; // one can't close the dialog
      break;
 }
 // Let the Dialog Manager process other events and
 return Info.DefDlgProc(hDlg,Msg,Param1,Param2);
}
```

e also:

DefDlgProc, DialogEx, SendDlgMessage

putBox

```
in | service functions
```

The **InputBox** function displays a simple dialog box allowing to enter one line of text.

```
int WINAPI InputBox(
    const char *Title,
    const char *Prompt,
    const char *HistoryName,
    const char *SrcText,
    char *DestText,
    int DestLength,
    const char *HelpTopic,
    DWORD Flags
);
```

arameters

Title

Iput dialog title. Can be NULL or "".

Prompt

Prompt text (text above the input line). Can be NULL or "".

HistoryName

Name of the "history" record for the input line. Set to NULL if history is not needed.

SrcText

The initial value of the input line. Can be NULL or "".

DestText

Points to the result string. Can point to the same buffer as **SrcText**, but you must reserve enough space.

DestLength

Size of destination buffer.

НеlрТоріс

<u>Help topic</u> for the inputbox in the format of "<FullPath>Topic", e.g.:

"<D:\\FAR\\Plugins\\Foo\\>FooInfo"

Set to NULL if help is not used.

Flags

Can be a combination of the following values (INPUTBOXFLAGS enum):

Constant	Description
FIB_ENABLEEMPTY	the function will return true even if the input line is empty.
FIB_PASSWORD	used to input passwords - entered text is represented by ' * ' on the screen.
FIB_EXPANDENV	after a successful return, any environment variables present in the input line will be replaced by their values in the DestText buffer, e.g. if the user entered '%TEMP%', then DestText will contain 'C : \TEMP'.
FIB_NOUSELASTHISTORY	if SrcText is empty and HistoryName is not NULL, then do not initialize the input line from the history.
FIB_BUTTONS	displays a separator and the [OK] and [Cancel] buttons below the input line. The dialog will grow by 2 lines.
FIB_NOAMPERSAND	the ampersand character will not be shown in the prompt string but can instead be used to define a hotkey.

eturn value

The function returns TRUE in case of successful user input, and FALSE in case of user interruption.

emarks

FAR Manager uses this function to promt the user when creating a folder:



xample

e also: Dialog

lessage

in | service functions

The **Message** function shows a message.

```
int WINAPI Message(
    int PluginNumber,
    DWORD Flags,
    const char *HelpTopic,
    const char * const *Items,
    int ItemsNumber,
    int ButtonsNumber
);
```

arameters

PluginNumber

Number of the plugin module. It is passed to the plugin in the <u>SetStartupInfo</u> function.

Flags

Can be a combination of the following values (FARMESSAGEFLAGS enum):

Flag	Description
FMSG_WARNING	Warning message colors are used (white text on red background by default).
FMSG_ERRORTYPE	If error type returned by <u>GetLastError</u> <u>is known to FAR</u> or Windows, the error description will be shown in the first message line. In that case, the text given by the plugin will be displayed below the error description.
FMSG_KEEPBACKGROUND	Do not redraw the message background.
FMSG_DOWN	Display the message two lines lower than usual.

1	
FMSG_LEFTALIGN	Left align the message lines instead of centering them.
FMSG_ALLINONE	In this case the <i>Items</i> parameter is not an array of string pointers. Instead it points to a single string in which the lines of the message are separated by the newline character '\n'.
	Minimal number of lines is - 2 - a title and one message line.
	If this flag is specified the <i>ItemsNumber</i> parameter is ignored and the number of lines shown is calculated automatically (taking into account the button flags -FMSG_MB_*). To suppress title output when this flag is specified, start the line with a '\n' character.
FMSG MB OK	Additional button: <ok></ok>
FMSG_MB_OKCANCEL	Additional buttons: <ok> and <cancel></cancel></ok>
FMSG_MB_ABORTRETRYIGNORE	Additional buttons: <abort>, <retry> and <ignore></ignore></retry></abort>
FMSG_MB_YESNO	Additional buttons: <yes> and <no></no></yes>
FMSG_MB_YESNOCANCEL	Additional buttons: <yes>, <no> and <cancel></cancel></no></yes>
FMSG_MB_RETRYCANCEL	Additional buttons: <retry> and <cancel></cancel></retry>

НеlрТоріс

The <u>help topic</u> associated with the message.Set to NULL if help is not used.

Items

Address of an array of pointers to null-terminated text strings. The first string is the message title, the last *ButtonsNumber* strings are buttons, and all other strings belong to the message body.

To draw a single border line start the string with a character with code 1 (x001).

To draw a double border line start the string with a character with code 2 (x002).

See also the description of the flag <u>FMSG_ALLINONE</u>

ItemsNumber

Number of strings in the array passed in the *Items* parameter. Minimal values - 2 lines.

ButtonsNumber

Number of strings which are shown as buttons. If one of the FMSG_MB_* flags is set, this value is ignored.

eturn value

This function returns either -1, if the user cancelled the message (or the sysrem could not allocate enough memory for internal buffers), or the number of the selected button (for the first button 0 is returned, for the second 1 is returned, and so on).

emarks

- 1. In FAR Manager versions up to (and including) 1.70 beta 4 the maximum number of items in a message (including the buttons) was limited to 13.
- 2. If *ButtonsNumber* is zero and none of the FMSG_MB_* flags is set the plugin <u>should</u> restore the screen either by using <u>RestoreScreen</u> or in any other way when the message output is no longer necessary
- 3. If *ButtonsNumber* is not equal to zero, the screen will be restored by FAR.
- 4. If *Items* is NULL or the total number of items is less than 2, the message is not shown.
- 5. When FMSG_MB_* button flags are specified the *ButtonsNumber* parameter is ignored.
- 6. It is possible to specify hotkeys for buttons.
- 7. When using the FMSG_ALLINONE flag you need to do an explicit typecast to achieve error free compilation:

```
Info.Message(Info.ModuleNumber,
FMSG_ALLINONE|FMSG_MB_OKCANCEL,
"HelpTopic",
(const char * const *)"Title\nItem1\nItem2\nItem3",
0,0);
```



or

```
const char *Msg="Title\nItem1\nItem2\nItem3\nOk\nCancel";
Info.Message(Info.ModuleNumber,
FMSG_ALLINONE,
"HelpTopic",
(const char * const *)Msg,
0,2);
```

xample

The following function displays a file deletion confirmation dialog:

```
Удаление
Вы хотите поместить в Корзину
bin.zip
Удалить Отменить
BOOL IsDeleted(char *filename)
{
  const char *Msg[5];
  Msg[0]=GetMsg(MTitle); // message title
  Msg[1]=GetMsg(MIsDeleted); // message body
  Msg[2]=filename;
                              // last ButtonsNumber
  Msg[3]=GetMsg(MDelete);
  Msg[4]=GetMsg(MCancel);
  return Info.Message(Info.ModuleNumber,
                 Θ,
                 "DeleteFile",
                 Msg,
                 sizeof(Msg)/sizeof(Msg[0]),
                 2) == 0;
}
```

Info is defined as a global variable:

struct PluginStartupInfo Info;

...and is initialized in the <u>SetStartupInfo</u> function:

```
void WINAPI _export SetStartupInfo(struct PluginStart
{
    ...
    ::Info=*Info;
    ...
}
```

```
e also:
Dialog
```

endDlgMessage

in | Dialog API Messages

The **SendDlgMessage** function is used to send a message to the dialog callback function.

```
LONG_PTR WINAPI SendDlgMessage(
   HANDLE hDlg,
   int Msg,
   int Param1,
   LONG_PTR Param2
);
```

rguments

hDlg

Dialog handle

Msg

One of the Dialog API messages.

Param1

The 1st parameter.

Param2

The 2nd parameter.

eturn value

Return value depends on Msg value.

xample

A fragment from the Reversi game dialog callback proc:

```
LONG_PTR WINAPI ReversiDialogProc(HANDLE hDlg, int Ms
{
...
case DN_INITDIALOG:
//get element info
Info.SendDlgMessage(hDlg,DM_GETDLGITEM,75,(LONG
ListItems=DialogItem.ListItems->Items;
...
```



e also: <u>DialogEx</u>, <u>DefDlgProc</u>

ddEndSlash

in | FarStandardFunctions

The **FSF.AddEndSlash** function is used to add a trailing backslash or slash to a path. The symbol that will be added depends on those used in the path.

```
int WINAPI AddEndSlash(
    char *Path
);
```

arameters

Path

A string containing the path to which you want to add a trailing slash or backslash.

Note that Path must have enough space for an additional character.

eturn value

On success return value is TRUE, otherwise return value is FALSE.

emarks

- 1. The string must be large enough to contain an additional character ('\' or '/').
- 2. This function works with both types of slashes normal and backslashes.
- 3. If a string already has a trailing slash it will be converted to the slash of such type which is more common in the *Path* string.
- 4. No slash will be added at the end of the string if the string already contains a trailing slash.

toi

in | FarStandardFunctions

The **FSF.atoi** function converts a string to a 32-bit integer.

```
int WINAPI atoi(
   const char *Str
);
```

arameters

Str

Points to a string to convert.

eturn value

If the function succeeds, return value is the converted value of the input string *Str*, otherwise it returns 0.

emarks

The *Str* string parameter must be in the following form:

[ws][sn][ddd] where

```
ws - space or tab characters (ignored)
```

sn - sign - '+' or '-'

```
ddd - one or more decimal digits - from '0' to '9'
```

The function stops reading the input string at the first character that it cannot recognize as part of a number. In case of overflow the return value is undefined.

xample

You can define and initialize a function pointer to use it later:

```
FARSTDATOI FarAtoi;
...
FarAtoi=Info.FSF->atoi;
...
I=FarAtoi(Str);
```

... or call the function directly:

I=Info.FSF->atoi(Str);

e also:

FSF.atoi64 | FSF.itoa | FSF.itoa64

toi64

in | FarStandardFunctions

The **FSF.atoi** function converts a string to a 64-bit integer (__**int64**).

```
__int64 WINAPI atoi64(
   const char *Str
);
```

arameters

Str

Points to a string to convert.

eturn value

If the function succeeds, return value is the converted value of the input string, otherwise it returns 0i64.

emarks

The *Str* string parameter must be in the following form:

[ws][sn][ddd] where

```
ws - space or tab characters (ignored)
```

sn - sign - '+' or '-'

```
ddd - one or more decimal digits - from '0' to '9'
```

The function stops reading the input string at the first character that it cannot recognize as part of a number. In case of overflow the return value is undefined.

xample

You can define and initialize a function pointer to use it later:

```
FARSTDAT0I64 FarAtoi64;
...
FarAtoi64=Info.FSF->atoi64;
...
I_64=FarAtoi64(Str);
```

...or call the function directly:

I_64=Info.FSF->atoi64(Str);

e also:

FSF.atoi | FSF.itoa | FSF.itoa64

search

in | FarStandardFunctions

The **FSF.bsearch** function allows to perform a binary search of a sorted array.

```
void* WINAPI bsearch(
  const void *key,
  const void *base,
  size_t nelem,
  size_t width,
  int (__cdecl *fcmp)(const void *, const void *)
);
```

arameters

key

Points to a value that you want to search for.

base

Points to an element from which you want the search to be started.

nelem

The number of elements in the array you want to search.

width

The size of each element in bytes.

fcmp

User-defined comparison function that must be declared with ___cdecl - Cstyle calling convention. This function must compare two accepted elements and return an integer value:

```
*elem1 < *elem2 - fcmp returns value < 0
*elem1 == *elem2 - fcmp returns value == 0
*elem1 > *elem2 - fcmp returns value > 0
```

eturn value

bsearch returns the address of the first occurrence of *key* value in the array **base** or NULL if no occurrence found.

emarks

See the C/C++ run-time library reference for more information.

xample

e also: FSF.qsort

onvertNameToReal

in | FarStandardFunctions

The **FSF.ConvertNameToReal** function converts a relative name of a file object to its full pathname and expands symbolic links (Windows 2000 reparse points).

```
int WINAPI ConvertNameToReal(
   const char *Src,
   char *Dest,
   int DestSize
);
```

arameters

Src

Source string - a full or relative name of a file or a directory.

Dest

Destination string - the expanded pathname will be stored here. Can be NULL.

DestSize

Length of the destination string. If *Dest*=NULL, *DestSize* is ignored.

eturn value

The actual size needed to store the expanded pathname in *Dest*.

- For mounted drives that do not have a drive letter assigned, the function will store into *Dest* a string similar to this one: "\\? \Volume{273872e0-5e49-11d5-b614-0080ad70bb9b}\Foo.bar"
- If, for example, the directory "D:\Foo\Bar" is a symbolic link to an existing directory "C:\work\doc", calling this function for the file "D:\Foo\Bar\1092\readme.txt" will return "C:\work\Doc\1092\readme.txt".

emarks

1. The function correctly determines the real pathname only under Windows 2000 or later. Under earlier operating systems, it is not possible to determine the real name of a symbolic link if one is encountered in the path.

2. Parameters *Src* and *Dest* can point to the same string.

opyToClipboard

in | FarStandardFunctions

The **FSF.CopyToClipboard** function copies a text string to the Windows clipboard.

```
int WINAPI CopyToClipboard(
   const char *Data
);
```

arameters

Data

Pointer to the string that you want to place into the clipboard.

eturn value

On success the return value is TRUE, otherwise the return value is FALSE.

emarks

See FAQ:"<u>There are dupes in the Clipboard...</u>"

eleteBuffer

in | FarStandardFunctions

The **FSF.DeleteBuffer** function is used to free an allocated buffer returned by the <u>PasteFromClipboard</u> function.

```
void WINAPI DeleteBuffer(
   void *Buffer
);
```

arameters

Buffer

Pointer to the buffer that needs to be freed.

eturn value

None.

emarks

This function must be used to free FAR memory blocks since plugin's memory manager can be different from the one used in FAR.

xpandEnvironmentStr

in | FarStandardFunctions

The **FSF.ExpandEnvironmentStr** function is used to expand environment variables in a string to their values.

```
DWORD WINAPI ExpandEnvironmentStr(
   const char *Src,
   char *Dest,
   size_t Size
);
```

arameters

Src

Pointer to a null-terminated string containing references to environment variables of the form: %VariableName%. For each such reference, the %VariableName% portion is replaced with the current value of that environment variable.

The replacement rules are the same as those used by the command interpreter. Case is ignored when looking up the environment-variable name. If the name is not found, the %VariableName% portion is left unchanged.

Dest

Pointer to the buffer that will receive the result of the expansion. May be the same as **Src**.

Size

Size of the destination buffer (**Dest**), including the trailing '\0'.

eturn value

The function returns the number of characters stored into the buffer. If the environment variable expansion fails, up to (Size-1) characters are copied from Src to Dest.

emarks

1. This function is just a "wrapper" for the **ExpandEnvironmentStrings**

Windows API function, so you can see Windows API documentation for the details.

- Unlike <u>ExpandEnvironmentStrings</u> FSF.ExpandEnvironmentStr always fills Dest buffer.
- 3. **Src** and **Dest** must be in the OEM code page.

arInputRecordToKey

in | FarStandardFunctions

The **FSF.FarInputRecordToKey** function is used to convert a key code from an <u>INPUT_RECORD</u> structure to an internal FAR key code.

```
int WINAPI FarInputRecordToKey(
    INPUT_RECORD *Rec
);
```

arameters

Rec

Pointer to an **INPUT_RECORD** structure you want to convert.

eturn value

Return value is an internal FAR key code.

emarks

arKeyToName

in | FarStandardFunctions

The **FSF.FarKeyToName** is used to convert an internal FAR <u>key code</u> to a string.

```
BOOL WINAPI FarKeyToName(
    int Key,
    char *KeyText,
    int Size
);
```

arameters

Key

Internal FAR key code to convert to a string.

KeyText

String that will receive the result of conversion.

Size

The size of converted string, not including a null character, that will be copied to *KeyText*, or 0 to use the whole length of converted string.

eturn value

If conversion succeeds TRUE is returned, otherwise if the passed *Key* is not known to FAR, FALSE is returned.

emarks

If you use 0 in *Size*, then *KeyText* must be at least 32 bytes long.

arNameToKey

in | FarStandardFunctions

The **FSF.FarNameToKey** is used to convert a literal key name to an internal FAR key code.

```
int WINAPI FarNameToKey(
   const char *Name
);
```

arameters

Name

Points to a string containing a literal key name that you want to convert to an internal FAR key code.

eturn value

If conversion succeeds an internal FAR <u>key code</u> is returned, otherwise if the passed *Name* is not known to FAR, -1 is returned.

emarks

- 1. If the literal key name contains Ctrl or Alt part and the "letter", this "letter" will be uppercased. For example, FarNameToKey("CtrlAltz") will return KEY_CTRLALTZ.
- 2. If the literal key name contains Ctrl or Alt and the "letter from the national alphabet" this "letter" will be converted to it's keyboard equivalent and uppercased. For example, FarNameToKey("Ctrly") will return KEY_CTRLE.
- 3. Shift-"letter" combination will be converted to the "LETTER" key (Shift part will be removed and the "letter" will be uppercased).

arRecursiveSearch

in | FarStandardFunctions

The **FSF.FarRecursiveSearch** function is used to find a file in a directory tree with a name matching the given mask.

```
void WINAPI FarRecursiveSearch(
   const char *InitDir,
   const char *Mask,
   <u>FRSUSERFUNC</u> UserFunc,
   DWORD Flags,
   void *Param
);
```

arameters

InitDir

Name of the directory where you want to start the search. ("c:\far\plugins" for example).

Mask

File mask to search for. Starting with FAR 1.70 beta 4, all standard features of FAR masks (multiple masks, character ranges, exclude masks and so on) are supported (see <u>File masks</u>).

UserFunc

Pointer to a user-defined callback function of **FRSUSERFUNC** type that is called for every found file. This function have to return **TRUE** to continue the search or **FALSE** to stop it.

Flags

Can be a combination of the following flags (FRSMODE enumeration):

Flag	Description
FRS_RETUPDIR	When the search is stopped, the FullName parameter of the <u>UserFunc</u> function will contain the directory name where the file is found, instead of the name of the file.
FRS_RECUR	Recurse the directory tree while searching for the specified file.
FRS_SCANSYMLINK	The search will follow symbolic links just as if they

If one of the scanned symbolic links is a "recursive" one (for example, symbolic link points to one of it's root directories) the search will continue until the maximum allowed length of the full path string will be reached.
be reached.

Param

Application-defined value to be passed to the callback function specified in the *UserFunc* parameter.

eturn value

None.

emarks

- 1. If you want to use the data passed in the parameters of the **UserFunc** function after the search is complete, you must copy it to an internal variable.
- 2. <u>In FAR versions up to and including 1.70 beta 3</u> the "*" *Mask* is to be used if you want to find all files.
- 3. <u>In FAR versions up to and including 1.70 beta 3</u>, when doing recursive search, the file mask given in the *Mask* parameter is also used to determine the directories searched. So, if you want to find a file recursively, in most cases you should pass "*" in the *Mask* parameter and use the callback function to stop the search when the needed file is found.
- 4. If the **FRS_SCANSYMLINK** flag is not set, symbolic links will not be parsed, in no dependence of FAR configuration.
etFileOwner

in | FarStandardFunctions

The **FSF.GetFileOwner** function is used to determine the owner of the given file.

```
int WINAPI GetFileOwner(
   const char *Computer,
   const char *Name,
   char *Owner
);
```

arameters

Computer

Name of the computer containing the file for which you want to determine the owner. If this value is NULL, then the owner of the file for the current system is determined.

Name

The name of the file for which you want to determine the owner.

Owner

Pointer to a buffer that receives the file owner. This buffer must be large enough to hold the returned string (minimum buffer size is - \underline{NM})

eturn value

If the function succeeds the return value is TRUE, otherwise the return value is FALSE.

emarks

etNumberOfLinks

in | FarStandardFunctions

The **FSF.GetNumberOfLinks** function returns the number of <u>hard links</u> to the specified file.

```
int WINAPI GetNumberOfLinks(
   const char *Name
);
```

arameters

Name

Name of the file for which you want to obtain the number of links.

eturn value

If the function succeeds, the return value is the number of links to the specified file, otherwise the return value is 0. On file systems other than NTFS the number of links to a file is always 1.

e also: <u>MkLink</u>

etPathRoot

in | FarStandardFunctions

The **FSF.GetPathRoot** function is used to get the root directory from a given path.

```
void WINAPI GetPathRoot(
   const char *Path,
   char *Root
);
```

arameters

Path

The path from which you want to get the root directory.

Root

Buffer that receives the root directory.

eturn value

None.

emarks

Root must be large enough to hold the resulting string:

- Local drives driver letter, colon, slash C:\
- Reparse point (on NTFS 5 filesystem Windows 2000) something like \\?\Volume{be877ec2-afd6-11d4-b5e3-806d6172696f}\ or \??\D:\
- UNC-path host and share **host\share**\

etReparsePointInfo

in | FarStandardFunctions

The **FSF.GetReparsePointInfo** function allows to determine the target (path to the target drive and directory) of a symbolic link (reparse point).

```
int WINAPI GetReparsePointInfo(
   const char *Src,
   char *Dest,
   int DestSize
);
```

arameters

Src

Source string. Must contain a full pathname to a symbolic link terminated with a backslash (' $\$ ').

Dest

Destination string. May be NULL.

DestSize

Length of the destination string (*Dest*). If *Dest*=NULL, *DestSize* is ignored.

eturn value

The actual size of the string stored in *Dest*, or the required buffer size if *Dest* is NULL.

The function returns 0 in case of an error:

- the function is not supported (the current operating system is not Windows 2000 or later);
- if the path is invalid (in this case, the system error code ERROR_PATH_NOT_FOUND is set).
- symbolic link is on a network drive and in most cases real directory name where symbolic link points to is useless.

emarks

1. Src and Dest can point to the same string.

- 2. The function works only under Windows 2000 or later.
- 3. Return value is practically useless for symbolic links on network drives.
- 4. Unlike <u>ConvertNameToReal</u>, this function can only be used for symbolic links.
- 5. You can obtain a bit of information about symbolic links <u>here</u>.

Oa <u>in | FarStandardFunctions</u>

The **FSF.itoa** function converts a 32-bit integer value into a string.

```
char * WINAPI itoa(
    int Value,
    char *Str,
    int Radix);
```

arameters

Value

Integer value to convert.

Str

Pointer to a buffer that will receive the resulting string. The size of the *Str* string must be large enough to hold the converted string (max value = 32 symbols + '\0').

Radix

Base of *Value*. Must be in then range of 2 - 36.

eturn value

The return value is a pointer to the resulting string (*Str*).

emarks

If *Radix* equals 10 and *Value* is negative, the first character of the result string will be the minus sign '-'.

xample

You can define and initialize a function pointer to use it later:

```
FARSTDITOA FarItoa;
...
FarItoa=Info.FSF->itoa;
...
FarItoa(Value,Str,10);
```

... or call the function directly:

Info.FSF->itoa(Value,Str,10);

e also: <u>FSF.atoi</u> | <u>FSF.atoi64</u> | <u>FSF.itoa64</u>

oa64

in | FarStandardFunctions

The **FSF.itoa64** function converts a 64-bit integer value into a string.

```
char * WINAPI itoa64(
   __int64 Value,
   char *Str,
   int Radix);
```

arameters

Value

64-bit integer value to convert.

Str

Pointer to a buffer that will receive the resulting string. The size of the *Str* string must be large enough to hold the converted string (max value = 64 symbols + '\0').

Radix

Base of *Value*. Must be in then range 2 - 36.

eturn value

The return value is a pointer to the resulting string (*Str*). There is no error return.

emarks

If *Radix* equals 10 and *Value* is negative, the first character of the resulting string will be the minus sign '-'.

xample

You can define and initialize a function pointer to use it later:

```
FARSTDITOA64 FarItoa64;
...
FarItoa64=Info.FSF->itoa64;
```

Faiit0a04-1110.F3F->1t0a04

FarItoa64(Value64,Str,10);

... or call the function directly:

```
Info.FSF->itoa64(Value64,Str,10);
```

e also: <u>FSF.atoi</u> | <u>FSF.atoi64</u> | <u>FSF.itoa</u>

IsAlpha

in | FarStandardFunctions

The **FSF.LIsAlpha** function tests whether the given character is a letter. This function works in OEM code page.

```
int WINAPI LISAlpha(
    unsigned Ch
);
```

arameters

Ch

The character you want to test.

eturn value

If the given character is a letter returns TRUE, otherwise returns FALSE.

emarks

IsAlphanum

in | FarStandardFunctions

The **FSF.LIsAlphanum** function tests whether the given character is a letter or a number. This function works in OEM code page.

```
int WINAPI LIsAlphanum(
    unsigned Ch
);
```

arameters

Ch

The character you want to test.

eturn value

If the given character is a letter or a number returns TRUE, otherwise returns FALSE.

emarks

IsLower

in | FarStandardFunctions

The **FSF.LIsLower** function tests whether the given character is in lower case. This function works in OEM code page.

```
int WINAPI LISLower(
    unsigned Ch
);
```

arameters

Ch

The character you want to test.

eturn value

If the given character is in lower case returns TRUE, otherwise returns FALSE.

emarks

IsUpper

in | FarStandardFunctions

The **FSF.LIsUpper** function tests whether the given character is in upper case. This function works in OEM code page.

```
int WINAPI LIsUpper(
    unsigned Ch
);
```

arameters

Ch

The character you want to test.

eturn value

If the given character is in upper case returns TRUE, otherwise returns FALSE.

emarks

Lower

```
in | FarStandardFunctions
```

The **FSF.LLower** function converts a character to lower case. This function works in OEM code page.

```
unsigned WINAPI LLower(
    unsigned UpperChar
);
```

arameters

UpperChar

The character you want to convert.

eturn value

This function returns the converted character.

emarks

LowerBuf

in | FarStandardFunctions

The **FSF.LLowerBuf** function converts an array of characters, including null ones, to lower case. This function works in OEM code page.

```
void WINAPI LLowerBuf(
   char *Buf,
   int Length
);
```

arameters

Buf

An array of characters you want to convert.

Length

Size of the array in bytes.

eturn value

None.

emarks

Stricmp

in | FarStandardFunctions

The **FSF.LStricmp** function compares two strings without case sensitivity. This function works in OEM code page.

```
int WINAPI LStricmp(
   const char *Str1,
   const char *Str2
);
```

arameters

Str1, Str2

The strings you want to compare.

eturn value

```
This function returns:
-1 - if s1 < s2
1 - if s1 > s2
0 - if s1 == s2
```

emarks

Strlwr

in | FarStandardFunctions

The **FSF.LStrlwr** function converts a null-terminated string to lower case. This function works in OEM code page.

```
void WINAPI LStrlwr(
   char *s1
);
```

arameters

s1

The string you want to convert.

eturn value

None.

emarks

Strnicmp

in | FarStandardFunctions

The **FSF.LStrnicmp** function compares portions of two strings without case sensitivity. This function works in OEM code page.

```
int WINAPI LStrnicmp(
   const char *Str1,
   const char *Str2,
   int Num
);
```

arameters

Str1, Str2

The strings you want to compare.

Num

Number of characters to compare.

eturn value

This function returns: -1 - if s1 < s2 1 - if s1 > s2 0 - if s1 == s2

emarks

Strupr

in | FarStandardFunctions

The **FSF.LStrupr** function converts a null-terminated string to upper case. This function works in OEM code page.

```
void WINAPI LStrupr(
   char *Str
);
```

arameters

Str

The string you want to convert.

eturn value

None.

emarks

Trim

in | FarStandardFunctions

The **FSF.LTrim** function removes all leading whitespaces from a string.

```
char* WINAPI LTrim(
   char *Str
);
```

arameters

Str

The string from which you want to remove leading whitespace. The result will be stored in the same string.

eturn value

Pointer to the resulting string.

e also: <u>FSF.RTrim</u> | <u>FSF.Trim</u>

Upper

in | FarStandardFunctions

The **FSF.LUpper** function converts a character to upper case. This function works in OEM code page.

```
unsigned WINAPI LUpper(
    unsigned LowerChar
);
```

arameters

LowerChar

The character you want to convert.

eturn value

This function returns the converted character.

emarks

UpperBuf

in | FarStandardFunctions

The **FSF.LUpperBuf** function converts an array of characters, including null ones, to upper case. This function works in OEM code page.

```
void WINAPI LUpperBuf(
   char *Buf,
   int Length
);
```

arameters

Buf

An array of characters you want to convert.

Length

Size of the array in bytes.

eturn value

None.

emarks

lkTemp

in | FarStandardFunctions

The **FSF.MkTemp** function is used to create a temporary file name with the path based on a specified template.

```
char* WINAPI MkTemp(
    char *Dest,
    const char *Prefix
);
```

arameters

Dest

Pointer to buffer to receive the temporary file name. It must be large enough to hold the resulting string (the path to the temporary directory + 12 characters for the name of the temporary file).

Prefix

Points to a null-terminated prefix string. At most four leading characters from that string will be used as the filename prefix. FAR will pad the prefix with zeroes if its length is less than 4 bytes.

If *Prefix* is NULL or points to an empty string, the standard prefix "FTMP" will be used.

eturn value

Pointer to *Dest* containing the temporary file name, or NULL if function has failed. A possible reason for the failure is that the temporary directory contains too many files and should be cleaned.

emarks

1. The temporary file name is obtained by concatenating the temporary directory path (returned by the <u>GetTempPath</u>

Windows API function), the prefix passed to the function and several random hexadecimal digits. The name has the following format:

PrefXXXP.PTT

where

Pref - Pref is the 4-character prefix;**XXX** - three random hexadecimal digits;**PP** - two hexagemical digits from process ID (returned by the

GetCurrentProcessId

Windows API

function);

 $\mathbf{T}\mathbf{T}$ - two hexagemical digits from thread ID (returned by the

<u>GetCurrentThreadId</u> function).

Windows API

- 2. Unlike in FAR 1.70 beta 3, this function does not create the file on the disk; it only generates the name.
- 3. In FAR 1.70 beta 3, this function used only the first three characters of the prefix.

xample

char TempName[NM];

<pre>FSF.MkTemp(TempName,NULL);</pre>	->	"FTMP000D.P50"
<pre>FSF.MkTemp(TempName,"");</pre>	->	"FTMP000D.P50"
<pre>FSF.MkTemp(TempName, "MY");</pre>	->	"MY00000D.P50"
<pre>FSF.MkTemp(TempName, "BaR");</pre>	->	"BAR0000D.P50"
<pre>FSF.MkTemp(TempName, "TstPlugin");</pre>	->	"TSTP000D.P50"

lkLink

in | FarStandardFunctions

The **FSF.MkLink** function supports creating hard and symbolic links, directory junctions and mounting local drives to the file system. The function works only under Windows NT 4 or higher.

```
int WINAPI MkLink(
   const char *Src,
   const char *Dest,
   DWORD Flags
);
```

arameters

Src

Name of the file object to which the link is created.

Dest

Name of the created link.

Flags

Operation mode. One of the following flags (MKLINKOP enum):

Operation	Description
FLINK_HARDLINK	Create a hard link.
FLINK_JUNCTION	Create a directory junction.
FLINK_VOLMOUNT	Mount a local drive to the file system.
FLINK_SYMLINKFILE	Create a file symbolic link.
FLINK_SYMLINKDIR	Create a directory symbolic link.

You can combine operation mode with one of the following flags:

Flag	Description
FLINK_SHOWERRMSG	Show error messages.
FLINK_DONOTUPDATEPANEL	Do not update the panel after the link has been created.

eturn value

- 1 the link was created successfully.
- 0 error creating link.

Possible error resons:

- For hard links:
 - Src and Dest are on different partitions;
 - the partition is not NTFS;
 - the partition is not local;
 - Src does not exist or is not a file;
 - Dest already exists;
- For directory junctions:
 - Src or Dest is not on the local partition;
 - the partition is not NTFS 5.0;
 - Src does not exist or is not a directory;
 - Dest exists, but is not an empty directory;
- For volume mounts:
 - Src or Dest is not on the local partition;
 - the partition for Dest is not NTFS 5.0;
 - Src does not exist or is not a local drive;
 - Dest exists, but is not an empty directory;

emarks

- 1. The links are created according to the following rules:
 - hard links are created only for files within a single NTFS partition (NT4/Win2K/XP);
 - Directory junctions are created only for directories within local NTFS partitions (**Win2K/NTFS 5.0**);
 - mounting local drives to the file system is possible only on NTFS partitions (**Win2K/NTFS 5.0**).
- 2. If the value of Src is, for example, "C:", a volume mount will be created instead of a junction.
- 3. If the destination directory for a volume mount operation is terminated with a backslash, a subdirectory "disk_N" will be created in Dest, where N is the letter of the drive being mounted.
- 4. On Windows 2000 you cannot create a junction which points to a CD-ROM folder, but you can mount this CD-ROM disk as an NTFS folder (see <u>Mount Points</u>) and then create the necessary junction.

xample

e also: <u>GetNumberOfLinks</u>

asteFromClipboard

in | FarStandardFunctions

The **FSF.PasteFromClipboard** function is used to get data from the Windows clipboard.

char* WINAPI PasteFromClipboard(void);

arameters

None.

eturn value

Pointer to string, or NULL if the function has failed.

emarks

The buffer returned from this function must be freed through a call to the <u>DeleteBuffer</u> function.

ointToName

in | FarStandardFunctions

The **FSF.PointToName** function is used to get a file name from a given file path.

```
char *WINAPI PointToName(
   const char *Path
);
```

arameters

Path

The file path from which you want to get the file name.

eturn value

Pointer to the file name in the given path.

emarks

rocessName

in | FarStandardFunctions | CmpName

The **FSF.ProcessName** function allows to perform various actions on a file name: compare with a mask, with a list of masks or to generate new file name using the mask.

```
int WINAPI ProcessName(
   const char *Param1,
   char *Param2,
   DWORD Flags
);
```

arameters

Param1

Depends on the Flags value.

Param2

Depends on the Flags value.

Flags

Specifies a command that can be one of the following values (PROCESSNAME_FLAGS enum):

Action	Description
PN_CMPNAME	Compares a file name with the specified mask. This flag works like the <u>CmpName</u> function - Param1 corresponds to Pattern, Param2 corresponds to String. In case of success TRUE is returned.
PN_CMPNAMELIST	Compares a file name with a list of masks delimited by commas. This flag works like PN_CMPNAME, but Param1 contains the list of masks. Note that this function doesn't support exclude masks that were first implemented in FAR 1.70 beta 4.
PN_SKIPPATH	This flag is a modifier for the PN_CMPNAME and PN_CMPNAMELIST flags. It specifies that the path to the file name must be ignored when comparing.
PN_GENERATENAME	Generates a file name based on the name contained in Param1 and a mask contained in Param2. The result is returned in Param2. In case of success TRUE is returned, otherwise FALSE. If there is a

	necessity to process only a part of Param1, there is a possibility to specify a size of this part (up to 255) by combining it with PN_GENERATENAME, for example: Param1 contains "dir1\\file1" but the user wants to change only "dir1", then flags must contain PN_GENERATENAME 4.
--	---

eturn value

The return value depends on the **Flags** parameter.

sort

in | FarStandardFunctions

The **FSF.qsort** function allows to sort an array of any type of data using the QuickSort algorithm.

```
void WINAPI qsort(
  void *Base,
  size_t NElem,
  size_t Width,
  int (__cdecl *fcmp)(const void *, const void *)
);
```

arameters

Base

Start of target array.

NElem

Array size in elements.

Width

The size of each element in bytes.

fcmp

```
User-defined comparison function that must be declared with ___cdecl - C- style calling convention. This function takes two arguments - elem1 and elem2. These arguments are the pointers to the array elements. fcmp function must compare these elements and return an integer value:
```

```
*elem1 < *elem2 - fcmp returns value < 0
*elem1 == *elem2 - fcmp returns value == 0
*elem1 > *elem2 - fcmp returns value > 0
```

eturn value

None.

emarks

- 1. See the C/C++ run-time library reference for more information.
- 2. If you need to pass user-defined data to the compare function, you should use the <u>qsortex</u> function instead.

3. The sort implemented by the qsort and qsortex functions is not stable. In other words, the order for the elements that are equal according to the compare function is not defined. The order can change when the array is sorted repeatedly.

xample

e also: <u>FSF.bsearch, FSF.qsortex</u>

sortex

in | FarStandardFunctions

The **FSF.qsortex** function allows to sort an array of any type of data using the QuickSort algorithm. Unlike the <u>qsort</u> function, it allows to pass user-defined data to the compare function.

```
void WINAPI qsortex(
   void *Base,
   size_t NElem,
   size_t Width,
   int (__cdecl *fcmp)(const void *, const void *, void
   void *User
);
```

arameters

Base

Start of target array.

NElem

Array size in elements.

Width

The size of each element in bytes.

fcmp

User-defined comparison function that must be declared with ___cdecl - Cstyle calling convention. This function takes three arguments - *elem1*, *elem2* (the pointers to the array of elements) and *user* (user-defined data passed in the *User* argument to the **qsortex** function). *fcmp* function must compare *elem1* and *elem2* elements and return an integer value:

```
*elem1 < *elem2 - fcmp returns value < 0
*elem1 == *elem2 - fcmp returns value == 0</pre>
```

*elem1 > *elem2 - fcmp returns value > 0

User

User-defined data passed as the third parameter to the comparison function.

eturn value

None.

emarks

The sort implemented by the <u>qsort</u> and qsortex functions is not stable. In other words, the order for the elements that are equal according to the compare function is not defined. The order can change when the array is sorted repeatedly.

e also: <u>FSF.bsearch</u>, <u>FSF.qsort</u>
uoteSpaceOnly

in | FarStandardFunctions

The **FSF.QuoteSpaceOnly** function encloses an input string in double quotes if it contains at least one space inside.

```
char* WINAPI QuoteSpaceOnly(
   char *Str
);
```

arameters

Str

String that you want to quote. The result will be placed in the same string.

eturn value

This function returns a pointer to the resulting string.

emarks

- 1. Note that Str must be large enough to hold the resulting string.
- 2. This function does nothing if the string is already enclosed in quotes.

Trim

in | FarStandardFunctions

The **FSF.RTrim** function removes all trailing whitespace from a string.

```
char* WINAPI RTrim(
   char *Str
);
```

arameters

Str

String from which you want to remove the trailing whitespace.

eturn value

On return Str contains a string with trailing spaces removed.

e also: <u>FSF.LTrim</u>, <u>FSF.Trim</u>

nprintf

in | FarStandardFunctions

The **FSF.snprintf** function allows to write formatted output to a string.

```
int WINAPI snprintf(
   char *Buffer,
   size_t Sizebuf,
   const char *Format,
   ...
);
```

arameters

Buffer

Buffer that receives the formatted string.

Sizebuf

The maximum allowed size of the receiving buffer.

Format

Format string.

•••

Series of arguments, in accordance with the format string.

eturn value

If the function succeeds the return value is the number of bytes put to *Buffer*, otherwise it returns -1.

emarks

See a C/C++ run-time library reference for more information.

Delphi:

You can use format() function. Please read Object Pascal (Delphi) language reference for more information on that function.

e also: FSF.sprintf

printf

in | FarStandardFunctions

The **FSF.sprintf** function allows to write formatted output to a string.

```
int WINAPI sprintf(
   char *Buffer,
   const char *Format,
   ...
);
```

arameters

Buffer

Buffer that receives the formatted string.

Format

Format string.

•••

Series of arguments, in accordance with the format string.

eturn value

If the function succeeds the return value is the number of bytes put to *Buffer*, otherwise it returns -1.

emarks

See a C/C++ run-time library reference for more information.

Delphi:

You can use format() function. Please read Object Pascal (Delphi) language reference for more information on that function.

e also: <u>FSF.snprintf</u>

scanf

in | FarStandardFunctions

The **FSF.sscanf** function allows to read formatted data from a string.

```
int WINAPI sscanf(
   const char *Buffer,
   const char *Format,
   [address, ...]
);
```

arameters

Buffer

Buffer that will be scanned.

Format

Format string

address

Series of arguments that receive data in accordance with the formatted string.

eturn value

If the function succeeds the return value is the number of fields successfully converted and assigned. -1 is returned if the number of format specifiers is greater than the number of fields in the scanned string. On error return value is 0.

emarks

See a C/C++ run-time library reference for more information.

rim

in | FarStandardFunctions

The **FSF.Trim** function removes all leading and trailing whitespace from a string.

```
char* WINAPI Trim(
   char *Str
);
```

arameters

Str

The string from which you want to remove the leading and trailing whitespace.

eturn value

On return **Str** contains the string with all leading and trailing whitespace removed.

e also: <u>FSF.LTrim</u>, <u>FSF.RTrim</u>

runcPathStr

in | FarStandardFunctions

The **FSF.TruncPathStr** function truncates a given path to specified length and, if needed, inserts into it an ellipsis to indicate the place of truncation.

```
char* WINAPI TruncPathStr(
   char *Str,
   int MaxLength
);
```

arameters

Str

Path that you want to truncate. The result will be placed into the same buffer.

MaxLength

Specifies the length to truncate the path to.

eturn value

On return **Str** contains a pointer to the truncated path.

e also: <u>FSF.TruncStr</u>

runcStr

in | FarStandardFunctions

The **FSF.TruncStr** function truncates a given string to the specified length and, if needed, inserts into its beginning an ellipsis instead of the truncated part.

```
char* WINAPI TruncStr(
   char *Str,
   int MaxLength
);
```

arameters

Str

String that you want to truncate. The result will be placed in the same buffer.

MaxLength

Specifies the length to truncate the string to.

eturn value

On return **Str** contains a pointer to truncated string.

emarks

xample

e also: <u>FSF.TruncPathStr</u>

nquote

in | FarStandardFunctions

The **FSF.Unquote** function removes all double quotes from a null-terminated string.

```
void WINAPI Unquote(
   char *Str
);
```

arameters

Str

The string from which you want quotes to be removed. The result will be placed into the same buffer.

eturn value

None.

emarks

In versions of FAR starting with 1.70 beta 1 and up to 1.70 beta 3, this function deleted <u>only leading and trailing</u> quotation marks.

Lat in | FarStandardFunctions

The **FSF.XLat** function is used to transliterate a string portion from one character set (for example Russian) to another character set (for example Latin).

```
char* WINAPI XLat(
   char *Line,
   int StartPos,
   int EndPos,
   const struct CharTableSet *TableSet,
   DOWRD Flags
);
```

arameters

Line

Pointer to a string a portion of which you want to transliterate.

StartPos

Starting position of the portion you want to transliterate.

EndPos

End position of the portion you want to transliterate.

TableSet

If it is necessary to convert a string to OEM code page before transliteration and then back, this field can contain a pointer to a <u>CharTableSet</u> structure. This field can also accept a NULL value.

Flags

Can be a combination of the following flags (XLATMODE enum):

Flag	Description
XLAT_SWITCHKEYBLAYOUT	Switches the keyboard layout after the transliteration.
	Attention! This function doesn't support switching the keyboard layout under Windows 95/98/Me.
XLAT_SWITCHKEYBBEEP	Sounds a beep after keyboard layout switching (works in conjunction with

eturn value

This function returns a pointer to the transliterated string.

eneral purpose structures

in | structures

Structure	Description
<u>ActlEjectMedia</u>	Eject media
<u>ActlKeyMacro</u>	Macro-oriented operations
<u>CharTableSet</u>	Character table
<u>CmdLineSelect</u>	Command-line text selection/deselection
<u>FarMenuItem</u>	Menu item
<u>FarSetColors</u>	FAR Manager color scheme manipulations
FarStandardFunctions	Useful functions from Far.exe
<u>KeySequence</u>	Description of a key code sequence
<u>OpenPluginInfo</u>	Information about the current plugin instance
PluginInfo	Information about a plugin module
PluginStartupInfo	Various pieces of important plugin information
<u>WindowInfo</u>	Information about the FAR Manager window

e also:

Exported functions, Service functions, Dialog API, Archive support, Addons, Delphi structures, Win32 structures

ctlEjectMedia

in | structures

The **ActlEjectMedia** structure is used in the <u>AdvControl</u> function to eject the medium from a removable drive (CD-ROM/USB/SUBST).

```
struct ActlEjectMedia {
   DWORD Letter;
   DWORD Flags;
};
```

lements

Letter

Drive letter of the removable drive.

Flags

Combination of the following flags (FAREJECTMEDIAFLAGS enum):

Flag	Description
EJECT_NO_MESSAGE	suppress error message display
EJECT_LOAD_MEDIA	attempt to "load/close" device (works only for CD- ROM drives, doesn't work under Windows 95/98/Me)

e also: <u>Structures</u> | <u>TActlEjectMedia</u>

ctlKeyMacro

in | structures

The **ActlKeyMacro** structure is used in the <u>AdvControl</u> function for operations with macro-commands.

```
struct ActlKeyMacro {
    int Command;
    union{
        struct {
            char *SequenceText;
            DWORD Flags;
        } PlainText;
        DWORD Reserved[3];
    } Param;
};
```

emnts

Command

One of the following commands (FARMACROCOMMAND enum):

Command	Description	
MCMD_LOADALL	Read all macros from the registry in Previous values are erased.	to FAR memory.
MCMD_POSTMACROSTRING	Pass a macro in text form to FAR (in as macros are stored in the registry). The <u>AdvControl</u> function returns TF is analyzed and placed into the queut start running when FAR gets control returned if the macro contains any e	n the same format RUE if the macro le (the macro will I). FALSE is rror.
MCMD_SAVEALL	Forces FAR to immediately save all memory to the registry.	macros from
MCMD_GETSTATE	Get macro execution status. Returns one of the following values FARMACROSTATE):	(enum
	Value	Description
	MACROSTATE_NOMACRO	no macro is being executed
	MACROSTATE_EXECUTING	a macro is being

	executed without sending key strokes to plugins
MACROSTATE_EXECUTING_COMMON	a macro is being executed; key strokes are sent to plugins
MACROSTATE_RECORDING	a macro is being recorded without sending key strokes to plugins
MACROSTATE_RECORDING_COMMON	a macro is being recorded; key strokes are sent to plugins
Param is ignored. The value is return AdvControl.	ned by

Param.PlainText.SequenceText

Pointer to a zero-terminated string containing a macro sequence in text form. OEM-encoding should be used to store macros. This member is used in the **MCMD_POSTMACROSTRING** command.

Param.PlainText.Flags

Combination of the following macro execution flags (FARKEYSEQUENCEFLAGS enum):

Flag	Description	
KSFLAGS_DISABLEOUTPUT	Disable screen output during macro playback.	
KSFLAGS_NOSENDKEYSTOPLUGINS	Don't send keystrokes to editor plugins (plugins, that export <u>ProcessEditorInput</u> function).	
KSFLAGS_REG_MULTI_SZ	The Param.PlainText.SequenceText parameter is represented in the REG_MULTI_SZ format. REG_MULTI_SZ in the registry:	
	line 1\x00 line 2\x00 line N\x00	

This member is used in the **MCMD_POSTMACROSTRING** command.

Reserved

Reserved for future use.

emarks

- 1. The **MCMD_LOADALL** and **MCMD_SAVEALL** commands won't execute during macro recording or playback.
- 2. The **KSFLAGS_REG_MULTI_SZ** flag can be discarded, if *Param.PlainText.SequenceText* contains '\n' instead of 0x00.

xample

MCMD_POSTMACROSTRING usage in FARCmds plugin:

```
command.Command=MCMD_POSTMACROSTRING;
command.Param.PlainText.SequenceText=(char *)malloc(s
if(command.Param.PlainText.SequenceText)
{
    command.Param.PlainText.Flags=KSFLAGS_DISABLEOUTPUT
    strcpy(command.Param.PlainText.SequenceText,pCmd);
    Info.AdvControl(Info.ModuleNumber,ACTL_KEYMACRO,&cc
    free(command.Param.PlainText.SequenceText);
}
```

MCMD_LOADALL usage in FARCmds plugin:

command.Command=MCMD_LOADALL; Info.AdvControl(Info.ModuleNumber,ACTL_KEYMACRO,&comm

MCMD_SAVEALL usage in FARCmds plugin:

command.Command=MCMD_SAVEALL; Info.AdvControl(Info.ModuleNumber,ACTL_KEYMACRO,&comm

e also:

Structures | KeySequence | TActlKeyMacro

harTableSet

in | <u>structures</u>

The **CharTableSet** structure contains a set of arrays describing a FAR character table. This structure is used by the <u>CharTable</u> function.

```
struct CharTableSet
{
    unsigned char DecodeTable[256];
    unsigned char EncodeTable[256];
    unsigned char UpperTable[256];
    unsigned char LowerTable[256];
    char TableName[128];
};
```

emnts

DecodeTable

Table to decode the given codepage to the DOS (OEM) codepage.

EncodeTable

Table to encode from DOS (OEM) codepage to the given codepage.

UpperTable

Lowercase to uppercase conversion table.

LowerTable

Uppercase to lowercase conversion table.

TableName

Name of the character table.

e also: <u>Structures</u> | <u>TCharTableSet</u>

ARINT64

in | structures

The FARINT64 structure is used to hold a 64 bit integer value.

```
typedef union
{
    ___int64 i64;
    struct
    {
        DWORD LowPart;
        LONG HighPart;
    } Part;
} FARINT64;
```

e also:

mdLineSelect

in | structures

The **CmdLineSelect** structure is used in the <u>Control</u> function for text selection/deselection on the FAR command line, or getting selection information thereof.

```
struct CmdLineSelect
{
    int SelStart;
    int SelEnd;
};
```

emnts

SelStart

Selection start position.

SelEnd

Selection end position.

emarks

To clear the selection, set SelStart and SelEnd to -1.

e also:

<u>Structures</u> | <u>TCmdLineSelect</u> | <u>FCTL_GETCMDLINESELECTION</u> | <u>FCTL_SETCMDLINESELECTION</u>.

arMenultem

in | <u>structures</u>

The **FarMenuItem** structure describes a single menu item. An array of these structures is passed to the <u>Menu</u> function to show a menu.

```
struct FarMenuItem
{
    char Text[128];
    int Selected;
    int Checked;
    int Separator;
};
```

lements

Text

Item text.

Selected

Item selection flag. There must be only one item for which **Selected** is equal to 1.

Checked

If nonzero, a selection mark is displayed before the item text. If **Checked** is 1, the standard mark is displayed, otherwise the value of *Checked* is used as the mark character.

Separator

If nonzero, the menu item is displayed as a separator line. The other fields are ignored in this case.

emarks

As the *FarMenuItem.Text* field is large, direct initialization of an array of **FarMenuItem** structures can significantly increase plugin size. To prevent this, the <u>InitMenuItem</u> non-standard structure can be used.

e also:

structures | InitMenuItem | TFarMenuItem | FarMenuItemEx

arMenultemEx

in | structures | Menu

The **FarMenuItemEx** structure describes a single menu item. An array of structures of this type is passed to the <u>Menu</u> function. In order to use the **FarMenuItemEx** structure the <u>FMENU_USEEXT</u> flag should be set when the **Menu** function is called.

```
struct FarMenuItemEx
{
    DWORD Flags;
    union {
        char Text[128];
        const char *TextPtr;
    } Text;
    DWORD AccelKey;
    DWORD Reserved;
    DWORD_PTR UserData;
};
```

lements

Flags

Combination of the following values (the FARMENUFLAGS enumeration):

Flag	Description
MIF_SELECTED	Denotes a selected menu item. Only one item can be selected.
MIF_CHECKED	Denotes a checked menu item. Check sign will be shown near the item.
MIF_SEPARATOR	The menu item is shown as delimiter. Unlike in the <u>FarMenuItem</u> structure, the delimiter can contain text. The text is center-aligned and is not selectable.
MIF_DISABLE	Denotes a disabled menu item.
MIF_GRAYED	If this flag is set, the menu item is shown, but cannot be selected.
MIF_HIDDEN	If this flag is set, the menu item is not shown.
MIF_USETEXTPTR	The menu item uses the <i>Text.TextPtr</i> field.

Text.Text

The text of the menu item.

Text.TextPtr

A pointer to the menu item text. Used in conjunction with the **MIF_USETEXTPTR** flag, if the *Text.Text* array has insufficient size or a string from a <u>language file</u> is used, for example:

struct FarMenuItemEx Item; Item.Flags=MIF_USETEXTPTR; Item.Text.TextPtr=Info.GetMsg(Info.ModuleNumber,MFo

AccelKey

The FAR Manager key code which will be used to activate the menu item. Example: in the "Commands" menu pressing Ctrl-O is used to the select the "Panels On/Off" menu item. As opposed to the *BreakKeys* parameter of the Menu function, the *AccelKey* field is analogous to the hotkey.

Reserved

Reserved for future use, should be set to 0.

UserData

User data associated with the menu item. FAR Manager does not use this field.

emarks

- 1. In the low order word of the *Flags* a field character code can be specified which will be displayed as the check mark. In this case, FAR will automatically set the **MIF_CHECKED** flag for this menu item.
- 2. If the low order word of the *Flags* field is 0 and the **MIF_CHECKED** flag is set, or if the low order word of the *Flags* field is 1, then the default check mark will be displayed: the character with code 0FBh (\checkmark).
- 3. If the **MIF_SEPARATOR** flag is used without text, the *Text.Text* or *Text.TextPtr* fields should be initialized to 0, for example:

```
struct FarMenuItemEx Item;
Item.Flags=MIF_SEPARATOR;
Item.Text.Text[0]=0;
```

or:

```
struct FarMenuItemEx Item;
```

Item.Flags=MIF_SEPARATOR|MIF_USETEXTPTR; Item.Text.TextPtr=NULL;

e also:

structures | FarMenuItem | TFarMenuItemEx

arSetColors

in | structures

The **FarSetColors** structure is used in the <u>AdvControl</u> function to change the color palette of FAR Manager.

```
struct FarSetColors {
   DWORD Flags;
   int StartIndex;
   int ColorItem;
   LPBYTE Colors;
};
```

ements

Flags

Can contain a combination of the following values (the FARCOLORFLAGS enumeration):

Flag	Description
FCLR_REDRAW	Redraw the screen after the color scheme has been changed.

StartIndex

Start index of the color scheme range that should be changed.

ColorItem

Number of the colors to be changed.

Colors

Points to the byte array containing the new <u>color attributes</u>.

emarks

e also:

structures | ACTL_SETARRAYCOLOR | TFarSetColors

IfoPanelLine

in | structures

The **InfoPanelLine** structure describes a single line n the information panel. An array of **InfoPanelLine** structures is passed to FAR by the <u>GetOpenPluginInfo</u> function.

```
struct InfoPanelLine
{
    char Text[80];
    char Data[80];
    int Separator;
};
```

lements

Text

Parameter header - left-aligned text displayed using the regular text color (<u>COL_PANELTEXT</u>).

Data

Parameter data - right-aligned text displayed using the selected text color (<u>COL_PANELINFOTEXT</u>)

Separator

If non-zero, a separator is displayed. *Text* is used as the separator header, *Data* is ignored.

e also:

structures | TInfoPanelLine

eyBarTitles

in | <u>structures</u>

The **KeyBarTitles** structure is used to redefine the function key labels in the key bar. An array of these structures can be passed to FAR in the <u>GetOpenPluginInfo</u> function.

```
struct KeyBarTitles
{
    char *Titles[12];
    char *CtrlTitles[12];
    char *AltTitles[12];
    char *ShiftTitles[12];
    // FAR Manager >= 1.70
    char *CtrlShiftTitles[12];
    char *AltShiftTitles[12];
    char *CtrlAltTitles[12];
};
```

emnts

```
Titles, CtrlTitles, AltTitles, ShiftTitles
```

Contain addresses of new key bar labels. *CtrlTitles*, *AltTitles* and *ShiftTitles* are used when <Ctrl>, <Alt> or <Shift> is pressed. If it is desired to leave some standard FAR labels unaltered, set the corresponding addresses to NULL.

```
CtrlShiftTitles, AltShiftTitles, CtrlAltTitles
```

Contain addresses of new key bar titles. *CtrlShiftTitles*, *AltShiftTitles* and *CtrlAltTitles* are used when <CtrlShift>, <AltShift> or <CtrlAlt> is pressed. Use NULL as above. *Available in FAR versions after 1.70*

e also: ECTL_SETKEYBAR | structures | TKeyBarTitles

eySequence

in | structures

The **KeySequence** structure is used in the <u>AdvControl</u> function to pass a sequence of key codes to FAR.

```
struct KeySequence {
   DWORD Flags;
   int Count;
   DWORD *Sequence;
};
```

lements

Flags

May be a combinaiton of the following flags (the FARKEYSEQUENCEFLAGS enumeration):

Flag	Description
KSFLAGS_DISABLEOUTPUT	Do not display the results of processing each key on the screen.
KSFLAGS_NOSENDKEYSTOPLUGINS	Do not pass keys to editor plugins (plugins, that export the <u>ProcessEditorInput</u> function).

Count

Count of key codes in the **Sequence** array.

Sequence

Points to an array of FAR key codes.

emarks

e also: <u>structures</u> | <u>ActlKeyMacro</u> | <u>TKeySequence</u>

penPluginInfo

in | <u>structures</u>

The **OpenPluginInfo** structure describes a plugin instance to FAR.

```
struct OpenPluginInfo
{
  int StructSize;
  DWORD Flags;
  const char *HostFile;
  const char *CurDir;
  const char *Format;
  const char *PanelTitle;
  const struct InfoPanelLine *InfoLines;
  int InfoLinesNumber;
  const char * const *DescrFiles;
  int DescrFilesNumber;
  const struct PanelMode *PanelModesArray;
  int PanelModesNumber;
  int StartPanelMode;
  int StartSortMode;
  int StartSortOrder;
  const struct KeyBarTitles *KeyBar;
  const char *ShortcutData;
  long Reserverd;
};
```

lements

StructSize

```
This field should contain size of the OpenPluginInfo structure:
C: Opi.StructSize = sizeof (struct OpenPluginInfo);
Pascal: Opi.StructSize := SizeOf(Info);
```

Flags

A combination of the following values (the OPENPLUGININFO_FLAGS enumeration):

Flag	Description
OPIF_USEFILTER	Use filter in the plugin panel.

OPIF_USESORTGROUPS	Use sort groups in the plugin panel.
OPIF_USEHIGHLIGHTING	Use file highlighting in the plugin panel.
OPIF_ADDDOTS	Add "" item automatically if it is absent.
OPIF_RAWSELECTION	Folders may be selected regardless of FAR settings.
OPIF_REALNAMES	Turns on the standard FAR file processing mechanism if requested operation is not supported by the plugin. If this flag is set, the items on the plugin panel should be real file names.
OPIF_SHOWNAMESONLY	Show file names without paths by default.
OPIF_SHOWRIGHTALIGNNAMES	Show file names right-aligned by default in all panel display modes.
OPIF_SHOWPRESERVECASE	Show file names using original case regardless of FAR settings.
OPIF_FINDFOLDERS	Apply "Find file" command for folders. The OPIF_FINDFOLDERS flag has no effect since FAR Manager 1.70 beta 4.
OPIF_COMPAREFATTIME	Convert timestamps to FAT format for the <i>Compare folders</i> operation. Set this flag if the plugin file system doesn't provide the time accuracy necessary for standard comparison operations.
OPIF_EXTERNALGET OPIF_EXTERNALPUT OPIF_EXTERNALDELETE OPIF_EXTERNALMKDIR	These flags can be used with OPIF_REALNAMES only. Forces usage of corresponding internal FAR functions, even if plugin exports such function.
OPIF_USEATTRHIGHLIGHTING	FAR Manager 1.70 build #963 and below: use attribute-based file highlighting. All file highlighting templates, except * and * . * will be ignored.FAR Manager 1.70 build #964 and above: use attributes only for file highlighting. File names will be ignored. Color is chosen from file color groups, which have templates excluded from analysis (i.e. option "[]] Match file mask(s)" in file highlighting setup dialog is off).

HostFile

File name on emulated file system. If plugin doesn't emulate a file system based on files, set this field to NULL.

CurDir

Current directory of plugin. If plugin returns empty string here, FAR will close plugin automatically if ENTER is pressed on ".." item.

Format

Plugin's format name. This is shown in the file copy dialog.

PanelTitle

Plugin panel header.

InfoLines

Pointer to an array of <u>InfoPanelLine</u> structures. Each structure describes one line in the information panel. If no plugin-dependent information needs to be shown in the information panel, set this field to NULL.

InfoLinesNumber

Number of structures in InfoPanelLine.

DescrFiles

Pointer to an array of pointers to strings with description file names. FAR tries to read these files (using the <u>GetFiles</u> function) when descriptions are shown and refresh them after file processing, if the **PPIF_PROCESSDESCR** flag in the <u>PluginPanelItem</u> structure was set. Depending on the plugin type, description processing can take significant time. If you don't need this functionality, set the field to NULL.

DescrFilesNumber

Number of description file names.

PanelModesArray

Pointer to an array of <u>PanelMode</u> structures. Panel display mode settings can be redefined using this field. The first structure describes display mode number 0, the second - number 1, etc. If new panel display modes are not required, set the field to NULL.

PanelModesNumber

Number of **<u>PanelMode</u>** structures.

StartPanelMode

The panel display mode to set on panel creation. Must be in the form '**0**'+ **<view mode number>**. For example, '1' (0x31) will set Brief view mode. If you don't want to change panel display mode at plugin startup, set the field

to 0.

StartSortMode

The sort mode to set on panel creation. One of the following values can be specified: SM_UNSORTED, SM_NAME, SM_EXT, SM_MTIME, SM_CTIME, SM_ATIME, SM_SIZE, SM_DESCR, SM_OWNER, SM_COMPRESSEDSIZE, SM_NUMLINKS. If you don't want to change sort mode at plugin startup, set the field to SM_DEFAULT or 0.

StartSortOrder

If *StartSortMode* is specified, this field must be used to set sort direction: 0 for ascending, 1 - for descedning.

KeyBar

Pointer to the <u>KeyBarTitles</u> structure. Function key labels are redefined using this field. Set to NULL if not required.

ShortcutData

Pointer to a null-terminated string, which describes the current state of the plugin. The length of string should be less than or equal to

MAXSIZE_SHORTCUTDATA. This string is passed to the <u>OpenPlugin</u> function, when the plugin is activated by the link to folder command. For example, an FTP client can place the server host name, login and password here. The current directory is not required as FAR will restore this itself. If no additional information is required for activation of links to a folder, set this field to NULL.

Reserverd

Reserved for future use, set to 0.

emarks

- 1. All data, passed through this structure should be valid after return from the <u>GetOpenPluginInfo</u> function. This means, for example, that pointers to the stack are not allowed, use static or global variables instead.
- 2. When this structure is passed to a plugin's <u>GetOpenPluginInfo</u> function, it can be assumed to be cleared to zeros.

anelInfo

in | structures

The **PanelInfo** structure contains information about a FAR panel. Use the <u>Control</u> function to populate this structure.

```
struct PanelInfo
{
  int PanelType;
  int Plugin;
  RECT PanelRect;
  struct PluginPanelItem *PanelItems;
  int ItemsNumber;
  struct PluginPanelItem *SelectedItems;
  int SelectedItemsNumber;
  int CurrentItem;
  int TopPanelItem;
  int Visible;
  int Focus;
  int ViewMode;
  char ColumnTypes[80];
  char ColumnWidths[80];
  char CurDir[NM];
  int ShortNames;
  int SortMode;
  DWORD Flags;
  DWORD Reserved;
};
```

emnts

PanelType

May be (the PANELINFOTYPE enumeration):

Туре	Description
PTYPE_FILEPANEL	Regular file panel
PTYPE_TREEPANEL	Tree panel
PTYPE_QVIEWPANEL	Quick view panel
PTYPE_INFOPANEL	Information panel

Plugin

TRUE if panel is supported by a plugin.

PanelRect

Panel geometry, the <u>RECT</u> structure.

PanelItems

Pointer to an array of all panel items (see <u>PluginPanelItem</u>). Plugins shouldn't change items (except those marked with <u>PPIF_SELECTED</u>) or free memory allocated for this array. Data is valid until return from the plugin's exported function, which retrieved it via a call to the

Control(FCTL_GET[ANOTHER]PANELINFO, ...) function, or until the next call to the <u>Control</u> function.

ItemsNumber

Number of items in the *PanelItems* array.

SelectedItems

Pointer to the array of selected panel items. (see <u>PluginPanelItem</u>). Plugin shouldn't change elements (except those marked with <u>PPIF_SELECTED</u>) or free memory allocated for this array. Data is valid until return from the plugin's exported function, which retrieved it via a call to the Control(FCTL_GET[ANOTHER]PANELINFO, ...) function, or until the next call to the <u>Control</u> function.

SelectedItemsNumber

Number of items in the *SelectedItems* array. When there is no selection, **SelectedItemsNumber** is equal to 1. When there is no selection and current element is ".." then **SelectedItemsNumber** is equal to 0.

CurrentItem

Index of the current item in the *PanelItems* array. Ensure that *ItemsNumber* > 0 before using **CurrentItem**.

TopPanelItem

Panel item which is at the first visible position in the panel.

Visible

If non-zero, panel is visible.

Focus

If non-zero, panel is active.

ViewMode

Number of panel view mode.

ColumnTypes

Null-terminated string, which describes <u>column types</u>. Column types are encoded with one or several letters, separated by commas, for example: **"N, SC, D, T"**.

ColumnWidths

Null-terminated string, which describes <u>columns width</u>.

CurDir

Current directory for panel. If *Type* = **PTYPE_TREEPANEL**, *CurDir* contains currently selected directory in panel.

ShortNames

Non-zero, if short file names mode is on.

SortMode

Panel sort mode. Can be SM_DEFAULT, SM_UNSORTED, SM_NAME, SM_EXT, SM_MTIME, SM_CTIME, SM_ATIME, SM_SIZE, SM_DESCR, SM_OWNER, SM_COMPRESSEDSIZE, SM_NUMLINKS.

Flags

Additional flags. Can be a combination of the following values (the PANELINFOFLAGS enumeration):

Flag	Description
PFLAGS_SHOWHIDDEN	Hidden and system files are displayed.
PFLAGS_HIGHLIGHT	File highlighting is used.
PFLAGS_REVERSESORTORDER	Descending sort is used.
PFLAGS_USESORTGROUPS	Sort groups are used.
PFLAGS_SELECTEDFIRST	Show selected files first.
PFLAGS_REALNAMES	Plugin panel items are shown with real file names (see also <u>OPIF_REALNAMES</u>).
PFLAGS_NUMERICSORT	Numeric sort is used.
PFLAGS_PANELLEFT	Left panel.

Reserved

Reserved for future use.

emarks

Additional information about panel can be retrieved using the <u>AdvControl</u> function (the <u>ACTL_GETPANELSETTINGS</u> command)

e also: <u>structures</u> | <u>TPanelInfo</u>

anelMode

<u>in | structures</u>

The **PanelMode** structure describes one panel view mode. An array of these structures should be passed to the <u>GetOpenPluginInfo</u> function.

```
struct PanelMode
{
    char *ColumnTypes;
    char *ColumnWidths;
    char **ColumnTitles;
    int FullScreen;
    int DetailedStatus;
    int AlignExtensions;
    int CaseConversion;
    char *StatusColumnTypes;
    char *StatusColumnWidths;
    DWORD Reserved[2];
};
```

emnts

ColumnTypes

Text string which describes <u>column types</u>. Column types are encoded by one or more letters separated by commas: "**N**, **SC**, **D**, **T**". To use standard Far panel view modes, set this field to NULL.

ColumnWidths

Text string which gives the <u>columns width</u>.

ColumnTitles

Pointer to an array of strings which specifies the column labels. To use standard column names, set this value to NULL.

FullScreen

If TRUE - resize panel to fill the entire window (instead of half).

DetailedStatus

If TRUE, the status line will display *name*, *size*, *date* and *time* of the file, if *StatusColumnTypes*=NULL and *StatusColumnWidths*=NULL. Otherwise, the
status line will only display the file name.

AlignExtensions

TRUE - align file extensions.

CaseConversion

FALSE - to preserve file names.

StatusColumnTypes

As *ColumnTypes*, but applies to the status line. To use the standard status line, set *StatusColumnTypes* to NULL.

StatusColumnWidths

As *ColumnWidths*, but applies to the status line.

Reserved

Reserved for future use. Set to 0.

e also: <u>structures</u> | <u>TPanelMode</u>

anelRedrawInfo

in | structures

The **PanelRedrawInfo** structure can be used to determine the new cursor position after a panel redraw caused by the <u>Control</u> function.

```
struct PanelRedrawInfo
{
    int CurrentItem;
    int TopPanelItem;
};
```

emnts

CurrentItem

Zero-based index of the current panel item.

TopPanelItem

Zero-based index of the first visible panel item.

e also:

structures | TPanelRedrawInfo

luginInfo

<u>in | structures</u>

The **PluginInfo** structure describes a FAR Manager plugin.

```
struct PluginInfo
{
    int StructSize;
    DWORD Flags;
    const char * const *DiskMenuStrings;
    int *DiskMenuNumbers;
    int DiskMenuStringsNumber;
    const char * const *PluginMenuStrings;
    int PluginMenuStringsNumber;
    const char * const *PluginConfigStrings;
    int PluginConfigStringsNumber;
    const char *CommandPrefix;
    DWORD Reserved;
};
```

lements

StructSize

This field should contain the size of the **PluginInfo** structure.

Flags

A combination of the following values (the PLUGIN_FLAGS enumeration).

Flag	Description
PF_PRELOAD	Disables plugin parameters caching and forces FAR to always load the plugin at startup. Must be specified if it is necessary to change items in "Disks", "Plugins" or "Plugins configuration" menus dynamically. This flag decreases efficiency of memory usage.
PF_DISABLEPANELS	Do not show the plugin in the "Plugin commands" menu called from panels.
PF_EDITOR	Show the plugin in the "Plugin commands" menu called from the FAR editor.
PF_VIEWER	Show the plugin in the "Plugin commands" menu called from the FAR viewer.

PF_DIALOG	Show the plugin in the "Plugin commands" menu called from the FAR dialog.
PF_FULLCMDLINE	Forces FAR to pass the full command line (with the prefix CommandPrefix) to the plugin. Use this flag when a plugin can handle multiple command line prefixes.

DiskMenuStrings

If the plugin adds items to the *Disks menu*, this field is set to the address of an array with pointers to menu items, otherwise set it to NULL.

DiskMenuNumbers

Preferred hotkey numbers for the items added to the *Disks menu*. To have FAR autoassign them, either set **DiskMenuNumbers** to NULL or set preferred numbers to 0.

If specified number is already in use by another plugin, FAR reassigns it itself.

DiskMenuStringsNumber

Number of items to be added to the Disks menu.

PluginMenuStrings

Similar to *DiskMenuStrings*, but items are added to the "**Plugin commands**" menu.

PluginMenuStringsNumber

Number of items to be added to the "**Plugin commands**" menu.

PluginConfigStrings

Similar to *DiskMenuStrings*, but items are added to the "**Plugins configuration**" menu.

PluginConfigStringsNumber

Numer of items to be added to the "Plugins configuration" menu.

CommandPrefix

This parameter can contain a string of command line prefixes that will be used to intercept FAR commands. For example, if the string passed in this field is **ftp** and the user enters anything beginning with **ftp:** in the command line, <u>OpenPlugin</u> with <u>OPEN_COMMANDLINE</u> will be called.

To define more than one prefix, the plugin must separate the prefixes with a colon. For example, if the plugin needs to process the edit:, goto: and view: prefixes, the string passed in this field should be to:

```
CommandPrefix="edit:view:goto";
```

If the plugin does not process FAR commands, set this parameter to NULL.

Reserved

Reserved for future use, should be set to NULL.

emarks

- 1. All data passed in this structure must be valid after returning from <u>GetPluginInfo</u>, therfore pointers to the stack are illegal, use static or global variables instead.
- 2. The structure passed to the <u>GetPluginInfo</u> function, is preinitialized to zeros.

e also: <u>structures</u> | <u>TPluginInfo</u>

luginPanelItem

<u>in | structures</u>

The **PluginPanelItem** structure describes a single item in a file system emulated by a plugin. It is used both to return information about the plugin file system and to pass a list of files to process to the plugin.

▲ Attention! The size of the PluginPanelItem structure should be 366 bytes.

```
struct PluginPanelItem
{
#ifndef _FAR_USE_WIN32_FIND_DATA
  struct FAR FIND DATA
                        FindData;
#else
  WIN32_FIND_DATA FindData;
#endif
  DWORD PackSizeHigh;
  DWORD PackSize;
  DWORD Flags;
  DWORD NumberOfLinks;
  char *Description;
  char *Owner;
  char **CustomColumnData;
  int CustomColumnNumber;
  DWORD_PTR UserData;
  DWORD CRC32;
  DWORD Reserved[2];
};
```

emnts

FindData

The **FindData** field contains many file parameters. See the description of the <u>WIN32_FIND_DATA</u> structure for detailed information.

PackSizeHigh

Contains the high-order 4 bytes of the file's packed size (in bytes). Currently unused.

PackSize

Contains the low-order 4 bytes of the file's packed size (in bytes).

Flags

A combination of the following values (the PLUGINPANELITEMFLAGS enumeration):

Flag	Description
PPIF_PROCESSDESCR	Use FAR's internal description processing. This flag can be set for processed files in the <u>DeleteFiles</u> , <u>GetFiles</u> and <u>PutFiles</u> functions. If set, FAR will update the description file contents using file names returned from the <u>GetOpenPluginInfo</u> function.
PPIF_SELECTED	In <u>Control</u> functions FCTL_GETPANELINFO, FCTL_GETANOTHERPANELINFO, FCTL_SETSELECTION and FCTL_SETANOTHERSELECTION this flag allows to check and set item selection. In <u>PutFiles</u> , <u>GetFiles</u> and <u>ProcessHostFile</u> functions, if an operation has failed, but some of the files were successfully processed, the plugin can remove selection only from the processed files. To perform this, the plugin should clear the PPIF_SELECTED flag in processed items in the <i>PluginPanelItem</i> list passed to the function.
PPIF_USERDATA	If this flag is set, FAR considers the <i>UserData</i> field a pointer to a user data structure. Cf. the description of the <i>UserData</i> field. The low order word of the Flags parameter can be used by a plugin for its own flags.

NumberOfLinks

Number of hard links.

Description

Points to a file description. Plugins can use this field to pass file descriptions to FAR. If not required, set theis field to NULL. If a plugin uses standard FAR description processing and has passed description file names to FAR in the <u>GetOpenPluginInfo</u> function, this field also must be NULL.

Owner

Points to a file owner name. Plugins can use this field to pass file owner names to FAR. If not used, set this field to NULL.

CustomColumnData

Points to an array of string addresses for plugin defined column types. The first string contains data for the **C0** column type, the second - for **C1** and so on. Up to 10 additional column types from **C0** to **C9** can be defined. If not used, set this field to NULL.

CustomColumnNumber

Number of data strings for additional column types.

UserData

This field can be used by the plugin to store either a 32-bit value or a pointer to a data structure. In the latter case, the first field of this structure must be a 32-bit value containing the structure size and the plugin must set PPIF_USERDATA in the *Flags* field. This allows FAR to copy the structure correctly to FAR internal buffers and later pass it to the plugin in *PluginPanelItem* lists. In the <u>FreeFindData</u> function the plugin must free the memory occupied by this additional structure.

CRC32

A 32-bit CRC (checksum) value. FAR does not use this field.

Reserved

Reserved for future use, should be set to 0.

emarks

- 1. All the data allocated by the plugin for *Description*, *Owner* and *CustomColumnData* members must be released by the plugin. FAR copies these data to its own structures, so it can be released at any time after passing these to FAR in the <u>GetFindData</u> function.
- 2. If the plugin uses the *UserData* field to store information about a file, the plugin writer should check the validity of that field. This is related to the following issue: When FAR calls the <u>GetFiles</u> function from the list of found files, only the *FindData* member of the **PluginPanelItem** structure is filled correctly. All other fields are equal to NULL.

e also: <u>structures</u> | <u>FAR USE WIN32 FIND DATA</u> | <u>FAR FIND DATA</u> | <u>TPluginPanelItem</u>

luginStartupInfo

<u>in | structures</u>

The **PluginStartupInfo** structure is used in the <u>SetStartupInfo</u> function to pass various important information to the plugin.

```
struct PluginStartupInfo
{
  int StructSize;
  char ModuleName[NM];
  int ModuleNumber;
  const char *RootKey;
  FARAPIMENU
                          Menu;
                          Dialog;
  FARAPIDIALOG
  FARAPIMESSAGE
                          Message;
  FARAPIGETMSG
                          GetMsg;
                          Control;
  FARAPICONTROL
  FARAPISAVESCREEN
                          SaveScreen;
  FARAPIRESTORESCREEN
                          RestoreScreen;
  FARAPIGETDIRLIST
                          GetDirList;
  FARAPIGETPLUGINDIRLIST GetPluginDirList;
                          FreeDirList;
  FARAPIFREEDIRLIST
  // FAR >= 1.50
  FARAPIVIEWER
                          Viewer;
                          Editor;
  FARAPIEDITOR
  FARAPICMPNAME
                          CmpName;
  // FAR >= 1.52
                          CharTable;
  FARAPICHARTABLE
  FARAPITEXT
                          Text;
  // FAR >= 1.60
                          EditorControl;
  FARAPIEDITORCONTROL
  // FAR >= 1.70
  FARSTANDARDFUNCTIONS
                         *FSF;
```

```
FARAPISHOWHELP
                        ShowHelp;
                        AdvControl;
FARAPIADVCONTROL
FARAPIINPUTBOX
                        InputBox;
FARAPIDIALOGEX
                        DialogEx;
FARAPISENDDLGMESSAGE
                        SendDlgMessage;
FARAPIDEFDLGPROC
                        DefDlgProc;
DWORD PTR
                        Reserved;
FARAPIVIEWERCONTROL
                        ViewerControl;
```

```
};
```

lements

StructSize

Structure size. If new fields are added, this field will allow detection of the version of the structure used.

ModuleName

Full name with path of the plugin module.

ModuleNumber

Number of the plugin module. Passed as parameter in some functions.

RootKey

Registry root key, where plugins can save their parameters. Valid both for **HKEY_CURRENT_USER** and **HKEY_LOCAL_MACHINE**. Usually it is "Software\Far\Plugins", but you must not specify this string directly, because it can be changed or can become optional in future. For example, if FAR is started with '/u user1' command line parameter, *RootKey* will contain "Software\Far\Users\user1\Plugins".

Do not save parameters directly in the *RootKey*, create your own subkey here.

Menu

Address of the <u>Menu</u> function.

Dialog

Address of the **Dialog** function.

Message

Address of the <u>Message</u> function.

GetMsg

Address of the <u>GetMsg</u> function.

Control

Address of the <u>Control</u> function.

SaveScreen

Address of the <u>SaveScreen</u> function.

RestoreScreen

Address of the <u>RestoreScreen</u> function.

GetDirList

Address of the GetDirList function.

GetPluginDirList

Address of the <u>GetPluginDirList</u> function.

FreeDirList

Address of the **<u>FreeDirList</u>** function.

Viewer

Address of the <u>Viewer</u> function.

Editor

Address of the Editor function.

CmpName

Address of the <u>CmpName</u> function.

CharTable

Address of the <u>CharTable</u> function.

Text

Address of the <u>Text</u> function.

EditorControl

Address of the <u>EditorControl</u> function.

FSF

Pointer to the <u>FarStandardFunctions</u> structure, which contains addresses of useful functions from far.exe.

ShowHelp

Address of the <u>ShowHelp</u>.

AdvControl

Address of the <u>AdvControl</u>.

InputBox

Address of the <u>InputBox</u>.

DialogEx

Address of the **DialogEx**.

SendDlgMessage

Address of the <u>SendDlgMessage</u>.

DefDlgProc

Address of the <u>DefDlgProc</u>.

DefDlgProc

Address of the <u>ViewerControl</u>.

emarks

FAR Manager versions:

- below 1.50 (with plugins support) do not contain fields after FreeDirList;
- below 1.52 do not contain fields after <u>CmpName</u>;
- below 1.60 do not contain fields after <u>Text;</u>
- below 1.65 do not contain fields after EditorControl

So, if you intend to use <u>Viewer</u> or <u>Editor</u> functions or any other function after these, you should first check the *StructSize* field, to determine if the required function is present in the running FAR version.

e also: structures | TPluginStartupInfo

/indowInfo

in | structures

The **WindowInfo** structure contains information about one FAR Manager window. A plugin can retrieve this information using the <u>ACTL_GETWINDOWINFO</u> command.

```
struct WindowInfo{
    int Pos;
    int Type;
    int Modified;
    int Current;
    char TypeName[64];
    char Name[NM];
};
```

lements

Pos

Zero-based number of the window to retrieve information for. Pos = -1 will return information for the current window.

Туре

Window type. Can be one of the following (the WINDOWINFO_TYPE enumeration):

Туре	Description
WTYPE_PANELS	File panels.
WTYPE_VIEWER	Internal viewer window.
WTYPE_EDITOR	Internal editor window.
WTYPE_DIALOG	Dialog.
WTYPE_VMENU	Menu.
WTYPE_HELP	Help window.

Modified

Modification flag. Can be set only if **Type = WTYPE_EDITOR**.

Current

Nonzero if the window is active.

TypeName

The name of the window type (Panels, View, Edit, Help, VMenu), depends on the current language setting of Far.

Name

Window title. For **WTYPE_VIEWER** and **WTYPE_EDITOR** windows this is a file name. For panels, the name of the currently selected file object. For the help window - full path to the opened HLF file. For menu and dialogs - header.

xample

```
void GetFarWindowInfo()
{
  WindowInfo WInfo;
  int CountWindow;
  int I;
  FILE *Fp;
  if((Fp=fopen("window.log", "a+t")) == NULL)
      return ;
  // request window count
  CountWindow=(int)Info.AdvControl(Info.ModuleNumber,
  fprintf(Fp, "WindowCount=%i\n", CountWindow);
  for ( I=0; I < CountWindow; I++ )</pre>
  {
    WInfo.Pos=I;
    // request window information
    Info.AdvControl(Info.ModuleNumber,ACTL_GETWINDOWI
    // output it
    fprintf(Fp, "Window[%i], Type=%i (%s), File=[%s] C
                I, WInfo. Type, WInfo. TypeName, WInfo. Name
               WInfo.Current,WInfo.Modified);
  }
  fclose(Fp);
```

```
// set window number 1 (i.e. second)
Info.AdvControl(Info.ModuleNumber,ACTL_SETCURRENTWI
}
```

e also:

structures | AdvControl | TWindowInfo

ditor plugin structures

in | structures

Structure	Description
<u>EditorBookMarks</u>	Information about bookmarks in the currently edited file
<u>EditorColor</u>	Information about color regions
<u>EditorConvertPos</u>	Conversion between real and screen positions of the cursor
<u>EditorConvertText</u>	Text conversion between the OEM and the internal Far character set
EditorGetString	Editor line retrieval
<u>EditorInfo</u>	Current Far editor state
<u>EditorSaveFile</u>	Editor file saving
<u>EditorSelect</u>	Text selection/deselection in the editor
EditorSetParameter	Cditor parameter control
<u>EditorSetPosition</u>	Position control in the Far editor
EditorSetString	Change or insert a string in the Far editor

e also:

Exported functions, Service functions, Dialog API, Archive support, Addons, Delphi structures, Win32 structures

ditorBookMarks

in | structures

The **EditorBookMarks** structure is used in the <u>EditorControl</u> function to retrieve information about bookmarks in the current editor (<u>ECTL_GETBOOKMARKS</u> command).

```
struct EditorBookMarks
{
    long *Line;
    long *Cursor;
    long *ScreenLine;
    long *LeftPos;
    DWORD Reserved[4];
};
```

lements

Line

Pointer to an array of line numbers for each bookmark.

Cursor

Pointer to an array of cursor positions for each bookmark.

ScreenLine

Points to an array of line numbers specifying the first line visible on the screen for each bookmark.

LeftPos

Points to an array of positions specifying the leftmost character visible on the screen for each bookmark.

Reserved

Reserved for future use.

emarks

1. Before retrieving information about bookmarks you should determine the length of the arrays **Line**, **Cursor**, **ScreenLine** and **LeftPos** and allocate the required amount of memory for them. The bookmark count is stored in the *BookMarkCount* member of the <u>EditorInfo</u> structure.

2. If a plugin does not need the information stored in one of the arrays, it should set the respective member of the structure to NULL.

e also: <u>Structures</u> | <u>TEditorBookMarks</u>

ditorColor

in | <u>structures</u>

The **EditorColor** structure is used in the <u>EditorControl</u> function to get or set information about color regions in FAR editor.

```
struct EditorColor
{
    int StringNumber;
    int ColorItem;
    int StartPos;
    int EndPos;
    int Color;
};
```

lements

StringNumber

Line number to process or **-1** for current string.

ColorItem

Ordinal number of the color region whose information is to be retrieved. A line consists of a set of segments (regions) all of which can have a different color. Set ColorItem to **0** to retrieve information about the first segment, **1** for the second, and so on.

This field isn't used with the <u>ECTL_ADDCOLOR</u> command. The new color is applied to the whole region between positions StartPos and EndPos, regardless of color regions which existed there before.

StartPos, EndPos

Line region bounds.

Input parameters for the <u>ECTL_ADDCOLOR</u> command and output parameters for the <u>ECTL_GETCOLOR</u> command.

Color

Character color.

Input parameter for the <u>ECTL_ADDCOLOR</u> command and output parameter for the <u>ECTL_GETCOLOR</u> command.

For the <u>ECTL_ADDCOLOR</u> command, if **Color** is set to **0**, **EndPos** is ignored and the command will remove all existing color regions starting at

position **StartPos**. In the latter case, if **StartPos** is **-1**, all color regions for the line will be removed.

Starting with build 1957 FAR highlights the tabulation character to its full length. If it is needed to highlight the tab character as a character of width 1 then in addition to specifying the color for the *Color* member set the **ECF_TAB1** flag (EDITORCOLORFLAGS enum):

Flag	Description
ECF_TAB1	If a tab character is to be found inside a color region, highlight it as a character of width 1.

emarks

Partial deletion of color regions is not very useful, as FAR does not renormalize the segments as new ones are added, this is the programmer's responsibility. Normally, it is necessary to clear all regions in line, and recreate them as normalized --- non-overlapping --- segments (otherwise the regions will accumulate causing increased memory usage).

Using the *Color* parameter, the background color can be set also. Furthermore, regardless of line length, you can set a color region starting from the first visible position and ending at the rightmost; this will change the editor background color. Unfortunately this method won't work for files with a line count less than the editor screen height.

e also: <u>structures</u> | <u>TEditorColor</u>

ditorConvertPos

in | structures

The **EditorConvertPos** structure is used in the <u>EditorControl</u> function to convert a position in the line to a screen position (and vice versa). This is useful for lines which contain tab characters.

```
struct EditorConvertPos
{
    int StringNumber;
    int SrcPos;
    int DestPos;
};
```

lements

StringNumber

Line number or -1 for current line.

SrcPos

Source position.

DestPos

Converted position.

e also:

structures | TEditorConvertPos

ditorConvertText

in | structures

The **EditorConvertText** structure is used in the <u>EditorControl</u> function for text conversion from OEM-encoding to the current FAR editor's codepage (and vice versa).

```
struct EditorConvertText
{
    char *Text;
    int TextLength;
};
```

emnts

Text

Points to text for conversion.

TextLength

Length of text for conversion.

e also: structures | TEditorConvertText

ditorGetString

in | <u>structures</u>

The **EditorGetString** structure is used in the <u>EditorControl</u> function to retrieve a text line from the FAR editor.

```
struct EditorGetString
{
    int StringNumber;
    const char *StringText;
    const char *StringEOL;
    int StringLength;
    int SelStart;
    int SelEnd;
};
```

ements

StringNumber

Zero-based index of the line to retrieve. Can be set to -1 to retrieve the current line (see <u>this article</u> regarding the -1 value).

StringText

Pointer to line data. Cannot be modified. Note, that line data is not a zero-terminated string and can contain ASCII NULL-s.

StringEOL

End-of-line sequence. Can be the empty string, r n n

StringLength

Size of data pointed to by *StringText*.

SelStart

Start position of selection in the line. If line doesn't contain selection this field has a value of -1.

SelEnd

End position of selection in the line. If selection includes the *StringEOL* sequence this field has a value of -1.

e also:

structures | TEditorGetString

ditorInfo

<u>in | structures</u>

The **EditorInfo** structure is used in the <u>EditorControl</u> function to get information about the current FAR editor state.

```
struct EditorInfo
{
  int EditorID;
  const char *FileName;
  int WindowSizeX;
  int WindowSizeY;
  int TotalLines;
  int CurLine;
  int CurPos;
  int CurTabPos;
  int TopScreenLine;
  int LeftPos;
  int Overtype;
  int BlockType;
  int BlockStartLine;
  int AnsiMode;
  int TableNum;
  DWORD Options;
  int TabSize;
  int BookMarkCount;
  DWORD CurState;
  DWORD Reserved[6];
};
```

lements

EditorID

Identifier of the editor instance. Each editor instance has a unique identifier during the lifetime of a FAR session.

FileName

Full path and name of the edited file .

WindowSizeX, WindowSizeY

Width and height of the editor window.

TotalLines

Total number of lines in the edited text.

CurLine

Number of the current line.

CurPos

Cursor position in the current line.

CurTabPos

Cursor screen position in the current line. If the line does not contain tab characters, *CurTabPos* is equal to *CurPos*.

TopScreenLine

Number of the line at the top of the screen.

LeftPos

Position of the left border of the editor window in the edited text.

Overtype

Overtype mode state. 0 - insert mode, 1 - overtype mode.

BlockType

Type of the selected block. One of the following values (the EDITOR_BLOCK_TYPES enum):

Block type	Description
BTYPE_NONE	no selection,
BTYPE_STREAM	stream block
BTYPE_COLUMN	column (rectangular) block.

BlockStartLine

Number of the first line in the selected block.

AnsiMode

ANSI text mode state. This field is nonzero only when no character table is used and the text is in the ANSI codepage.

TableNum

Number of FAR character table currently used in the editor. -1 if no table is used and the text is in OEM format (in this case see AnsiMode). If this field is

not -1, you can pass it to the <u>CharTable</u> function to get the table.

Options

Describes the state of editor options. Can be a combination of the following flags (the EDITOR_OPTIONS enumeration):

Flag	Description
EOPT_EXPANDTABS	The "Expand all tabs to spaces" option is selected. While editing a file convert all entered and existing Tab characters to the corresponding number of spaces. This parameter can be changed using the <u>ECTL_SETPARAM</u> command (<u>ESPT_EXPANDTABS</u>).
EOPT_EXPANDONLYNEWTABS	The "Expand newly entered tabs to spaces" option is selected. While editing a file convert all newly entered Tab characters to the corresponding number of spaces. Existing Tab characters won't be converted. This parameter can be changed using the <u>ECTL SETPARAM (ESPT EXPANDTABS</u>).
EOPT_PERSISTENTBLOCKS	"Persistent blocks" option is on.
EOPT_DELREMOVESBLOCKS	"Del removes blocks" option is on.
EOPT_AUTOINDENT	"Auto indent" option is on. This parameter can be changed using the <u>ECTL_SETPARAM</u> command.
EOPT_AUTODETECTTABLE	"Autodetect character table" option is on.
EOPT_CURSORBEYONDEOL	"Cursor beyond end of line" option is on. This parameter can be changed using the <u>ECTL_SETPARAM</u> command.
EOPT_SAVEFILEPOSITION	"Save file position" option is on. This parameter can be changed using the <u>ECTL_SETPARAM</u> command.

TabSize

Tab size. This parameter can be changed using the <u>ECTL_SETPARAM</u> command.

BookMarkCount

Number of editor bookmarks. To retrieve information about bookmarks, use the <u>ECTL_GETBOOKMARKS</u> command.

CurState

Current state of the text in the editor. Can contain one or more of the following flags (the EDITOR_CURRENTSTATE enumeration):

Flag	Description
ECSTATE_MODIFIED	modified
ECSTATE_SAVED	saved
ECSTATE_LOCKED	locked (Ctrl-L)

Reserved

Reserved for future use.

emarks

To determine current character set in the editor exactly, the following table can be used:

	EditorInfo.TableNum	Editor
DOS	-1	
WIN	-1	
Other	Table number	

For "Other", the <u>CharTable</u> function should be used to retrieve the character table name.

e also:

structures | TEditorInfo

ditorSelect

in | <u>structures</u>

The **EditorSelect** is used in the <u>EditorControl</u> function to select or deselect text in the FAR editor.

```
struct EditorSelect
{
    int BlockType;
    int BlockStartLine;
    int BlockStartPos;
    int BlockWidth;
    int BlockHeight;
};
```

emnts

BlockType

One of the following values (the EDITOR_BLOCK_TYPES enumeration):

Block type	Description
BTYPE_NONE	deselect block
BTYPE_STREAM	select stream block
BTYPE_COLUMN	select column (rectangular) block

If *BlockType* is equal to BTYPE_NONE, other fields of the structure are ignored.

BlockStartLine

First line of the selection. The field can be **-1** - current line will be the start of the block.

BlockStartPos

Start position of the selection. If *BlockStartPos* = -1, text is deselected.

BlockWidth

Block width. Can be negative.

BlockHeight

Block height, should be ≥ 1

emarks

- 1. <u>EditorControl</u> function returns FALSE, if:
 - size of transferred variable is less than EditorSelect structure;
 - *BlockHeight* is less than 1;
 - *BlockStartLine* is greater than number of lines in editor.

xample

Stream block from (X1,Y1) to (X2,Y2) can be selected in this way:

```
es.BlockType=BTYPE_STREAM;
es.BlockStartLine=min(Y2,Y1);
es.BlockStartPos=(Y1 < Y2?X1:X2);</pre>
// small correction if positions are equal
if(X1 == X2)
  es.BlockStartPos+=(Y1 < Y2?1:-1);</pre>
es.BlockHeight=max(Y1,Y2)-min(Y1,Y2)+1;
if(Y1 < Y2)
  es.BlockWidth=X2-X1+1;
else
  es.BlockWidth=X1-X2+1;
if(X1 == X2)
{
  if(Y1 < Y2)
    es.BlockStartPos--;
  else
    es.BlockStartPos++;
}
```

Info.EditorControl(ECTL_SELECT, (void*)&es);

e also:

structures | TEditorSelect

ditorSetParameter

in | structures

The **EditorSetParameter** structure is used in the <u>EditorControl</u> function to change the settings of the current FAR editor.

```
struct EditorSetParameter
{
    int Type;
    union {
        int iParam;
        char *cParam;
        DWORD Reserved1;
    } Param;
    DWORD Flags;
    DWORD Reserved2;
};
```

lements

Туре

Which setting to change. Can have one of the following values (the EDITOR_SETPARAMETER_TYPES enum):

Option	Description
ESPT_AUTOINDENT	Sets the " <i>Auto indent</i> " mode according to iParam =TRUE or FALSE.
ESPT_CHARCODEBASE	 Display format of the current character code in the editor status line. iParam can have one of the following values: 0 - octal (3 characters with leading zeros) 1 - decimal (3 characters with leading spaces) 2 - hexadecimal (2 digits + the character 'h')
ESPT_CHARTABLE	 Sets the current character table in the editor. The value of iParam can be: 1 - OEM 2 - ANSI 3 - character table with the index 0

	• N - character table with the index (N-3)	
	In case of an error, FALSE is returned by <u>EditorControl</u> and the character table is not changed.	
ESPT_CURSORBEYONDEOL	Sets the " <i>Cursor beyond end of line</i> " mode according to iParam =TRUE or FALSE.	
ESPT_EXPANDTABS	Controls the behaviour of tabs to spaces convertion. iParam can be one of the following flags (EXPAND_TABS enum):	
	Flag	Description
	EXPAND_NOTABS	
	EXPAND_ALLTABS	
	EXPAND_NEWTABS	
	Attention! This operation is not reversible; that is, if a file contained tabs instead of spaces, the sequences of spaces in the file will not be converted to tabs when the "Expand tabs to spaces" mode is disabled.	
ESPT_LOCKMODE	Prohibit or allow user to modify the text in the editor (similar to Ctrl-L) according to iParam =TRUE or FALSE.	
ESPT_SAVEFILEPOSITION	Sets the " <i>Save file position</i> " option according to iParam =TRUE or FALSE.	
ESPT_SETWORDDIV	Changes the word delimiter set (using the cParam field) for the current editor instance. If cParam is NULL or the empty string, the default delimiter set - "~!%^&*()+ {}:"<>?`-=\[];',./" - is used. The word delimiter set cannot contain more than 255 characters.	
ESPT_GETWORDDIV	Retrieves the word delimiter set (using the cParam field) for the current editor instance. cParam should point to a buffer of at least 256 characters.	
ESPT_TABSIZE	Changes the tabsize. iParam is the new value - between 1 and 512. If iParam is out of range, tabsize is set to 8.	

iParam

Contains a numeric setting; see the description of the individual settings. *cParam*

Contains a pointer to a null-terminated text string; see the description of individual settings.

Reserved1

Not used; reserved for future use.

Flags

Contains additional flags or data; see the description of individual settings.

Reserved2

Not used; reserved for future use. Must be set to 0.

emarks

If a plugin changes the values of the "*Tabsize*" and "*Expand tabs to spaces*" parameters at the same time, it is recommended to set the tabsize first and then set the "Expand tabs to spaces" mode.

e also:

structures | TEditorSetParameter

ditorSetPosition

in | structures

The **EditorSetPosition** structure is used in the <u>EditorControl</u> function to set the cursor position and state in the FAR editor.

```
struct EditorSetPosition
{
    int CurLine;
    int CurPos;
    int CurTabPos;
    int TopScreenLine;
    int LeftPos;
    int Overtype;
};
```

ements

CurLine

New value of the current line index, or -1 to retain the current value.

CurPos

New value of the cursor position in the line, or -1 to retain the current value.

CurTabPos

New value of the cursor position on the screen, or -1 to keep the current value. If the current line doesn't contain tab characters, *CurTabPos* has the same meaning, as *CurPos*. Both *CurPos* and *CurTabPos* should not be specified, either one or the other must be set to -1.

TopScreenLine

New value of the first visible line index, or -1 to keep the current value.

LeftPos

New value of the leftmost visible position of the text on the screen, or -1 to keep the current value.

Overtype

Set to 0 for insert mode, 1 for overtype mode, -1 to keep the current mode.

e also:

structures | TEditorSetPosition

ditorSetString

in | <u>structures</u>

The **EditorSetString** structure is used in the <u>EditorControl</u> function to change the value of a text line in the internal FAR editor.

```
struct EditorSetString
{
    int StringNumber;
    char *StringText;
    char *StringEOL;
    int StringLength;
};
```

lements

StringNumber

Number of the text line to change, **-1** indicates the current line.

StringText

Pointer to the line text.

StringEOL

End-of-line sequence. Can be an empty string, r n r n. You can place this sequence either in *StringEOL* or directly in *StringText*. This field can also be set to NULL to use the default sequence.

StringLength

Length of data pointed to by *StringText*.

e also: structures | <u>TEditorSetString</u>
ialog API structures

in | structures

Structure	Description
<u>FarDialogEvent</u>	Information about dialog event
<u>FarDialogItem</u>	Dialog item
<u>FarDialogItemData</u>	Passing data to a dialog item
<u>FarList</u>	The DI_LISTBOX list
<u>FarListColors</u>	Describes color schemes for DI_COMBOBOX and DI_LISTBOX controls
<u>FarListDelete</u>	Parameters for deletion from a DI_COMBOBOX or DI_LISTBOX
<u>FarListFind</u>	Search in a DI_COMBOBOX or DI_LISTBOX
<u>FarListGetItem</u>	Retrieval of one element from a DI_COMBOBOX or DI_LISTBOX
<u>FarListInfo</u>	Retrieval of information about a DI_COMBOBOX or DI_LISTBOX
<u>FarListInsert</u>	Item insertion into a DI_COMBOBOX or DI_LISTBOX
<u>FarListItem</u>	List item
<u>FarListPos</u>	Positioning in the list
<u>FarListItemData</u>	Association of a list item with data
FarListTitles	Set or get list labels
<u>FarListUpdate</u>	List item update data
<u>OpenDlgPluginData</u>	Information about dialog and activated plugin item.

e also:

Exported functions, Service functions, Dialog API, Archive support, Addons, Delphi structures, Win32 structures

arDialogEvent

in | <u>Dialog API</u>

FarDialogEvent structure describes an event, sent to the <u>ProcessDialogEvent</u> function.

```
struct FarDialogEvent
{
    HANDLE hDlg;
    int Msg;
    int Param1;
    LONG_PTR Param2;
    LONG_PTR Result;
};
```

lements

hDlg Dialog handle

Msg

One of the <u>messages or events</u>

Param1

Param 1

Param2

Param 2

Result

Dialog handler return code.

emarks

e also: <u>Structures</u>, <u>ProcessDialogEvent</u>, <u>Dialog API</u>

arDialogItem

in | Dialog API | Dialog items

The **FarDialogItem** structure describes one dialog item. An array of those structures is passed to the <u>Dialog</u> or <u>DialogEx</u> functions to show a dialog.

```
struct FarDialogItem
{
  int Type;
  int X1;
  int Y1;
  int X2;
  int Y2;
  int Focus;
  union
  {
    DWORD_PTR Reserved;
    int Selected;
    const char *History;
    const char *Mask;
    struct FarList *ListItems;
    int ListPos;
    CHAR_INFO *VBuf;
  }
#ifdef _FAR_NO_NAMELESS_UNIONS
  Param
#endif
  ;
  DWORD Flags;
  int DefaultButton;
  union
  {
    char Data[512];
    struct
    {
      DWORD PtrFlags;
          PtrLength;
      int
      char *PtrData;
      char PtrTail[1];
```

```
} Ptr;
}
#ifdef _FAR_NO_NAMELESS_UNIONS
Data
#endif
;
};
```

embers

Туре

Dialog item type. Can be one of the following values, described in the "<u>Dialog</u> <u>items</u>" topic.

X1,Y1,X2,Y2

Dialog item coordinates, calculated relative to the top left dialog corner (coordinates start from 0,0). For more information about coordinates see the description of a specific <u>dialog item</u>.

Focus

Keyboard focus flag. You must specify one item with *Focus* equal to TRUE.

Param.Reserved

Size of this members always equals the size of the Param union, which allows to use this member for platform independent initialization.

Param.Selected

Applicable for <u>DI_CHECKBOX</u> and <u>DI_RADIOBUTTON</u> controls. Allows to set their initial state and get their state after closing the dialog.

Param.History

Contains the address of a null-terminated text string that will be used as the internal history name when an edit control has the <u>DIF_HISTORY</u> flag. If several edit controls have the same history name, they will share the same history list.

Param.Mask

Contains the address of a null-terminated string that serves as a mask for a <u>DI_FIXEDIT</u> control.

Param.ListItems

Pointer to a FarList structure that describes the list of items for a

DI_COMBOBOX or DI_LISTBOX control.

Param.ListPos

Current list position in a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> control.

Param.VBuf

Pointer to an array of <u>CHAR_INFO</u> structures describing a virtual buffer for the <u>DI_USERCONTROL</u> control.

Flags

Combination of values described in the "<u>Dialog item flags</u>" topic.

DefaultButton

Define current item as the "default control". If while pressing **<Enter>** the focus is not set on a button, the <u>Dialog</u> function (or <u>DialogEx</u>) will return the number of the item with *DefaultButton* set to 1. The *DefaultButton* flag can be set not only for a button, but for any other dialog item.

Data.Data

Buffer to exchange data with the dialog (without <u>DIF_VAREDIT</u> flag). See <u>dialog items</u> descriptions for details.

Data.Ptr.PtrFlags

Additional flags (not used in Dialog API 1.0). *Ptr.PtrFlags* must be 0.

Data.Ptr.PtrLength

Length of the data pointed to by *PtrData*.

Data.Ptr.PtrData

Points to the user buffer for the edit data.

Data.Ptr.PtrTail

"Tail" - provides access to the remaining part of the *Data* member.

emarks

- 1. All information about the **FarDialogItem** structure is described with named unions taken into account. See details in the <u>FAR NO NAMELESS UNIONS</u> macro description.
- 2. Because the *Data.Data* member size is large, direct initialization of a **FarDialogItem** structures array can be very memory consuming. To avoid it, you can, for example, initialize a temporary structure type array with **Data* instead of *Data*[512] and then convert it to a **FarDialogItem** array

using a simple function.

- 3. The example function to do the abovementioned conversion can be found in the sources of example plugins supplied with FAR Manager (the <u>InitDialogItems</u> function and the <u>InitDialogItem</u> structure).
- 4. The **Ptr.*** members are used only for <u>DI_EDIT</u> controls with the <u>DIF_VAREDIT</u> flag.

e also:

<u>Structures</u>, <u>FAR NO NAMELESS UNIONS</u>, <u>InitDialogItem</u>, <u>TFarDialogItem</u>, <u>Dialog item flags</u>

arDialogItemData

in | Dialog API

The **FarDialogItemData** structure describes data, being sent (received) to the dialog item by a <u>DM_SETTEXT</u> message (<u>DM_GETTEXT</u>) as Param2 parameter.

```
struct FarDialogItemData
{
    int PtrLength;
    char *PtrData;
};
```

embers

PtrLength

Data size in *PtrData* without ending NULL character.

PtrData

Pointer to the data being sent.

emarks

e also:

<u>DM GETTEXT</u>, <u>DM SETTEXT</u>, <u>TFarDialogItemData</u>, <u>Structures</u>

arList

in | <u>Dialog API</u>

The **FarList** structure contains a pointer to an array of <u>FarListItem</u> structures for <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> controls.

```
struct FarList
{
    DWORD ItemsNumber;
    struct FarListItem *Items;
};
```

embers

ItemsNumber

The number of list items.

Items

Pointer to an array of <u>FarListItem</u> structures containing a list of combobox, dropdown list or list box items.

emarks

e also: <u>TFarList, FarListItem, Structures</u>

arListColors

in | <u>Dialog API</u>

The **FarListColors** structure describes the color scheme for <u>DI_LISTBOX</u> and <u>DI_COMBOBOX</u> controls.

```
struct FarListColors
{
    DWORD Flags;
    DWORD Reserved;
    int ColorCount;
    LPBYTE Colors;
};
```

embers

Flags

Falgs. Must be 0.

Reserved

Reserved for future use. Must be 0.

ColorCount

Amount of items in the **Colors** array.

Colors

Byte array of the color attributes (background_color+text_color). By default the folowing attributes are set:

for **DI_LISTBOX**

Index	Constant	Description
0	COL_DIALOGLISTBOX	background
1	COL_DIALOGLISTBOX	border
2	COL_DIALOGLISTTITLE	titles - top and bottom
3	COL_DIALOGLISTTEXT	normal item
4	COL_DIALOGLISTHIGHLIGHT	hotkey
5	COL_DIALOGLISTBOX	separator
6	COL_DIALOGLISTSELECTEDTEXT	selected item

7	COL_DIALOGLISTSELECTEDHIGHLIGHT	selected hotkey
8	COL_DIALOGLISTSCROLLBAR	scrollbar
9	COL_DIALOGLISTDISABLED	disabled item
10	COL_DIALOGLISTARROWS	long string indicators
11	COL_DIALOGLISTARROWSSELECTED	selected long string indicators
12	COL_DIALOGLISTARROWSDISABLED	disabled long string indicators
13	COL_DIALOGLISTGRAY	grayed out item
14	COL_DIALOGLISTSELECTEDGRAYTEXT	selected grayed out item

for <u>DI_COMBOBOX</u>

Index	Constant	Description
0	COL_DIALOGCOMBOBOX	background
1	COL_DIALOGCOMBOBOX	border
2	COL_DIALOGCOMBOTITLE	titles - top and bottom
3	COL_DIALOGCOMBOTEXT	normal item
4	COL_DIALOGCOMBOHIGHLIGHT	hotkey
5	COL_DIALOGCOMBOBOX	separator
6	COL_DIALOGCOMBOSELECTEDTEXT	selected item
7	COL_DIALOGCOMBOSELECTEDHIGHLIGHT	selected hotkey
8	COL_DIALOGCOMBOSCROLLBAR	scrollbar
9	COL_DIALOGCOMBODISABLEDTEXT	disabled item
10	COL_DIALOGCOMBOARROWS	long string indicators
11	COL_DIALOGCOMBOARROWSSELECTED	selected long string indicators
12	COL_DIALOGCOMBOARROWSDISABLED	disabled long string indicators
13	COL_DIALOGCOMBOGRAY	grayed out item
14	COL_DIALOGCOMBOSELECTEDGRAYTEXT	selected grayed out item

emarks

arListDelete

in | Dialog API

The **FarListDelete** structure specifies settings for deletion of items from a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> list.

```
struct FarListDelete
{
    int StartIndex;
    int Count;
};
```

embers

StartIndex

Index of the first item to be deleted.

Count

Number of items to be deleted.

emarks

If the value of **Count** is less or equal to 0, all items will be deleted.

e also: <u>DM_LISTDELETE, TFarListDelete, Structures</u>

arListFind

in | <u>Dialog API</u>

The **FarListFind** structure specifies settings for item search in a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> list.

```
struct FarListFind
{
    int StartIndex;
    const char *Pattern;
    DWORD Flags;
    DWORD Reserved;
};
```

embers

StartIndex

Index of the item from which the search is started.

Pattern

Pattern to find in the format used by the CmpName function

Flags

Search options. Combination of zero or more of the following values (FARLISTFINDFLAGS enum):

Flag	Description
LIFIND_EXACTMATCH	The Pattern field is not a pattern (mask) but a literal string. It must be an exact match of the listbox string (that is, not only the beginning should match).

Reserved

Reserved

emarks

e also: DM_LISTFINDSTRING, TFarListFind, Structures

arListGetItem

in | <u>Dialog API</u>

The **FarListGetItem** structure describes one item of a <u>FarListItem</u> structure for <u>DI_LISTBOX</u> and <u>DI_COMBOBOX</u> dialog items.

```
struct FarListGetItem
{
    int ItemIndex;
    struct FarListItem Item;
};
```

embers

ItemIndex

Index of the list item.

Item

List item represented by a <u>FarListItem</u> structure.

emarks

e also: <u>TFarListGetItem</u>, <u>FarListItem</u>, <u>DM_LISTGETITEM</u>, <u>Structures</u>

arListInfo

in | Dialog API

The **FarListInfo** structure is used to retrieve information about a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> control.

```
struct FarListInfo
{
    DWORD Flags;
    int ItemsNumber;
    int SelectPos;
    int TopPos;
    int MaxHeight;
    int MaxLength;
    DWORD Reserved[6];
};
```

embers

Flags

A combination of zero or more of the following flags (FARLISTINFOFLAGS enum):

Flag	Description
LINFO_SHOWNOBOX	A <u>DI_LISTBOX</u> control is drawn without a frame when the <u>DIF_LISTNOBOX</u> flag is set.
LINFO_AUTOHIGHLIGHT	Hotkeys will be assigned automatically, starting with the first item.
LINFO_REVERSEHIGHLIGHT	Hotkeys will be assigned automatically, starting with the last item.
LINFO_WRAPMODE	Trying to move the cursor above the first item or below the last item will move the cursor to the bottom or the top of the list, respectively.
LINFO_SHOWAMPERSAND	Show ampersands (&). If this flag is not set, ampersands are used to define hot keys for list items.

ItemsNumber

Number of items in the list.

SelectPos

Index of the selected item in the list.

TopPos

Index of the topmost visible item in the list.

MaxHeight

Maximum height of the list.

MaxLength

Maximum length of a list item line.

Reserved

Reserved for future use.

emarks

e also: <u>DM_LISTINFO</u>, <u>TFarListInfo</u>, <u>Structures</u>

arListInsert

in | Dialog API

The **FarListInsert** structure specifies the settings for insertion of an item into a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> list.

```
struct FarListInsert
{
    int Index;
    struct FarListItem Item;
};
```

embers

Index

Position at which the item is to be inserted. If the position is greater than the count of items in the list, the item is appended to the end of the list. If the position is negative, the item is inserted at the beginning of the list.

Item

Pointer to a <u>FarListItem</u> structure describing the item to insert.

emarks

e also:

FarListItem, DM_LISTINSERT, TFarListInsert, Structures

arListItem

in | <u>Dialog API</u>

The **FarListItem** structure describes one item in a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> list.

```
struct FarListItem
{
    DWORD Flags;
    char Text[128];
    DWORD Reserved[3];
};
```

embers

Flags

Can be a combination of the following values (LISTITEMFLAGS enum):

Flag	Description
LIF_SELECTED	Item selection flag. It must be set only for one item.
LIF_CHECKED	If this flag is set, a selection mark is displayed before the item's text. (character with code 0xFB, for example).
LIF_SEPARATOR	If this flag is set, the menu item is displayed as a separator. If Text is not empty, it is drawn over the separator line.
LIF_DISABLE	If this flag is set, the list item becomes disabled.
LIF_GRAYED	If this flag is set, the list item is shown, but cannot be selected.
LIF_HIDDEN	If this flag is set, the list item is not shown.
LIF_DELETEUSERDATA	This flag is set in the Item parameter of the <u>DM_LISTUPDATE</u> message, if it is needed to delete related data when list item is being updated.

Text

Item text.

Reserved

Reserved. Must be 0.

emarks

e also:

<u>FarList</u>, <u>DM_LISTGETITEM</u>, <u>FarListInsert</u>, <u>FarListUpdate</u>, <u>TFarListItem</u>, <u>Structures</u>

arListPos

in | Dialog API

The **FarListPos** structure contains information about the cursor position in a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> list.

```
struct FarListPos
{
    int SelectPos;
    int TopPos;
};
```

embers

SelectPos

Index of the current item.

TopPos

Index of the first visible item, or -1 if the first visible item should be determined automatically according to *SelectPos*.

emarks

e also:

DM LISTGETCURPOS, DM LISTSETCURPOS, TFarListPos, Structures

arListItemData

in | Dialog API

The **FarListItemData** structure describes the data that will be associated with an item in a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> list.

```
struct FarListItemData
{
    int Index;
    int DataSize;
    void *Data;
    DWORD Reserved;
};
```

embers

Index

Index of the list item to which the data is associated.

DataSize

Size of *Data* or 0 if a null-terminated string is being associated with the item.

Data

Pointer to the data.

Reserved

Reserved.

emarks

Dialog manager allocates memory for the data associated with the list item using the following rules:

DataSize	Data
0	Pointer to a null-terminated string. Memory area of strlen(Data)+1 bytes is alocated. The data is then copied to the allocated space.
<= sizeof(DWORD)	Character array of 4 or less elements or any pointer (HWND window handle, for example). No memory is allocated. Data is placed in the local area of the list item.
> sizeof(DWORD)	Arbitrary data. Memory area of <i>DataSize</i> bytes is

allocated. The data is then copied to the allocated
space.

e also:

FarList, DM_LISTGETDATA, DM_LISTSETDATA, TFarListItemData, Structures

arListTitles

in | structures | Dialog API

The **FarListTitles** structure specifies the titles (top and bottom) of a <u>DI_LISTBOX</u> list.

```
struct FarListTitles
{
    int TitleLen;
    char *Title;
    int BottomLen;
    char *Bottom;
};
```

embers

TitleLen

Length of the top title string.

Title

Top title string. May be NULL; in this case, the title is not shown.

BottomLen

Length of the bottom title string.

Bottom

Bottom title string. May be NULL; in this case, the title is not shown.

emarks

TitleLen and **BottomLen** members are required only while getting list titles (see <u>DM_LISTGETTITLES</u>)

e also:

TFarListTitles, DM LISTGETTITLES, DM LISTSETTITLES

arListUpdate

in | <u>Dialog API</u>

The **FarListUpdate** structure specifies the settings for updating an item in a <u>DI_LISTBOX</u> or <u>DI_COMBOBOX</u> list: the index of the item to update and the updated item's data.

```
struct FarListUpdate
{
    int Index;
    struct FarListItem Item;
};
```

embers

Index

Index of the item to update.

Items

A <u>FarListItem</u> structure being updated.

emarks

User data associated with a list item is not deleted. Use <u>LIF_DELETEUSERDATA</u> for autodeletion of user data.

e also: DM_LISTUPDATE, TFarListUpdate, FarListItem, Structures

penDlgPluginData

in | <u>structures</u>

Pointer to an **OpenDlgPluginData** structure is passed to an <u>OpenPlugin</u> function, when plugin is called from dialog.

```
struct OpenDlgPluginData
{
    int ItemNumber;
    HANDLE hDlg;
};
```

emnts

ItemNumber

Position of the activated plugin item in the exported items list in plugins menu.

hDlg

Dialog handle.

emarks

e also: <u>OpenPlugin</u>, <u>structures</u>

tructure - Viewer

in | structures

iewer specific structures

Structure	Description
FARINT64	used to hold a 64 bit integer value.
<u>ViewerInfo</u>	current viewer state
<u>ViewerMode</u>	information about the current view mode
<u>ViewerSelect</u>	block selection in the internal viewer
<u>ViewerSetMode</u>	set the working mode of the current viewer instance
<u>ViewerSetPosition</u>	position setting in the viewer

e also:

Exported functions, Service functions, Delphi structures, Win32 structures

iewerInfo

in | <u>structures</u>

The **ViewerInfo** structure is used to get information about the current state of the internal viewer.

```
struct ViewerInfo
{
       StructSize;
  int
  int ViewerID;
  const char *FileName;
  FARINT64 FileSize;
  FARINT64 FilePos;
  int WindowSizeX;
  int WindowSizeY;
  DWORD Options;
     TabSize;
  int
  struct ViewerMode CurMode;
  int LeftPos;
  DWORD Reserved3;
};
```

emnts

StructSize

Size of the ViewerInfo structure. You must set this field.

ViewerID

Identifier of the viewer instance. Each viewer instance has a unique identifier that cannot be repeated during a FAR session.

FileName

Full path and name of the file beeing viewed.

FileSize

File size. Veriable of <u>FARINT64</u> type.

FilePos

Current file position (absolute offset in bytes). Variable of <u>FARINT64</u> type. *WindowSizeX*, *WindowSizeY*

Width and height of the viewer window.

Options

Describes viewer options state. Can be a combination of the following flags (VIEWER_OPTIONS enum):

Flag	Description
VOPT_SAVEFILEPOSITION	"Save file position"
VOPT_AUTODETECTTABLE	"Autodetect character table"

TabSize

Tabulation size.

CurMode

A variable of <u>ViewerMode</u> type - additional information about the view mode.

LeftPos

Position of the left border of the viewer window in the viewed text.

Reserved3

Reserved for future use.

e also:

Structures, ViewerControl, ViewerMode

iewerMode

in | structures

The **ViewerMode** structure is used to receive additional information about the state of the current viewer instance.

```
struct ViewerMode
{
    int UseDecodeTable;
    int TableNum;
    int AnsiMode;
    int Unicode;
    int Wrap;
    int WordWrap;
    int Hex;
    DWORD Reserved[4];
};
```

emnts

UseDecodeTable

If 1, then a decoding table is used.

TableNum

The number of the user character table. Use only if UseDecodeTable is 1.

AnsiMode

If 1, then the current charset is ANSI.

Unicode

If 1, then the current charset is Unicode.

Wrap

Text wrapping is on - 1, off - 0.

TypeWrap

Text wrapping mode: 1 - word wrap is on, 0 - line wrap.

Hex

If 1, then the viewer is in hex-mode.

Reserved

Reserved for future use.

e also: <u>Structures, ViewerControl, ViewerInfo</u>

iewerSelect

in | structures

The **ViewerSelect** structure is used to select a block in the internal viewer.

```
struct ViewerSelect
{
    FARINT64 BlockStartPos;
    int BlockLen;
};
```

emnts

BlockStartPos

Selection start - in characters, not in bytes. This means that if the viewer is in **Unicode** mode, **BlockStartPos** will equal - position in file / 2.

BlockLen

Selection length in characters.

e also:

Structures, ViewerControl, FARINT64

iewerSetMode

in | structures

The **ViewerSetMode** structure is used to set the view mode of the current viewer instance.

```
struct ViewerSetMode
{
    int Type;
    union {
        int iParam;
        char *cParam;
        Param;
        DWORD Flags;
        DWORD Reserved;
};
```

lements

Туре

Mode type. Can be one of the following values (VIEWER_SETMODE_TYPES enum):

Mode	Description
VSMT_HEX	Text/Hex mode: <i>iParam</i> =1 - turn Hex mode on, <i>iParam</i> =0 - text mode.
VSMT_WRAP	Line wrap: <i>iParam</i> =1 - line wrap is on, <i>iParam</i> =0 - off)
VSMT_WORDWRAP	Word wrap: <i>iParam</i> =1 - word wrap is on, <i>iParam</i> =0 - off.

iParam

Integer value, see details above.

cParam

Pointer to a null terminated string, see details above.

Flags

Additional flags (VIEWER_SETMODEFLAGS_TYPES enum):

	Mode	Description
_ 8		

VSMFL_REDRAW	Redraw the screen. Otherwise use the
	VCTL REDRAW command to redraw the screen
	after changing the mode.

Reserved

Reserved for future use. Should be 0.

emarks

e also: <u>structures</u> | <u>TViewerSetMode</u>

iewerSetPosition

in | structures

The **ViewerSetPosition** structure is used to change the current position in the current viewer instance.

```
struct ViewerSetPosition
{
    DWORD Flags;
    FARINT64 StartPos;
    int LeftPos;
};
```

emnts

Flags

Flags, defining the position change process. Can be a confination of the following flags (VIEWER_SETPOS_FLAGS enum):

Flag	Description
VSP_NOREDRAW	Do not redraw the screen.
VSP_PERCENT	The offset is given in percents not bytes.
VSP_RELATIVE	The offset is relative not absolute.
VSP_NORETNEWPOS	Do not return the real position (see <i>StartPos</i>).

StartPos

New file positions (in bytes or percents - depends on the **VSP_PERCENT** flag, can be negative if the **VSP_RELATIVE** flag is specified). Generaly it is not possible to set the exact position in the viewer, so the new position may not coninside with the one in *StartPos*. The new real position is then stored in *StartPos* (if the **VSP_NORETNEWPOS** is not specified). Use this property if needed.

LeftPos

Position of the left border of the viewer window in the viewed text.

e also:

Structures, ViewerControl, FARINT64

ActlEjectMedia

in | structures | ActlEjectMedia

The **ActlEjectMedia** stucture for Delphi:

```
TActlEjectMedia = packed record
Letter: DWORD;
Flags: DWORD;
end;
PActlEjectMedia = ^TActlEjectMedia;
```

ActlKeyMacro

in | structures | ActlKeyMacro

The **ActlKeyMacro** stucture for Delphi:

```
TActlKeyMacro = packed record
  Command: Integer;
  Reserved: packed array[0..2] of DWORD;
end;
PActlKeyMacro = ^TActlKeyMacro;
```

ArcInfo

chive support | structures | ArcInfo

The **ArcInfo** stucture for Delphi:

```
TArcInfo = packed record
SFXSize: Integer;
Volume: Integer;
Comment: Integer;
Recovery: Integer;
Lock: Integer;
Flags: Integer;
end;
PArcInfo = ^TArcInfo;
```
Arciteminfo

chive support | structures | TArcItemInfo

The **ArcItemInfo** stucture for Delphi:

```
TArcItemInfo = packed record
HostOS: packed array[0..31] of char;
Description: packed array[0..255] of char;
Solid: integer;
Comment: integer;
Encrypted: integer;
DictSize: integer;
UnpVer: integer;
end;
PArcItemInfo = ^TArcItemInfo;
```

CharTableSet

in | structures | CharTableSet

The **CharTableSet** stucture for Delphi:

```
TCharTableSet = packed record
DecodeTable: packed array[0..255] of BYTE;
EncodeTable: packed array[0..255] of BYTE;
UpperTable: packed array[0..255] of BYTE;
LowerTable: packed array[0..255] of BYTE;
TableName: packed array[0..127] of char;
end;
PCharTableSet = ^TCharTableSet;
```

CmdLineSelect

in | structures | CmdLineSelect

The **CmdLineSelect** stucture for Delphi:

```
TCmdLineSelect = packed record
  SelStart: integer;
  SelEnd: integer;
end;
PEditorSelect = ^TCmdLineSelect;
```

EditorBookMarks

in | structures EditorBookMarks

The **EditorBookMarks** stucture for Delphi:

```
TEditorBookMarks = packed record
Line: ^Integer;
Cursor: ^Integer;
ScreenLine: ^Integer;
LeftPos: ^Integer;
Reserved: packed array[0..3] of DWORD;
end;
PEditorBookMarks = ^TEditorBookMarks;
```

EditorColor

in | structures

The **EditorColor** stucture for Delphi:

```
TEditorColor = packed record
StringNumber: integer;
ColorItem: integer;
StartPos: integer;
EndPos: integer;
Color: integer;
end;
PEditorColor = ^TEditorColor;
```

EditorConvertPos

in | structures | EditorConvertPos

The **EditorConvertPos** stucture for Delphi:

```
TEditorConvertPos = packed record
StringNumber: integer;
SrcPos: integer;
DestPos: integer;
end;
PEditorConvertPos = ^TEditorConvertPos;
```

EditorConvertText

in | structures | EditorConvertText

The **EditorConvertText** stucture for Delphi:

```
TEditorConvertText = packed record
Text: PChar;
TextLength: integer;
end;
PEditorConvertText = ^TEditorConvertText;
```

EditorGetString

in | structures | EditorGetString

The **EditorGetString** stucture for Delphi:

```
TEditorGetString = packed record
StringNumber: integer;
StringText: PChar;
StringEOL: PChar;
StringLength: integer;
SelStart: integer;
SelEnd: integer;
end;
PEditorGetString = ^TEditorGetString;
```

EditorInfo

<u>in</u> | <u>structures</u> | <u>EditorInfo</u>

The **EditorInfo** stucture for Delphi:

```
TEditorInfo = packed record
  EditorID: integer;
  FileName: PChar;
  WindowSizeX: integer;
  WindowSizeY: integer;
  TotalLines: integer;
  CurLine: integer;
  CurPos: integer;
  CurTabPos: integer;
  TopScreenLine: integer;
  LeftPos: integer;
  Overtype: integer;
  BlockType: integer;
  BlockStartLine: integer;
  AnsiMode: integer;
  TableNum: integer;
  Options: DWORD;
  TabSize: integer;
  BookMarkCount: integer;
  Reserved: packed array[0..6] of DWORD;
end;
PEditorInfo = ^TEditorInfo;
```

EditorSelect

in | structures | EditorSelect

The **EditorSelect** stucture for Delphi:

```
TEditorSelect = packed record
BlockType: integer;
BlockStartLine: integer;
BlockStartPos: integer;
BlockWidth: integer;
BlockHeight: integer;
end;
PEditorSelect = ^TEditorSelect;
```

EditorSetParameter

in | structures | EditorSetParameter

The **EditorSetParameter** stucture for Delphi:

```
TEditorSetParameter = packed record
ParamType: integer;
case integer of
0: (iParam: integer);
1: (cParam: PChar);
2: (Reserved1: DWORD);
Flags: DWORD;
Reserved2: DWORD;
end;
PEditorSetParameter = ^TEditorSetParameter;
```

EditorSetPosition

in | structures | EditorSetPosition

The **EditorSetPosition** stucture for Delphi:

```
TEditorSetPosition = packed record
CurLine: integer;
CurPos: integer;
CurTabPos: integer;
TopScreenLine: integer;
LeftPos: integer;
Overtype: integer;
end;
PEditorSetPosition = ^TEditorSetPosition;
```

EditorSetString

in | structures | EditorSetString

The **EditorSetString** stucture for Delphi:

```
TEditorSetString = packed record
StringNumber: integer;
StringText: PChar;
StringEOL: PChar;
StringLength: integer;
end;
PEditorSetString = ^TEditorSetString;
```

FarSetColors

in | structures | FarSetColors

The **FarSetColors** stucture for Delphi:

```
TFarSetColors = packed record
Flags: DWORD;
StartIndex: integer;
ColorCount: integer;
Colors: PChar;
end;
PFarSetColors = ^TFarSetColors;
```

FarDialogItem

in | structures

The **FarDialogItem** stucture for Delphi:

```
TFarPtr = packed record
    PtrFlags: DWORD;
    PtrLength: integer;
    PtrData: PChar;
    PtrTail: array[0..0] of char;
  end;
TFarDialogItem = packed record
  ItemType: integer;
  X1: integer;
  Y1: integer;
  X2: integer;
  Y2: integer;
  Focus: integer;
  case integer of
    0: (History: PChar);
    1: (Mask: PChar);
    2: (ListItems: PFarListItemArr);
    3: (VBuf: PCharInfo);
    4: (Selected: integer;
        Flags: DWORD;
        DefaultButton: integer;
        case integer of
          0: (Data: packed array[0..511] of char);
          1: (Ptr: TFarPtr)
       );
end;
PFarDialogItem = ^TFarDialogItem;
```

FarDialogItemData

in | structures | FarDialogItemData

The **FarDialogItemData** stucture for Delphi:

```
TFarDialogItemData = packed record
DataLength: Integer;
DataPtr: PChar;
end;
PFarDialogItemData = ^TFarDialogItemData;
```

FarList

<u>in | structures | FarList</u>

The **FarList** stucture for Delphi:

```
TFarList = packed record
  ItemsNumber: integer;
  Items: PFarListItemArr;
end;
PFarList = ^TFarList;
```

FarListColors

in | structures | FarListColors

The **FarListColors** stucture for Delphi:

```
TFarListColors = packed record
Flags: DWORD;
Reserved: integer;
ColorCount: integer;
Colors: PChar;
end;
PFarListColors = ^TFarListColors;
```

FarListDelete

in | structures | FarListDelete

The **FarListDelete** stucture for Delphi:

```
TFarListDelete = packed record
   StartIndex: integer;
   Count: integer;
end;
PFarListDelete = ^TFarListDelete;
```

FarListGetItem

in | structures | FarListGetItem

The **FarListGetItem** stucture for Delphi:

```
TFarListGetItem = packed record
  ItemIndex: integer;
  Item: FarListItem;
end;
PFarListGetItem = ^TFarListGetItem;
```

FarListItem

in | structures FarListItem

The **FarListItem** stucture for Delphi:

```
TFarListItem = packed record
Flags: DWORD;
Text: packed array[0..127] of char;
Reserved: array[0..2] of DWORD;
end;
PFarListItem = ^TFarListItem;
```

FarListPos

in | structures FarListPos

The **FarListPos** stucture for Delphi:

```
TFarListPos = packed record
  SelectPos: Integer;
  TopPos: Integer;
end;
PFarListPos = ^TFarListPos;
```

FarListItemData

in | structures FarListItemData

The **FarListItemData** stucture for Delphi:

```
TFarListItemData = packed record
Index: integer;
DataSize: integer;
Data: PChar;
Reserved: DWORD;
end;
PFarListItemData = ^TFarListItemData;
```

FarListTitles

in | structures | FarListTitles

The **FarListTitles** stucture for Delphi:

```
TFarListTitles = packed record
TitleLen: Integer;
Title: PChar;
BottomLen: Integer;
Bottom: PChar;
end;
PFarListTitles = ^TFarListTitles;
```

FarMenultem

in | structures

The **FarMenuItem** stucture for Delphi:

```
TFarMenuItem = packed record
  Text: packed array[0..127] of char;
  Selected: integer;
  Checked: integer;
  Separator: integer;
end;
PFarMenuItem = ^TFarMenuItem;
```

FarMenultemEx

in | structures FarMenuItemEx

The **FarMenuItemEx** stucture for Delphi:

```
TFarMenuItemEx = packed record
Flags: DWORD;
Text: packed array[0..127] of char;
UserData: DWORD;
end;
PFarMenuItemEx = ^TFarMenuItemEx;
```

FarListFind

in | structures | FarListFind

The **FarListFind** stucture for Delphi:

```
TFarListFind = packed record
StartIndex: integer;
Pattern: PChar;
Flags: DWORD;
Reserved: DWORD;
end;
PFarListFind = ^TFarListFind;
```

FarListInfo

in | structures | FarListInfo

The **FarListInfo** stucture for Delphi:

```
TFarListInfo = packed record
Flags: DWORD;
ItemsNumber: integer;
SelectPos: integer;
TopPos: integer;
MaxHeight: integer;
MaxLength: integer;
Reserved: array[0..5] of DWORD;
end;
PFarListInfo = ^TFarListInfo;
```

FarListInsert

in | structures | FarListInsert

The **FarListInsert** stucture for Delphi:

```
TFarListInsert = packed record
  Index: integer;
  Item: TFarListItem;
end;
PFarListInsert = ^TFarListInsert;
```

InfoPanelLine

in | structures | InfoPanelLine

The **InfoPanelLine** stucture for Delphi:

```
TInfoPanelLine = packed record
Text: packed array[0..79] of char;
Data: packed array[0..79] of char;
Separator: integer;
end;
PInfoPanelLine = ^TInfoPanelLine;
```

KeyBarTitles

in | structures | KeyBarTitles

The **KeyBarTitles** stucture for Delphi:

```
TKeyBarTitles = packed record
Titles: packed array[0..11] of PChar;
CtrlTitles: packed array[0..11] of PChar;
AltTitles: packed array[0..11] of PChar;
ShiftTitles: packed array[0..11] of PChar;
CtrlShiftTitles: packed array[0..11] of PChar;
AltShiftTitles: packed array[0..11] of PChar;
CtrlAltTitles: packed array[0..11] of PChar;
end;
PKeyBarTitles = ^TKeyBarTitles;
```

KeySequence

in | structures | KeySequence

The **KeySequence** stucture for Delphi:

```
TKeySequence = packed record
Flags: DWORD;
Reserved: DWORD;
Count: Integer;
Sequence: ^DWORD;
end;
PKeySequence = ^TKeySequence;
```

OpenPluginInfo

in | structures

The **OpenPluginInfo** stucture for Delphi:

```
TOpenPluginInfo = packed record
    StructSize: integer;
    Flags: DWORD;
    HostFile: PChar;
    CurDir: PChar;
    Format: PChar;
    PanelTitle: PChar;
    InfoLines: PInfoPanelLineArr;
    InfoLinesNumber: integer;
    DescrFiles: PPCharArray;
    DescrFilesNumber: integer;
    PanelModesArray: PPanelModeArr;
    PanelModesNumber: integer;
    StartPanelMode: integer;
    StartSortMode: integer;
                   (*
                     SM DEFAULT,
                     SM UNSORTED,
                     SM_NAME,
                     SM_EXT,
                     SM_MTIME,
                     SM_CTIME,
                     SM_ATIME,
                     SM_SIZE,
                     SM DESCR,
                     SM_OWNER,
                     SM_COMPRESSEDSIZE,
                     SM NUMLINKS
                   * )
    StartSortOrder: Integer;
    KeyBar: PKeyBarTitles;
    ShortcutData: PChar;
    Reserved: DWORD;
end;
```

POpenPluginInfo = ^TOpenPluginInfo;

PanelInfo

in | structures

The **PanelInfo** stucture for Delphi:

```
TPanelInfo = packed record
    PanelType: integer;
               (*
                 PTYPE FILEPANEL,
                PTYPE TREEPANEL,
                 PTYPE OVIEWPANEL,
                 PTYPE INFOPANEL
               * )
    Plugin: integer;
    PanelRect: TRect;
    PanelItems: PPluginPanelItemArr;
    ItemsNumber: integer;
    SelectedItems: PPluginPanelItemArr;
    SelectedItemsNumber: integer;
    CurrentItem: integer;
    TopPanelItem: integer;
    Visible: integer;
    Focus: integer;
    ViewMode: inetegr;
    ColumnTypes: packed array[0..79] of char;
    ColumnWidths: packed array[0..79] of char;
    CurDir: packed array[0..Pred(NM)] of char;
    ShortNames: integer;
    SortMode: integer;
              (*
                SM DEFAULT,
                SM_UNSORTED,
                SM NAME,
                SM EXT,
                SM MTIME,
                SM CTIME,
                SM_ATIME,
                SM_SIZE,
                SM_DESCR,
```

```
SM_OWNER,
SM_COMPRESSEDSIZE,
SM_NUMLINKS
*)
Reserved: packed array[0..1] of DWORD;
end;
PPanelInfo = ^TPanelInfo;
```
PanelMode

in | structures

The **PanelMode** stucture for Delphi:

```
TPanelMode = packed record
  ColumnTypes: PChar;
  ColumnWidths: PChar;
  ColumnTitles: PPCharArray;
  FullScreen: integer;
  DetailedStatus: integer;
  AlignExtensions: integer;
  CaseConversion: integer;
  StatusColumnTypes: PChar;
  StartusColumnWidths: PChar;
  Reserved: array[0..1] of DWORD;
end;
PPanelMode = ^TPanelMode;
```

PanelRedrawInfo

in | structures

The **PanelRedrawInfo** stucture for Delphi:

```
TPanelRedrawInfo = packed record
    CurrentItem: integer;
    TopPanelItem: integer;
end;
PPanelRedrawInfo = ^TPanelRedrawInfo;
```

PluginInfo

in | structures

The **PluginInfo** stucture for Delphi:

```
TPluginInfo = packed record
StructSize: Integer;
Flags: DWORD;
DiskMenuStrings: PPCharArray;
DiskMenuNumbers: PIntegerArray;
DiskMenuStringsNumber: integer;
PluginMenuStrings: PPCharArray;
PluginMenuStringsNumber: integer;
PluginConfigStrings: PPCharArray;
PluginConfigStringsNumber: integer;
commandPrefix: PChar;
end;
PPluginInfo = ^TPluginInfo;
```

PluginPanelItem

in | structures | PluginPanelItem

The **PluginPanelItem** stucture for Delphi:

```
TPluginPanelItem = packed record
FindData: TWin32FindDataEx;
PackSizeHigh: DWORD;
PackSize: DWORD;
Flags: DWORD;
NumberOfLinks: DWORD;
Description: PChar;
Owner: PChar;
CustomColumnData: PPCharArray;
CustomColumnNumber: integer;
UserData: DWORD;
Reserved: array[0..2] of DWORD;
end;
PPluginPanelItem = ^TPluginPanelItem;
```

PluginStartupInfo

in | structures | PluginStartupInfo

The **PluginStartupInfo** stucture for Delphi:

```
TPluginStartupInfo = packed record
  StructSize: Integer;
  ModuleName: array[0..Pred(NM)] of char;
  ModuleNumber: integer;
  RootKey: PChar;
  Menu: TFarApiMenu;
  Dialog: TFarApiDialog;
  Message: TFarApiMessage;
  GetMsg: TFarApiGetMsg;
  Control: TFarApiControl;
  SaveScreen: TFarApiSaveScreen;
  RestoreScreen: TFarApiRestoreScreen;
  GetDirList: TFarApiGetDirList;
  GetPluginDirList: TFarApiGetPluginDirList;
  FreeDirList: TFarApiFreeDirList;
  Viewer: TFarApiViewer;
  Editor: TFarApiEditor;
  CmpName: TFarApiCmpName;
  CharTable: TFarApiCharTable;
  Text: TFarApiText;
  EditorControl: TFarApiEditorControl;
  FSF: <u>PFarStandardFunctions;</u>
  ShowHelp: TFarApiShowHelp;
  AdvControl: TFarApiAdvControl;
  InputBox: TFarApiInputBox;
  DialogEx: TFarApiDialogEx;
  SendDlgMessage: TFarApiSendDlgMessage;
  DefDlgProc: TFarApiDefDlgProc;
  Reserved1: DWORD;
  Reserved2: DWORD;
end;
PPluginStartupInfo = ^TPluginStartupInfo;
```

Where:

```
TFarApiMenu = function(
 PluginNumber: integer;
 X, Y: integer;
  MaxHeight: integer;
  Flags: DWORD;
  Title: PChar;
  Bottom: PChar;
 HelpTopic: PChar;
  BreakKeys: PIntArr;
 BreakCode: PIntArr;
 Items: PFarMenuItemArr;
 ItemsNumber: integer): integer; stdcall;
TFarApiDialog = function(
  PluginNumber: integer;
 X1, Y1: integer;
 X2, Y2: integer;
 HelpTopic: PChar;
 Items: PFarDialogItemArr;
 ItemsNumber: integer): integer; stdcall;
TFarApiMessage = function(
 PluginNumber: integer;
 Flags: DWORD;
 HelpTopic: PChar;
  Items: PPCharArr;
  ItemsNumber: integer;
 ButtonsNumber: integer): integer; stdcall;
TFarApiGetMsg = function(
 PluginNumber: integer;
 MsgId: integer): PChar; stdcall;
TFarApiControl = function(
  hPlugin: THandle;
 Command: integer;
 Param: pointer): integer; stdcall;
```

```
TFarApiSaveScreen = function(
  X1, Y1: integer;
  X2, Y2: integer): THandle; stdcall;
TFarApiRestoreScreen = procedure(
  hScreen: THandle); stdcall;
TFarApiGetDirList = function(
  Dir: PChar;
  var PanelItems: PPluginPanelItemArr;
  var ItemsNumber: integer): integer; stdcall;
TFarApiGetPluginDirList = function(
  PluginNumber: integer;
  hPlugin: THandle;
  Dir: PChar;
  var PanelItems: PPluginPanelItemArr;
  var ItemsNumber: integer): integer; stdcall;
TFarApiFreeDirList = procedure(
  PanelItems: PPluginPanelItemArr); stdcall;
TFarApiViewer = function(
  FileName: PChar;
  Title: PChar;
  X1, Y1: integer;
  X2, Y2: integer;
  Flags: DWORD): integer; stdcall;
TFarApiEditor = function(
  FileName: PChar;
  Title: PChar;
  X1, Y1: integer;
  X2, Y2: integer;
  Flags: DWORD;
  StartLine: integer;
  StartChar: integer): integer; stdcall;
TFarApiCmpName = function(
```

```
Pattern: PChar;
  FileName: PChar;
  SkipPath: integer): integer; stdcall;
TFarApiCharTable = function(
  Command: integer;
  Buffer: PChar;
  BufferSize: integer): integer; stdcall;
TFarApiText = procedure(
  X, Y: integer;
  Color: integer;
  Str: PChar); stdcall;
TFarApiEditorControl = function(
  Command: integer;
  Param: pointer): integer; stdcall;
TFarApiAdvControl = function(
  ModuleNumber: integer;
  Command: integer;
  Param: pointer): integer; stdcall;
TFarApiDialogEx = function(
  PluginNumber: integer;
  X1, Y1: integer;
  X2, Y2: integer;
  HelpTopic: PChar;
  Items: PFarDialogItemArr;
  ItemsNumber: integer;
  Reserved: DWORD;
  Flags: DWORD;
  DlgProc: TFarApiWndProc;
  Param: integer): integer; stdcall;
TFarApiSendDlgMessage = function(
  hDlg: THandle;
  Msg: integer;
  Param1: integer;
```

```
Param2: integer): integer; stdcall;
TFarApiDefDlgProc = function(
  hDlg: THandle;
  Msg: integer;
  Param1: integer;
  Param2: integer): integer; stdcall;
TFarApiInputBox = function(
  Title: PChar;
  SubTitle: PChar;
  HistoryName: PChar;
  SrcText: PChar;
  DstText: PChar;
  DstLength: integer;
  HelpTopic: PChar;
  Flags: DWORD): integer; stdcall;
TFarApiShowHelp = function(
  ModuleName: PChar;
  HelpTopic: PChar;
  Flags: DWORD): BOOL; stdcall;
```

Win32FindDataEx

in | structures

The **WIN32_FIND_DATA** stucture for Delphi:

```
TWin32FindDataEx = packed record
dwFileAttributes: DWORD;
ftCreationTime: TFileTime;
ftLastAccessTime: TFileTime;
ftLastWriteTime: TFileTime;
nFileSizeHigh: DWORD;
nFileSizeLow: DWORD;
dwReserved0: DWORD;
dwReserved1: DWORD;
cFileName: packed array[0..MAX_PATH - 1] of AnsiChar
cAlternateFileName: packed array[0..13] of AnsiChar;
end;
```

e also: <u>WIN32 FIND DATA</u>

WindowInfo

in | structures | WindowInfo

The **WindowInfo** stucture for Delphi:

```
TWindowInfo = packed record
  Pos: Integer;
  Type: Integer;
  Modified: Integer;
  Current: Integer;
  TypeName: array[0..63] of char;
  Name: array[0..Pred(NM)] of char;
end;
PWindowInfo = ^TWindowInfo;
```

FarInt64

in | structures

FARINT64 structure for Delphi:

```
TFarInt64Part = packed record
LowPart : DWORD;
HighPart : DWORD;
end;
TFarInt64 = packed record
case Integer of
{$IFDEF USE_DELPHI4}
0 : (i64 : Int64);
{$ENDIF}
1 : (Part : TFarInt64Part);
end;
```

ViewerInfo

in | structures

ViewerInfo structure for Delphi:

```
TViewerInfo = packed record
StructSize : Integer;
ViewerID : Integer;
FileName : PChar;
FileSize : TFarInt64;
FilePos : TFarInt64;
WindowSizeX : Integer;
WindowSizeY : Integer;
Options : DWORD;
TabSize : Integer;
CurMode : TViewerMode;
LeftPos : Integer;
Reserved3 : DWORD;
end;
PViewerInfo = ^TViewerInfo;
```

ViewerMode

in | structures

ViewerMode structure for Delphi:

```
TViewerMode = packed record
UseDecodeTable : Integer;
TableNum : Integer;
AnsiMode : Integer;
Unicode : Integer;
Wrap : Integer;
WordWrap : Integer;
Hex : Integer;
Reserved : array [0..3] of DWORD;
end;
```

PViewerMode = ^TViewerMode;

ViewerSelect

in | structures

ViewerSelect structure for Delphi:

```
TViewerSelect = packed record
BlockStartPos : TFarInt64;
BlockLen : Integer;
end;
PViewerSelect = ^TViewerSelect;
```

ViewerSetMode

in | structures

ViewerSetMode structure for Delphi:

```
TViewerSetMode = packed record
ParamType : Integer;
Param : record case Integer of
0 : (iParam : Integer);
1 : (cParam : PChar);
end;
Flags : DWORD;
Reserved : DWORD;
end;
PViewerSetMode = ^TViewerSetMode;
```

ViewerSetPosition

in | structures

ViewerSetPosition structure for Delphi:

```
TViewerSetPosition = packed record
Flags : DWORD;
StartPos : TFarInt64;
LeftPos : Integer;
end;
PViewerSetPosition = ^TViewerSetPosition;
```

olor indexes

in | types and definitions

This table lists the FAR Manager color scheme indexes, located in the registry at HKCU\Software\Far\Colors\CurrentPalette (see farcolor.hpp, PaletteColors enum).

The hexadecimal color values of the default color scheme are provided in the "Color" column.

Constant	Co	olor	Description
COL_DIALOGBOXTITLE	Text	0x70	Dialog.Title
COL_DIALOGHIGHLIGHTBOXTITLE	Text	0x7E	Dialog.Highlighted title
COL_DIALOGBOX	Text	0x70	Dialog.Border
COL_DIALOGTEXT	Text	0x70	Dialog.Normal text
COL_DIALOGHIGHLIGHTTEXT	Text	0x7E	Dialog.Highlighted text
COL_DIALOGDISABLED	Text	0x78	Dialog.Disabled text
COL_DIALOGSELECTEDBUTTON	Text	0x30	Dialog.Button.Selected text
COL_DIALOGHIGHLIGHTSELECTEDBUTTON	Text	0x3E	Dialog.Button.Selected highlighted text
COL_DIALOGHIGHLIGHTBUTTON	Text	0x7E	Dialog.Button.Highlighted text
COL_DIALOGBUTTON	Text	0x70	Dialog.Button.Normal text
COL_DIALOGEDITUNCHANGED	Text	0x38	Dialog.Input.Unchanged text
COL_DIALOGEDITSELECTED	Text	0x0F	Dialog.Input.Selected text
COL_DIALOGEDITDISABLED	Text	0x38	Dialog.Input.Disabled test
COL_DIALOGEDIT	Text	0x30	Dialog.Input.Normal text
COL_DIALOGLISTTITLE	Text	0x70	Dialog.Listbox.Title
COL_DIALOGLISTSELECTEDTEXT	Text	0x0F	Dialog.Listbox.Selected text
COL_DIALOGLISTSELECTEDHIGHLIGHT	Text	0x0E	Dialog.Listbox.Selected highlighted text
COL_DIALOGLISTHIGHLIGHT	Text	0x7E	Dialog.Listbox.Highlighted text
COL_DIALOGLISTBOX	Text	0x70	Dialog.Listbox.Border
COL_DIALOGLISTDISABLED	Text	0x78	Dialog.Listbox.Disabled text
COL_DIALOGLISTSCROLLBAR	Text	0x70	Dialog.Listbox.Scrollbar
COL_DIALOGLISTTEXT	Text	0x70	Dialog.Listbox.Normal text
COL_DIALOGCOMBOTITLE	Text	0x3F	Dialog.Combobox.Title
COL_DIALOGCOMBOSELECTEDTEXT	Text	0x0F	Dialog.Combobox.Selected text
COL_DIALOGCOMBOSELECTEDHIGHLIGHT	Text	0x0E	Dialog.Combobox.Selected

			highlighted text
COL_DIALOGCOMBOHIGHLIGHT	Text	0x3E	Dialog.Combobox.Highlighted text
COL_DIALOGCOMBOBOX	Text	0x3F	Dialog.Combobox.Border
COL_DIALOGCOMBODISABLED	Text	0x38	Dialog.Combobox.Disabled text
COL_DIALOGCOMBOSCROLLBAR	Text	0x3F	Dialog.Combobox.Scrollbar
COL_DIALOGCOMBOTEXT	Text	0x3F	Dialog.Combobox.Normal text
COL_MENUDISABLEDTEXT	Text	0x38	Menu.Disabled item
COL_MENUTITLE	Text	0x3F	Menu.Title
COL_MENUSELECTEDHIGHLIGHT	Text	0x0E	Menu.Selected highlighted text
COL_MENUSELECTEDTEXT	Text	0x0F	Menu.Selected text
COL_MENUHIGHLIGHT	Text	0x3E	Menu.Highlighted text
COL_MENUBOX	Text	0x3F	Menu.Border
COL_MENUSCROLLBAR	Text	0x3F	Menu.Scrollbar
COL_MENUTEXT	Text	0x3F	Menu.Normal text
COL_HMENUSELECTEDTEXT	Text	0x0F	Horizontal menu.Selected text
COL_HMENUSELECTEDHIGHLIGHT	Text	0x0E	Horizontal menu.Selected highlighted text
COL_HMENUHIGHLIGHT	Text	0x3E	Horizontal menu.Highlighted text
COL_HMENUTEXT	Text	0x30	Horizontal menu.Normal text
COL_KEYBARTEXT	Text	0x30	Key bar.Key names
COL_KEYBARNUM	Text	0x07	Key bar.Key numbers
COL_KEYBARBACKGROUND	Text	0x07	Key bar.Background
COL_WARNDIALOGBOXTITLE	Text	0x4F	Warning message.Title
COL_WARNDIALOGHIGHLIGHTTEXT	Text	0x4E	Warning message.Highlighted text
COL_WARNDIALOGLISTTITLE	Text	0x4F	Warning message.Listbox.Title
COL_WARNDIALOGLISTSELECTEDTEXT	Text	0x70	Warning message.Listbox.Selected text
COL_WARNDIALOGLISTSELECTEDHIGHLIGHT	Text	0x7E	Warning message.Listbox.Selected highlighted text
COL_WARNDIALOGLISTHIGHLIGHT	Text	0x4E	Warning message.Listbox.Highlighted text
COL_WARNDIALOGLISTBOX	Text	0x4F	Warning message.Listbox.Border
COL_WARNDIALOGLISTDISABLED	Text	0x48	Warning message.Listbox.Disabled text
COL_WARNDIALOGLISTSCROLLBAR	Text	0x4F	Warning

			message.Listbox.Scrollbar
COL_WARNDIALOGLISTTEXT	Text	0x4F	Warning message.Listbox.Normal text
COL_WARNDIALOGHIGHLIGHTBOXTITLE	Text	0x4E	Warning message.Highlighted title
COL_WARNDIALOGBOX	Text	0x4F	Warning message.Border
COL_WARNDIALOGDISABLED	Text	0x48	Warning message.Disabled text
COL_WARNDIALOGSELECTEDBUTTON	Text	0x70	Warning message.Button.Selected text
COL_WARNDIALOGHIGHLIGHTSELECTEDBUTTON	Text	0x7E	Warning message.Button.Selected highlighted text
COL_WARNDIALOGHIGHLIGHTBUTTON	Text	0x4E	Warning message.Button.Highlighted text
COL_WARNDIALOGBUTTON	Text	0x4F	Warning message.Button.Normal text
COL_WARNDIALOGEDITUNCHANGED	Text	0x38	Warning message.Input.Unchanged text
COL_WARNDIALOGEDITSELECTED	Text	0x0F	Warning message.Input.Selected text
COL_WARNDIALOGEDITDISABLED	Text	0x38	Warning message.Input.Disabled text
COL_WARNDIALOGEDIT	Text	0x30	Warning message.Input.Normal text
COL_WARNDIALOGCOMBOTITLE	Text	0x3F	Warning message.Combobox.Title
COL_WARNDIALOGCOMBOSELECTEDTEXT	Text	0x0F	Warning message.Combobox.Selected text
COL_WARNDIALOGCOMBOSELECTEDHIGHLIGHT	Text	0x0E	Warning message.Combobox.Selected highlighted text
COL_WARNDIALOGCOMBOHIGHLIGHT	Text	0x3E	Warning message.Combobox.Highlighted text
COL_WARNDIALOGCOMBOBOX	Text	0x3F	Warning message.Combobox.Border
COL_WARNDIALOGCOMBODISABLED	Text	0x38	Warning message.Combobox.Disabled text
COL_WARNDIALOGCOMBOSCROLLBAR	Text	0x3F	Warning message.Combobox.Scrollbar
COL_WARNDIALOGCOMBOTEXT	Text	0x3F	Warning message.Combobox.Normal text
COL_WARNDIALOGTEXT	Text	0x4F	Warning message.Normal text

COL_VIEWERSELECTEDTEXT	Text	0x30	Viewer.Selected text
COL_VIEWERARROWS	Text	0x1E	Viewer.Screen scrolling arrows
COL_VIEWERSTATUS	Text	0x30	Viewer.Status line
COL_VIEWERSCROLLBAR	Text	0x1B	Viewer.Scrollbar
COL_VIEWERTEXT	Text	0x1B	Viewer.Normal text
COL_PANELHIGHLIGHTTEXT	Text	0x17	Panel.(not used)
COL_PANELCOLUMNTITLE	Text	0x1E	Panel.Column title
COL_PANELSELECTEDTEXT	Text	0x1E	Panel.Selected text
COL_PANELSELECTEDTITLE	Text	0x30	Panel.Selected title
COL_PANELSELECTEDCURSOR	Text	0x3E	Panel.Selected cursor
COL_PANELINFOTEXT	Text	0x1E	Panel.Highlighted info
COL_PANELBOX	Text	0x1B	Panel.Border
COL_PANELDRAGTEXT	Text	0x3E	Panel.Dragging text
COL_PANELTOTALINFO	Text	0x1B	Panel.Total info
COL_PANELSCREENSNUMBER	Text	0x0B	Panel.Number of background screens
COL_PANELSELECTEDINF0	Text	0x3E	Panel.Selected info
COL_PANELSCROLLBAR	Text	0x1B	Panel.Scrollbar
COL_PANELTEXT	Text	0x1B	Panel.Normal text
COL_PANELTITLE	Text	0x1B	Panel.Normal title
COL_PANELCURSOR	Text	0x30	Panel.Nirmal cursor
COL_EDITORSELECTEDTEXT	Text	0x30	Editor.Selected text
COL_EDITORSTATUS	Text	0x30	Editor.Status line
COL_EDITORTEXT	Text	0x1B	Editor.Normal text
COL_COMMANDLINESELECTED	Text	0x70	Command line.Selected text
COL_COMMANDLINEPREFIX	Text	0x07	Command line.Prefix text
COL_COMMANDLINE	Text	0x07	Command line.Normal text
COL_HELPBOXTITLE	Text	0x30	Help.Title
COL_HELPSELECTEDTOPIC	Text	0x0F	Help.Selected link
COL_HELPHIGHLIGHTTEXT	Text	0x3F	Help.Highlighted text
COL_HELPTOPIC	Text	0x3E	Help.Link
COL_HELPBOX	Text	0x30	Help.Border
COL_HELPSCROLLBAR	Text	0x30	Help.Scrollbar
COL_HELPTEXT	Text	0x30	Help.Normal text
COL_VIEWERCLOCK	Text	0x30	Clock.Viewer
COL_EDITORCLOCK	Text	0x30	Clock.Editor

COL_CLOCK	Text	0x30	Clock.Panel
COL_DIALOGLISTARROWS	Text	0x30	Dialog.List box.Long string indicators
COL_DIALOGLISTARROWSDISABLED	Text	0x30	Dialog.List box.Long string indicators.Disabled item
COL_DIALOGLISTARROWSSELECTED	Text	0x0E	Dialog.List box.Long string indicators.Selected item
COL_DIALOGCOMBOARROWS	Text	0x3E	Dialog.Combobox.Long string indicators
COL_DIALOGCOMBOARROWSDISABLED	Text	0x38	Dialog.Combobox.Long string indicators.Disabled item
COL_DIALOGCOMBOARROWSSELECTED	Text	0x0E	Dialog.Combobox.Long string indicators.Selected item
COL_WARNDIALOGLISTARROWS	Text	0x4E	Warning message.List box.Long string indicators
COL_WARNDIALOGLISTARROWSDISABLED	Text	0x48	Warning message.List box.Long string indicators.Disabled item
COL_WARNDIALOGLISTARROWSSELECTED	Text	0x7E	Warning message.List box.Long string indicators.Selected item
COL_WARNDIALOGCOMBOARROWS	Text	0x3E	Warning message.Combobox.Long string indicators
COL_WARNDIALOGCOMBOARROWSDISABLED	Text	0x38	Warning message.Combobox.Long string indicators.Disabled item
COL_WARNDIALOGCOMBOARROWSSELECTED	Text	0x0E	Warning message.Combobox.Long string indicators.Selected item
COL_MENUARROWS	Text	0x3E	Menu.Long string indicators
COL_MENUARROWSDISABLED	Text	0x38	Menu.Long string indicators.Disabled item
COL_MENUARROWSSELECTED	Text	0x0E	Menu.Long string indicators.Selected item
COL_COMMANDLINEUSERSCREEN	Text	0x07	Command line.User screen
COL_EDITORSCROLLBAR	Text	0x1B	Editor.Scrollbar
COL_MENUGRAYTEXT	Text	0x38	Menu.Gray text
COL_MENUSELECTEDGRAYTEXT	Text	0x07	Menu.Selected gray text
COL_DIALOGCOMBOGRAY	Text	0x38	Dialog.Combobox.Gray text
COL_DIALOGCOMBOSELECTEDGRAYTEXT	Text	0x07	Dialog.Combobox.Selected gray text
COL_DIALOGLISTGRAY	Text	0x78	Dialog.List box.Gray text

COL_DIALOGLISTSELECTEDGRAYTEXT	Text	0x07	Dialog.List box.Selected gray text
COL_WARNDIALOGCOMBOGRAY	Text	0x38	Warning message.Combobox.Gray text
COL_WARNDIALOGCOMBOSELECTEDGRAYTEXT	Text	0x07	Warning message.Combobox.Selected gray text
COL_WARNDIALOGLISTGRAY	Text	0x48	Warning message.List box.Gray text
COL_WARNDIALOGLISTSELECTEDGRAYTEXT	Text	0x70	Warning message.List box.Selected gray text
COL_RESERVED0		0x00	(reserved for internal needs)

AR Manager key codes

in | types and definitions | virtual key codes

This table shows the hexadecimal key codes used in FAR manager (in farkeys.hpp the BaseDefKeyboard enum).

asic set:

Key	KEY_*	Hex	Remarks
Ctrl	KEY_CTRL	01000000	Left Ctrl
Alt	KEY_ALT	02000000	Left Alt
Shift	KEY_SHIFT	0400000	
Right Ctrl	KEY_RCTRL	1000000	Right Ctrl
Right Alt	KEY_RALT	20000000	Right Alt
[KEY_BRACKET	0000005B	
]	KEY_BACKBRACKET	0000005D	
,	KEY_COMMA	0000002C	
"	KEY_QUOTE	00000022	
	KEY_DOT	0000002E	
/	KEY_SLASH	0000002F	
:	KEY_COLON	000003A	
\	KEY_BACKSLASH	0000005C	
Backspace	KEY_BS	0000008	
Tab	KEY_TAB	00000009	
Enter	KEY_ENTER	0000000D	
Esc	KEY_ESC	0000001B	
Space	KEY_SPACE	00000020	
Break	KEY_BREAK	00000103	Ctrl-Pause
Page Up	KEY_PGUP	00000121	
Page Down	KEY_PGDN	00000122	
End	KEY_END	00000123	
Home	KEY_HOME	00000124	
Left	KEY_LEFT	00000125	
UP	KEY_UP	00000126	
Right	KEY_RIGHT	00000127	
Down	KEY_DOWN	00000128	
Insert	KEY_INS	0000012D	
Delete	KEY_DEL	0000012E	
Left Win	KEY_LWIN	0000015B	
Right Win	KEY_RWIN	0000015C	
Apps	KEY APPS	0000015D	

Numpad 0	KEY_NUMPAD0	00000160	Numeric keypad (if "UseNumPad" option is on)
Numpad 1	KEY_NUMPAD1	00000161	Numeric keypad (if "UseNumPad" option is on)
Numpad 2	KEY_NUMPAD2	00000162	Numeric keypad (if "UseNumPad" option is on)
Numpad 3	KEY_NUMPAD3	00000163	Numeric keypad (if "UseNumPad" option is on)
Numpad 4	KEY_NUMPAD2	00000164	Numeric keypad (if "UseNumPad" option is on)
Numpad 5	KEY_NUMPAD5	00000165	Numeric keypad
Numpad 6	KEY_NUMPAD6	00000166	Numeric keypad (if "UseNumPad" option is on)
Numpad 7	KEY_NUMPAD7	00000167	Numeric keypad (if "UseNumPad" option is on)
Numpad 8	KEY_NUMPAD8	00000168	Numeric keypad (if "UseNumPad" option is on)
Numpad 9	KEY_NUMPAD9	00000169	Numeric keypad (if "UseNumPad" option is on)
	KEY_CLEAR	00000165	Same as KEY_NUMPAD5
Gray *	KEY_MULTIPLY	0000016A	Numeric keypad
Gray +	KEY_ADD	0000016B	Numeric keypad
Gray -	KEY_SUBTRACT	0000016D	Numeric keypad
Gray /	KEY_DIVIDE	0000016F	Numeric keypad
F1	KEY_F1	00000170	
F2	KEY_F2	00000171	
F3	KEY_F3	00000172	
F4	KEY_F4	00000173	
F5	KEY_F5	00000174	
F6	KEY_F6	00000175	
F7	KEY_F7	00000176	
F8	KEY_F8	00000177	
F9	KEY_F9	00000178	
F10	KEY_F10	00000179	
F11	KEY_F11	0000017A	
F12	KEY_F12	0000017B	
F13	KEY_F13	0000017C	
F14	KEY_F14	0000017D	
F15	KEY_F15	0000017E	
F16	KEY_F16	0000017F	
F17	KEY_F17	00000180	
F18	KEY_F18	00000181	
	_		

F19	KEY_F19	00000182	
F21	KEY_F20	00000183	
F22	KEY_F22	00000184	
F23	KEY_F23	00000185	
F24	KEY_F24	00000186	
	KEY_BROWSER_BACK	000001A6	Same as VK_BROWSER_BACK
	KEY_BROWSER_FORWARD	000001A7	Same as VK_BROWSER_FORWARD
	KEY_BROWSER_REFRESH	000001A8	Same as VK_BROWSER_REFRESH
	KEY_BROWSER_STOP	000001A9	Same as VK_BROWSER_STOP
	KEY_BROWSER_SEARCH	000001AA	Same as VK_BROWSER_SEARCH
	KEY_BROWSER_FAVORITES	000001AB	Same as VK_BROWSER_FAVORITES
	KEY_BROWSER_HOME	000001AC	Same as VK_BROWSER_HOME
	KEY_VOLUME_MUTE	000001AD	Same as VK_VOLUME_MUTE
	KEY_VOLUME_DOWN	000001AE	Same as VK_VOLUME_DOWN
	KEY_VOLUME_UP	000001AF	Same as VK_VOLUME_UP
	KEY_MEDIA_NEXT_TRACK	000001B0	Same as VK_MEDIA_NEXT_TRACK
	KEY_MEDIA_PREV_TRACK	000001B1	Same as VK_MEDIA_PREV_TRACK
	KEY_MEDIA_STOP	000001B2	Same as VK_MEDIA_STOP
	KEY_MEDIA_PLAY_PAUSE	000001B3	Same as VK_MEDIA_PLAY_PAUSE
	KEY_LAUNCH_MAIL	000001B4	Same as VK_LAUNCH_MAIL
	KEY_LAUNCH_MEDIA_SELECT	000001B5	Same as VK_LAUNCH_MEDIA_SELECT
	KEY_LAUNCH_APP1	000001B6	Same as VK_LAUNCH_APP1
	KEY_LAUNCH_APP2	000001B7	Same as VK_LAUNCH_APP2
		000001XX	Other special keys that have a <u>virtual code</u> other than 0xFF, are formed by the following formula: "KEY_FKEY_BEGIN" + "Virtual code". In <u>macros</u> such keys are stored as "OemXXXXX" (here XXXXX is the virtual key code).
	KEY_CTRLALTSHIFTPRESS	00000201	All three keys are pressed
	KEY_CTRLALTSHIFTRELEASE	00000202	All the three keys were released
	KEY_MSWHEEL_UP	00000203	The mouse wheel is rotated one notch up
	KEY_MSWHEEL_DOWN	00000204	The mouse wheel is rotated one notch down
	KEY_NUMDEL	00000209	Del on the numpad when NumLock is off
	KEY_DECIMAL	0000020A	Del on the numpad when NumLock is on
	KEY_NUMENTER	0000020B	Enter on the numpad
	KEY_MSWHEEL_LEFT	0000020C	The mouse wheel is rotated one notch left
	KEY_MSWHEEL_RIGHT	0000020D	The mouse wheel is rotated one notch right
	KEY_STANDBY	0000020E	Same as VK_SLEEP
	KEY_MSLCLICK	0000020F	Click left mouse button (only for macros - shortcuts and within macro sequences)

K	EY_MSRCLICK	00000210	Click right mouse button (only for macros - shortcuts and within macro sequences)
K	EY_MSM1CLICK	00000211	Click middle (next to left) mouse button (only for macros - shortcuts and within macro sequences)
K	EY_MSM2CLICK	00000212	Click third after left mouse button (only for macros - shortcuts and within macro sequences)
K	EY_MSM3CLICK	00000213	Click fourth after left mouse button (only for macros - shortcuts and within macro sequences)
K	EY_VK_0xFF_BEGIN	00000300	Beginning of special keys definitions that have a <u>virtual code</u> of 0xFF (i.e. misc. multimedia keys which are added by keyboard manufacturers). Key code is formed using the following formula: "KEY_VK_0xFF_BEGIN" + ScanCode. In <u>macros</u> those keys are saved as "SpecXXXXX" (here XXXXX is the scan code of the key).
K	EY_VK_0xFF_END	000003FF	Ending of special keys definitions.
K	EY_NONE	00001001	Idle
K	EY_IDLE	00001002	Idle
K	EY_END_SKEY	0000FFFF	The end of basic set

irtual key codes

in | FAR Manager key codes

The following table shows the symbolic constant names, hexadecimal values, and mouse or keyboard equivalents for the <u>virtual-key codes</u> used by the system. The codes are listed in numeric order.

Symbolic constant name	Value (hex)	Mouse or keyboard equivalents
VK_LBUTTON	01	Left mouse button
VK_RBUTTON	02	Right mouse button
VK_CANCEL	03	Control-break processing
VK_MBUTTON	04	Middle mouse button (three-button mouse)
VK_XBUTTON1	05	Windows 2000/XP/2003/Vista/2008/7: X1 mouse button
VK_XBUTTON2	06	Windows 2000/XP/2003/Vista/2008/7: X2 mouse button
-	07	Undefined
VK_BACK	08	BACKSPACE key
VK_TAB	09	TAB key
-	0A-0B	Reserved
VK_CLEAR	0C	CLEAR key
VK_RETURN	0D	ENTER key
-	0E-0F	Undefined
VK_SHIFT	10	SHIFT key
VK_CONTROL	11	CTRL key
VK_MENU	12	ALT key
VK_PAUSE	13	PAUSE key
VK_CAPITAL	14	CAPS LOCK key
VK_KANA	15	Input Method Editor (IME) Kana mode
VK_HANGUEL	15	IME Hanguel mode (maintained for compatibility; use VK_HANGUL)
VK_HANGUL	15	IME Hangul mode
-	16	Undefined
VK_JUNJA	17	IME Junja mode
VK_FINAL	18	IME final mode
VK_HANJA	19	IME Hanja mode
VK_KANJI	19	IME Kanji mode
-	1A	Undefined
VK_ESCAPE	1B	ESC key
VK_CONVERT	1C	IME convert (Reserved for Kanji systems)
VK_NONCONVERT	1D	IME nonconvert (Reserved for Kanji systems)
VK_ACCEPT	1E	IME accept (Reserved for Kanji systems)
VK_MODECHANGE	1F	IME mode change request (Reserved for Kanji systems)
VK_SPACE	20	SPACEBAR

VK_PRIOR	21	PAGE UP key
VK_NEXT	22	PAGE DOWN key
VK_END	23	END key
VK_HOME	24	HOME key
VK_LEFT	25	LEFT ARROW key
VK_UP	26	UP ARROW key
VK_RIGHT	27	RIGHT ARROW key
VK_DOWN	28	DOWN ARROW key
VK_SELECT	29	SELECT key
VK_PRINT	2A	PRINT key
VK_EXECUTE	2B	EXECUTE key
VK_SNAPSHOT	2C	PRINT SCREEN key for Windows 3.0 and later
VK_INSERT	2D	INS key
VK_DELETE	2E	DEL key
VK_HELP	2F	HELP key
VK_0	30	0 key
VK_1	31	1 key
VK_2	32	2 key
VK_3	33	3 key
VK_4	34	4 key
VK_5	35	5 key
VK_6	36	6 key
VK_7	37	7 key
VK_8	38	8 key
VK_9	39	9 key
-	3A-40	Undefined
VK_A	41	A key
VK_B	42	B key
VK_C	43	C key
VK_D	44	D key
VK_E	45	E key
VK_F	46	F key
VK_G	47	G key
VK_H	48	H key
VK_I	49	I key
VK_J	4A	J key
VK_K	4B	K key
VK_L	4C	L key
VK_M	4D	M key
VK_N	4E	N key
VK_O	4F	O key

VK_P	50	P key
VK_Q	51	Q key
VK_R	52	R key
VK_S	53	S key
VK_T	54	T key
VK_U	55	U key
VK_V	56	V key
VK_W	57	W key
VK_X	58	X key
VK_Y	59	Y key
VK_Z	5A	Z key
VK_LWIN	5B	Left Windows key (Microsoft Natural Keyboard)
VK_RWIN	5C	Right Windows key (Microsoft Natural Keyboard)
VK_APPS	5D	Applications key (Microsoft Natural Keyboard)
-	5E	Reserved
VK_SLEEP	5F	Computer Sleep key
VK_NUMPAD0	60	Numeric keypad 0 key
VK_NUMPAD1	61	Numeric keypad 1 key
VK_NUMPAD2	62	Numeric keypad 2 key
VK_NUMPAD3	63	Numeric keypad 3 key
VK_NUMPAD4	64	Numeric keypad 4 key
VK_NUMPAD5	65	Numeric keypad 5 key
VK_NUMPAD6	66	Numeric keypad 6 key
VK_NUMPAD7	67	Numeric keypad 7 key
VK_NUMPAD8	68	Numeric keypad 8 key
VK_NUMPAD9	69	Numeric keypad 9 key
VK_MULTIPLY	6A	Multiply key
VK_ADD	6B	Add key
VK_SEPARATOR	6C	Separator key
VK_SUBTRACT	6D	Subtract key
VK_DECIMAL	6E	Decimal key
VK_DIVIDE	6F	Divide key
VK_F1	70	F1 key
VK_F2	71	F2 key
VK_F3	72	F3 key
VK_F4	73	F4 key
VK_F5	74	F5 key
VK_F6	75	F6 key
VK_F7	76	F7 key
VK_F8	77	F8 key
VK_F9	78	F9 key

VK_F10	79	F10 key
VK_F11	7A	F11 key
VK_F12	7B	F12 key
VK_F13	7C	F13 key
VK_F14	7D	F14 key
VK_F15	7E	F15 key
VK_F16	7F	F16 key
VK_F17	80H	F17 key
VK_F18	81H	F18 key
VK_F19	82H	F19 key
VK_F20	83H	F20 key
VK_F21	84H	F21 key
VK_F22	85H	F22 key
VK_F23	86H	F23 key
VK_F24	87H	F24 key
-	88-8F	Unassigned
VK_NUMLOCK	90	NUM LOCK key
VK_SCROLL	91	SCROLL LOCK key
VK_OEM_NEC_EQUAL	92	NEC PC-9800 kbd definitions: '=' key on numpad
VK_OEM_FJ_JISHO	92	Fujitsu/OASYS kbd definitions: 'Dictionary' key
VK_OEM_FJ_MASSHOU	93	Fujitsu/OASYS kbd definitions: 'Unregister word' key
VK_OEM_FJ_TOUROKU	94	Fujitsu/OASYS kbd definitions: 'Register word' key
VK_OEM_FJ_LOYA	95	Fujitsu/OASYS kbd definitions: 'Left OYAYUBI' key
VK_OEM_FJ_ROYA	96	Fujitsu/OASYS kbd definitions: 'Right OYAYUBI' key
-	97-9F	Unassigned
VK_LSHIFT	A0	Left SHIFT key
VK_RSHIFT	A1	Right SHIFT key
VK_LCONTROL	A2	Left CONTROL key
VK_RCONTROL	A3	Right CONTROL key
VK_LMENU	A4	Left MENU key
VK_RMENU	A5	Right MENU key
VK_BROWSER_BACK	A6	Windows 2000/XP/2003/Vista/2008/7: Browser Back key
VK_BROWSER_FORWARD	A7	Windows 2000/XP/2003/Vista/2008/7: Browser Forward key
VK_BROWSER_REFRESH	A8	Windows 2000/XP/2003/Vista/2008/7: Browser Refresh key
VK_BROWSER_STOP	A9	Windows 2000/XP/2003/Vista/2008/7: Browser Stop key
VK_BROWSER_SEARCH	AA	Windows 2000/XP/2003/Vista/2008/7: Browser Search key
VK_BROWSER_FAVORITES	AB	Windows 2000/XP/2003/Vista/2008/7: Browser Favorites key

VK_BROWSER_HOME	AC	Windows 2000/XP/2003/Vista/2008/7: Browser Start and Home key
VK_VOLUME_MUTE	AD	Windows 2000/XP/2003/Vista/2008/7: Volume Mute key
VK_VOLUME_DOWN	AE	Windows 2000/XP/2003/Vista/2008/7: Volume Down key
VK_VOLUME_UP	AF	Windows 2000/XP/2003/Vista/2008/7: Volume Up key
VK_MEDIA_NEXT_TRACK	B0	Windows 2000/XP/2003/Vista/2008/7: Next Track key
VK_MEDIA_PREV_TRACK	B1	Windows 2000/XP/2003/Vista/2008/7: Previous Track key
VK_MEDIA_STOP	B2	Windows 2000/XP/2003/Vista/2008/7: Stop Media key
VK_MEDIA_PLAY_PAUSE	B3	Windows 2000/XP/2003/Vista/2008/7: Play/Pause Media key
VK_LAUNCH_MAIL	B4	Windows 2000/XP/2003/Vista/2008/7: Start Mail key
VK_LAUNCH_MEDIA_SELECT	B5	Windows 2000/XP/2003/Vista/2008/7: Select Media key
VK_LAUNCH_APP1	B6	Windows 2000/XP/2003/Vista/2008/7: Start Application 1 key
VK_LAUNCH_APP2	B7	Windows 2000/XP/2003/Vista/2008/7: Start Application 2 key
-	B8-B9	Reserved
VK_OEM_1	BA	Windows 2000/XP/2003/Vista/2008/7: For the US standard keyboard, the ';:' key
VK_OEM_PLUS	BB	Windows 2000/XP/2003/Vista/2008/7: For any country/region, the '+' key
VK_OEM_COMMA	BC	Windows 2000/XP/2003/Vista/2008/7: For any country/region, the ',' key
VK_OEM_MINUS	BD	Windows 2000/XP/2003/Vista/2008/7: For any country/region, the '-' key
VK_OEM_PERIOD	BE	Windows 2000/XP/2003/Vista/2008/7: For any country/region, the '.' key
VK_OEM_2	BF	Windows 2000/XP/2003/Vista/2008/7: For the US standard keyboard, the '/?' key
VK_OEM_3	C0	Windows 2000/XP/2003/Vista/2008/7: For the US standard keyboard, the '`~' key
-	C1-D7	Reserved
-	D8-DA	Unassigned
VK_OEM_4	DB	Windows 2000/XP/2003/Vista/2008/7: For the US standard keyboard, the '[{' key
VK_OEM_5	DC	Windows 2000/XP/2003/Vista/2008/7: For the US standard keyboard, the '\ ' key
VK_OEM_6	DD	Windows 2000/XP/2003/Vista/2008/7: For the US standard keyboard, the ']}' key
VK_OEM_7	DE	Windows 2000/XP/2003/Vista/2008/7: For the US standard keyboard, the 'single-quote/double-quote' key
VK_OEM_8	DF	Used for miscellaneous characters; it can vary by keyboard.
-	E0	Reserved

	E1	OEM specific
VK_OEM_102	E2	Windows 2000/XP/2003/Vista/2008/7: Either the angle bracket key or the backslash key on the RT 102-key keyboard
-	E3-E4	OEM specific
VK_PROCESSKEY	E5	Windows 95/98/Me, Windows NT/2000/XP/2003/Vista/2008/7: IME PROCESS key
-	E6	OEM specific
VK_PACKET	E7	Windows 2000/XP/2003/Vista/2008/7: Used to pass Unicode characters as if they were keystrokes. The VK_PACKET key is the low word of a 32-bit Virtual Key value used for non-keyboard input methods. For more information, see Remark in <u>KEYBDINPUT</u> ,
		<u>SendInput</u> ,
		WM KEYDOWN , and
		<u>WM KEYUP</u>

	50	
	E8	Unassigned
VK_OEM_RESET	E9	Only used by Nokia.
VK_OEM_JUMP	EA	Only used by Nokia.
VK_OEM_PA1	EB	Only used by Nokia.
VK_OEM_PA2	EC	Only used by Nokia.
VK_OEM_PA3	ED	Only used by Nokia.
VK_OEM_WSCTRL	EE	Only used by Nokia.
VK_OEM_CUSEL	EF	Only used by Nokia.
VK_OEM_ATTN	F0	Only used by Nokia.
VK_OEM_FINNISH	F1	Only used by Nokia.
VK_OEM_COPY	F2	Only used by Nokia.
VK_OEM_AUTO	F3	Only used by Nokia.
VK_OEM_ENLW	F4	Only used by Nokia.
VK_OEM_BACKTAB	F5	Only used by Nokia.
VK_ATTN	F6	Attn key
VK_CRSEL	F7	CrSel key
VK_EXSEL	F8	ExSel key
VK_EREOF	F9	Erase EOF key
VK_PLAY	FA	Play key
VK_ZOOM	FB	Zoom key
VK_NONAME	FC	Reserved for future use.
VK_PA1	FD	PA1 key
VK_OEM_CLEAR	FE	Clear key
	FF	Multimedia keys. See ScanCode keys.

peration mode

in | types and definitions

The **OpMode** parameter passes to plugin additional information about function operation mode and place, from which it was called. It can be a combination of the following values (OPERATION_MODES enum):

Mode	Description
OPM_SILENT	Plugin should minimize user requests if possible, because the called function is only a part of a more complex file operation.
OPM_FIND	Plugin function is called from Find file or another directory scanning command. Screen output has to be minimized.
OPM_VIEW	Plugin function is called as part of a file view operation. If file is viewed on quickview panel, than both OPM_VIEW and OPM_QUICKVIEW are set.
OPM_QUICKVIEW	Plugin function is called as part of a file view operation activated from the quick view panel (activated by pressing Ctrl-Q in the file panels).
OPM_EDIT	Plugin function is called as part of a file edit operation.
OPM_DESCR	Plugin function is called to get or put file with file descriptions.
OPM_TOPLEVEL	All files in host file of file based plugin should be processed. This flag is set when executing Shift-F2 and Shift-F3 FAR commands outside of host file. Passed to plugin functions files list also contains all necessary information, so plugin can either ignore this flag or use it to speed up processing.

e also: <u>SetDirectory</u>, <u>PutFiles</u>, <u>ProcessHostFile</u>, <u>GetFiles</u>, <u>DeleteFiles</u>,

GetFindData, MakeDirectory

orting methods

in | types and definitions

Sorting method can be one of the following values (OPENPLUGININFO_SORTMODES enum):

Method	Description
SM_DEFAULT	Default sort mode
SM_UNSORTED	Unsorted
SM_NAME	Sort by name
SM_EXT	Sort by extension
SM_MTIME	Sort by file modification time
SM_CTIME	Sort by file creation time
SM_ATIME	Sotr by last file access time
SM_SIZE	Sort by size
SM_DESCR	sotr by description
SM_OWNER	Sort by owner
SM_COMPRESSEDSIZE	Sort be compressed size
SM_NUMLINKS	Sort by number of hard file links

e also:

Compare | OpenPluginInfo | PanelInfo
AR_PKF_FLAGS

in | types and definitions

The members of the FAR_PKF_FLAGS enumeration describe the state of the shift keys of an event sent to the <u>ProcessKey</u> function.

Flag	Description
PKF_CONTROL	Ctrl is pressed
PKF_ALT	Alt is pressed
PKF_SHIFT	Shift is pressed
PKF_PREPROCESS	Preprocessing: - FAR passes a "raw" keystroke. This flag is applicable <u>only</u> to the <u>virtual key code</u> (second parameter of the <u>ProcessKey</u> function).

emarks

- Since FAR Manager 1.70 build 2052 keyboard events are sent to the plugins with no exclusions (refer to the remarks on the <u>ProcessKey</u> function). If the **PKF_PREPROCESS** flag is set, plugin may ignore calls to the <u>ProcessKey</u> function. In this case after the input is complete FAR will form the needed command and pass it to the plugin. For example, if a user enters "Cd ...Enter" in the command line the plugin receives the sequence
 "80043h 80044h 80020h 800BEh 800BEh 8000Dh" (every virtual code has the **PKF_PREPROCESS** flag set). The plugin may behave in two ways:
 - 1. process the sequence by itself;
 - ignore the calls to <u>ProcessKey</u> with **PKF_PREPROCESS** set and wait for FAR to call <u>SetDirectory</u> with Dir = "..".

e also:

ProcessKey

FAR_NO_NAMELESS_UNIONS

in | types and definitions

The macro _FAR_NO_NAMELESS_UNIONS controls whether the <u>FarDialogItem</u> structure uses anonymous unions. Anonymous unions are a language feature that is allowed by the C++ standard but not supported in ANSI C.

If the macro **_FAR_NO_NAMELESS_UNIONS** is **not defined**, the FarDialogItem structure will be compatible with FAR Manager versions prior to FAR 1.70 beta 3 (inclusive). So the <u>FarDialogItem</u> structure will have the following form:

```
struct FarDialogItem
{
  . . .
  union {
    int Selected;
    char *History;
    char *Mask;
    struct FarList *ListItems;
    CHAR_INFO *VBuf;
  };
  . . .
  union {
    char Data[512];
    struct {
      DWORD PtrFlags;
      int PtrLength;
      char *PtrData;
             PtrTail[1];
      char
    } Ptr;
  };
};
```

So to access the Data member of the <u>FarDialogItem</u> structure it will be suficient to write Data, and to access the Selected member - Selected.

If the macro **_FAR_NO_NAMELESS_UNIONS** is **defined**, the structure will

use named unions. Then it will be compatible with ANSI C compilers, but will not be source-level compatible with plugins written for FAR 1.65. The structure will have the following form:

```
struct FarDialogItem
{
  . . .
  union {
    int Selected;
    char *History;
    char *Mask;
    struct FarList *ListItems;
    CHAR_INFO *VBuf;
  } Param;
  . . .
  union {
    char Data[512];
    struct {
      DWORD PtrFlags;
            PtrLength;
      int
      char *PtrData;
      char
             PtrTail[1];
    } Ptr;
  } Data;
};
```

In this case to access the Data member of the structure you will have to write Data.Data, and to access the Selected member - Param.Selected.

The macro must be defined before the **#include** "plugin.hpp" directive:

```
#define _FAR_NO_NAMELESS_UNIONS
#include "plugin.hpp"
```

▲ Attention! In FAR 1.70 beta 4, the default variant is compatible with old plugins (_FAR_NO_NAMELESS_UNIONS is not defined). However, in FAR 1.70 release the new default will be _FAR_NO_NAMELESS_UNIONS. So if you want your plugins to be source-level compatible with FAR 1.70 release, you can right now define the **_FAR_NO_NAMELESS_UNIONS** macro and modify the source code of your plugins accordingly.

e also: <u>GetMinFarVersion</u>

ARMANAGERVERSION

in | types and definitions

The **FARMANAGERVERSION** constant defines the current FAR Manager version and has the following format - **0**×**BBBBXXYY**:

BBBB = build number (343 = 0x0157) XX = major version (FAR 1.70 = 0x01) YY = minor version (FAR 1.70 = 0x46)

So for FAR Manager 1.70 beta 3 build 343 this constant will be: 0x01570146

The **FARMANAGERVERSION** constant is formed using the <u>MAKEFARVERSION</u> macro.

e also: <u>GetMinFarVersion</u>

IAKEFARVERSION

in | types and definitions

The macro **MAKEFARVERSION** is intended to be used in the <u>GetMinFarVersion</u> function to return the minimal FAR Manager version needed to run the plugin.

MAKEFARVERSION(major,minor,build)

e also: <u>GetMinFarVersion</u>

arConfirmationsSettings

in | types and definitions

Information about the confirmation settings (FarConfirmationsSettings enum). Corresponds to options in the "Confirmations" dialog.

Constant	Description
FCS_COPYOVERWRITE	"Overwrite files when copying"
FCS_MOVEOVERWRITE	"Overwritte files when moving"
FCS_DRAGANDDROP	"Drag and drop"
FCS_DELETE	"Delete"
FCS_DELETENONEMPTYFOLDERS	"Delete non-empty folders"
FCS_INTERRUPTOPERATION	"Interrupt operation"
FCS_DISCONNECTNETWORKDRIVE	"Disconnect network drive"
FCS_RELOADEDITEDFILE	"Reload edited file"
FCS_CLEARHISTORYLIST	"Clear history list"
FCS_EXIT	Exit

e also:

ACTL GETCONFIRMATIONS

arInterfaceSettings

in | types and definitions

Information about the interface settings (FarInterfaceSettings enum). Corresponds to options in the "Interface settings" dialog.

Constant	Description
FIS_CLOCKINPANELS	"Clock in panels"
FIS_CLOCKINVIEWERANDEDITOR	"Clock in viewer and editor"
FIS_MOUSE	"Mouse"
FIS_SHOWKEYBAR	"Show key bar"
FIS_ALWAYSSHOWMENUBAR	"Always show menu bar, even when it's inactive"
FIS_USERIGHTALTASALTGR	"Use right Alt as AltGr"
FIS_SHOWTOTALCOPYPROGRESSINDICATOR	"Show total copy progress indicator"
FIS_SHOWCOPYINGTIMEINFO	"Show copying time information"
FIS_USECTRLPGUPTOCHANGEDRIVE	"Use Ctrl-PgUp to change drive"

e also:

ACTL GETINTERFACESETTINGS

arDialogSettings

in | types and definitions

Information about the dialog settings (FarDialogSettings enum). Corresponds to options in the "Dialog Settings" dialog.

Constant	Description
FDIS_AUTOCOMPLETEININPUTLINES	"AutoComplete in edit controls"
FDIS_HISTORYINDIALOGEDITCONTROLS	"History in dialog edit controls" (applies to some internal dialogs)
FDIS_PERSISTENTBLOCKSINEDITCONTROLS	"Persistent blocks in edit controls"
FDIS_BSDELETEUNCHANGEDTEXT	"Backspace deletes unchanged text". If this option is turned on, pressing BackSpace inside an input line with unchanged text will delete the whole line as if Del was pressed.
FDIS_DELREMOVESBLOCKS	"Del removes blocks in edit controls"
FDIS_MOUSECLICKOUTSIDECLOSESDIALOG	"Mouse click outside a dialog closes it"

e also:

ACTL GETDIALOGSETTINGS

arDescriptionSettings

in | types and definitions

Information about the file description settings (FarDescriptionSettings enum). Corresponds to options in the "File descriptions" dialog.

Constant	Description
FDS_SETHIDDEN	"Set "Hidden" attribute to new description lists"
FDS_UPDATEALWAYS	"Always update descriptions"
FDS_UPDATEIFDISPLAYED	"Update descriptions if displayed"
FDS_UPDATEREADONLY	"Update read only description file"

emarks

The **FDS_UPDATEALWAYS** and **FDS_UPDATEIFDISPLAYED** flags are mutually exclusive.

e also:

ACTL_GETDESCSETTINGS

arSystemSettings

in | types and definitions

Information about the system settings (FarSystemSettings enum). Corresponds to options in the "System settings" dialog.

Constant	Description
FSS_CLEARROATTRIBUTE	"Clear R/O attribute from CD files"
FSS_DELETETORECYCLEBIN	"Delete to Recycle Bin"
FSS_USESYSTEMCOPYROUTINE	"Use system copy routine"
FSS_COPYFILESOPENEDFORWRITING	"Copy files opened for writing"
FSS_SCANSYMLINK	"Scan symbolic links"
FSS_CREATEFOLDERSINUPPERCASE	"Create folders in uppercase"
FSS_SAVECOMMANDSHISTORY	"Save commands history"
FSS_SAVEFOLDERSHISTORY	"Save folders history"
FSS_SAVEVIEWANDEDITHISTORY	"Save view and edit history"
FSS_USEWINDOWSREGISTEREDTYPES	"Use Windows registered types"
FSS_AUTOSAVESETUP	"Auto save setup"

e also:

ACTL GETSYSTEMSETTINGS

arPanelSettings

in | types and definitions

Information about the panel settings (FarPanelSettings enum). Corresponds to options in the "Panel settings" dialog.

Constant	Description
FPS_SHOWHIDDENANDSYSTEMFILES	"Show hidden and system files"
FPS_HIGHLIGHTFILES	"Highlight files"
FPS_AUTOCHANGEFOLDER	"Auto change folder"
FPS_SELECTFOLDERS	"Select folders"
FPS_ALLOWREVERSESORTMODES	"Allow reverse sort modes"
FPS_SHOWCOLUMNTITLES	"Show column titles"
FPS_SHOWSTATUSLINE	"Show status line"
FPS_SHOWFILESTOTALINFORMATION	"Show files total information"
FPS_SHOWFREESIZE	"Show free space"
FPS_SHOWSCROLLBAR	"Show scrollbar"
FPS_SHOWBACKGROUNDSCREENSNUMBER	"Show background screens number"
FPS_SHOWSORTMODELETTER	"Show sort mode letter"

e also: ACTL GETPANELSETTINGS

ile masks

in

File masks are frequently used in FAR commands to select a single file and/or folder or a group files and/or folders. Masks may contain common valid file name symbols, wildcards ('*' and '?') and special expressions:

Expression	Description
*	Zero or more characters.
?	Any single character.
[c,x-z]	Any character enclosed in the brackets. Both lists and ranges of characters are allowed.

For example, files ftp.exe, fc.exe and f.ext may be selected using the mask f*.ex?, the mask *co* will select both color.ini and edit.com, the mask [c-f,t]*.txt will select config.txt, demo.txt, faq.txt and tips.txt.

In many FAR commands you may enter several file masks separated by commas or semicolons. For example, to select all the documents, you can enter *.doc, *.txt, *.wri in the "Select" command.

It is allowed to put any of the masks (in a list) in quotes (but not the whole list). For example, you have to do this when a mask contains any of the delimiting characters (a comma or a semicolon), so that the mask isn't confused with a list of masks.

In some commands (find files, file selection, file associations, sort groups and file highlighting) you may use exclude masks. An **exclude mask is one or multiple file masks that must not be matched by the needed files. The exclude mask is delimited from the main mask by the '|' character.**

Usage examples of exclude masks:

- 1. ***.cpp**All files with the cpp extension.
- 2. *.*|*.bak,*.tmp

All files except for the files with bak and tmp extensions.

3. *.*|

This mask has an error - the character '|', is entered, but the mask itself is not specified.

4. *.* [*.bak|*.tmp

Also an error - the character '|' may not be specified in the mask more than once.

5. **|*.bak**

The same as '* | * . bak'

elp topic syntax

<u>in | Help files</u>

The *HelpTopic* parameter describes a help topic and can be in one of the following formats:

Format	Description
"Topic"	Reference to a topic in the plugins help file.
":Topic"	Reference to a topic from the main FAR Manager help file.
" <fullpath>Topic"</fullpath>	Reference to a topic in a help file located in a folder with full or relative path of FullPath . An ending backslash must be added. The reference must not be split on mutiple lines. For example, the plugin Foo is located in folder 'D:\FAR\Plugins\Foo' and we need to show the topic 'FooInfo' from its help file: " <d:\far\plugins\foo\>FooInfo"</d:\far\plugins\foo\>
" <fullmodulename>Topic"</fullmodulename>	Reference to a topic in a help file located in the same folder as the plugin with the relative or full path of FullModuleName . The reference must not be split on mutiple lines. For example, we need to show the help topic 'FooInfo' from the help file of the plugin Foo 'D:\FAR\Plugins\Foo\Foo.dll':
	" <d:\far\plugins\foo\foo.dll>FooInfo"</d:\far\plugins\foo\foo.dll>

e also:

ShowHelp, Dialog, DialogEx, DN_HELP, InputBox, Menu, Message

ontrol statements

in | language and help files

In the beginning, language and help files can contain the following control statements, starting from a dot character.

Control statement	Description	
.Language	.Language= <language name="">,<language description=""></language></language>	
	This statement must be present at the beginning of all language and help files.	
	<language name=""></language>	
	describes the file language and must be a standard language name in English. All file <language name=""> field.</language>	
	<language description=""></language>	
	can contain a language description in arbitrary form. It will be displayed in the Lanç	
.PluginContents	.PluginContents= <contents name="" topic=""></contents>	
	This optional statement can be used to add the <i><contents name="" topic=""></contents></i> entry to the plug Shift-F2 is pressed. After choosing this entry, the topic Contents of the plugin he Contents topic).	
.Options	.Options <keyname>=<value> This optional statement can be used to specify additional options in help files. There ar</value></keyname>	
	<keyname></keyname>	
	One of the following options:	
	 CtrlColorChar <value> contains the character that will be used to specify the <u>files</u>" about the CtrlColorChar option). For example specifying:</value> 	
	.Options CtrlColorChar=\	
	sets the ∖ character to be the color specifying character. • TabSize	
	 <value> specifies the tab size in the HLF file. Must be in the range of 1 to 16,</value> CtrlStartPosChar 	
	< value > contains the character that will be used to mark a block alignment po	
	. Options CtristartPoschar=&	
	means that the '&' character will mark a block alignment position, then the blc	
	item 1 - &Joe's father is strong in math, he stud	
	will be aligned as follows:	

xample:

```
.Language=Engish,English
.PluginContents=FTP client
@Contents
$ #FTP client#
   ~Connecting to an FTP server~@FTPConnect@
   ~Working with server names~@FTPNames@
   ~FTP client commands~@FTPCmd@
   ~FTP client configuration~@FTPCfg@
   ~FTP client panel modes~@FTPPanel
```

e also:

Language and Help files, Help files

anguage files

in | language and help files | GetMsg

The language file (a text file with the . LNG extension) is intended to store language resources used by the plugin to output messages in dialogs and menus.

Messages in language files must be enclosed in double quotes. You can use the double quote character inside messages as well.

All lines not beginning with a dot or a double quote are ignored. Leading spaces are ignored.

```
.Language=English,English
"Please register your copy"
"Registered"
"Yes"
"No"
:
//functional keys - 6 characters max
"Help"
"UserMn"
:
"Group"
"SelUp"
//End of functional keys
```

emarks

1. When using double quotes inside messages it is not obligatory to prepend them with a backslash. While processing each line of language file, FAR Manager checks only the opening and closing quotes. For example:

```
"Option "Autodetect character table" is off."
"Copy \"%.55s\" to"
```

both lines are correct;

- 2. a line may not be longer than 1000 characters;
- 3. messages may not be split on mutiple lines;
- 4. the following control charachters are allowed: '\n', '\r', '\\', '\b', '\t'

e also:

Control statements, Language and Help files, GetMsg

elp files

in | language and help files

elp file syntax.

The following control statements can be used in help files.

Control statement	Description
<i>@</i> Topic (at the beginning of a line)	Starts a topic definition. There are four topics with special names:
	 The topic with the name Contents has a special meaning. I the FAR command line when the plugin is active or when a plugins help list. If a plugin can be configured, it is recommended to specify topic for the configuration dialog. This topic will be shown plugins configuration menu (Options Plugins configuration If a plugin can be invoked both in the panels and in the edit functions depending on where it was invoked, it is recomm Viewer for describing the operation in the viewer and editc be shown when Shift-F1 is pressed in the list of plugin editor.
\$Text (at the beginning of a line)	Defines a non-scrolling region. All lines starting with \$ must be in (immediately after the line starting with @) and will be shown in from the rest of the text with a horizontal line.
~Text~@Topic@	Reference to a topic. If you wish to access a topic from the main I plugins help, precede the topic name with a colon (':'). The reference must not be split on mutiple lines.
~Text~@ <fullpath>Topic@</fullpath>	Reference to a topic in a help file located at a folder with full or reending backslash must be added. The reference must not be split on mutiple lines. For example, the plugin Foo is located in folder 'D:\FAR\Plugins\' topic 'FooInfo' from its help file.
	~About Foo~@ <d:\far\plugins\foo\>FooInfo@</d:\far\plugins\foo\>
~Text~@ <fullmodulename>Topic@</fullmodulename>	Reference to a topic in a help file located at the same folder as the path of FullModuleName . The reference must not be split on mutiple lines. For example, we need to show the help topic 'FooInfo' from the help 'D:\FAR\Plugins\Foo\Foo.dll'.
	~ADOUT FOO~@ <d:\fak\p1ug1ns\foo\foo.dll>FO</d:\fak\p1ug1ns\foo\foo.dll>
~Text~@URL@	URL activator, allowing to run applications that support URL pro- the protocols that can be used in help files:

	<pre>~File access protocol~@file://C:\Program F ~File transfer protocol~@ftp://ftp.kgb.ru/ ~HTTP~@http://plugring.farmanager.com/@ ~MailTo~@mailto:vskirdin@@mail.ru@ ~News~@news://fido7.far.support@ ~Telnet~@telnet://fido7.far.support@</pre>	
	The reference must not be split on mutiple lines.	
#Text#	Highlights the text Text .	
^ (at the beginning of a line or after\$)	Centers the line.	
<i>@</i> - (at the beginning of a line)	Disables text auto format. By default FAR formats all lines in whi position (is not indented). Must be placed in a separate line.	
@+ (at the beginning of a line)	Enables text auto format. Must be placed in a separate line.	
<ctrlcolorchar>XX</ctrlcolorchar>	Specifies a color attribute that will be used to display the text fold consists of two hexadecimal digits (0-9A-F). For example, the backslash ('\') character is set to be the color spec (CtrlColorChar >). Then the statement \4F will force the help is text with white letters on dark red background. (see <u>CtrlColorChar</u>)	
<ctrlcolorchar>-</ctrlcolorchar>	Specifies that the following text must be displayed in default colo (see <u>CtrlColorChar</u>)	

emarks

- 1. The length of a text string in a help file must not exceed 300 characters.
- If you need to display the characters ~, # or @, duplicate them (~~, ##, @@).
- 3. In the <URL> field of URL activators, the ~ and # characters may be duplicated or specified once, but the @ character must be always duplicated. If the URL must contain a sequence of two ~ or # characters, specify a sequence of 3 or 4 characters (for example, ~~~ and ~~~~ will be shown as ~~).
- 4. Don't use special characters ~, # or @ inside a reference that is not an URL activator.

xample

The following example is taken from the FarEng.hlf file.

```
@FolderShortcuts
$ #Folder shortcuts#
```

Folder shortcuts are designed to provide fast acc used folders. Press Ctrl-Shift-0..9, to create a shor to the current folder. To change to the folder record press RightCtrl-0..9. If RightCtrl-0..9 pressed in ed the shortcut path into the line.

The #Show folder shortcuts# item in the **~Commands** used to view, set, edit and delete folder shortcuts.

It looks like this:

Ссылки на папки
Ссылки на папки позволяют обеспечить быстрый доступ к часто используемым папкам. Для создания ссылки на текущую папку нужно нажать Ctrl-Shift- <n>, где N - '0''9'. После этого, чтобы перейти в папку, записанную в ссылке, достаточно нажать ПравыйCtrl-<n>. Если ПравыйCtrl-<n> нажат в строке редактирования, то путь ссылки будет вставлен в эту строку.</n></n></n>
Пункт Ссылки на папки в <mark>Меню команд</mark> позволяет просматривать, устанавливать, редактировать и удалять ссылки на папки.

The following examples demonstrates usage of the URL activator. E-mail client activation:

~vskirdin@@mail.ru~@mailto:vskirdin@@mail.ru@

Browser activation:

```
~http://plugring.farmanager.com/~@plugring.farmanager
```

or

~http://www.uic.nnov.ru/~~ruiv/plugring/~@http://www.

The following example demonstrates usage of color attributes:

```
.Language=English,English
.PluginContents=Reversi - Game
.Options CtrlColorChar=\
```

```
@-
    Reversi, also known as Othello, is a strateg
    \70 B \-\2F W \- played by two players: Black and
    \2F W \-\70 B \- #White#. It is played on an 8x8 k
        using 64 disks with different color on each
@+
```

It looks like this:



e also: <u>Control statements</u>, <u>Language files</u>

/in32 structures

in

Structure	Description		
CHAR_INFO	specifies the character and its attributes		
CONSOLE CURSOR INFO	contains information about the console cursor		
COORD	defines the coordinates of a character cell in a console screen buffer		
<u>FILETIME</u>	the 64-bit number of 100-nanosecond intervals since January 1, 1601 (UTC)		
FOCUS EVENT RECORD	reports focus events in a console <u>INPUT RECORD</u> structure		
INPUT RECORD	reports input events in the console input buffer		
KEY EVENT RECORD	reports keyboard input events in a console <u>INPUT_RECORD</u> structure		
MENU EVENT RECORD	reports menu events in a console <u>INPUT_RECORD</u> structure		
MOUSE EVENT RECORD	reports mouse input events in a console <u>INPUT RECORD</u> structure		
RECT	defines the coordinates of the upper-left and lower- right corners of a rectangle		
SMALL RECT	defines the coordinates of the upper-left and lower- right corners of a rectangle		
<u>SYSTEMTIME</u>	represents a date and time using individual members for the month, day, year, weekday, hour, minute, second, and millisecond		
WIN32 FIND DATA	describes a file found by the FindFirstFile , FindFirstFileEx , or FindNextFile function		
WINDOW BUFFER SIZE RECORD	reports changes in the size of the screen buffer in a console INPUT RECORD structure		

e also:

Exported functions Service functions Addons

HAR_INFO

in | <u>structures</u> | <u>win32 structures</u>

The **CHAR_INFO** structure specifies the Unicode or ANSI character and the colour attributes of the screen character cell. This structure is used by console functions to read from and write to a console screen buffer.

```
typedef struct _CHAR_INFO {
   // Unicode or ANSI character
   union {
     WCHAR UnicodeChar;
     CHAR AsciiChar;
   } Char;
   // Text and background colors
   WORD Attributes;
} CHAR_INFO, *PCHAR_INFO;
```

embers

Char

Unicode (wide-character) or ANSI character of a screen buffer character cell, depending on whether it is used with the Unicode or ANSI version of a function.

Attributes

Character attributes. There are two classes of the attributes - colour and DBCS. This member can be zero or any combination of the following attributes: (all of them are defined in Wincon.h).

Attribute	Description
FOREGROUND_BLUE	Text color contains blue.
FOREGROUND_GREEN	Text color contains green.
FOREGROUND_RED	Text color contains red.
FOREGROUND_INTENSITY	Text color is intensified.
BACKGROUND_BLUE	Background color contains blue.
BACKGROUND_GREEN	Background color contains green.
BACKGROUND_RED	Background color contains red.

BACKGROUND_INTENSITY	Background color is intensified.
COMMON_LVB_LEADING_BYTE	DBCS: Leading byte.
COMMON_LVB_TRAILING_BYTE	DBCS: Trailing byte.
COMMON_LVB_GRID_HORIZONTAL	DBCS: Grid attribute: top horizontal.
COMMON_LVB_GRID_LVERTICAL	DBCS: Grid attribute: left vertical.
COMMON_LVB_GRID_RVERTICAL	DBCS: Grid attribute: right vertical.
COMMON_LVB_REVERSE_VIDEO	DBCS: Reverse foreground and background attributes.
COMMON_LVB_UNDERSCORE	DBCS: Underscore.

The foreground attributes (FOREGROUND_*) define the colour of the text symbols. The background attributes (BACKGROUND_*) define the colour of the background of the text cell. Other attributes (COMMON_LVB_*) are used with **DBCS**.

,

,

emarks

e also: <u>ReadConsoleOutput</u>

<u>ScrollConsoleScreenBuffer</u>

<u>WriteConsoleOutput</u>

ONSOLE_CURSOR_INFO

in | structures | win32 structures

The **CONSOLE_CURSOR_INFO** structure contains information about the console cursor.

```
typedef struct _CONSOLE_CURSOR_INFO {
   DWORD dwSize;
   BOOL bVisible;
} CONSOLE_CURSOR_INFO, *PCONSOLE_CURSOR_INFO;
```

embers

dwSize

Percentage of the character cell that is filled by the cursor. This value is between 1 and 100. The cursor appearance varies, ranging from completely filling the cell to showing up as a horizontal line at the bottom of the cell.

Windows 9X/Me To show a fully filled cursor in Windows 9x/Me set this value to 99.

bVisible

Visibility of the cursor. If the cursor is visible, this member is TRUE.

e also:

<u>GetConsoleCursorInfo</u>

<u>SetConsoleCursorInfo</u>

OORD

in | structures | win32 structures

The **COORD** structure defines the coordinates of a character cell in a console screen buffer. The origin of the coordinate system (0,0) is at the top, left cell of the buffer.

```
typedef struct _COORD {
   SHORT X;
   SHORT Y;
} COORD;
```

embers

X

Horizontal coordinate or column value.

Y

Vertical coordinate or row value.

emarks

e also:

ILETIME

in | structures | win32 structures

The **FILETIME** data structure is a 64-bit value representing the number of 100nanosecond intervals since January 1, 1601. It is the means by which Win32 determines the date and time.

```
typedef struct _FILETIME {
   DWORD dwLowDateTime;
   DWORD dwHighDateTime;
} FILETIME;
```

embers

dwLowDateTime

Specifies the low-order 32 bits of the Win32 date/time value.

dwHighDateTime

Specifies the high-order 32 bits of the Win32 date/time value.

emarks

It is not recommended that you add or substract values from this structure to obtain relative times. Instead, you should do the following:

• Copy this structure to a <u>ULARGE_INTEGER</u>

structure.

• Use standard 64-bit arithmetic on the <u>ULARGE_INTEGER</u>

value or cast a variable of

```
FILETIME type to the __int64 type:
FILETIME WriteTime1, WriteTime2;
if(*(__int64*) & WriteTime1 == *(__int64*) & Write
...
```

Not all file systems can record creation and last access time and not all file systems record them in the same manner. For example, on NT FAT, create time has a resolution of 10 milliseconds, write time has a resolution of 2 seconds, and access time has a resolution of 1 day (really, the access date). On NTFS, access time has a resolution of 1 hour. Therefore, the GetFileTime function may not return the same file time information set using the SetFileTime function. Furthermore, FAT records times on disk in local time. However, NTFS records times on disk in UTC.

e also: <u>CompareFileTime</u>

, <u>GetFileTime</u>,

SetFileTime, ULARGE INTEGER

OCUS_EVENT_RECORD

in | structures | win32 structures | INPUT_RECORD

The **FOCUS_EVENT_RECORD** structure is used to report focus events in a console **INPUT_RECORD** structure. These events are used internally and should be ignored.

```
typedef struct _FOCUS_EVENT_RECORD {
    BOOL bSetFocus;
} FOCUS_EVENT_RECORD;
```

embers

bSetFocus

Reserved.

e also:

INPUT RECORD

IPUT_RECORD

in | structures | win32 structures

The **INPUT_RECORD** structure is used to report input events in the console input buffer. These records can be read from the input buffer by using the <u>ReadConsoleInput</u> or <u>PeekConsoleInput</u> function, or written to the input buffer by using the <u>WriteConsoleInput</u> function.

```
typedef struct _INPUT_RECORD {
  WORD EventType;
  union {
     KEY_EVENT_RECORD KeyEvent;
     MOUSE_EVENT_RECORD MouseEvent;
     WINDOW_BUFFER_SIZE_RECORD WindowBufferSizeEvent;
     MENU_EVENT_RECORD MenuEvent;
     FOCUS_EVENT_RECORD FocusEvent;
   } Event;
} INPUT_RECORD;
```

embers

EventType

Handle to the type of input event and the event record stored in the Event member.

This member can be one of the following values.

Value	Description
KEY_EVENT	The Event member contains a <u>KEY EVENT RECORD</u> structure with information about a keyboard event.
MOUSE_EVENT	The Event member contains a <u>MOUSE EVENT RECORD</u> structure with information about a mouse movement or button press event.
WINDOW_BUFFER_SIZE_EVENT	The Event member contains a <u>WINDOW BUFFER SIZE RECORD</u> structure with information about the new size of the screen buffer.
MENU_EVENT	The Event member contains a <u>MENU EVENT RECORD</u> structure. These events are used internally and should be ignored.

FOCUS_EVENT	The Event member contains a <u>FOCUS EVENT RECORD</u> structure. These events are used internally and should be ignored.
FARMACRO_KEY_EVENT	The Event member contains a <u>KEY EVENT RECORD</u> structure with information about a keyboard event. Plugin receives this specific message from FAR manager version 1.70 build 1663 and higher while playing keyboard macro.

Event

Event information. The format of this member depends on the event type specified by the **EventType** member.

e also:

KEY EVENT RECORD, MOUSE EVENT RECORD, WINDOW BUFFER SIZE RECORD, MENU EVENT RECORD, FOCUS EVENT RECORD, PeekConsoleInput, ReadConsoleInput, WriteConsoleInput

EY_EVENT_RECORD

in | structures | win32 structures | INPUT_RECORD

The **KEY_EVENT_RECORD** structure is used to report keyboard input events in a console **INPUT_RECORD** structure.

```
typedef struct _KEY_EVENT_RECORD {
   BOOL bKeyDown;
   WORD wRepeatCount;
   WORD wVirtualKeyCode;
   WORD wVirtualScanCode;
   union {
        WCHAR UnicodeChar;
        CHAR AsciiChar;
        } uChar;
        DWORD dwControlKeyState;
} KEY_EVENT_RECORD;
```

embers

bKeyDown

Indicates whether a key is down. This member is TRUE if the key is pressed, or FALSE if the key is released.

wRepeatCount

Count indicating that a key is being held down. For example, when a key is held down, you might get five events with this member equal to 1, one event with this member equal to 5, or multiple events with this member greater than or equal to 1.

wVirtualKeyCode

<u>Virtual-key code</u> that identifies the given key in a device-independent manner.

wVirtualScanCode

Virtual scan code of the given key that represents the device-dependent value generated by the keyboard hardware.

uChar

Translated Unicode or ASCII character, depending on whether the widecharacter (Unicode) or ANSI version of the <u>ReadConsoleInput</u> function was used. *dwControlKeyState*

Indicates the state of the control keys. This member can be one or more of the following values.

Key	Value	Description
CAPSLOCK_ON	0x0080	The CAPS LOCK light is on.
ENHANCED_KEY	0x0100	The key is enhanced.
LEFT_ALT_PRESSED	0x0002	The left ALT key is pressed.
LEFT_CTRL_PRESSED	0x0008	The left CTRL key is pressed.
NUMLOCK_ON	0x0020	The NUM LOCK light is on.
RIGHT_ALT_PRESSED	0x0001	The right ALT key is pressed.
RIGHT_CTRL_PRESSED	0x0004	The right CTRL key is pressed.
SCROLLLOCK_ON	0x0040	The SCROLL LOCK light is on.
SHIFT_PRESSED	0x0010	The SHIFT key is pressed.

emarks

Enhanced keys for the IBM[®] 101- and 102-key keyboards are the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and direction keys in the clusters to the left of the keypad; and the divide (/) and ENTER keys in the keypad. Keyboard input events are generated when any key, including control keys, is pressed or released. However, the ALT key when pressed and released without combining with another character, has special meaning to the system and is not passed through to the application. Also, the CTRL+C key combination is not passed through if the input handle is in processed mode (ENABLE_PROCESSED_INPUT).

e also: <u>PeekConsoleInput</u>, <u>ReadConsoleInput</u>, <u>WriteConsoleInput</u>, <u>INPUT RECORD</u>

IENU_EVENT_RECORD

in | structures | win32 structures | INPUT_RECORD

The **MENU_EVENT_RECORD** structure reports menu events in a console **INPUT_RECORD** structure. These events are used internally and should be ignored.

typedef struct _MENU_EVENT_RECORD {
 UINT dwCommandId;
} MENU_EVENT_RECORD, *PMENU_EVENT_RECORD;

embers

dwCommandId

Reserved.

e also:

INPUT RECORD
IOUSE_EVENT_RECORD

in | structures | win32 structures | input_record

The **MOUSE_EVENT_RECORD** structure is used in a console **INPUT_RECORD** structure to report mouse input events.

```
typedef struct _MOUSE_EVENT_RECORD {
    COORD dwMousePosition;
    DWORD dwButtonState;
    DWORD dwControlKeyState;
    DWORD dwEventFlags;
} MOUSE_EVENT_RECORD;
```

embers

dwMousePosition

Location of the cursor, in terms of the screen buffer's character-cell coordinates (see <u>COORD</u> structure).

dwButtonState

Indicates the status of the mouse buttons. The least significant bit corresponds to the leftmost mouse button. The next least significant bit corresponds to the rightmost mouse button. The next bit indicates the next-to-leftmost mouse button. The bits then correspond left to right to the mouse buttons. A bit is 1 if the button was pressed.

The following constants are defined for the first five mouse buttons: FROM_LEFT_1ST_BUTTON_PRESSED RIGHTMOST_BUTTON_PRESSED FROM_LEFT_2ND_BUTTON_PRESSED FROM_LEFT_3RD_BUTTON_PRESSED FROM_LEFT_4TH_BUTTON_PRESSED

dwControlKeyState

Indicates the state of the control keys. This member can be one or more of the following values.

Value	Description
CAPSLOCK_ON	The CAPS LOCK light is on.
ENHANCED_KEY	The key is enhanced.
LEFT_ALT_PRESSED	The left ALT key is pressed.

LEFT_CTRL_PRESSED	The left CTRL key is pressed.
NUMLOCK_ON	The NUM LOCK light is on.
RIGHT_ALT_PRESSED	The right ALT key is pressed.
RIGHT_CTRL_PRESSED	The right CTRL key is pressed.
SCROLLLOCK_ON	The SCROLL LOCK light is on.
SHIFT_PRESSED	The SHIFT key is pressed.

dwEventFlags

Indicates the type of mouse event. If this value is zero, it indicates a mouse button being pressed or released. Otherwise, this member is one of the following values.

Value	Description
DOUBLE_CLICK	The second click (button press) of a double-click occurred. The first click is returned as a regular button-press event.
MOUSE_MOVED	A change in mouse position occurred.
MOUSE_WHEELED	Windows 2000/XP/2003/Vista/2008/7: The vertical mouse wheel was moved. If this flag is set, high word of dwButtonState indicates the distance the wheel is rotated, expressed in multiples or divisions of WHEEL_DELTA.
MOUSE_HWHEELED	Windows Vista/2008/7: The horizontal mouse wheel was moved. If this flag is set, high word of dwButtonState indicates the distance the wheel is rotated, expressed in multiples or divisions of WHEEL_DELTA.

emarks

Mouse events are placed in the input buffer when the console is in mouse mode (ENABLE_MOUSE_INPUT).

Mouse events are generated whenever the user moves the mouse, or presses or releases one of the mouse buttons. Mouse events are placed in the console input buffer only when the console group has the keyboard focus and the cursor is within the borders of the console window.

e also: <u>COORD</u>, <u>PeekConsoleInput</u>, <u>ReadConsoleInput</u>, <u>WriteConsoleInput</u>, <u>INPUT_RECORD</u>

ECT

The **RECT** structure defines the coordinates of the upper-left and lower-right corners of a rectangle.

```
typedef struct _RECT {
  LONG left;
  LONG top;
  LONG right;
  LONG bottom;
} RECT, *PRECT;
```

embers

left

Specifies the X-coordinate of the upper-left corner of the rectangle.

top

Specifies the Y-coordinate of the upper-left corner of the rectangle.

right

Specifies the X-coordinate of the lower-right corner of the rectangle.

bottom

Specifies the Y-coordinate of the lower-right corner of the rectangle.

emarks

e also: SMALL RECT

MALL_RECT

in | structures | win32 structures

The **SMALL_RECT** structure defines the coordinates of the upper-left and lower-right corners of a rectangle.

```
typedef struct _SMALL_RECT {
   SHORT Left;
   SHORT Top;
   SHORT Right;
   SHORT Bottom;
} SMALL_RECT;
```

embers

Left

X-coordinate of the upper left corner of the rectangle.

Тор

Y-coordinate of the upper left corner of the rectangle.

Right

X-coordinate of the lower right corner of the rectangle.

Bottom

Y-coordinate of the lower right corner of the rectangle.

emarks

This structure is used by console functions to specify rectangular areas of console screen buffers, where the coordinates specify the rows and columns of screen-buffer character cells.

e also: <u>RECT</u>

YSTEMTIME

<u>in | structures | win32 structures</u>

The **SYSTEMTIME** structure represents a date and time using individual members for the month, day, year, weekday, hour, minute, second, and millisecond.

```
typedef struct _SYSTEMTIME {
  WORD wYear;
  WORD wMonth;
  WORD wDayOfWeek;
  WORD wDay;
  WORD wHour;
  WORD wHour;
  WORD wMinute;
  WORD wSecond;
  WORD wMilliseconds;
} SYSTEMTIME;
```

embers

wYear

Specifies the current year. The year must be greater than 1601.

```
Windows XP, Windows Server
The year cannot be greater than 30827.
```

wMonth

Specifies the current month; January = 1, February = 2, and so on.

wDayOfWeek

Specifies the current day of the week; Sunday = 0, Monday = 1, and so on.

wDay

Specifies the current day of the month.

wHour

```
Specifies the current hour (0-23).
```

wMinute

```
Specifies the current minute (0-59).
```

wSecond

Specifies the current second (0-59).

wMilliseconds

Specifies the current millisecond (0-999).

emarks

It is not recommended that you add or substract values from this structure to obtain relative times. Instead, you should do the following:

• Convert the **SYSTEMTIME** structure to a **<u>FILETIME</u>** structure using the

SystemTimeToFileTime function.
Copy the resulting FILETIME structure to a <u>ULARGE INTEGER</u>

Use standard 64-bit arithmetic on the <u>ULARGE_INTEGER</u>

value or cast a variable of

FILETIME type to the **__int64** type:

```
FILETIME WriteTime1, WriteTime2;
if(*(__int64*) & WriteTime1 == *(__int64*) & Write
...
```

e also: <u>FILETIME, LARGE INTEGER</u>

,

,

<u>GetSystemTime</u>

<u>SetSystemTime</u>

/IN32_FIND_DATA

in | structures | win32 structures

The **WIN32_FIND_DATA** structure describes a file found by the **FindFirstFile**, **FindFirstFileEx** or **FindNextFile** function.

```
typedef struct _WIN32_FIND_DATA {
   DWORD dwFileAttributes;
   FILETIME ftCreationTime;
   FILETIME ftLastAccessTime;
   FILETIME ftLastWriteTime;
   DWORD nFileSizeHigh;
   DWORD nFileSizeLow;
   DWORD dwReserved0;
   DWORD dwReserved1;
   TCHAR cFileName[ MAX_PATH ];
   TCHAR cAlternateFileName[ 14 ];
} WIN32_FIND_DATA;
```

embers

dwFileAttributes

Specifies the file attributes of the file found. This member can be one or more of the following values.

Attribute	Description
FILE_ATTRIBUTE_ARCHIVE	The file or directory is an archive file or directory. Applications use this attribute to mark files for backup or removal.
FILE_ATTRIBUTE_COMPRESSED	The file or directory is compressed. For a file, this means that all of the data in the file is compressed. For a directory, this means that compression is the default for newly created files and subdirectories.
FILE_ATTRIBUTE_DIRECTORY	The handle identifies a directory.
FILE_ATTRIBUTE_ENCRYPTED	The file or directory is encrypted. For a file, this means that all data in the file is encrypted. For a directory, this means that encryption is the

	default for newly created files and subdirectories.
FILE_ATTRIBUTE_HIDDEN	The file or directory is hidden. It is not included in an ordinary directory listing.
FILE_ATTRIBUTE_NORMAL	The file or directory has no other attributes set. This attribute is valid only if used alone.
FILE_ATTRIBUTE_OFFLINE	The file data is not immediately available. This attribute indicates that the file data has been physically moved to offline storage.
FILE_ATTRIBUTE_READONLY	The file or directory is read-only. Applications can read the file but cannot write to it or delete it. In the case of a directory, applications cannot delete it.
FILE_ATTRIBUTE_REPARSE_POINT	The file has an associated <u>reparse</u> <u>point</u> .
FILE_ATTRIBUTE_SPARSE_FILE	The file is a <u>sparse file</u> .
FILE_ATTRIBUTE_SYSTEM	The file or directory is part of the operating system or is used exclusively by the operating system.
FILE_ATTRIBUTE_TEMPORARY	The file is being used for temporary storage. File systems attempt to keep all of the data in memory for quicker access, rather than flushing it back to mass storage. A temporary file should be deleted by the application as soon as it is no longer needed.
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	The file or directory is not be indexed by the content indexing service.
FILE_ATTRIBUTE_VIRTUAL	A file is a virtual file.

ftCreationTime

A **FILETIME** structure that specifies when the file or directory was created. If the underlying file system does not support creation time, **ftCreationTime** is zero.

ftLastAccessTime

A **FILETIME** structure. For a file, the structure specifies when the file was last read from or written to. For a directory, the structure specifies when the directory was created. For both files and directories, the specified date will be correct, but the time of day will always be set to midnight. If the underlying file system does not support last access time, **ftLastAccessTime** is zero.

ftLastWriteTime

A **FILETIME** structure. For a file, the structure specifies when the file was last written to. For a directory, the structure specifies when the directory was created. If the underlying file system does not support last write time, ftLastWriteTime is zero.

nFileSizeHigh

Specifies the high-order DWORD value of the file size, in bytes. This value is zero unless the file size is greater than MAXDWORD. The size of the file is equal to (*nFileSizeHigh* * (1+MAXDWORD)) + *nFileSizeLow*.

nFileSizeLow

Specifies the low-order DWORD value of the file size, in bytes.

dwReserved0

If the *dwFileAttributes* member includes the

FILE_ATTRIBUTE_REPARSE_POINT attribute, this member specifies the reparse tag. Otherwise, this value is undefined and should not be used.

dwReserved1

Reserved.

cFileName

A null-terminated string that is the name of the file.

cAlternateFileName

A null-terminated string that is an alternative name for the file. This name is in the classic 8.3 (filename.ext) file name format.

emarks

• If a file has a long file name, the complete name appears in the *cFileName* field, and the 8.3 format truncated version of the name appears in the *cAlternateFileName* field. Otherwise *cAlternateFileName* is empty. As an alternative, you can use the <u>GetShortPathName</u>

function to find the 8.3 format

version of a file name.

• Not all file systems can record creation and last access time and not all file systems record them in the same manner. For example, on NT FAT, create time has a resolution of 10 milliseconds, write time has a resolution of 2 seconds, and access time has a resolution of 1 day (really, the access date). On NTFS, access time has a resolution of 1 hour.

e also:

FAR USE WIN32 FIND DATA, FAR FIND DATA, FILETIME, TWin32FindData

/INDOW_BUFFER_SIZE_RECORD

in | structures | win32 structures

The **WINDOW_BUFFER_SIZE_RECORD** structure is used in a console <u>INPUT_RECORD</u> structure to report changes in the size of the screen buffer.

typedef struct _WINDOW_BUFFER_SIZE_RECORD {
 COORD dwSize;
} WINDOW_BUFFER_SIZE_RECORD;

embers

dwSize

Size of the screen buffer, in character cell columns and rows.

emarks

Buffer size events are placed in the input buffer when the console is in windowaware mode (ENABLE_WINDOW_INPUT).

e also: INPUT RECORD, COORD, ReadConsoleInput

lin32 functions

in

Function	Description
<u>GetFileTime</u>	retrieves the date and time that a file was created, last accessed, and last modified
PeekConsoleInput	reads data from the specified console input buffer without removing it from the buffer
<u>ReadConsoleInput</u>	reads data from a console input buffer and removes it from the buffer
<u>SetFileApisToAnsi</u>	causes the file I/O functions to use the ANSI character set code page
<u>SetFileApisToOem</u>	causes the file I/O functions to use the OEM character set code page
<u>SetFileTime</u>	sets the date and time that a file was created, last accessed, or last modified
<u>WriteConsoleInput</u>	writes data directly to the console input buffer

e also:

Exported functions Service functions Addons

etFileTime

```
in | structures | win32 structures
```

The **GetFileTime** function retrieves the date and time that a file was created, last accessed, and last modified.

```
BOOL GetFileTime(
   HANDLE hFile,
   CONST FILETIME *lpCreationTime,
   CONST FILETIME *lpLastAccessTime,
   CONST FILETIME *lpLastWriteTime
);
```

arameters

hFile

Handle to the file for which to get dates and times. The file handle must have been created with the GENERIC_READ access to the file.

lpCreationTime

Pointer to a **FILETIME** structure to receive the date and time the file was created. This parameter can be NULL if the application does not require this information.

lpLastAccessTime

Pointer to a **FILETIME** structure to receive the date and time the file was last accessed. The last access time includes the last time the file was written to, read from, or, in the case of executable files, run. This parameter can be NULL if the application does not require this information.

lpLastWriteTime

Pointer to a **FILETIME** structure to receive the date and time the file was last written to. This parameter can be NULL if the application does not require this information.

eturn value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information,

call GetLastError

emarks

The FAT and NTFS file systems support the file creation, last access, and last write time values.

Time precision The file time precision can vary depending on operating system, file system, network configuration. See remarks for the **FILETIME** for details.

Windows NT family When Windows NT creates a list of folders (e.g. DIR command) in a NTFS volume, it modifies last access date/time for all found folders. It can degrade effectiveness if the number of folders is very large. This behaviour can be controled, see Disable the NTFS Last Access Time Stamp for details.

> If you rename or delete a file, then restore it shortly thereafter, Windows NT searches the cache for file information to restore. Cached information includes its short/long name pair and creation time.

xample

The following example demonstrates how to retrieve last-write time for a file in string form (Windows NT/2000).

```
BOOL GetLastWriteTime(HANDLE hFile, LPSTR lpszString)
Ł
  FILETIME ftCreate, ftAccess, ftWrite;
  SYSTEMTIME stUTC, stLocal;
  // get file time and date
  if (!GetFileTime(hFile, &ftCreate, &ftAccess, &ftWr
      return FALSE;
```

```
// convert modification time to local time.
FileTimeToSystemTime(&ftWrite, &stUTC);
SystemTimeToTzSpecificLocalTime(NULL, &stUTC, &stLc
// convert retrieved time to string
wsprintf(lpszString, "%02d/%02d/%d %02d:%02d",
    stLocal.wDay, stLocal.wMonth, stLocal.wYear,
    stLocal.wHour, stLocal.wMinute);
return TRUE;
```

e also: <u>FILETIME, GetFileSize</u>

}

, <u>SetFileTime</u>,

<u>GetFileType</u>

eekConsoleInput

in | <u>structures</u> | <u>win32 structures</u>

The **PeekConsoleInput** function reads data from the specified console input buffer without removing it from the buffer.

```
BOOL PeekConsoleInput(
   HANDLE hConsoleInput,
   PINPUT_RECORD lpBuffer,
   DWORD nLength,
   LPDWORD lpNumberOfEventsRead
);
```

arameters

hConsoleInput

Handle to the input buffer. The handle must have GENERIC_READ access.

lpBuffer

Pointer to an <u>INPUT_RECORD</u> buffer that receives the input buffer data.

nLength

Specifies the size, in records, of the buffer pointed to by the **lpBuffer** parameter.

lpNumberOfEventsRead

Pointer to a variable that receives the number of input records read.

eturn value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information,

call GetLastError

emarks

If the number of records requested exceeds the number of records available in the buffer, the number available is read. If no data is available, the function returns immediately.

Windows NT/2000/XP/2003/Vista/2008/7 This function uses either Unicode characters or 8-bit characters from the console's current code page. The console's code page defaults initially to the system's OEM code page. To change the console's code page, use

the SetConsoleCP

or

SetConsoleOutputCP functions, or use the chcp or mode con cp select = commands.

,

,

e also:

INPUT_RECORD, ReadConsoleInput, SetConsoleCP

<u>SetConsoleOutputCP</u>

WriteConsoleInput

eadConsoleInput

in | structures | win32 structures

The **ReadConsoleInput** function reads data from a console input buffer and removes it from the buffer.

```
BOOL ReadConsoleInput(
   HANDLE hConsoleInput,
   PINPUT_RECORD lpBuffer,
   DWORD nLength,
   LPDWORD lpNumberOfEventsRead
);
```

arameters

hConsoleInput

Handle to the input buffer. The handle must have GENERIC_READ access.

lpBuffer

Pointer to an <u>INPUT_RECORD</u> buffer that receives the input buffer data.

nLength

Specifies the size, in input records, of the buffer pointed to by the **lpBuffer** parameter.

lpNumberOfEventsRead

Pointer to a variable that receives the number of input records read.

eturn value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information,

call GetLastError

emarks

If the number of records requested in the **nLength** parameter exceeds the number of records available in the buffer, the number available is read. The function does not return until at least one input record has been read.

A process can specify a console input buffer handle in one of the *wait functions* to determine when there is unread console input. When the input buffer is not empty, the state of a console input buffer handle is signaled.

To determine the number of unread input records in a console's input buffer, use the **GetNumberOfConsoleInputEvents** function. To read input records from a console input buffer without affecting the number of unread records, use the PeekConsoleInput function. To discard all unread records in a console's input

buffer, use the FlushConsoleInputBuffer function.

Windows NT/2000/XP/Vista/2008/7 This function uses either Unicode characters or 8-bit characters from the console's current code page. The console's code page defaults initially to the system's OEM code page. To change the console's code page, use

the SetConsoleCP

or

SetConsoleOutputCP

functions,

or use the chcp or mode con cp select = commands.

e also: INPUT RECORD, SetConsoleCP

<u>SetConsoleOutputCP</u>

, <u>WriteConsoleInput</u>, <u>PeekConsoleInput</u>, <u>FlushConsoleInputBuffer</u>

,

,

<u>GetNumberOfConsoleInputEvents</u>

, <u>ReadConsole</u>

, <u>ReadFile</u>

etFileApisToANSI

in | structures | win32 structures

The **SetFileApisToANSI** function causes the file I/O functions to use the ANSI character set code page. This function is useful for 8-bit console input and output operations.

VOID SetFileApisToANSI(VOID);

arameters

This function has no parameters.

eturn value

This function has no return value.

emarks

The **SetFileApisToANSI** function complements the <u>SetFileApisToOEM</u> function, which causes file I/O functions to use the OEM character set code page.

The 8-bit console functions use the OEM code page by default. All other functions use the ANSI code page by default. This means that strings returned by the console functions may not be processed correctly by other functions, and vice versa. For example, if the **FindFirstFileA** function returns a string that contains certain extended ANSI characters, and the 8-bit console functions are set to use the OEM code page, then the **WriteConsoleA** function does not display the string properly.

Use the **AreFileApisANSI** function to determine which code page the set of file I/O functions is currently using. Use the **SetConsoleCP** and **SetConsoleOutputCP** functions to set the code page for the 8-bit console functions.

To solve the problem of code page incompatibility, it is best to use Unicode for console applications. Console applications that use Unicode are much more versatile than those that use 8-bit console functions. Barring that solution, a console application can call the **SetFileApisToOEM** to cause the set of file I/O functions to use OEM character set strings rather than ANSI character set strings. Use the **SetFileApisToANSI** to set those functions back to the ANSI code page.

The **SetFileApisToANSI** and **SetFileApisToOEM** functions affect the following set of Win32 file functions.

Function	Description
<u>lopen</u>	open existing file
CopyFile	copy file
CreateDirectory	create directory
CreateFile	create/open an object (file, pipe, etc.)
CreateProcess	create process

DeleteFile	delete file
FindFirstFile	start searching for a file object
<u>FindNextFile</u>	continue searching
GetCurrentDirectory	get the current directory
CatDialtEreas	get disk info

GetDriveType	get drive type
<u> </u>	
CatEilaAttributes	get file attributes
GettineAttributes	
C. (E. III), (I. N	
GetFullPatnName	get path to a file
<u>GetModuleFileName</u>	get full name of the module

GetModuleHandle	get handle of the module
	Set minute of the module
CatSystemDirectory	get path to the system directory.
GetSystemDirectory	get pain to the system directory
<u>GetTempFileName</u>	get name for a temporary file
GetTempPath	get path to the TEMP directory

CotVolumoInformation	got file system information
Get volumennormation	get me system mormation
CotWindowsDirectory	got path to the Windows directory
GetwindowsDirectory	get path to the whidows directory
LoadLibrary	load library (DLL)
LoadLibraryEx	load library (DLL)

MoveFile	move/rename file or directory
	move/rename me of uncerory
<u>MoveFileEx</u>	move/rename file or directory
OpenFile	create/open/delete file
<u>openine</u>	create/open/acreate file
RemoveDirectory	delete directory
	acicic unceiory

<u>SearchPath</u>	search for a file
SetCurrentDirectory	set current directory
SetFileAttributes	set file attributes

When dealing with command lines, a console application should obtain the command line in Unicode form and then convert it to OEM form using the relevant character-to-OEM functions. Note also that the array in the **argv** parameter contains ANSI character set strings in this case.

e also: <u>SetFileApisToOEM</u>

etFileApisToOEM

in | <u>structures</u> | <u>win32 structures</u>

The **SetFileApisToOEM** function causes the file I/O functions to use the OEM character set code page. This function is useful for 8-bit console input and output operations.

VOID SetFileApisToOEM(VOID);

arameters

This function has no parameters.

eturn value

This function has no return value.

emarks

The **SetFileApisToOEM** function complements the <u>SetFileApisToANSI</u> function, which causes file I/O functions to use the ANSI character set code page.

The 8-bit console functions use the OEM code page by default. All other functions use the ANSI code page by default. This means that strings returned by the console functions may not be processed correctly by other functions, and vice versa. For example, if the **FindFirstFileA** function returns a string that contains certain extended ANSI characters, and the 8-bit console functions are set to use the OEM code page, then the **WriteConsoleA** function does not display the string properly.

Use the **AreFileApisANSI** function to determine which code page the set of file I/O functions is currently using. Use the **SetConsoleCP** and **SetConsoleOutputCP** functions to set the code page for the 8-bit console functions.

To solve the problem of code page incompatibility, it is best to use Unicode for console applications. Console applications that use Unicode are much more versatile than those that use 8-bit console functions. Barring that solution, a console application can call the **SetFileApisToOEM** to cause the set of file I/O functions to use OEM character set strings rather than ANSI character set strings. Use the **SetFileApisToANSI** to set those functions back to the ANSI code page.

The **SetFileApisToANSI** and **SetFileApisToOEM** functions affect the following set of Win32 file functions.

Function	Description
<u>lopen</u>	open existing file
CopyFile	copy file
CreateDirectory	create directory
CreateFile	create/open an object (file, pipe, etc.)
CreateProcess	create process

delete file
start searching for a file object
start scarching for a file object
continue searching
get the current directory

GetDriveType	get drive type
<u> </u>	
CatEilaAttributes	get file attributes
GettineAttributes	
C. (E. III), (I. N	
GetFullPatnName	get path to a file
<u>GetModuleFileName</u>	get full name of the module

GetModuleHandle	get handle of the module
CatSystemDirectory	get path to the system directory.
GetsystemDirectory	get path to the system directory
<u>GetTempFileName</u>	get name for a temporary file
<u>GetTempPath</u>	get path to the TEMP directory

CatValume Laformation	
GetvolumeInformation	get file system information
GetWindowsDirectory	get path to the Windows directory
LoadLibrary	load library (DLL)
LoadLibraryEx	load library (DLL)
MoveFile	move/rename file or directory
-------------------	-------------------------------
	move/rename me of uncerory
<u>MoveFileEx</u>	move/rename file or directory
OpenFile	create/open/delete file
<u>openine</u>	create/open/acreate file
RemoveDirectory	delete directory
	acicic unceiory

<u>SearchPath</u>	search for a file
SetCurrentDirectory	set current directory
SetFileAttributes	set file attributes

When dealing with command lines, a console application should obtain the command line in Unicode form and then convert it to OEM form using the relevant character-to-OEM functions. Note also that the array in the **argv** parameter contains ANSI character set strings in this case.

e also: <u>SetFileApisToANSI</u>

etFileTime

```
in | structures | win32 structures
```

The **SetFileTime** function sets the date and time that a file was created, last accessed, or last modified.

```
BOOL SetFileTime(
   HANDLE hFile,
   CONST FILETIME *lpCreationTime,
   CONST FILETIME *lpLastAccessTime,
   CONST FILETIME *lpLastWriteTime
);
```

arameters

hFile

Handle to the file for which to set the dates and times. The file handle must have been created with GENERIC_WRITE access to the file.

lpCreationTime

Pointer to a **FILETIME** structure that contains the date and time the file was created. This parameter can be NULL if the application does not need to set this information.

lpLastAccessTime

Pointer to a **FILETIME** structure that contains the date and time the file was last accessed. The last access time includes the last time the file was written to, read from, or (in the case of executable files) run. This parameter can be NULL if the application does not need to set this information.

lpLastWriteTime

Pointer to a **FILETIME** structure that contains the date and time the file was last written to. This parameter can be NULL if the application does not want to set this information.

eturn value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information,

call <u>GetLastError</u>

emarks

The FAT and NTFS file systems support the file creation, last access, and last write time values.

The file time precision can vary depending on operating system, file system, network configuration. See remarks for the **FILETIME** for details.

xample

The following example sets the last-write time for a file to the current system time.

e also: <u>FILETIME, GetFileSize</u>

, <u>GetFileTime</u>,

<u>GetFileType</u>

/riteConsoleInput

in | structures | win32 structures

The **WriteConsoleInput** function writes data directly to the console input buffer.

```
BOOL WriteConsoleInput(
   HANDLE hConsoleInput,
   CONST INPUT_RECORD *lpBuffer,
   DWORD nLength,
   LPDWORD lpNumberOfEventsWritten
);
```

arameters

hConsoleInput

Handle to the console input buffer. The handle must have GENERIC_WRITE access.

lpBuffer

Pointer to an <u>INPUT_RECORD</u> buffer containing data to be written to the input buffer.

nLength

Specifies the number of input records to be written.

lpNumberOfEventsWritten

Pointer to a variable that receives the number of input records actually written.

eturn value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information,

call <u>GetLastError</u>

emarks

WriteConsoleInput places input records into the input buffer behind any pending events in the buffer. The input buffer grows dynamically, if necessary, to hold as many events as are written.

Windows NT/2000/XP/2003/Vista/2008/7 This function uses either Unicode characters or 8-bit characters from the console's current code page. The console's code page defaults initially to the system's OEM code page. To change the console's code page, use

the SetConsoleCP

or

SetConsoleOutputCP functions, or use the chcp or mode con cp select = commands.

,

,

e also:

INPUT_RECORD, ReadConsoleInput, SetConsoleCP

<u>SetConsoleOutputCP</u>

PeekConsoleInput

/in32 definitions

in

Constant	Description
Virtualkeycodes	virtual key codes

e also:

Exported functions Service functions Addons

ialog functions

in | Dialog API

The following functons are used in the dialog API.

Function	Description
Dialog	Shows "uncontrollable" dialog
<u>DialogEx</u>	Shows extended dialog
<u>SendDlgMessage</u>	Sends a message to the dialog callback function
<u>DefDlgProc</u>	Calls the default dialog callback function
DlgProc	Dialog callback function template

e also:

<u>Structures</u>, <u>Dialog items</u>, <u>Dialog item flags</u>, <u>Events and Messages</u>, <u>Exported functions</u>, <u>Service functions</u>, <u>Structures</u>, <u>Archive</u> <u>support</u>, <u>Addons</u>

ARWINDOWPROC

in | Dialog API | macros and types

The **FARWINDOWPROC** type describes the dialog window callback function.

```
typedef LONG_PTR (WINAPI *FARWINDOWPROC)(
  HANDLE hDlg,
  int Msg,
  int Param1,
  LONG_PTR Param2
);
```

```
e also:
```

<u>DialogEx</u>

I BUTTON

in | Dialog API | Dialog items

The **DI_BUTTON** dialog item describes a Push Button control.

```
struct FarDialogItem
{
  int Type
                    = DI BUTTON
  int X1
                     = X
  int Y1
                    = Y
  int X2
                    = 0 (not used in Dialog API 1.0)
  int Y2
int Focus
int Selected
                    = Y (equals to Y1)
                    = Focus
                    = Selected
  DWORD Flags = Flags
  int DefaultButton = DefaultButton
  char Data[512] = Button caption
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

DCUS

Keyboard focus flag.

elected

If the button had focus when the user pressed <Enter> this field is set to 1.

lags

There are several flags applicable to the **DI_BUTTON** item:

Flag	Description
DIF_BTNNOCLOSE	Disables dialog closing after pressing the button.
DIF_CENTERGROUP	Sequential items having this flag set and equal vertical coordinates will be horizontally centered in the dialog. Their X1 and X2 coordinates are ignored. Useful for centering button groups.
DIF_NOBRACKETS	Display button titles without brackets.

DIF_SETCOLOR	The low byte of Flags will be used as the item color.
DIF_DISABLE	Disables user access to the control.
DIF NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.
DIF SHOWAMPERSAND	Show ampersand symbol in caption instead of using it for defining hotkeys.

vent

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function just before the button condrawn.
DN CTLCOLORDLGITEM	The plugin should pass the color attributes of the button when this event an Param2 argument (foreground+background):
	LoWord LoByte - color of the caption (<u>COL_WARNDIALOGBUTTON</u> or
	COL DIALOGBUTTON)
	LOWORD HIBYTE - COLOR OF NIGNLIGHTED TEXT (COL WARNDIAL OGHIGHLIGHTEDUTION OR
	<u>COL DIALOGHIGHLIGHTBUTTON</u>)
	Param2, when the button has focus:
	LoWord LoByte - color of the caption (<u>COL_WARNDIALOGSELECTEDBUTTON</u> or
	COL_DIALOGSELECTEDBUTTON)
	COL WARNDIALOGHIGHLIGHTSELECTEDBU
	COL_DIALOGHIGHLIGHTSELECTEDBUTTON)
	HiWord LoByte - 0
	HiWord HiByte - 0
	If the special <u>DIF_SETCOLOR</u> flag is used, then the button's caption (LoV LoByte) will be drawn according to the above settings.
DN KEY	This event comes after the user has pressed a key in the dialog.
DN HOTKEY	A hotkey was pressed (Alt- <letter>).</letter>
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outsi dialog with a mouse button.
DN BTNCLICK	The button was pressed.
DN KILLFOCUS	This event is sent before the button loses the focus, if the flag <u>DIF_NOFO</u> was not used.

The **FarDialogItem** structure is described assuming the use of named unions. For more information, see <u>FAR_NO_NAMELESS_UNIONS</u>.

e also:

DI CHECKBOX, DI RADIOBUTTON, FarDialogItem

I CHECKBOX

in | Dialog API | Dialog items

The **DI_CHECKBOX** dialog item describes a Check Box control. It is also known as a "button with independent fixation" or an "on/off switch". The switch is considered to be ON when the dialog item has non-zero Selected field.

```
struct FarDialogItem
{
                    = DI CHECKBOX
  int Type
  int X1
                    = X
  int Y1
                    = Y
                    = 0 (not used in Dialog API 1.0)
  int X2
                    = Y (equals to Y1)
  int Y2
  int Focus
                    = Focus
  int Selected
                    = Selected
  DWORD Flags
                    = Flags
  int DefaultButton = DefaultButton
  char Data[512]
                    = Checkbox caption
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

DCUS

Keyboard focus flag.

elected

This field reflects current state of the checkbox control: is it switched on or off.

ags

There are several flags applicable to the **DI_CHECKBOX** item:

Flag	Description
DIF CENTERGROUP	Sequential items having this flag set and equal vertical coordinates will be horizontally centered in the dialog. Their X1 and X2 coordinates are ignored. Useful for centering checkbox groups.
DIF SETCOLOR	The low byte of Flags will be used as the item color.

DIF_DISABLE	Disables user access to the control.
DIF NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.
DIF 3STATE	The checkbox will have 3 possible states: "off", "on", "undefined".
DIF SHOWAMPERSAND	Show ampersand symbol in caption instead of using it for defining hotkeys.

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function just before the control is a
DN_CTLCOLORDLGITEM	The plugin should pass the color attributes of the checbox when this event comes. Param2 argument (foreground+background):
	LoWord LoByte - color of the caption (COL WARNDIALOGBUTTON or COL DIALOGBUTTON) LoWord HiByte - color of highlighted text (COL WARNDIALOGHIGHLIGHTBUTTON or COL DIALOGHIGHLIGHTBUTTON)
	Param2, when the checkbox has focus:
	LoWord LoByte - color of the caption (COL WARNDIALOGSELECTEDBUTTON or COL DIALOGSELECTEDBUTTON) LoWord HiByte - color of highlighted text (COL WARNDIALOGHIGHLIGHTSELECTEDBL COL DIALOGHIGHLIGHTSELECTEDBUTTON) HiWord LoByte - 0 HiWord HiByte - 0 If the special flag <u>DIF SETCOLOR</u> is used, the checkbox's caption (LoWe LoByte) will be drawn according to the above settings.
DN HOTKEY	A hotkey was pressed (Alt- <letter>).</letter>
DN_BTNCLICK	The state of the checkbox was changed.
DN KEY	This event comes after the user has pressed a key in the dialog.
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outsi dialog with a mouse button.
DN_KILLFOCUS	This event is sent before the button loses the focus, if the flag <u>DIF_NOFO</u> was not used.

It is strongly recommended to set correct values for *X2* and *Y2* fields, though they aren't used in Dialog API 1.0.

e also:

DI RADIOBUTTON, DI BUTTON, FarDialogItem

in | Dialog API | Dialog items

The **DI_COMBOBOX** dialog item describes an edit box with a drop-down list (Combo Box).

```
struct FarDialogItem
{
 int Type
                      = DI COMBOBOX
 int X1
                      = X1
 int Y1
                      = Y
                      = X2
 int X2
                      = Y (equals to Y1)
 int Y2
                      = Focus
 int Focus
 union {
                                   [passed to]
   FarList *ListItems = ListItems
   int ListPos = ListPos
                                   [returned]
 };
             = Flags
 DWORD Flags
 int DefaultButton = DefaultButton
 union {
   char Data[512]; = the text for editing (without
    struct {
     DWORD PtrFlags; = advanced flags (not used in D
         PtrLength; = size of the user buffer point
     int
     char *PtrData; = the pointer to the user buffe
     char PtrTail[1];= the remainder part of Data
                      = the text for editing (with DI
   } Ptr;
  };
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

DCUS

Keyboard focus flag.

stltems

This is the pointer to the <u>FarList</u> structure containing fields for the combo box

initialization. The "edit" part of the combo box gets the value of the first item in the list with the <u>LIF_SELECTED</u> flag set, if any.

If this field is set to NULL, then the list box will not be shown.

stPos

Current position in the <u>ListItems.Items</u> list. The index of the item selected by the user is stored in this field after the dialog has been closed.

lags

There are several flags applicable to the **DI_COMBOBOX** control (for the flags of the list box, see <u>FarListItem</u>):

Flag	Description
DIF DROPDOWNLIST	Shows non-editable drop-down list instead of a common combo box.
DIF EDITEXPAND	Expand environment variables.
DIF LISTAUTOHIGHLIGHT	Assigns hotkeys for the list elements automatically, starting with the first item.
DIF LISTNOAMPERSAND	Shows a hotkey instead of showing the ampersand itself.
DIF_LISTWRAPMODE	If this flag is set, trying to move the cursor up from the first element or down from the last element will move the cursor to the bottom or the top of the list, respectively.
DIF DISABLE	Disables user access to the combo box.
DIF_READONLY	Sets read-only state for the edit control.
DIF_SELECTONENTRY	Makes the edit control always select the text when it receives focus.
DIF NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.
<u>DIF VAREDIT</u>	If this flag is set, the dialog manager will use only the Ptr.* members instead of Data in the FarDialogItem structure. The use of this flag allows to exceed the 512-byte limit for the edit control.

Event	Description
DN_DRAWDLGITEM	This event is sent to the dialog callback function before the combo box is

	drawn.	
DN CTLCOLORDLGITEM	I Plugin should pass the color attributes of the edit item of the combo bo this event comes. Param2 parameter:	
	LoWord LoByte - color of the text in the edit cont (<u>COL WARNDIALOGEDIT</u> or COL DIALOGEDIT)	
	LoWord HiByte - color of selected text (<u>COL DIALOGEDITSELECTED</u>)	
	HiWord LoByte - color of unchanged text (<u>COL DIALOGEDITUNCHANGED</u>)	
	HiWord HiByte - color of the drop-down arrow (<u>COL DIALOGTEXT</u>)	
DN_CTLCOLORDLGLIST	When this event comes, the plugin may change the color attributes of the litem of the combo box to be drawn.	
DN KEY	This event comes after the user has pressed a key in the dialog.	
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outsi the dialog with the mouse button.	
DN EDITCHANGE	The text in the edit field has been changed.	
DN KILLFOCUS	This event is sent before the combo box loses the focus, if the flag <u>DIF NOFOCUS</u> was not used.	
DN GOTFOCUS	This event is sent after the button has received the keyboard focus, if the fl <u>DIF NOFOCUS</u> was not used.	

e also:

DI_LISTBOX, FarDialogItem

I DOUBLEBOX

in | Dialog API | Dialog items

The **DI_DOUBLEBOX** dialog item describes a double line frame.

```
struct FarDialogItem
{
  int Type
                      = DI DOUBLEBOX
  int X1
                      = X1
  int Y1
                      = Y1
  int X2
                      = X2
  int Y2
                      = Y2
  int Focus = 0
int Selected = 0
DWORD Flags = Flags
  int DefaultButton = 0
  char Data[512] = Caption
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

lags

There are several flags applicable to the **DI_DOUBLEBOX** control:

Flag	Description
DIF_SETCOLOR	The low byte of Flags will be used as the frame color.
DIF_LEFTTEXT	The caption of the frame will be left aligned.
DIF SHOWAMPERSAND	Show ampersand symbol in caption instead of using it for defining hotkeys.

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function just before the double line drawn.
DN_CTLCOLORDLGITEM	The plugin should pass the color attributes of the frame item when this ever comes. Param2 parameter:

	LoWord LoByte - color of text in the caption (COL WARNDIALOGBOXTITLE or COL DIALOGBOXTITLE) LoWord HiByte - color of highlighted text in the c (COL WARNDIALOGHIGHLIGHTBOXTITLE (COL DIALOGHIGHLIGHTBOXTITLE) HiWord LoByte - color of the frame lines (COL WARNDIALOGBOX or COL DIALOGB(HiWord HiByte - 0 (not used)
DN HOTKEY	Hotkey was pressed (Alt- <letter>).</letter>
DN_MOUSECLICK	This event comes after the user has clicked one of the dialog items or outsi dialog with the mouse button.

- 1. If this item is the first in the dialog items array, its caption is copied into the FAR console window title.
- 2. When X1==X2 or Y1==Y2, a line (vertical or horizontal) will be drawn instead of the frame.

e also:

DI SINGLEBOX, FarDialogItem

in | Dialog API | Dialog items

The **DI_EDIT** dialog item describes an edit box.

```
struct FarDialogItem
{
                                                                                                                                   = DI EDIT
            int Type
            int X1
                                                                                                                                   = X1
            int Y1
                                                                                                                                   = Y
            int X2
                                                                                                                                   = X2
                                                                                                                                   = Y (equals to Y1)
            int Y2
          Building = Focus
Building = Focus
Building = Flags
Building = Defaulte
Building = Bu
                                                                                                                                   = DefaultButton
                       char Data[512]; = the text for editing (without
                        struct {
                                  DWORD PtrFlags; = advanced flags (not used in D
                                                                     PtrLength; = size of the user buffer point
                                   int
                                  char *PtrData; = the pointer to the user buffe
                                  char PtrTail[1];= the remainder part of Data
                       } Ptr;
                                                                                                                                   = the text for editing (with DI
           };
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

DCUS

Keyboard focus flag.

istory

Contains the address of a null-terminated text string that will be used as the internal history name when an edit control has the **DIF HISTORY** flag. If several edit fields have the same history name, they will share the same history list.

lags

There are several flags applicable to the **DI_EDIT** control:

Flag	Description
DIF EDITOR	Sequential edit controls with this flag are grouped into a simple editor with the ability to insert and delete lines.
DIF_HISTORY	Adds a history list to an edit control. If this flag is set, the History field must contain the address of a text string that will be used as the internal name of the history.
DIF MANUALADDHISTORY	Specifies that items will be added to the history list of an edit box only manually, not automatically. Must be used together with DIF_HISTORY.
DIF USELASTHISTORY	The initial value will be set to the last history element.
DIF_EDITEXPAND	Expand environment variables.
DIF_DISABLE	Disables user access to the edit control.
DIF READONLY	Sets read-only state for the edit control.
DIF SELECTONENTRY	Makes the edit control always select the text when it receives the focus.
DIF NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.
DIF_VAREDIT	If this flag is set, the dialog manager will use only the Ptr.* members instead of Data in the FarDialogItem structure. The use of this flag allows to exceed the 512-byte limit for the edit control.

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function just before the edit box is drawn.
DN_CTLCOLORDLGITEM	The plugin should pass the color attributes of the edit control when this even comes. Param2 parameter: LoWord LoByte - color of the text (<u>COL WARNDIALOGEDIT</u> or <u>COL DIALOGEDIT</u>) LoWord HiByte - color of selected text (<u>COL DIALOGEDITSELECTED</u>) HiWord LoByte - color of unchanged text

	(<u>COL_DIALOGEDITUNCHANGED</u>) HiWord HiByte - color of the History drop-down arr (<u>COL_DIALOGTEXT</u>)
DN KEY	This event comes after the user has pressed a key in the dialog.
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outsi the dialog with the mouse button.
DN EDITCHANGE	The text in the edit field has been changed.
DN KILLFOCUS	This event is sent before the combo box loses the focus, if the flag <u>DIF_NOFOCUS</u> was not used.
DN GOTFOCUS	This event is sent after the button has received the keyboard focus, if the fl DIF NOFOCUS was not used.

e also: <u>DI PSWEDIT, DI FIXEDIT, FarDialogItem</u>

in | Dialog API | Dialog items

The **DI_FIXEDIT** dialog item describes a fixed size edit box. It is the same as **DI_EDIT**, except the text in the **DI_FIXEDIT** cannot be scrolled.

```
struct FarDialogItem
{
  int Type
                    = DI FIXEDIT
  int X1
                    = X1
  int Y1
                    = Y
                    = X2
  int X2
                    = Y (equals to Y1)
  int Y2
  int Focus
                    = Focus
  union{
    char *History = History
char *Mask; = Mask
  };
              = Flags
  DWORD Flags
  int DefaultButton = DefaultButton
  char Data[512] = the text for editing
};
```

Attention! This is an exemplary structure; read full description here.

DCUS

Keyboard focus flag.

istory

Contains the address of a null-terminated text string that will be used as the internal history name when an edit control has the **DIF** HISTORY flag. If several edit fields have the same history name, they will share the same history list.

ask

Contains the address of a null-terminated string that serves as a mask for user input when the **DIF_MASKEDIT** flag is set. NULL value means the edit field has no input mask.

The DIF_HISTORY flag has higher priority than the DIF_MASKEDIT flag.

lags

There are several flags applicable to **DI_FIXEDIT**:

Flag	Description
DIF HISTORY	Adds a history list to an edit control. If this flag is set, the History field must contain the address of a text string that will be used as the internal name of the history.
DIF MANUALADDHISTORY	Specifies that items will be added to the history list of an edit box only manually and not automatically. Must be used together with DIF_HISTORY.
DIF USELASTHISTORY	The initial value will be set to the last history element.
DIF MASKEDIT	Uses the null-terminated string in the Mask field as a filter for user input.
DIF_DISABLE	Disables user access to the edit control.
DIF READONLY	Sets read-only state for the edit control.
DIF SELECTONENTRY	Makes the edit control always select the text when it receives the focus.
DIF NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function just before the edit box is drawn.
DN_CTLCOLORDLGITEM	Plugin should pass the color attributes of the edit control when this event comes. Param2 parameter: LoWord LoByte - color of the text (<u>COL WARNDIALOGEDIT</u> or <u>COL DIALOGEDIT</u>) LoWord HiByte - color of selected text
	HiWord LoByte - color of unchanged text (<u>COL DIALOGEDITUNCHANGED</u>) HiWord HiByte - color of the History drop-down arr (<u>COL DIALOGTEXT</u>)

DN KEY	This event comes after the user has pressed a key in the dialog.
DN_MOUSECLICK	This event comes after the user has clicked one of the dialog items or outsi the dialog with the mouse button.
DN EDITCHANGE	The text in the edit field has been changed.
DN KILLFOCUS	This event is sent before the combo box loses the focus, if the flag <u>DIF NOFOCUS</u> was not used.
DN GOTFOCUS	This event is sent after the button has received the keyboard focus, if the fl DIF NOFOCUS was not used.

The text cursor in the **DI_FIXEDIT** edit field will initially be in overwrite mode.

e also:

DI EDIT, DI PSWEDIT, FarDialogItem

I LISTBOX

in | Dialog API | Dialog items

The **DI_LISTBOX** dialog item describes a list box.

```
struct FarDialogItem
{
 int Type
                   = DI LISTBOX
  int X1
                   = X1
  int Y1
                   = Y1
  int X2
                   = X2
                   = Y2
 int Y2
  int Focus
                   = Focus
  union {
   FarList *ListItems= ListItems [passed to]
   int ListPos = ListPos
                                  [returned]
 };
 DWORD Flags
             = Flags
 int DefaultButton = DefaultButton
  char Data[512]; = Caption
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

stltems

This is a pointer to the **FarList** structure containing fields for the list box initialization.

stPos

Current position in the ListItems.Items list. The index of the item selected by the user will be stored in this filed when the dialog is closed.

lags

There are several flags applicable to the **DI_LISTBOX** (for the flags of the list box, see <a>FarListItem):

Flag	Description
DIF LISTAUTOHIGHLIGHT	Assigns hotkeys for the list elements automatically,

	starting with the first item.
DIF LISTNOAMPERSAND	Shows a hotkey when a letter is preceded with an ampersand instead of showing the ampersand itself.
DIF LISTWRAPMODE	If this flag is set, trying to move the cursor up from the first element or down from the last element will move the cursor to the bottom or the top of the list, respectively.
DIF_DISABLE	Disables user access to the list box.
DIF NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.
DIF LISTNOBOX	Disables the drawing of a frame around the list. The Data field is ignored in this case.

vents

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function just before the list box is drawn.
DN CTLCOLORDLGLIST	Plugin should pass the color attributes of the list box when this event comes.
DN LISTCHANGE	Position in the list was changed.
DN_LISTHOTKEY	This event comes after the user has pressed a hotkey in the list.

e also: <u>DI COMBOBOX, FarDialogItem</u>

I PSWEDIT

in | Dialog API | Dialog items

DI_PSWEDIT dialog item describes a password edit control. It is the same as **DI EDIT**, except the text in the **DI_PSWEDIT** is hidden with '*' symbols.

```
struct FarDialogItem
{
  int Type
                          = DI PSWEDIT
  int X1
                          = X1
  int Y1
                          = Y
  int X2 = X2
int Y2 = Y (equals to Y1)
int Focus = Focus
  int Selected = 0
DWORD Flags = Flags
int DefaultButton = DefaultButton
  char Data[512] = the text for editing
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

DCUS

Keyboard focus flag.

lags

There are several flags applicable to the **DI_PSWEDIT**:

Flag	Description
DIF_DISABLE	Disables user access to the password control.
DIF READONLY	Sets read-only state for the password control.
DIF SELECTONENTRY	Makes the password control always select the text when it receives focus.
DIF NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.

Event	Description
DN_DRAWDLGITEM	This event is sent to the dialog callback function just before the password box is drawn.
DN CTLCOLORDLGITEM	Plugin should pass the color attributes of the password control when this event comes. Param2 parameter:
	LoWord LoByte - color of the text (<u>COL WARNDIALOGEDIT</u> or <u>COL DIALOGEDIT</u>)
	LoWord HiByte - color of selected text (<u>COL_DIALOGEDITSELECTED</u>)
	HiWord LoByte - color of unchanged text (COL DTALOGEDITUNCHANGED)
	HiWord HiByte - 0 (not used)
<u>DN KEY</u>	This event comes after the user has pressed a key in the dialog.
DN_MOUSECLICK	This event comes after the user has clicked one of the dialog items or outside the dialog with the mouse button.
DN EDITCHANGE	The text in the password field was changed.
DN_KILLFOCUS	This event is sent before the password control loses focus, if the flag <u>DIF NOFOCUS</u> was not used.
DN GOTFOCUS	This event is sent after the password control receives keyboard focus, if the flag <u>DIF_NOFOCUS</u> was not used.

The **DI_PSWEDIT** control doesn't allow copying password text to the clipboard.

e also:

DI EDIT, DI FIXEDIT, FarDialogItem

I RADIOBUTTON

in | Dialog API | Dialog items

The **DI_RADIOBUTTON** dialog item describes a Radio Button control. It is also known as the "button with dependent fixation". It acts like a Check Box, except that if the first radio button item in a group of several consequent radio button items have the **DIF GROUP** flag set, they become mutually exclusive if one of them has been switched on all others will be switched off.

```
struct FarDialogItem
{
  int Type
                   = DI RADIOBUTTON
  int X1
                    = X
  int Y1
                    = Y
  int X2
                    = 0 (not used in Dialog API 1.0)
                    = Y (equals to Y1)
  int Y2
  int Focus
                    = Focus
  int Selected
                   = Selected
  DWORD Flags
                   = Flags
  int DefaultButton = DefaultButton
  char Data[512] = Radio button caption
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

DCUS

Keyboard focus flag.

elected

The field reflects current state of the radio button control: is it switched on or off.

ags

There are several flags applicable to the **DI_RADIOBUTTON** control:

Flag	Description
DIF_SETCOLOR	The low byte of Flags will be used as the item's color.
DIF GROUP	This flag should be set for the first radio button item in

	a group.
DIF_MOVESELECT	Change selection in a radio button group when focus is moved. Radio buttons with this flag set are also drawn without parentheses around the selection mark (example: FAR color selection dialog).
DIF CENTERGROUP	Sequential items with this flag set and equal vertical coordinates will be horizontally centered in the dialog. Their X1 and X2 coordinates are ignored.
DIF DISABLE	Disables user access to the control.
DIF_NOFOCUS	The dialog item cannot receive keyboard focus, but can handle other user events.
DIF SHOWAMPERSAND	Show ampersand symbol in caption instead of using it for defining hotkeys.

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function before the control is drawn.
DN CTLCOLORDLGITEM	Plugin should pass the color attributes of the control when this event comes. Param2 parameter (foreground+background):
	LoWord LoByte - color of the caption (<u>COL_WARNDIALOGBUTTON</u> or COL_DIALOGTEXT)
	LoWord HiByte - color of highlighted text (<u>COL_WARNDIALOGHIGHLIGHTTEXT</u>)
	HiWord LoByte - 0 HiWord HiByte - 0
	If a special attribute is used (<u>DIF_SETCOLOR</u> flag), the caption (LoWord LoByte) will be drawn according to the parameters.
DN_HOTKEY	Hotkey was pressed (Alt- <letter>).</letter>
DN BTNCLICK	State of the radiobutton was changed.
DN KEY	This event comes after the user has pressed a key in the dialog.
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outside the dialog with the mouse button.
DN KILLFOCUS	This event is sent just before the button loses focus, if the flag <u>DIF NOFOCUS</u> was not used.
DN_GOTFOCUS	This event is sent after the button has received keyboard focus, if the flag <u>DIF NOFOCUS</u> was not used.

It is strongly recommended to set correct values for *X2* and *Y2* fields, although they aren't used in Dialog API 1.0.

e also: <u>DI CHECKBOX, DI BUTTON, FarDialogItem</u>

I_SINGLEBOX

in | Dialog API | Dialog items

The **DI_SINGLEBOX** dialog item describes a single line frame.

```
struct FarDialogItem
{
  int Type
                    = DI SINGLEBOX
  int X1
                    = X1
  int Y1
                    = Y1
  int X2
                    = X2
  int Y2
                    = Y2
  int Focus = 0
int Selected = 0
  DWORD Flags = Flags
  int DefaultButton = 0
  char Data[512] = Caption
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

lags

There are several flags applicable to the **DI_SINGLEBOX** control:

Flag	Description
DIF_SETCOLOR	The low byte of Flags will be used as the frame color.
DIF LEFTTEXT	The caption of the frame will be left aligned.
DIF SHOWAMPERSAND	Show ampersand symbol in caption instead of using it for defining hotkeys.

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function before the double line bo drawn.
DN CTLCOLORDLGITEM	Plugin should pass the color attributes of the frame when this event comes parameter:
	LoWord LoByte - color of text in the caption
	<pre>(COL_WARNDIALOGBOXTITLE or COL_DIALOGBOXTITLE) LoWord HiByte - color of highlighted text in the c (COL_WARNDIALOGHIGHLIGHTBOXTITLE c COL_DIALOGHIGHLIGHTBOXTITLE) HiWord LoByte - color of the frame lines (COL_WARNDIALOGBOX or COL_DIALOGB(HiWord HiByte - 0 (not used)</pre>
---------------	---
DN HOTKEY	Hotkey was pressed (Alt- <letter>).</letter>
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outsi dialog with the mouse button.

emarks

- 1. If this item is the first in the dialog items array, its caption is copied into the FAR console window title.
- 2. When X1==X2 or Y1==Y2, a line (vertical or horizontal) will be drawn instead of a frame.

e also:

DI DOUBLEBOX, FarDialogItem

I_TEXT

in | Dialog API | Dialog items

The **DI_TEXT** dialog item describes a static text label.

```
struct FarDialogItem
{
  int Type
                   = DI TEXT
  int X1
                   = X1
  int Y1
                   = Y
  int X2
                   = X2
  int Y2
                   = Y (not used in Dialog API 1.0, m
  int Focus
                   = 0
  int Selected = 0
             = Flags
  DWORD Flags
  int DefaultButton = 0
  char Data[512] = text label
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

lags

There are several flags applicable to the **DI_TEXT** control:

Flag	Description
DIF_SETCOLOR	The low byte of Flags will be used as the item's color.
DIF BOXCOLOR	The text item will be displayed using frame color (COL_DIALOGBOX or COL_WARNDIALOGBOX)
DIF_CENTERGROUP	Sequential strings with this flag set and equal vertical coordinates will be horizontally centered in the dialog. Their X1 and X2 coordinates are ignored.
DIF_SEPARATOR	Draws a single-line separator. You may write any text on the separator line, just use Data and the coordinate fields.
DIF_SEPARATOR2	Draws a double-line separator. You may write any text on the separator line, just use Data and the coordinate fields.
DIF SHOWAMPERSAND	Show ampersand symbol in caption instead of using it for defining hotkeys.

vents

Event	Description
DN DRAWDLGITEM	This event is sent to the dialog callback function just before the text item is drawn.
DN_CTLCOLORDLGITEM	Plugin should pass the color attributes of the text item when this event comes. Param2 parameter (foreground+background):
	LoWord LoByte - color of the text (<u>COL WARNDIALOGTEXT</u> or <u>COL DIALOGTEXT</u>). If <u>DIF BOXCOLOR</u> flag is set: <u>COL WARNDIALOGBOX</u> or <u>COL DIALOGBOX</u> LoWord HiByte - color of highlighted text (<u>COL WARNDIALOGHIGHLIGHTTEXT</u>) HiWord LoByte - 0 (not used) HiWord HiByte - 0 (not used)
	If a special attribute is used (<u>DIF_SETCOLOR</u> flag), the text (LoWord LoByte) will be drawn according to the parameters.
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outside the dialog with the mouse button.
DN HOTKEY	Hotkey was pressed (Alt- <letter>).</letter>

emarks

- 1. It is recommended to set the *Y*² coordinate correctly, although it's not used in Dialog API 1.0. Just set it equal to the *Y*¹ coordinate.
- 2. If the <u>DIF_CENTERTEXT</u> flag is set, the *X2* field must have adequate value for the correct text centering.
- 3. If the <u>DIF_CENTERTEXT</u> flag is not set, FAR will calculate *X2* and *Y2* coordinates automatically.
- 4. If this text item is the first in the dialog items array, the text string is copied into the FAR console window title.
- 5. If the text of the **DI_TEXT** item has a hotkey and <u>DIF_SHOWAMPERSAND</u> flag isn't set, then pressing **Alt-Letter** causes the <u>keyboard focus</u> to move to the next available dialog item.

e also: <u>DI VTEXT, FarDialogItem</u>

I USERCONTROL

in | Dialog API | Dialog items

The **DI_USERCONTROL** dialog item describes an user-defined control controlled completely by the plugin: initialize, draw etc.

```
struct FarDialogItem
{
  int Type
                      = DI USERCONTROL
  int X1
                      = X1
  int Y1
                      = Y1
  int X2
                      = X2
                      = Y2
  int Y2
                      = User Defined
  int Focus
  CHAR_INFO *VBuf = Virtual Draw Buffer
DWORD Flags = Flags
  int DefaultButton = 0
  char Data[512] = User Defined
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

irtual Draw Buffer

VBuf parameter points to an array of <u>CHAR_INFO</u> structures that contain characters and their attributes to be drawn in the dialog.

If **VBuf** is NULL, the plugin itself must draw the control using the Text service function when a DN DRAWDLGITEM event comes.

If **VBuf** is not NULL, the plugin must fill the **VBuf** array when it receives the **DN_DRAWDLGITEM** event, and the dialog manager will then copy the contents of the buffer to the screen.

This is the typical scenario for using the **DI_USERCONTROL** (see the source code of **Reversi** plugin):

```
// allocate memory for the virtual buffer before call
#define DIM(Item) (((Item).X2-(Item).X1+1)*((Item).Y2
CHAR_INFO *VBuf=new CHAR_INFO[DIM(DialogItems[11])];
DialogItems[11].VBuf=VBuf;
```

```
// in the dialog callback function
struct FarDialogItem DialogItem;
case DN_DRAWDLGITEM:
  if(Param1 == 11)
  {
                     ";
    char Face[4]="
    SMALL_RECT Rect;
    BYTE AddColor=0x00;
    // get coordinates of the dialog and description
    Info.SendDlgMessage(hDlg,DM_GETDLGRECT,0,(LONG_PT
    Info.SendDlgMessage(hDlg,DM_GETDLGITEM,11,(LONG_F
    // drawing the game field
    for(Y=0; Y < 8; ++Y)
    {
      for(X=0; X < 8; ++X)
      {
        // prepare one rectangle
        // if the cell is not used...
        if (GAME[0].Field[Y*8+X]==0)
          Face[1]=' ';
        // for the white player
        else if (GAME[0].Field[Y*8+X]==GAME[0].Pl1)
        {
          Face[1]=FaceWhite;
          AddColor=0x00;
        }
        // for the black player
        else if (GAME[0].Field[Y*8+X]==GAME[0].Pl2)
        {
          Face[1]=FaceBlack;
          AddColor=0x0F;
```

```
// if the memory couldn't be allocated,
      // draw with the Text function
      if(!DialogItem.VBuf)
      {
        Info.Text(Rect.Left+DialogItem.X1+X*3,
                 Rect.Top+DialogItem.Y1+Y,
                ColorsPanel[Y&1][X&1]|AddColor,
                Face);
      }
      else // if the memory was allocated, use virt
      {
        CHAR_INFO *VBuf=&DialogItem.VBuf[Y*8*3+X*3]
        VBuf[0].Char.AsciiChar=Face[0];
        VBuf[1].Char.AsciiChar=Face[1];
        VBuf[2].Char.AsciiChar=Face[2];
        VBuf[0].Attributes=
           VBuf[1].Attributes=
           VBuf[2].Attributes=ColorsPanel[Y&1][X&1]
      }
   }
  }
}
return TRUE;
```

lags

There are several flags applicable to the **DI_USERCONTROL** control:

Flag	Description
DIF NOFOCUS	The user-defined dialog control cannot receive keyboard focus, but can handle other user events.
DIF DISABLE	Disables user access to the control.
DIF NOTCVTUSERCONTROL	do not convert characters (CHAR_INFO::Char) while writing the virtual buffer to the screen.

vents

Event	Description
DN_DRAWDLGITEM	This event is sent to the dialog callback function just before the item is drawn.
<u>DN KEY</u>	This event comes after the user has pressed a key in the dialog.
DN MOUSECLICK	This event comes after the user has clicked a mouse button; coordinates are counted from upper left corner of the item
DN KILLFOCUS	This event is sent before the button loses focus, if the flag <u>DIF_NOFOCUS</u> was not used.
DN_GOTFOCUS	This event is sent after the button has received keyboard focus, if the flag <u>DIF NOFOCUS</u> was not used.

emarks

- 1. For a description of how to manage the text cursor in a user-defined dialog control, see <u>DM_GETCURSORPOS</u> or <u>DM_SETCURSORPOS</u>.
- 2. The **DI_USERCONTROL** item is not supported by the <u>Dialog</u> function.

e also: <u>FarDialogItem</u>

I VTEXT

in | Dialog API | Dialog items

The **DI_VTEXT** dialog item describes a vertical static text label.

```
struct FarDialogItem
{
  int Type
                    = DI_VTEXT
  int X1
                    = X
  int Y1
                    = Y1
  int X2
                    = X (not used in Dialog API 1.0, m
  int Y2
                    = Y2
  int Focus = 0
int Selected = 0
              = Flags
  DWORD Flags
  int DefaultButton = 0
  char Data[512] = text label
};
```

Attention! This is an exemplary structure; read full description <u>here</u>.

lags

There are several flags applicable to the **DI_VTEXT** control:

Flag	Description
DIF_SETCOLOR	The low byte of Flags will be used as the item's color.
DIF BOXCOLOR	The text item will be displayed using box colors (COL_DIALOGBOX or COL_WARNDIALOGBOX)
DIF CENTERGROUP	Centers two vertical text labels with equal horizontal position (Y1 is ignored)
DIF SEPARATOR	Draws a single-line vertical separator; Y1 is ignored if Data is empty; if Data is not empty, it will also be drawn with Y1 as its starting position.
DIF_SEPARATOR2	Draws a double-line vertical separator Y1 is ignored if Data is empty; if Data is not empty, it will also be drawn with Y1 as its starting position.
DIF SHOWAMPERSAND	Show ampersand symbol in caption instead of using it

	for defining hotkeys.
DIF_CENTERTEXT	Centers the text between the Y1 and Y2 coordinates.

vents

Event	Description	
DN_DRAWDLGITEM	This event is sent to the dialog callback function just before the text item is drawn.	
DN CTLCOLORDLGITEM	Plugin should pass the color attributes of the text item when this event comes. Param2 parameter (foreground+background):	
	LoWord LoByte - color of the text (<u>COL WARNDIALOGTEXT</u> or <u>COL DIALOGTEXT</u>). If <u>DIF BOXCOLOR</u> flag is set: <u>COL WARNDIALOGBOX</u> or <u>COL DIALOGBOX</u>	
	LoWord HiByte - color of highlighted text (<u>COL WARNDIALOGHIGHLIGHTTEXT</u> o <u>COL DIALOGHIGHLIGHTTEXT</u>)	
	HiWord HiByte - 0 (not used)	
	If a special attribute is used (<u>DIF_SETCOLOR</u> flag), the text (LoWord LoByte) will be drawn according to the parameters.	
DN MOUSECLICK	This event comes after the user has clicked one of the dialog items or outside the dialog with the mouse button.	

emarks

- 1. It is recommended to set the *X2* coordinate correctly, although it isn't used in Dialog API 1.0. Just set it equal to the *X1* coordinate.
- 2. If the <u>DIF_CENTERTEXT</u> flag is set, the *Y*2 field must have an adequate value for the correct text centering.
- 3. If the <u>DIF_CENTERTEXT</u> flag is not set, FAR will calculate the *X2* and *Y2* coordinates automatically.
- 4. If the text of the **DI_TEXT** item has a hotkey and <u>DIF_SHOWAMPERSAND</u> flag isn't set, then pressing **Alt-Letter** causes the <u>keyboard focus</u> to move to the next available dialog item.

e also: <u>DI TEXT</u>, <u>FarDialogItem</u>

IF_3STATE

in | Dialog API | Dialog item flags

The **DIF_3STATE** flag indicates that a <u>DI_CHECKBOX</u> element will have three states:

[]	-	off	(FarDialogItem.Selected	=	0)
[X]	-	on	(FarDialogItem.Selected	=	1)
[?]	-	undefined	(FarDialogItem.Selected	=	2)

ontrols

The **DIF_3STATE** flag is applicable to the following dialog items:

Control	Description
DI CHECKBOX	Check box.

emarks

When using the **DIF_3STATE** flag it is necessary to remember what the user expects while working with such checkbox. That is, if the user have selected the undefined checkbox state, then the option being controlled by this checkbox must be completely ignored during further processing. File attributes dialog is the striking example:

[] - clear attribute
[+] - set attribute
[?] - don't do anything with this attribute

IF_BOXCOLOR

in | Dialog API | Dialog item flags

The **DIF_BOXCOLOR** flag allows to specify initial element color corresponding to the dialog frame color (<u>COL_WARNDIALOGBOX</u>) or <u>COL_DIALOGBOX</u>).

ontrols

The DIF_BOXCOLOR flag is applicable to the following dialog items:

Control	Description
<u>DI_TEXT</u>	Text string.
<u>DI VTEXT</u>	Vertical text string.

emarks

It is possible to change the color value upon receiving the <u>DN_CTLCOLORDLGITEM</u> event in the <u>dialog callback</u> function.

e also: <u>DIF_SETCOLOR, Color indexes</u>

IF_BTNNOCLOSE

in | Dialog API | Dialog item flags

The **DIF_BTNNOCLOSE** flag directs the button not to close the dialog when pressed. Default behavior of the buttons is to end dialog processing. Another way to change the normal behavior is to return FALSE for the <u>DN_CLOSE</u> event.

ontrols

The **DIF_BTNNOCLOSE** flag is applicable to the following dialog items:

Control	Description
DI BUTTON	Push Button.

emarks

The **DIF_BTNNOCLOSE** flag has no meaning for a dialog with no callback function.

e also: <u>DM_CLOSE</u>

IF_CENTERGROUP

in | Dialog API | Dialog item flags

Sequentially declared elements with the **DIF_CENTERGROUP** flag and with the same vertical position will be centered in the dialog. Horizontal coordinates of those elements (*X1* and *X2*) are ignored.

ontrols

The **DIF_CENTERGROUP** flag is applicable to the following dialog items:

Control	Description
<u>DI BUTTON</u>	Push button.
DI CHECKBOX	Check box.
DI RADIOBUTTON	Radio button.
<u>DI_TEXT</u>	Text string .
<u>DI VTEXT</u>	Vertical text string.

emarks

- 1. It is convenient for centering a group of buttons.
- 2. FAR itself handles group centering upon dialog resize.

IF_CENTERTEXT

in | Dialog API | Dialog item flags

The **DIF_CENTERTEXT** flag allows to align text in the <u>DI_TEXT</u> and <u>DI_VTEXT</u> elements centering it relatively to element's geometry.

ontrols

The **DIF_CENTERTEXT** flag is applicable to the following dialog items:

Control	Description
DI TEXT	Text string.
<u>DI VTEXT</u>	Vertical text string.

emarks

If you specify this flag, it is necessary to fill the *X2* coordinate (for the <u>DI_TEXT</u> control) and the *Y2* coordinate (for the <u>DI_VTEXT</u> control) correctly.

IF_DISABLE

in | <u>Dialog API</u> | <u>Dialog item flags</u>

The **DIF_DISABLE** flag directs the Dialog Manager to disable this item. It means that such dialog element will not receive input focus and will not respond to mouse, but at the same time a control with this flag set can be changed programmatically.

ontrols

The **DIF_DISABLE** flag is applicable to the following dialog items:

Control	Description
All	All dialog items can be disabled.

emarks

- 1. To change Enabled/Disabled state of a control, send the <u>DM_ENABLE</u> message to the Dialog Manager (using the <u>SendDlgMessage</u> function).
- 2. Disabled edit controls are displayed using the <u>COL_DIALOGEDITDISABLED</u> color. If you want to change the color of a control, you must add the following code to the dialog handler:

```
if(msg == DN_CTLCOLORDLGITEM)
{
    if(Param1 >= 9 && Param1 <= 13)
    {
        int Lo=(int)Info.AdvControl(Info.ModuleNumber,
        int Hi=(int)Info.AdvControl(Info.ModuleNumber,
        return (LONG_PTR)MAKELONG(MAKEWORD(Lo,Hi),MAKE
    }
}....</pre>
```

In this example, predefined colors for disabled items are modified to match colors corresponding to the normal state of the edit control.

IF_DROPDOWNLIST

in | Dialog API | Dialog item flags

The **DIF_DROPDOWNLIST** flag specifies that a <u>DI_COMBOBOX</u> control is a read-only drop-down list.

ontrols

The **DIF_DROPDOWNLIST** flag is applicable to the following dialog items:

Control	Description
DI COMBOBOX	Combo box.

emarks

IF_EDITEXPAND

in | Dialog API | Dialog item flags

The **DIF_EDITEXPAND** flag "expands" environment variables after completion of dialog execution (for example, %TEMP% will be expanded to C:\TEMP)

ontrols

The **DIF_EDITEXPAND** flag is applicable to the following dialog items:

Control	Description
DI COMBOBOX	Combo box.
DI EDIT	Edit box.

emarks

Environment variables "expansion" is done by using the

ExpandEnvironmentStrings

function, so if

some %VariableName% value doesn't exist in the environment, it will remain untouched.

IF_EDITOR

in | Dialog API | Dialog item flags

Sequentially declared edit controls (<u>**DI_EDIT</u>**) with the **DIF_EDITOR** flag set are grouped into an editor capable of insertion and removal of lines.</u>

ontrols

The **DIF_EDITOR** flag is applicable to the following dialog items:

Control	Description
<u>DI EDIT</u>	Edit box.

emarks

In order to create an edit area with the size of 5 rows it is necessary to place 5 elements of type **DI_EDIT** with **DIF_EDITOR** flag set, as shown in the following example:

DI_EDIT,5,3,29,3,1,1,DIF_EDITOR,0,"", DI_EDIT,5,4,29,4,0,1,DIF_EDITOR,0,"", DI_EDIT,5,5,29,5,0,1,DIF_EDITOR,0,"", DI_EDIT,5,6,29,6,0,1,DIF_EDITOR,0,"", DI_EDIT,5,7,29,7,0,1,DIF_EDITOR,0,"",

IF_GROUP

in | Dialog API | Dialog item flags

The **DIF_GROUP** flag, if specified with the first <u>DI_RADIOBUTTON</u> control, and groups sequentially declared radio buttons:

```
// color selection dialog in FAR Manager:
...
// first group of radiobuttons
DI_RADIOBUTTON, 6, 3, 0, 0, 0, 0, F_LIGHTGRAY|B_BLACK|DIF_GF
DI_RADIOBUTTON, 6, 4, 0, 0, 0, 0, F_BLACK|B_RED|DIF_SETCOLOF
...
// second group of radiobuttons
DI_RADIOBUTTON, 21, 3, 0, 0, 0, 0, F_LIGHTGRAY|B_BLACK|DIF_G
DI_RADIOBUTTON, 21, 4, 0, 0, 0, 0, F_BLACK|B_RED|DIF_SETCOLC
...
```

ontrols

The **DIF_GROUP** flag is applicable to the following dialog items:

Control	Description
DI RADIOBUTTON	Radio button.

emarks

Set this flag for the first item in the group.

IF_HIDDEN

in | Dialog API | Dialog item flags

The **DIF_HIDDEN** flag hides a dialog item.

ontrols

The **DIF_HIDDEN** flag is applicable to the following dialog items:

Control	Description
All	All dialog elements can be hidden.

emarks

IF_HISTORY

in | Dialog API | Dialog item flags

DIF_HISTORY flag allows to keep a history list for edit controls. When this flag is set, the **History** field must contain the address of a text string that will be used as the internal history name. If several edit controls have the same history name, they will share the same history list. For the following example (ARCPROC.CPP file from MultiArc):

```
const char *PathHistoryName="ExtrDestPath";
struct InitDialogItem InitItems[]={
   ...
   DI_EDIT, 5, 3, 70, 3, 1, (DWORD)PathHistoryName, DIF_HISTOF
   ...
};
```

the history will be stored in the registry (under

HKCU\Software\Far\SavedDialogHistory*ExtrDestPath* key), one line (key names look like "Line<number>", of REG_SZ type) for each value entered by the user.

Kyes with names looking like "Locked<number>" (REG_DWORD) are intended for marking lines that cannot be deleted (while in the list, this state can be changed with the <Insert> key).

ontrols

The **DIF_HISTORY** flag is applicable to the following dialog items:

Control	Description
<u>DI EDIT</u>	Edit box.
<u>DI FIXEDIT</u>	Fixed size edit box.

emarks

- 1. The **DIF_HISTORY** flag **DOES NOT WORK** with the **DI_PSWEDIT** control!
- 2. **DIF_HISTORY** has higher priority than the **DIF_MASKEDIT** flag.
- 3. **FAR 1.70 beta 3**: If a plugin has an edit box with a history list, it is assumed that the user will leave the dialog using the Esc key (Enter is reserved for other needs), then the data will not be stored in the history

because Esc means rejection of further dialog processing. In such case, plugin can add necessary strings to the history list. It can be carried out by sending the <u>DM_ADDHISTORY</u> message to the Dialog Manager.

- 4. If an item has the <u>DIF_MANUALADDHISTORY</u> flag, then the Dialog Manager will not add strings to the history list when the dialog closes.
- 5. Also, a plugin can keep united history lists using predefined names:

name	purpose
"SearchText"	search edit box
"ReplaceText"	replace edit box
"PersPath"	personal plugins' paths
"Сору"	destination edit box in copy dialog
"LineNumber"	editor goto (Alt-F8)
"ViewerOffset"	viewer goto (Alt-F8)
"NewEdit"	edited files (Shift-F4/Shift-F2)
"Masks"	file masks (selection, associations, filters, file search)
"UserVarN"	user variables
"ApplyCmd"	"apply command" (Ctrl-G)
"DizText"	file description edit box
"NewFolder"	folder creation

e also: <u>DIF_USELASTHISTORY, DM_ADDHISTORY,</u> DIF_MANUALADDHISTORY.

IF_LEFTTEXT

in | Dialog API | Dialog item flags

The **DIF_LEFTTEXT** flag allows to left-align the title of a frame. Frame title is center-aligned by default.

ontrols

The **DIF_LEFTTEXT** flag is applicable to the following dialog items:

Control	Description
DI DOUBLEBOX	Double frame.
DI SINGLEBOX	Single frame.

emarks

IF_LISTAUTOHIGHLIGHT

in | Dialog API | Dialog item flags

If the **DIF_LISTAUTOHIGHLIGHT** flag is set, then hot keys will be assigned automatically starting from the first item.

ontrols

The **DIF_LISTAUTOHIGHLIGHT** flag is applicable to the following dialog items:

Control	Description
<u>DI LISTBOX</u>	List box.
DI COMBOBOX	Combo box.

emarks

IF_LISTNOAMPERSAND

in | Dialog API | Dialog item flags

The **DIF_LISTNOAMPERSAND** flag allows to display hot keys in the list. By default, ampersands in the list are shown on the screen, and are not used for hot key assignment.

ontrols

The **DIF_LISTNOAMPERSAND** flag is applicable to the following dialog items:

Control	Description
<u>DI LISTBOX</u>	List box.
DI COMBOBOX	Combo box.

emarks

IF_LISTNOBOX

in | Dialog API | Dialog item flags

The **DIF_LISTNOBOX** flag turns off the frame around a <u>DI_LISTBOX</u> control.

ontrols

The **DIF_LISTNOBOX** flag is applicable to the following dialog items:

Control	Description
<u>DI LISTBOX</u>	List box.

emarks

IF_LISTNOCLOSE

in | Dialog API | Dialog item flags

The **DIF_LISTNOCLOSE** flag directs a list not to close the dialog after item selection. Default list behavior after item selection is to end dialog processing. Another way to change the default behavior is to return FALSE for the <u>DN_CLOSE</u> event.

ontrols

The **DIF_LISTNOCLOSE** flag is applicable to the following dialog items:

Control	Description
<u>DI LISTBOX</u>	List box.
DI COMBOBOX	Combo box.

IF_LISTWRAPMODE

in | Dialog API | Dialog item flags

If the **DIF_LISTWRAPMODE** flag is set, then attempts to move the cursor up from the first item or down from the last item will result in movement to the last or the first item, respectively.

ontrols

The **DIF_LISTWRAPMODE** flag is applicable to the following dialog items:

Control	Description
<u>DI LISTBOX</u>	List box.
DI COMBOBOX	Combo box.

emarks

IF_MANUALADDHISTORY

in | Dialog API | Dialog item flags

The **DIF_MANUALADDHISTORY** flag informs the Dialog Manager that the dialog handler will manually add strings to the history list.

ontrols

The **DIF_MANUALADDHISTORY** flag is applicable to the following dialog items:

Control	Description
<u>DI EDIT</u>	Edit box.
<u>DI FIXEDIT</u>	Fixed size edit box.

emarks

- 1. Adding a string to the history list is carried out by the <u>DM_ADDHISTORY</u> message.
- 2. This flag only extends the <u>DIF_HISTORY</u> flag and has no meaning on its own.

This flag allows to obtain the most complete control on addition of strings to the history list.

e also: <u>DIF HISTORY</u>, <u>DM ADDHISTORY</u>

IF_MASKEDIT

in | Dialog API | Dialog item flags

The **DIF_MASKEDIT** flag allows to set a mask for a <u>DI_FIXEDIT</u> control. If this flag is set, **Mask** must contain the address of a text string with the mask.

For now, the following mask characters are supported:

'X' allows to enter any character at the given line position;

'#' allows to enter digits, spaces, and the minus sign at the given line position;

'9' allows to enter only digits at the given line position;

'A' allows to enter only letters at the given line position;

'H' allows to enter only hexadecimal digits at the given line position.

It is possible to create an infinite variety of masks, for example: "(###) #99-99-99". In this mask, parenthesis and hyphens will be static (i.e., they cannot be deleted) line elements, and it is possible to enter digits or spaces in the parenthesis (e.g., city phone code), but it is possible to enter only digits in the positions with "9" digits.

One note about the usage of the **DIF_MASKEDIT** flag. When you set a mask string (e.g., to "99.99.9999"), remember that processing is organized so that the edit string length is forcibly set to the mask string length.

ontrols

The **DIF_MASKEDIT** flag is applicable to the following dialog items:

Control	Description
<u>DI FIXEDIT</u>	Fixed size edit box.

emarks

The <u>**DIF_HISTORY</u>** flag has a higher priority than the **DIF_MASKEDIT** flag.</u>

IF_MOVESELECT

in | Dialog API | Dialog item flags

The **DIF_MOVESELECT** flag allows to change the selected item in a group of <u>DI_RADIOBUTTON</u> controls upon change of input focus, and the element will have a different look -- drawen without the round brackets. For an example of this flag usage see FAR Manager colors setup dialog.

ontrols

The **DIF_MOVESELECT** flag is applicable to the following dialog items:

Control	Description
DI RADIOBUTTON	Radio button.

emarks

IF_NOAUTOCOMPLETE

in | Dialog API | Dialog item flags

The **DIF_NOAUTOCOMPLETE** flag disables autocompletion for input lines.

ontrols

The **DIF_NOAUTOCOMPLETE** flag is applicable to the following dialog items::

Control	Description
DI COMBOBOX	Combo box.
<u>DI EDIT</u>	Edit box.
<u>DI FIXEDIT</u>	Fixed size edit box.
<u>DI PSWEDIT</u>	Password edit box.

enarks

IF_NOBRACKETS

in | Dialog API | Dialog item flags

The **DIF_NOBRACKETS** flag forces the Dialog Manager to display a <u>DI_BUTTON</u> title without square brackets.

ontrols

The **DIF_NOBRACKETS** flag is applicable to the following dialog items:

Control	Description
DI BUTTON	Push button.

emarks

IF_NOFOCUS

in | Dialog API | Dialog item flags

A control item with the **DIF_NOFOCUS** flag set cannot receive <u>keyboard input</u> <u>focus</u>. It is useful if only mouse control must be provided (see "<u>Reversi</u>" plugin for usage example).

ontrols

The **DIF_NOFOCUS** flag is applicable to the following dialog items:

Control	Description
DI BUTTON	Push button.
DI CHECKBOX	Check box.
<u>DI EDIT</u>	Edit box.
<u>DI_FIXEDIT</u>	Fixed size edit box.
DI COMBOBOX	Combo box.
DI RADIOBUTTON	Radio button.
<u>DI PSWEDIT</u>	Password edit box.
DI LISTBOX	List box.
DI USERCONTROL	User control.

emarks

IF_NOTCVTUSERCONTROL

in | Dialog API | Dialog item flags

A dialog item with the **DIF_NOTCVTUSERCONTROL** flag set will not convert characters (CHAR_INFO::Char) from the virtual buffer before writing them to the screen. Without this flag only the CHAR_INFO::Char.AsciiChar member is used by Far and converted according to the current output method (OEM or Unicode).

ontrols

The **DIF_NOTCVTUSERCONTROL** flag is applicable to the following dialog items:

Control	Description
DI USERCONTROL	User control.

emarks
IF_READONLY

in | Dialog API | Dialog item flags

The **DIF_READONLY** flag sets an edit box into a "read only" state.

ontrols

The **DIF_READONLY** flag is applicable to the following dialog items:

Control	Description
DI COMBOBOX	Combo box.
<u>DI EDIT</u>	Edit box.
<u>DI_FIXEDIT</u>	Fixed size edit box.
<u>DI PSWEDIT</u>	Password edit box.

emarks

e also:

IF_SELECTONENTRY

in | Dialog API | Dialog item flags

The contents of an edit box with the **DIF_SELECTONENTRY** flag set will be selected upon receiving input focus.

ontrols

The **DIF_SELECTONENTRY** flag is applicable to the following dialog items:

Control	Description
DI COMBOBOX	Combo box.
<u>DI EDIT</u>	Edit box.
<u>DI FIXEDIT</u>	Fixed size edit box.
<u>DI_PSWEDIT</u>	Password edit box.

emarks

Selection in an edit box with the **DIF_SELECTONENTRY** flag will be cleared upon <u>input focus</u> loss.

e also:

IF_SEPARATOR

in | Dialog API | Dialog item flags

A text string <u>**DI_TEXT</u>** control with the **DIF_SEPARATOR** flag is displayed as a single horizontal line with double line on edges.</u>

ontrols

The **DIF_SEPARATOR** flag is applicable to the following dialog items:

Control	Description
DI TEXT	Text string.
<u>DI VTEXT</u>	Vertical text string.

emarks

- 1. Separator line drawing coordinates are (left dialog edge = 0): X1 =
 X1_Dialog + 3; X2 = X2_Dialog 3.
- 2. X1 coordinate affects only text string positioning (is displayed above separator line) from *Data*. If you need a separator line with centered text you must specify X1 = -1.
- 3. If *Data* contains an empty string, element is displayed simply as a separator line.

e also: <u>DIF_SEPARATOR2</u>

IF_SEPARATOR2

in | Dialog API | Dialog item flags

A text string <u>**DI_TEXT</u>** control with the **DIF_SEPARATOR2** flag is displayed as a double horizontal line with double line on edges.</u>

ontrols

The **DIF_SEPARATOR2** flag is applicable to the following dialog items:

Control	Description
<u>DI TEXT</u>	Text string.
<u>DI VTEXT</u>	Vertical text string.

emarks

- 1. Separator line drawing coordinates are (left dialog edge = 0): X1 =
 X1_Dialog + 3; X2 = X2_Dialog 3.
- 2. X1 coordinate affects only text string positioning (is displayed above separator line) from *Data*. If you need a separator line with centered text you must specify X1 = -1.
- 3. If *Data* contains an empty string, element is displayed simply as a separator line.

e also: <u>DIF_SEPARATOR</u>

IF_SETCOLOR

in | Dialog API | Dialog item flags

The **DIF_SETCOLOR** flag allows to set the initial color of an item, placing the necessary color into the low-order byte of the <u>FarDialogItem.Flags</u> variable.

The DIF_COLORMASK value is a mask used by the Dialog Manager to select the color component passed in.

ontrols

The **DIF_SETCOLOR** flag is applicable to the following dialog item:

Control	Description
DI BUTTON	Push button.
DI CHECKBOX	Check box.
DI RADIOBUTTON	Radio button.
<u>DI_TEXT</u>	Text string.
<u>DI VTEXT</u>	Vertical text string.
DI DOUBLEBOX	Single frame.
DI SINGLEBOX	Double frame.

emarks

Color value can be changed upon receiving the <u>DN_CTLCOLORDLGITEM</u> event in the <u>dialog handler</u>.

e also:

DIF BOXCOLOR, Color indexes

IF_SHOWAMPERSAND

in | Dialog API | Dialog item flags

The **DIF_SHOWAMPERSAND** flag forces the Dialog Manager to show ampersands (&) in text items and frames, not using it for the hot key definition.

ontrols

The **DIF_SHOWAMPERSAND** flag is applicable to the following dialog item:

Control	Description
<u>DI BUTTON</u>	Push button.
DI CHECKBOX	Check box.
DI RADIOBUTTON	Radio button.
<u>DI TEXT</u>	Text string.
<u>DI VTEXT</u>	Vertical text string.
DI DOUBLEBOX	Double frame.
DI SINGLEBOX	Single frame.

emarks

e also:

IF_USELASTHISTORY

in | Dialog API | Dialog item flags

An edit box with the <u>DIF_HISTORY</u> and **DIF_USELASTHISTORY** flags set is set to the initial value from the last item of the history list, if the initial value is not specified.

ontrols

The **DIF_USELASTHISTORY** flag is applicable to the following dialog item:

Control	Description
<u>DI EDIT</u>	Edit box.
<u>DI FIXEDIT</u>	Fixed size edit box.

emarks

e also: <u>DIF HISTORY</u>

IF_VAREDIT

in | Dialog API | Dialog item flags

The **DIF_VAREDIT** flag is the "512 bytes barrier overcoming for an edit box". The Dialog Manager, while working with an edit box or a combo box (with **DIF_VAREDIT** flag set) will take into account only **Ptr.*** members of the <u>FarDialogItem</u> structure.

ontrols

The **DIF_VAREDIT** flag is applicable to the following dialog items:

Control	Description
DI COMBOBOX	Combo box.
<u>DI EDIT</u>	Edit box.

emarks

- 1. The plugin itself must take care of memory allocation for <u>FarDialogItem.Ptr.PtrData</u> and fill in the size of this buffer correctly (FarDialogItem.Ptr.PtrLength).
- 2. If the size of the data being used does not exceed 512 bytes (or *sizeof(FarDialogItem.Data)*), then there is no sence in using the **DIF_VAREDIT** flag.

e also:

ialog API Messages

Messages	Description
DM_ADDHISTORY	add an item to the history
DM_CLOSE	a signal that the dialog is about to close
DM EDITUNCHANGEDFLAG	controlling the "unchanged" state of a text input box
<u>DM ENABLE</u>	enable or disable a dialog item or to determine if a dialog item is enabled
DM ENABLEREDRAW	enable or disable dialog redrawing
DM GETCHECK	retrieve the state of DI_CHECKBOX or DI_RADIOBUTTON items
DM GETCOMBOBOXEVENT	determine the state of event sending for an open combo box
DM_GETCURSORPOS	get cursor position
DM_GETCURSORSIZE	get cursor size
DM GETDLGDATA	retrieve a data value associated with the dialog
DM_GETDLGITEM	retrieve complete information about a dialog item
DM_GETDLGRECT	retrieve the screen coordinates of the dialog window
DM GETDROPDOWNOPENED	determine if there is an open combo box or history list in the dialog
DM_GETEDITPOSITION	get cursor position in edit controls
DM GETFOCUS	retrieve the ID of the dialog item that has the keyboard focus
DM_GETITEMDATA	retrieve a data value associated with a dialog item
DM_GETITEMPOSITION	retrieve the size and position of a dialog item
DM_GETSELECTION	retrieve selection parameters in dialog edit lines
<u>DM GETTEXT</u>	retrieve the text of an edit string or the caption of an item
DM GETTEXTLENGTH	get text string length
DM GETTEXTPTR	retrieve the text of an edit string or the caption of a dialog item
DM KEY	and a key codes array to the dialog manager
DM LISTADD	add new item to a list
DM LISTADDSTR	add a string to a list

DM_LISTDELETE	delete list items
DM_LISTFINDSTRING	find list item
DM LISTGETCURPOS	get current position in a list
DM_LISTGETDATA	retrieve a data value associated with a list item
DM LISTGETDATASIZE	retrieve the size of the data value associated with a list item
DM LISTGETITEM	retrieve a list item
DM_LISTGETTITLES	retrieve the titles of a list
DM_LISTINFO	retrieve information about a list
DM_LISTINSERT	insert an item to a list
DM LISTSET	replace a list with new list items
DM LISTSETCURPOS	set position in a list
DM_LISTSETDATA	set the data value associated with a list item
DM LISTSETMOUSEREACTION	set the behavior for handling mouse movement in a DI_LISTBOX
DM_LISTSETTITLES	set list titles
DM_LISTSORT	sort list items
DM LISTUPDATE	update a list item
DM MOVEDIALOG	move the dialog
DM_REDRAW	redraw the whole dialog
DM_RESIZEDIALOG	change dialog size
DM SET3STATE	change the style of a DI_CHECKBOX
DM SETCHECK	change the state of DI_CHECKBOX and DI_RADIOBUTTON items
DM SETCOMBOBOXEVENT	enable or disable the sending of DN_KEY or DN_MOUSEEVENT events for an open combo box
DM_SETCURSORPOS	set cursor position in a dialog item
DM SETCURSORSIZE	set cursor size
DM SETDLGDATA	associate a data vaule with the dialog
DM SETDLGITEM	change a specified dialog item
DM SETDROPDOWNOPENED	open or close a combo box or history list
DM SETEDITPOSITION	set cursor position in edit controls
DM SETFOCUS	set the keyboard focus to the given dialog item

DM_SETHISTORY	manage availability of history in edit lines
DM_SETITEMDATA	associate a data vaule with a dialog item
DM SETITEMPOSITION	change position of a dialog item
DM SETMAXTEXTLENGTH	set the maximum length of an edit string
DM_SETMOUSEEVENTNOTIFY	control initial non-altered mouse events
DM_SETSELECTION	select a block in dialog edit lines
DM SETTEXT	set a new string value for an edit line or a new caption for an item
DM SETTEXTPTR	set a new string value for an edit line or a new caption for an item
DM SHOWDIALOG	show/hide the dialog window
DM SHOWITEM	show/hide a dialog item
DM USER	starting value for user defined messages

e also:

Dialog API Events

M_ADDHISTORY

essages | Dialog API

The **DM_ADDHISTORY** message is sent to the dialog manager to add an item to the history of a text input string.

aram1

The ID of the dialog item for which the history item is added.

aram2

Pointer to a NULL-terminated string to be added to the history.

eturn

TRUE - data was successfully added. FALSE - the specified dialog item doesn't have a history.

ontrols

Control	Description
<u>DI EDIT</u>	Text input string
<u>DI FIXEDIT</u>	Fixed width text input string

emarks

The message applies only to the <u>DI_EDIT</u> and <u>DI_FIXEDIT</u> items with the <u>DIF_HISTORY</u> flag set. Also, if the <u>DIF_MANUALADDHISTORY</u> flag is not set, items will be added to the history automatically.

xample

For example, in a calculator pressing Enter computes an expression. To save the entered expression to history after Enter was pressed it is necessary to get that string and add it to history:

```
FarDialogItem dialog[] = {
    ...
    { DI_EDIT, 10, 3, 49, 0, 1, (int)"foo_history", DIF
    ...
};
LONG_PTR WINAPI FooDlgProc(HANDLE hDlg, int Msg, int
```

```
{
...
if (Msg == DM_KEY && Param2 == KEY_ENTER){
    Info.SendDlgMessage(hDlg, DM_GETTEXTPTR, 2, (LC
    Info.SendDlgMessage(hDlg, DM_ADDHISTORY, 2, (LC
    Res = AData.Parse(Text);
...
}
```

e also: <u>DialogEx, DIF_HISTORY, DIF_MANUALADDHISTORY</u>.

M_CLOSE

essages | Dialog API

The **DM_CLOSE** message is sent to the dialog manager when a plugin notifies the Dialog API kernel that it wants to close the dialog.

aram1

ID of the item that will be returned from the <u>DialogEx</u> function. If Param1 is equal to -1, DialogEx will return the ID of the item that currently has the focus.

aram2

0

eturn

Value which is returned by dialog handler as the answer to <u>DN_CLOSE</u>.

emarks

The <u>DN_CLOSE</u> event is received immediately after the **DM_CLOSE** message is sent.

xample

e also: <u>DialogEx</u>, <u>DN</u> CLOSE

M_EDITUNCHANGEDFLAG

essages | Dialog API

The **DM_EDITUNCHANGEDFLAG** message allows to control the state of the "unchanged text" flag for edit boxes.

aram1

The ID of the dialog item for which the operation is performed.

aram2

One of the following values:

- -1 get the current value of the flag for an edit box;
- 0 clear the "unchanged text" flag;
- 1 set the "unchanged text" flag.

eturn

Previous state of the "unchanged text" flag.

ontrols

Control	Description
DI COMBOBOX	combined list (without <u>DIF_DROPDOWNLIST</u> flag set)
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field
<u>DI PSWEDIT</u>	password input field

emarks

- 1. When the dialog is initialized, the "unchanged text" flag is set for all items mentioned above, except for <u>DI_FIXEDIT</u>.
- 2. This message also clears the selection for the edit box for which it is called.

e also: DialogEx

M_ENABLE

essages | Dialog API

The **DM_ENABLE** message is sent to the Dialog API kernel to enable or disable a dialog item or to determine if a dialog item is enabled.

aram1

ID of the dialog item.

aram2

Item state: TRUE - enable item FALSE - disable item -1 - get current item state

eturn

Previous state of the item: TRUE - enabled FALSE - disabled

ontrols

Control	Description
All	All dialog items

emarks

xample

e also: <u>DialogEx</u> <u>DIF_DISABLE</u>

M_ENABLEREDRAW

essages | Dialog API

The **DM_ENABLEREDRAW** message is sent to the dialog manager to enable or disable dialog redrawing.

aram1

TRUE - enable dialog redrawingFALSE - disable dialog redrawing-1 - get current state of counter of output locking.

aram2

0

eturn

Previous state of counter of output locking.

emarks

This function is used to prevent excessive dialog redraws when modifying multiple dialog items. Calling this function with **Param1** = **TRUE** increments an internal counter, and calling with **Param1** = **FALSE** decrements the counter. The dialog is drawn when the value of the counter is zero. This is normal behavior for embedded manipulations. Typical usage:

```
Info.SendDlgMessage(hDlg,DM_ENABLEREDRAW,FALSE,0);
```

// here we change lots of headers

Info.SendDlgMessage(hDlg,DM_ENABLEREDRAW,TRUE,0);

e also: DialogEx

M_GETCHECK

essages | Dialog API

The **DM_GETCHECK** message is sent to the dialog manager to retrieve the state of <u>DI_CHECKBOX</u> or <u>DI_RADIOBUTTON</u> items.

aram1

The ID of the dialog item for which you want to retrieve the state.

aram2

0

eturn

State of the item: <u>BSTATE_UNCHECKED</u>, <u>BSTATE_CHECKED</u>, <u>BSTATE_3STATE</u>.

ontrols

Control	Description
DI CHECKBOX	Check Box
DI RADIOBUTTON	Radio Button

emarks

xample

e also: DialogEx | DM_SETCHECK | DM_SET3STATE.

M_GETCOMBOBOXEVENT

essages | Dialog API

The **DM_GETCOMBOBOXEVENT** message allows to determine if the sending of <u>DN_KEY</u> or <u>DN_MOUSEEVENT</u> events for an open <u>DI_COMBOBOX</u> is enabled or disabled.

aram1

ID of a <u>DI_COMBOBOX</u>.

aram2

0

eturn

State flags (<u>FARCOMBOBOXEVENTTYPE</u> enum).

ontrols

Control	Description
DI COMBOBOX	combined list

emarks

xample

e also: <u>DM_SETCOMBOBOXEVENT</u>

M_GETCURSORPOS

<u>essages | DM_SETCURSORPOS | Dialog API</u>

The **DM_GETCURSORPOS** message is sent to the dialog handler to retrieve the cursor position in edit lines or <u>DI_USERCONTROL</u> item.

aram1

The ID of the dialog item for which you want to retrieve the cursor position.

aram2

Pointer to a <u>COORD</u> structure where dialog manager will place information about cursor position (COORD.Y = 0 for line editor).

eturn

FALSE is returned if the given ID does not specify an edit control or DI_USERCONTROL item.

TRUE is returned when <u>COORD</u> contains cursor position.

ontrols

Control	Description
DI COMBOBOX	combined list
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field
<u>DI PSWEDIT</u>	password input field
DI USERCONTROL	custom (defined by programmer) control

emarks

xample

e also: DialogEx | DM_SETCURSORPOS|

M_GETCURSORSIZE

<u>ssages | DM_SETCURSORSIZE| Dialog API</u>

The **DM_GETCURSORSIZE** message is sent to the dialog handler to retrieve the cursor size in edit lines or in <u>DI_USERCONTROL</u> controls.

aram1

The ID of the dialog item for which you want to retrieve the cursor position.

aram2

0

eturn

The high word is the cursor size (from 0 to 100, as used in the <u>CONSOLE_CURSOR_INFO</u> structure), the low word is the cursor visibility flag (1 - the cursor is visible, 0 - the cursor is hidden).

ontrols

Control	Description
DI COMBOBOX	combined list (without <u>DIF_DROPDOWNLIST</u> flag)
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field
<u>DI PSWEDIT</u>	password input field
DI USERCONTROL	custom (defined by programmer) control

emarks

xample

e also:

```
DialogEx | DM_SETCURSORSIZE| DM_GETCURSORPOS|
DM_SETCURSORPOS|
```

M_GETDLGDATA

essages | Dialog API

The **DM_GETDLGDATA** returns to the dialog handler a 32-bit data value associated with the dialog. Each dialog has a corresponding 32-bit value designed for use by the plugin that created the dialog.

aram1

0

aram2

0

eturn

The dialog manager returns 32-bit value passed by plugin earlier.

emarks

Initially the 32-bit data value contains the data passed in Param to the <u>DialogEx</u> function.

xample

e also: DialogEx, DM SETDLGDATA

M_GETDLGITEM

essages | Dialog API

The **DM_GETDLGITEM** message is sent to the dialog handler to retrieve complete information about a dialog item.

aram1

Dialog item ID.

aram2

Pointer to a FarDialogItem structure.

eturn

TRUE is returned if the data was retrieved successfully. If Param1 contains an invalid item ID, FALSE is returned.

ontrols

Control	Description
All	All dialog items

emarks

xample

// retrieve parameters of 12th item
struct FarDialogItem DialogItem;

Info.SendDlgMessage(hDlg,DM_GETDLGITEM,12,(LONG_PTR)8

e also:

DialogEx DM GETITEMPOSITION DM GETDLGRECT

M_GETDLGRECT

essages | Dialog API

The **DM_GETDLGRECT** allows to retrieve the screen coordinates of the dialog window.

aram1

0

aram2

Pointer to a <u>SMALL RECT</u> structure that receives the dialog coordinates.

eturn

TRUE is returned if the data was retrieved successfully. If **Param2** is NULL, FALSE is returned.

xample

```
struct InitDialogItem InitDlg[]={
       DI_USERCONTROL, 9, 4, 32, 11, 0, 0, DIF_NOFOCUS, 0, ""
/*11*/
. . .
};
long WINAPI ReversiDialogProc(HANDLE hDlg, int Msg, ir
{
. . .
    case DN_DRAWDLGITEM:
      if(Param1 == 11)
      {
         SMALL_RECT Rect;
        Info.SendDlgMessage(hDlg,DM_GETDLGRECT,0,(LON
         . . .
               Info.Text(Rect.Left+DialogItem.X1+X*3,
                         Rect.Top+DialogItem.Y1+Y,
                        ColorsPanel[Y&1][X&1]|AddColor,
                        Face);
         . . .
. . .
}
```

e also: <u>DialogEx</u> <u>DM_GETDLGITEM</u> <u>DM_GETITEMPOSITION</u>

M_GETDROPDOWNOPENED

essages | Dialog API

The **DM_GETDROPDOWNOPENED** message is sent to the dialog manager to determine: "*if there is an open combo box or history list in the dialog*".

aram1

0

aram2

0

eturn

FALSE - there is no open combo box or history list. TRUE - there is an open combo box or history list.

ontrols

Control	Description
Dialog	message concerning dialog only

emarks

xample

e also: <u>DialogEx</u> <u>DM_SETDROPDOWNOPENED</u>

M_GETEDITPOSITION

essages | DM_SETEDITPOSITION | Dialog API

The **DM_GETEDITPOSITION** message is sent to the dialog manager to get the cursor position and state in edit controls.

aram1

Dialog item ID

aram2

Pointer to a EditorSetPosition structure.

eturn

FALSE - the given dialog item ID is not an edit control. TRUE - the <u>EditorSetPosition</u> structure was filled.

ontrols

Control	Description
DI COMBOBOX	combined list
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed size input field
DI PSWEDIT	password input field

emarks

xample

e also: <u>DialogEx</u> | <u>DM_GETCURSORPOS</u>| <u>DM_SETCURSORPOS</u>| <u>DM_SETEDITPOSITION</u>

M_GETFOCUS

essages | Dialog API

The **DM_GETFOCUS** message is sent to the dialog manager to retrieve the ID of the dialog item that has the keyboard focus.

aram1

0

aram2

0

eturn

The ID of the dialog item that has the keyboard focus.

ontrols

Control	Description
All	All items that could have <u>keyboard input focus</u> .

emarks

xample

e also: **DialogEx**

M_GETITEMDATA

essages | Dialog API

The **DM_GETITEMDATA** returns a 32-bit data value associated with a dialog item. Every dialog item has a corresponding 32-bit data value designed for use by the plugin that created the dialog.

aram1

The ID of the dialog item for which the data is retrieved.

aram2

0

eturn

The dialog manager returns 32-bit value that was previously passed by a plugin (or 0, if the plugin didn't associate data with this dialog item).

emarks

xample

e also: DialogEx, DM SETITEMDATA, DM GETDLGDATA DM SETDLGDATA,

M_GETITEMPOSITION

essages | Dialog API

The **DM_GETITEMPOSITION** message is sent to the dialog manager to retrieve the size and position of a dialog item.

aram1

The ID of the dialog item for which the position is retrieved.

aram2

Pointer to a <u>SMALL_RECT</u> structure that will contain the size and position of the item.

eturn

TRUE if the data has been copied or FALSE if **Param2** is NULL or the item does not exist.

xample

e also:	
Dial	<u>ogEx</u>
DM	SETITEMPOSITION
DM	GETDLGITEM
DM	GETDLGRECT

M_GETSELECTION

essages | DM_SETSELECTION| Dialog API

The **DM_GETSELECTION** message is sent to the dialog handler to retrieve selection parameters in dialog edit lines.

aram1

Dialog item ID.

aram2

Pointer to a <u>EditorSelect</u> structure, where dialog manager will put the information about selection in edit line.

eturn

FALSE is returned if the item ID is not a line editor. TRUE is returned if the <u>EditorSelect</u> structure contains selection parameters.

ontrols

Control	Description
DI COMBOBOX	combined list
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field
<u>DI PSWEDIT</u>	password input field

emarks

xample

e also: DialogEx | DM SETSELECTION

M_GETTEXT

essages | Dialog API

The **DM_GETTEXT** message is sent to the dialog manager to retrieve the text of an edit string or the caption of an item.

aram1

The ID of the dialog item for which you want to retrieve the text.

aram2

Pointer to a <u>FarDialogItemData</u>structure that receives the text.

If **Param2** is NULL this is yet another way to retrieve necessary data size (see <u>DM_GETTEXTLENGTH</u>):

LenData=SendDlgMessage(hDlg,DM_GETTEXT,1,(LONG_PTR)NL

is equal to

```
LenData=SendDlgMessage(hDlg,DM_GETTEXTLENGTH,1,0);
```

eturn

Data size without terminating character '\0'.

ontrols

Control	Description
All	String data for edit lines, captions for other items. Everything contained in FarDialogItem.Data or FarDialogItem.PtrData.

emarks

The **DM_GETTEXT** message in FAR versions up to (and including) 1.70 beta 3 returned the size INCLUDING the terminating character.

xample

e also: <u>DialogEx</u> | <u>DM_SETTEXTPTR</u>

M_GETTEXTPTR

essages | Dialog API

The **DM_GETTEXTPTR** message is sent to the dialog manager to retrieve the text of an edit string or the caption of a dialog item. Unlike the <u>DM_GETTEXT</u> message, this one works with a string pointer.

aram1

The ID of the dialog item for which you want to retrieve the text.

aram2

Pointer to destination string that receives the text. If **Param2** is NULL this is yet another way to get the size of the text (see <u>DM_GETTEXTLENGTH</u>).

eturn

Size of the specified dialog item string NOT including the terminating character '\0'.

ontrols

Control	Description
All	String data is returned for edit lines, captions for any other items. Everything contained in FarDialogItem.Data or FarDialogItem.PtrData.

emarks

The **DM_GETTEXTPTR** message in FAR versions up to 1.70 beta 3 returned the string length INCLUDING the terminating character.

xample

e also: DialogEx | DM_GETTEXT

M_GETTEXTLENGTH

essages | Dialog API

The **DM_GETTEXTLENGTH** message is sent to the dialog manager to retrieve the size of data from <u>FarDialogItem.Data</u> or value of FarDialogItem.Ptr.PtrLength field.

aram1

The ID of the dialog item for which you want to retrieve the text size.

aram2

0

eturn

Length of the specified dialog item string NOT including the terminating character '\0'.

ontrols

Control	Description
All	All dialog items.

emarks

The **DM_GETTEXTLENGTH** message in FAR version up to 1.70 beta 3 returned the string length INCLUDING the terminating character.

xample

e also: DialogEx

M_KEY

essages | Dialog API

Using the **DM_KEY** message a plugin sends a <u>key codes</u> array to the dialog manager.

aram1

The number of passed keys.

aram2

Pointer to an array (DWORD type) of internal FAR key codes.

eturn

0

emarks

The dialog manager will not receive the <u>DN_KEY</u> event in response to the **DM_KEY** message.

xample

Example of sending the key input to the dialog (focus item):

```
// add semicolon to the end of edit string
DWORD Keys[2]={KEY_END,';'};
```

Info.SendDlgMessage(hDlg,DM_KEY,sizeof(Keys)/sizeof(K

e also: <u>DialogEx</u>, <u>DN KEY</u>

M_LISTADD

essages | Dialog API

The **DM_LISTADD** message is sent to the dialog manager to add items to a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item to which items should be added.

aram2

Pointer to a <u>FarList</u> structure describing the data to be added.

eturn

TRUE if the items have been added to the list. FALSE in case of error while adding the items.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

e also: DialogEx
M_LISTADDSTR

essages | Dialog API

The **DM_LISTADDSTR** message is sent to the dialog manager to add a string to a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item to which the string should be added.

aram2

Pointer to the string to be added.

eturn

Index of the string added to the list.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

e also: DialogEx

M_LISTDELETE

essages | Dialog API

The **DM_LISTDELETE** message is sent to the dialog manager to delete items from a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item from which the items should be deleted.

aram2

Pointer to a <u>FarListDelete</u> structure describing the delete parameters, or NULL.

eturn

TRUE if the items have been deleted from the list. FALSE if an error occurred while deleting data.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

If **Param2** is equal to NULL, all items will be deleted from the list.

xample

```
// *** delete first two list items ***
struct FarListDelete FLDItem;
FLDItem.StartIndex=0;
FLDItem.Count=2;
Info.SendDlgMessage(hDlg,DM_LISTDELETE,ID,(LONG_PTR)&
// *** clear all list ***
   // Method 1:
   struct FarListDelete FLDItem={0,0};
   Info.SendDlgMessage(hDlg,DM_LISTDELETE,ID,(LONG_PTF
   // Method 2:
   Info.SendDlgMessage(hDlg,DM_LISTDELETE,ID,(LONG_PTF
```

e also: <u>DialogEx</u>, <u>FarListDelete</u>

M_LISTFINDSTRING

essages | Dialog API

The **DM_LISTFINDSTRING** message is sent to the dialog manager to find an element by pattern in a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item in which the item is searched.

aram2

Pointer to a <u>FarListFind</u>structure specifying the element to find.

eturn

Index of the found element (0-based), or -1 if the specified item was not found.

ontrols

Control	Description
<u>DI_LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

e also: <u>FarListFind</u> | <u>DialogEx</u>

M_LISTGETCURPOS

essages | Dialog API

The **DM_LISTGETCURPOS** message is sent to the dialog manager to determine the current cursor position in a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item for which the cursor position is determined.

aram2

Pointer to a <u>FarListPos</u>, structure that will contain additional information about the current position, or NULL if the additional information is not required.

eturn

Current cursor position in the list.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

```
// get current position
```

CurPos=Info.SendDlgMessage(hDlg,DM_LISTGETCURPOS,ID,C

e also: DialogEx, DM_LISTSETCURPOS

M_LISTGETDATA

essages | Dialog API

The **DM_LISTGETDATA** message is sent to the dialog manager to retrieve data associated with an item in a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item for which the item data is retrieved.

aram2

Index of the item for which the data is retrieved.

eturn

Data value the plugin has associated with the list item using <u>DM_LISTSETDATA</u> message or NULL if no data was associated. See notes for <u>FarListItemData</u> structure for more details about data storing logic.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

e also: DialogEx, DM_LISTSETDATA

M_LISTGETDATASIZE

essages | Dialog API

The **DM_LISTGETDATASIZE** message is sent to the dialog manager to retrieve the size of the data associated with an item in a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item for which the item data is retrieved.

aram2

Index of the item for which the data is retrieved.

eturn

Size of the data value the plugin has associated with the item using <u>DM_LISTSETDATA</u> messages.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

e also: DialogEx, DM_LISTSETDATA DM_LISTGETDATA

M_LISTGETITEM

essages | Dialog API

The **DM_LISTGETITEM** message is sent to the dialog manager to retrieve a single item from a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item from which the elements should be retrieved.

aram2

Pointer to a <u>FarListGetItem</u>, structure that will contain the retrieved data.

eturn

TRUE - data has been retrieved. FALSE - error retrieving data.

ontrols

Control	Description
DI LISTBOX	list
DI COMBOBOX	combined list

emarks

Before retrieving a list element, you should fill the <u>FarListGetItem</u>structure. The **FarListGetItem.ItemIndex** field should contain the index of the element to be retrieved.

```
struct FarListGetItem List;
List.ItemIndex=Index;
Info.SendDlgMessage(hDlg,DM_LISTGETITEM,ID,(LONG_PTR)
```

e also: <u>DialogEx, FarListGetItem</u>

M_LISTGETTITLES

essages | Dialog API

The **DM_LISTGETTITLES** message is sent to the dialog manager to retrieve the titles (header and footer) of a <u>DI_LISTBOX</u> list.

aram1

he ID of the dialog item for which the titles are retrieved.

aram2

Pointer to a <u>FarListTitles</u> structure where the titles will be stored.

eturn

TRUE - titles have been retrieved successfully. FALSE - the item is not a list.

ontrols

Control	Description
<u>DI LISTBOX</u>	list

emarks

You must initialize the <u>FarListTitles</u>structure before sending the message:

```
char Title[100];
char Bottom[100];
FarListTitles ListTitle;
ListTitle.Title=Title;
ListTitle.TitleLen=sizeof(Title);
ListTitle.Bottom=Bottom;
ListTitle.BottomLen=sizeof(Bottom);
Info.SendDlgMessage(hDlg,DM_LISTGETTITLES,ID,(LONG_PT
```

e also: DialogEx, FarListTitles, DM_LISTSETTITLES

M_LISTINFO

essages | Dialog API

The **DM_LISTINFO** message is sent to the dialog manager to retrieve information about a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u>list.

aram1

The ID of the dialog item for which the information is retrieved.

aram2

Pointer to a <u>FarListInfo</u> structure that will contain the information about the list.

eturn

TRUE - the <u>FarListInfo</u> structure has been filled successfully. FALSE - error retrieving information (Param2 is NULL).

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

```
// get info about the list
struct FarListInfo ListInfo;
Info.SendDlgMessage(hDlg,DM_LISTINFO,ID,(LONG_PTR)&Li
```

e also: <u>FarListInfo</u> | <u>DialogEx</u>

M_LISTINSERT

essages | Dialog API

The **DM_LISTINSERT** message is sent to the dialog manager to insert an element into a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item in which the element is inserted.

aram2

Pointer to a <u>FarListInsert</u>structure specifying the element to insert.

eturn

New number of elements in the list, or -1 in case of an error during insert.

ontrols

Control	Description
<u>DI_LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

e also: <u>DialogEx</u>

M_LISTSET

essages | Dialog API

The **DM_LISTSET** message is sent to the dialog manager to replace the contents of a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list with the given list of elements.

aram1

The ID of the dialog item in which the elements are replaced.

aram2

Pointer to a <u>FarList</u> structure specifying the new contents of the list.

eturn

TRUE if the elements have been added successfully FALSE in case of an error.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

```
FarList List;
FarListItem *ListItems;
...
ListItems=new FarListItem[MenuItemsNumber];
List.ItemsNumber=MenuItemsNumber;
List.Items=ListItems;
...
Info.SendDlgMessage(hDlg,DM_LISTSET,0,(LONG_PTR)&List
```

e also: <u>DialogEx</u>

M_LISTSETCURPOS

essages | Dialog API

The **DM_LISTSETCURPOS** message is sent to dialog message to set the cursor position in <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u>.

aram1

The ID of the dialog item.

aram2

Param2 contains pointer to FarListPos structure.

eturn

Modified cursor position in the list taking into account separators and unavailable items.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

```
// set current position
```

```
RealPos=Info.SendDlgMessage(hDlg,DM_LISTSETCURPOS,ID,
```

e also:

DialogEx, DM LISTGETCURPOS

M_LISTSETDATA

essages | Dialog API

The **DM_LISTSETDATA** message is sent to the dialog manager to associate the user data with an element of a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u>.

aram1

The ID of the dialog item for which the data is associated.

aram2

Pointer to a <u>FarListItemData</u>, structure describing the data added.

eturn

Size of the data added, or 0 if the data is incorrect or there is not enough memory.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

e also: DialogEx, DM LISTGETDATA

M_LISTSETMOUSEREACTION

essages | Dialog API

The **DM_LISTSETMOUSEREACTION** message is sent to the dialog manager to set the behavior for handling mouse movement in a <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item for which the behaviour is set.

aram2

Can have one of the following values (listing of FARLISTMOUSEREACTIONTYPE):

Value	Description
LMRT_ONLYFOCUS	The list will react on mouse movement (change cursor position in the list) only when the item is in focus
LMRT_ALWAYS	The list will always react on mouse movement (change cursor position in the list). The example of such behavior is a list of found files in search results (Alt-F7)
LMRT_NEVER	The list will not react on mouse movement

eturn

Previous value.

ontrols

Control	Description
DI LISTBOX	list

emarks

- 1. By default, a nonfocused <u>DI_LISTBOX</u> item reacts to mouse movement.
- 2. <u>Far Manager 1.70 beta 5 and earlier</u>: Param2 different of LMRT_ONLYFOCUS or LMRT_ALWAYS is similar to LMRT_ALWAYS value.

xample

e also: <u>DialogEx</u>

M_LISTSETTITLES

essages | Dialog API

The **DM_LISTSETTITLES** message is sent to the dialog manager to set the titles (header and footer) of a <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item for which the titles are set.

aram2

Pointer to a <u>FarListTitles</u>, structure containing the titles to be set.

eturn

TRUE - titles have been set. FALSE - the item is not a list.

ontrols

Control	Description
<u>DI LISTBOX</u>	list

emarks

You must fill the <u>FarListTitles</u> structure before sending the message:

```
FarListTitles ListTitle;
ListTitle.Title="Some title";
ListTitle.Bottom=NULL; // reset footer
Info.SendDlgMessage(hDlg,DM_LISTSETTITLES,ID,(LONG_PT
```

e also: <u>DialogEx</u>, <u>FarListTitles</u>, <u>DM_LISTGETTITLES</u>

M_LISTSORT

essages | Dialog API

The **DM_LISTSORT** message is sent to the dialog manager to sort the elements of a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list.

aram1

The ID of the dialog item to sort.

aram2

Sort direction: 0 - ascending, 1 - descending.

eturn

TRUE - the list has been sorted. FALSE - error sorting list.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

xample

```
// sort the list descending (from 'Z' to 'A')
Info.SendDlgMessage(hDlg,DM_LISTSORT,ID,1);
```

e also: DialogEx

M_LISTUPDATE

essages | Dialog API

The **DM_LISTUPDATE** message is sent to the dialog manager to update an element of a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u>.

aram1

The ID of the dialog item to be updated.

aram2

Pointer to a <u>FarListUpdate</u>structure specifying the element to update.

eturn

TRUE - the list element has been successfully updated. FALSE - error updating data.

ontrols

Control	Description
<u>DI LISTBOX</u>	list
DI COMBOBOX	combined list

emarks

It is also possible to use the <u>DM_SETTEXTPTR</u> (or <u>DM_SETTEXT</u>) message to update the text of the current item of the <u>DI_LISTBOX</u> list.

xample

e also: <u>DialogEx</u>

M_MOVEDIALOG

essages | Dialog API

The **DM_MOVEDIALOG** message is sent to the dialog manager to move the dialog window.

aram1

TRUE - **Param2** contains absolute coordinates. FALSE - **Param2** contains relative coordinates.

aram2

Pointer to a <u>COORD</u> structure containing the new coordinates of the upper left corner of the dialog window.

eturn

The <u>COORD</u> structure containing the current coordinates of the upper left corner of the dialog window.

emarks

In order to center the dialog window on the screen fill the <u>COORD</u> structure with -1 values and set **Param1** = TRUE.

xample

```
// center the dialog
COORD c={-1,-1};
Info.SendDlgMessage(hDlg,DM_MOVEDIALOG,TRUE,(LONG_PTF
// move the dialog to the top left corner of the scre
COORD c={1,1};
Info.SendDlgMessage(hDlg,DM_MOVEDIALOG,TRUE,(LONG_PTF
// move the dialog by 2 positions to the right
COORD c={2,0};
Info.SendDlgMessage(hDlg,DM_MOVEDIALOG,FALSE,(LONG_PT
```

e also: DM RESIZEDIALOG DialogEx

M_REDRAW

essages | Dialog API

The **DM_REDRAW** message is sent to the dialog manager to redraw the entire dialog window.

aram1

0

aram2

0

eturn

0

emarks

In FAR versions up to 1.70 beta 3 **DM_REDRAW** was called DM_SETREDRAW.

e also:

<u>DialogEx</u>

M_RESIZEDIALOG

essages | Dialog API

The **DM_RESIZEDIALOG** message is sent to the dialog manager to resize the dialog window.

aram1

0

aram2

Pointer to a <u>COORD</u> structure containing the new size of the dialog window.

eturn

The <u>COORD</u> structure containing the new size of the dialog window.

emarks

xample

e also: <u>DM_MOVEDIALOG</u> <u>DialogEx</u>

M_SET3STATE

essages | Dialog API

The **DM_SET3STATE** message is sent to the dialog manager to change the style of a <u>DI_CHECKBOX</u> item.

aram1

The ID of the dialog item that you want to change.

aram2

- TRUE the check box can have 3 states
- FALSE the check box can have only 2 states.

eturn

Previous style of the item.

ontrols

Control	Description
DI CHECKBOX	Check Box

emarks

xample

e also: <u>DialogEx</u> | <u>DM_GETCHECK</u> | <u>DM_SETCHECK</u>.

M_SETCHECK

essages | Dialog API

The **DM_SETCHECK** message is sent to the dialog manager to change the state of <u>DI_CHECKBOX</u> and <u>DI_RADIOBUTTON</u> items.

aram1

The ID of the dialog item that you want to change.

aram2

Param2 applies only <u>DI_CHECKBOX</u> and can have one of the following values (listing of FARCHECKEDSTATE):

Flag	Description
BSTATE_UNCHECKED	Off - [] or ()
BSTATE_CHECKED	On - [X] or (*)
BSTATE_3STATE	Set to undefined state - [?] (applies only to the items with <u>DIF_3STATE</u> flag set).
BSTATE_TOGGLE	Toggle the item state. For elements with <u>DIF_3STATE</u> flag set the state will be toggled sequentially between "on", "off", "undefined").

eturn

Previous state of the item. For <u>DI_RADIOBUTTON</u> returns the ID of the dialog item in the radio button group that was previously checked. (see <u>DIF_GROUP</u>).

ontrols

Control	Description
DI CHECKBOX	Check Box
DI RADIOBUTTON	Radio Button

emarks

xample

e also: DialogEx | DM_GETCHECK | DM_SET3STATE.

M_SETCOMBOBOXEVENT

essages | Dialog API

The **DM_SETCOMBOBOXEVENT** message allows to control the sending of <u>DN_KEY</u> or <u>DN_MOUSEEVENT</u> events for an open <u>DI_COMBOBOX</u>.

aram1

ID of a <u>DI_COMBOBOX</u>.

aram2

Param2 is a combination of the following flags (FARCOMBOBOXEVENTTYPE enum):

Flag	Description
CBET_KEY	Enable <u>DN KEY</u> events
CBET_MOUSE	Enable DN MOUSEEVENT events

eturn

Previous state.

ontrols

Control	Description
DI COMBOBOX	combined list

emarks

By default, the <u>DN_KEY</u> and <u>DN_MOUSEEVENT</u> events are not sent for an open <u>DI_COMBOBOX</u>.

xample

e also: DM GETCOMBOBOXEVENT

M_SETCURSORPOS

essages | Dialog API

The **DM_SETCURSORPOS** message is sent to dialog manager to position the cursor in dialog edit lines and <u>DI_USERCONTROL</u> items.

aram1

The ID of the dialog item

aram2

Pointer to a <u>COORD</u> structure containing information about cursor position (COORD.Y is ignored for line editor).

eturn

FALSE - the item is not a line editor or <u>DI_USERCONTROL</u> item.

ontrols

Control	Description
DI COMBOBOX	combined list
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field
<u>DI PSWEDIT</u>	password input field
DI USERCONTROL	custom (defined by programmer) control

emarks

To switch off the cursor in <u>DI_USERCONTROL</u> item set fields of <u>COORD</u> structure to -1.

xample

e also: <u>DialogEx</u>

M_SETCURSORSIZE

essages | Dialog API

The **DM_SETCURSORSIZE** message is sent to the dialog manager to set the cursor size and its visibility flag in edit lines or <u>DI_USERCONTROL</u> item.

aram1

The ID of the dialog item for which you want to retrieve the cursor position.

aram2

The high word is the cursor size (from 0 to 100, as used in the <u>CONSOLE_CURSOR_INFO</u>) structure), the low word is the cursor visibility flag (1 - the cursor is visible, 0 - the cursor is hidden).

eturn

Previous values:

The high word is the cursor size (from 0 to 100, as used in the <u>CONSOLE_CURSOR_INFO</u> structure), the low word is the cursor visibility flag (1 - the cursor is visible, 0 - the cursor is hidden).

ontrols

Control	Description
DI COMBOBOX	combined list (without <u>DIF_DROPDOWNLIST</u> flag)
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field
<u>DI_PSWEDIT</u>	password input field
DI USERCONTROL	custom (defined by programmer) control

emarks

1. To show the cursor in a <u>DI_USERCONTROL</u> dialog item, it is also necessary to move the cursor to the necessary position:

```
Coord.X=X;
Coord.Y=Y;
Info.SendDlgMessage(hDlg,DM_SETCURSORPOS,3,(LONG_F
```

The default cursor position in a <u>DI_USERCONTROL</u> is {-1, -1}.

2. To show a cursor filling the entire character cell under Windows 9x/Me set

the high word of **Param2** to 99.

xample

// set the cursor of 3rd element as fully filled cell
Info.SendDlgMessage(hDlg,DM_SETCURSORSIZE,3,(LONG_PTF

e also:

DialogEx | DM_GETCURSORSIZE| DM_GETCURSORPOS| DM_SETCURSORPOS|

M_SETDLGDATA

essages | Dialog API

The **DM_SETDLGDATA** message is sent to the dialog manager to set the 32bit data value associated with the dialog. This value is designed for use by the plugin that created the dialog.

aram1

0

aram2

New 32-bit value.

eturn

Previous value.

emarks

Initially the Param value of <u>DialogEx</u> is here.

xample

e also: <u>DialogEx</u> <u>DM_GETDLGDATA</u>

M_SETDLGITEM

essages | Dialog API

The **DM_SETDLGITEM** message is sent to the dialog manager to change a specified dialog item.

aram1

The ID of the dialog item that you want to change.

aram2

Pointer to a <u>FarDialogItem</u> structure containing the new item data.

eturn

TRUE if the new dialog item data has been set successfully FALSE if the item with such ID does not exist or the item type was changed.

ontrols

Control	Description
All	All dialog items

emarks

You can't change item type.

e also:

DialogEx

M_SETDROPDOWNOPENED

essages | Dialog API

The **DM_SETDROPDOWNOPENED** message is sent to the dialog manager to open or close a combo box or history list.

aram1

The ID of the element for which the combo box (<u>DI_COMBOBOX</u>) or history list (<u>DI_EDIT</u> or <u>DI_FIXEDIT</u> with the <u>DIF_HISTORY</u> flag) should be opened.

aram2

TRUE - open the list for the element with the given ID. FALSE - close the open combo box or history list. In this case *Param1* is ignored.

eturn

If **Param2**=FALSE the returned value is always TRUE.

If **Param2**=TRUE, the returned value is TRUE if the combo box or history list has been opened successfully. FALSE is returned if ID of not suppored type was set.

ontrols

Control	Description
DI_COMBOBOX	combined list
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field

emarks

Only one combo box or history list can be open in a dialog at the same time. Therefore, sending this message with *Param2*=TRUE closes an already opened combo box before opening the specified one, and when sending the message with *Param2*=FALSE it is not needed to specify the item *ID*.

xample

e also: <u>DialogEx</u> <u>DM_GETDROPDOWNOPENED</u>

M_SETEDITPOSITION

essages | DM_SETEDITPOSITION | Dialog API

The **DM_SETEDITPOSITION** message is sent to the dialog manager to set the cursor position and state in edit controls.

aram1

Dialog item ID

aram2

Pointer to a EditorSetPosition structure.

eturn

FALSE - the given dialog item ID is not an edit control. TRUE - cursor position is set.

ontrols

Control	Description
DI COMBOBOX	combined list
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed size input field
<u>DI PSWEDIT</u>	password input field

emarks

xample

Example of a mouse selection support function for edit controls from the "Visual renaming files" plugin

```
static void MouseSelect(HANDLE hDlg, DWORD idStr, DWC
{
    SMALL_RECT dlgRect, itemRect;
    Info.SendDlgMessage(hDlg, DM_GETDLGRECT, 0, (LONG_F
    Info.SendDlgMessage(hDlg, DM_GETITEMPOSITION, idStr
    EditorSetPosition esp;
    Info.SendDlgMessage(hDlg, DM_GETEDITPOSITION, idStr
```

```
int length=Info.SendDlgMessage(hDlg, DM_GETTEXTLENC
int CurPos=dwMousePosX - ( dlgRect.Left + itemRect.
if ( dwMousePosX <= ( dlgRect.Left + itemRect.Left</pre>
  esp.LeftPos-=1;
else if ( dwMousePosX >= ( dlgRect.Left + itemRect.
  esp.LeftPos+=1;
if (CurPos+esp.LeftPos < 0)
  CurPos=0;
else if (CurPos+esp.LeftPos > length)
  CurPos=length;
else
  CurPos+=esp.LeftPos;
esp.CurPos=esp.CurTabPos=CurPos;
if (bStartSelect)
{
  StartPosX=CurPos;
 bStartSelect=false;
}
EditorSelect es;
es.BlockType=BTYPE_COLUMN;
es.BlockStartLine=es.BlockHeight=0;
if (CurPos > StartPosX)
{
 es.BlockStartPos=StartPosX;
 es.BlockWidth=CurPos-StartPosX;
}
else
{
  es.BlockStartPos=CurPos;
  es.BlockWidth=StartPosX-CurPos;
}
Info.SendDlgMessage(hDlg, DM_SETSELECTION, idStr, (
```

Info.SendDlgMessage(hDlg, DM_SETEDITPOSITION, idStr }

e also:

DialogEx | DM_GETCURSORPOS| DM_SETCURSORPOS| DM_GETEDITPOSITION

M_SETFOCUS

essages | Dialog API

The **DM_SETFOCUS** message sets the keyboard focus to the given dialog item.

aram1

The ID of the dialog item that receives the input focus.

aram2

0

eturn

FALSE - the given dialog item cannot receive the keyboard focus TRUE - the focus has been set successfully.

ontrols

Control	Description
All	All items that can have <u>keyboard input focus</u> .

emarks

The **DM_SETFOCUS** spawns the following events: <u>DN_KILLFOCUS</u> for the element that loses focus and <u>DN_GOTFOCUS</u> for the element that gets focus. The DN_KILLFOCUS and DN_GOTFOCUS events are not generated when *Param1* is equal to the item currently having focus.

e also:

<u>DialogEx</u>

M_SETHISTORY

essages | Dialog API

The **DM_SETHISTORY** allows to manage availability of history in edit lines.

aram1

The ID of the dialog item for which the history is changed.

aram2

Pointer to a string containing the history list ID, or NULL, if the history should be disabled.

eturn

FALSE - the dialog item specified in *Param1* does not support history lists. TRUE - the history has been assigned successfully.

ontrols

Control	Description
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field

emarks

If the edit line is empty and the flag <u>DIF_USELASTHISTORY</u> for the <u>DI_EDIT</u> item is set, FAR will automatically subsitute the first value from the history into an empty edit line.

xample

Implementation of this message can be seen in MultiArc plugin sources.

This example sets history and empties edit line:

```
...
else if(Msg == MAM_ARCSWITCHES)
{
    static char SwHistoryName[NM];
    FSF.sprintf(SwHistoryName,"ArcSwitches\\%s",pdd->
    Info.SendDlgMessage(hDlg,DM_SETHISTORY,4,(LONG_PT
    Info.SendDlgMessage(hDlg,DM_SETTEXTPTR,4,(LONG_PT
  }
```
. . .

In order to make FAR substitute the first line from keys histiry the code should be rewritten in the following way:

```
else if(Msg == MAM_ARCSWITCHES)
{
    static char SwHistoryName[NM];
    FSF.sprintf(SwHistoryName, "ArcSwitches\\%s",pdd->
    Info.SendDlgMessage(hDlg,DM_SETTEXTPTR,4,(LONG_PT
    Info.SendDlgMessage(hDlg,DM_SETHISTORY,4,(LONG_PT
    }
...
```

The edit line is being emptied and the item history is set after.

e also: DialogEx, DM_ADDHISTORY, DIF_HISTORY

M_SETITEMDATA

essages | Dialog API

The **DM_SETITEMDATA**message is sent to the dialog manager to set the 32bit data value associated with a dialog item. Every dialog item has a corresponding 32-bit data value designed for use by the plugin that created the dialog.

aram1

The ID of the dialog item for which the data is retrieved.

aram2

The new 32-bit data value.

eturn

Previous data value.

emarks

xample

e also: <u>DialogEx</u> <u>DM GETITEMDATA, DM GETDLGDATA,</u> <u>DM SETDLGDATA</u>

M_SETITEMPOSITION

essages | Dialog API

The **DM_SETITEMPOSITION** message is sent to the dialog manager to change the position of a dialog item.

aram1

The ID of the dialog item for which the position is changed.

aram2

Pointer to a <u>SMALL_RECT</u> structure containing the new item coordinates.

eturn

TRUE - item position has been changed. FALSE - item wiht such ID does not exist.

ontrols

Control	Description
All	All dialog items

e also: <u>DialogEx</u> <u>DM GETITEMPOSITION</u> <u>DM RESIZEDIALOG</u>

M_SETMAXTEXTLENGTH

essages | Dialog API

The **DM_SETMAXTEXTLENGTH** message is sent to the dialog manager to set the maximum length of an edit string.

aram1

The ID of a dialog item.

aram2

New maximum length of the string.

eturn

The previous value of the length of the string or 0 in case of error.

ontrols

Control	Description
DI COMBOBOX	combined list
<u>DI EDIT</u>	edit line
DI PSWEDIT	password input field

emarks

- 1. For <u>DI_COMBOBOX</u> items affects only the edit line (has no effect for items with the <u>DIF_DROPDOWNLIST</u> flag set)
- 2. In FAR versions before 1.70 beta 4, this message was called DM_SETTEXTLENGTH.

xample

e also:

<u>DialogEx</u>

M_SETMOUSEEVENTNOTIFY

essages | Dialog API

The **DM_SETMOUSEEVENTNOTIFY** message allows to control the mechanism for sending initial non-altered mouse event (<u>DN_MOUSEEVENT</u>) to a dialog manager before they are processed by the dialog kernel.

aram1

0 - off 1 - on -1 - get state

aram2

0

eturn

Previous state.

ontrols

Control	Description
Dialog	message applies to the dialog only

emarks

xample

e also: <u>DialogEx</u> <u>DN_MOUSEEVENT</u>

M_SETSELECTION

<u>ssages | DM_GETSELECTION | Dialog API</u>

The **DM_SETSELECTION** is sent to dialog manager to select a block in dialog edit lines.

aram1

The ID of dialog item

aram2

Pointer to an EditorSelect structure.

eturn

FALSE - the item is not an edit line. TRUE - the text was selected.

ontrols

Control	Description
DI COMBOBOX	combined list
<u>DI EDIT</u>	edit line
<u>DI FIXEDIT</u>	fixed-size input field
DI PSWEDIT	password input field

emarks

xample

e also: DialogEx | DM GETSELECTION

M_SETTEXT

essages | Dialog API

The **DM_SETTEXT** message is sent to the dialog manager to set a new string value for an edit line or a new caption for an item.

aram1

The ID of the dialog item for which the text should be changed.

aram2

Pointer to the <u>FarDialogItemData</u> structure that is used to set the dialog item text.

If this parameter is equal to NULL, the message does nothing.

eturn

Size of the new dialog item string NOT including the terminating character '\0'.

ontrols

Control	Description
All	All dialog items

emarks

- 1. For <u>DI_COMBOBOX</u> items affects only the edit line (has no effect for items with the <u>DIF_DROPDOWNLIST</u> flag set).
- 2. The **DM_SETTEXT** message in FAR version prior to 1.70 beta 4 returned the string length INCLUDING the terminating NULL character.

xample

e also: DialogEx | DM_SETTEXTPTR

M_SETTEXTPTR

essages | Dialog API

The **DM_SETTEXTPTR** message is sent to the dialog manager to set a new string value for an edit line or a new caption for an item. Unlike the <u>DM_SETTEXT</u> message, this message works with a string pointer.

aram1

The ID of the dialog item for which the text should be changed.

aram2

Pointer to the string containing the new text. If this parameter is equal to NULL the message does nothing.

eturn

Size of the new dialog item string NOT including terminating character '\0'.

ontrols

Control	Description
All	All dialog items

emarks

- 1. For <u>DI_COMBOBOX</u> items affects only the edit line (has no effect for items with the <u>DIF_DROPDOWNLIST</u> flag set).
- 2. The **DM_SETTEXTPTR** message in FAR versions prior to 1.70 beta 4 returned the string length INCLUDING the terminating NULL character.

xample

e also: DialogEx | DM_SETTEXT

M_SHOWDIALOG

essages | Dialog API

The **DM_SHOWDIALOG** message is sent to the dialog manager to show or hide the dialog window on the screen.

aram1

TRUE - show the dialog window. FALSE - hide the dialog window.

aram2

0

eturn

0

e also:

DM_SHOWITEM DialogEx

M_SHOWITEM

essages | Dialog API

The **DM_SHOWITEM** message is sent to the dialog manager to show or hide a dialog item.

aram1

The ID of the dialog item to be shown or hidden.

aram2

One of the following values:

- -1 get the state of the item
- 0 hide the item
- 1 show the item

eturn

Previous state of the dialog item

ontrols

Control	Description
All	All dialog items

emarks

- 1. Dialog item is force redrawn if it is visible and Param2=1.
- 2. Dialog item is not redrawn if Param2=-1.

xample

e also: <u>DM_SHOWDIALOG</u> <u>DialogEx</u>

M_USER

essages | Dialog API

The **DM_USER** message is intended to specify the starting value for userdefined messages.

aram1

User-defined value.

aram2

User-defined value.

eturn

User-defined value.

ontrols

Control	Description
All	All dialog items

emarks

xample

```
#define DMREV_RESULT DM_USER+1
LONG_PTR WINAPI ReversiDialogProc(HANDLE hDlg, int Ms
{
...
Info.SendDlgMessage(hDlg,DMREV_RESULT,0,0);
...
case DMREV_RESULT: // user's definition
...
return 0;
...
}
```

ialog API Events

Event	Description
DN_CLOSE	Sent before the dialog is closed.
DN BTNCLICK	Sent after a button, radiobutton or checkbox was clicked.
DN CTLCOLORDIALOG	Sent before the dialog background is drawn, for changing its color.
DN CTLCOLORDLGITEM	Sent before a dialog item is drawn, for changing its color.
DN CTLCOLORDLGLIST	Sent before a list is drawn, for changing its color.
DN DRAGGED	Notification of dialog dragging.
DN DRAWDIALOG	Sent before the dialog is drawn.
DN DRAWDIALOGDONE	Sent after the dialog has been drawn.
DN DRAWDLGITEM	Sent before a dialog item is drawn.
DN EDITCHANGE	Sent when the text in the dialog edit control has changed.
DN ENTERIDLE	Sent when the dialog enters the idle state.
DN GOTFOCUS	Sent when the dialog item gets input focus.
DN HELP	Sent before a help topic is displayed.
DN HOTKEY	Sent after the user has pressed an hotkey (Alt-).
DN INITDIALOG	Sent before initializing the dialog.
DN KEY	Sent after the user has pressed a key in the dialog.
DN KILLFOCUS	Sent before a dialog item loses the input focus.
DN LISTCHANGE	Sent when a list item is changed.
DN LISTHOTKEY	Sent after the user has pressed an hotkey (Alt-) inside a list.
DN MOUSECLICK	Sent after the user has clicked one of the dialog items with the mouse.
DN MOUSEEVENT	Sent before a mouse event is handled by the dialog kernel.
DN RESIZECONSOLE	Sent after the size of the console window has changed.

Dialog API Messages

N_BTNCLICK

<u>ents</u> | <u>Dialog API</u>

The **DN_BTNCLICK** event is received by the following items <u>DI_BUTTON</u>, <u>DI_CHECKBOX</u>, <u>DI_RADIOBUTTON</u> after the user pressed Enter or Space (for buttons), '+', '-' or '*'.

aram1

Dialog item ID.

aram2

State of the item after the keystroke: <u>DI_BUTTON</u> = 0 <u>DI_CHECKBOX</u> =0 (unchecked), 1 (checked) and 2 (undefined, if the <u>DIF_3STATE</u> flag is set) <u>DI_RADIOBUTTON</u> = 0 - for the previous element in the group, 1 - for the active element in the group.

eturn

For <u>**DI_BUTTON</u>**: TRUE - if the message has been handled and it should not be processed by the kernel.</u>

FALSE - continue processing the message by the kernel.

For other elements: TRUE - to apply the changes and redraw the control. FALSE - to discard changes.

ontrols

Control	Description
<u>DI BUTTON</u>	Push Button.
DI CHECKBOX	Check Box.
DI RADIOBUTTON	Radio Button.

emarks

xample

e also: <u>DialogEx</u>

N_CLOSE

ents | Dialog API

The **DN_CLOSE** event is sent to the dialog callback function as a notification before the dialog is closed - the user wants to close the dialog.

aram1

The ID of the dialog item that had the keyboard focus when Ctrl+Enter was pressed or that has the DefaultButton field set to 1.

- ID of the dialog item that had the keyboard focus when Ctrl+Enter was pressed or ID of the dialog item that has the DefaultButton field set to 1.
- -2 if KEY_BREAK was pressed (Ctrl-Pause or Ctrl-ScrollLock)
- -1 KEY_ESC or KEY_F10

aram2

0

eturn

TRUE - to allow closing the dialog FALSE - to continue working with the dialog.

emarks

1. The **DN_CLOSE** event is received immediately after the **DM_CLOSE** message is sent.

xample

e also: <u>DialogEx</u>, <u>DM</u> <u>CLOSE</u>

N_CTLCOLORDIALOG

ents | <u>Dialog API</u>

The **DN_CTLCOLORDIALOG** event is sent to the dialog callback function before the dialog background is drawn.

aram1

0

aram2

Color attribute (foreground+background) that the callback function wants to use to draw the dialog background. By default, this attribute is equal to COL_DIALOGTEXT for standard dialogs and COL_WARNDIALOGTEXT for dialogs with the <u>FDLG_WARNING</u> flag set.

eturn

Color attribute (foreground+background) that the callback function should use to draw the dialog background.

emarks

This event is received immediately after **DN_DRAWDIALOG**.

xample

e also:

<u>DialogEx</u>

N_CTLCOLORDLGITEM

ents | <u>Dialog API</u>

The **DN_CTLCOLORDLGITEM** event is sent to the dialog callback function before each dialog item is drawn.

aram1

ID of the dialog item that is about to be drawn.

aram2

Color attribute (foreground+background) that the callback function wants to use to draw dialog item:

DI_SINGLEBOX DI_DOUBLEBOX

LO(LO=Title,HI=HiText), HI(LO=Frame,0)

DI_TEXT

```
LO(LO=Title,HI=HiText), HI(LO=Frame,0)
The box color applies only to text items with the <u>DIF_SEPARATOR</u> and <u>DIF_SEPARATOR2</u> flags set.
```

<u>DI_VTEXT</u>

LO(LO=Title,HI=0), HI(0,0)

<u>DI_EDIT</u> <u>DI_FIXEDIT</u> <u>DI_PSWEDIT</u> <u>DI_COMBOBOX</u>

> LO(LO=EditLine,HI=SelectText), HI(LO=UnchangedColor, HI=History&ComboBox pointer)

<u>DI_CHECKBOX</u> <u>DI_RADIOBUTTON</u>

LO(LO=Title,HI=HiText), HI(0,0)

DI_BUTTON

```
LO(LO=Title,HI=HiText), HI(0,0)
```

DI_LISTBOX

For the <u>DI_LISTBOX</u> item only the <u>DN_CTLCOLORDLGLIST</u> event is recieved.

eturn

Color attribute (foreground+background) that the callback function should use to draw the dialog item.

ontrols

Control	Description
All	Applies to all dialog items with the exception of <u>DI_LISTBOX</u> and <u>DI_USERCONTROL</u>

emarks

xample

For example, this is how the macro assignment dialog in FAR Manager changes the color of input field:

```
/* 02 */ DI_EDIT,5,3,28,3,1,0,0,1,"",
case DN_CTLCOLORDLGITEM:
   // Unchanged resides in the Lo byte of the Hi word.
   Param2&=0xFF00FFFFU;
   Param2|=(Param2&0xFF)<<16;
   return Param2;
```

N_CTLCOLORDLGLIST

ents | <u>Dialog API</u>

The **DN_CTLCOLORDLGLIST** event is sent to the dialog callback function before each list item (<u>DI_COMBOBOX</u>, <u>DI_LISTBOX</u>, <u>DIF_HISTORY</u>) is drawn.

aram1

ID of the dialog item that is about to be drawn.

aram2

Pointer to a <u>FarListColors</u>, structure specifying the color scheme of the list.

eturn

TRUE - to apply the changes FALSE - to leave as is.

ontrols

Control	Description
<u>DI LISTBOX</u>	List
DI COMBOBOX	Combined List
<u>DI EDIT</u>	Text input line (with the <u>DIF_HISTORY</u> flag set)
<u>DI FIXEDIT</u>	Fixed sized text input line (with the <u>DIF_HISTORY</u> flag set)

emarks

The dialog callback function can use this event to change the colors used for drawing the list.

xample

e also: <u>DialogEx</u> | <u>FarListColors</u>

N_DRAGGED

<u>ents</u> | <u>Dialog API</u>

The **DN_DRAGGED** event is sent to the dialog callback function as a notification of the beginning and the ending of dragging the dialog.

ragging of the dialog has began

aram1

0

aram2

0

eturn

TRUE - allow dragging the dialog FALSE - forbid dragging the dialog

ragging of the dialog is finished

aram1

1

aram2

0 - dialog successfully displaced

1 - if the dragging of the dialog was cancelled by pressing Esc or the right mouse button.

eturn

Ignored.

emarks

1. The **DN_DRAGGED** event is sent before the dialog is redrawn.

xample

e also: <u>DialogEx</u>

N_DRAWDIALOG

ents | Dialog API

The **DN_DRAWDIALOG** event is sent to the dialog callback function before the whole dialog is drawn.

aram1

0

aram2

0

eturn

If the dialog callback function returns FALSE, the dialog is not drawn.

emarks

xample

e also: DialogEx, DN DRAWDIALOGDONE

N_DRAWDIALOGDONE

ents | Dialog API

The **DN_DRAWDIALOGDONE** event is sent to the dialog callback function after the whole dialog has been drawn.

aram1

0

aram2

0

eturn

Ignored.

emarks

Although this event is sent at the end of the drawing process, the virtual screen buffer is not yet "flushed". Meaning that if a plugin wants to draw something ontop the dialog, the changes might be erased from the screen when returning from the dialog callback function.

To prevent this, you must force the virtual screen buffer to be flushed using the Text(0, 0, 0, NULL) function, and only the "draw" your changes. For example:

```
case DN_DRAWDIALOGDONE:
    Info.Text(0,0,0,NULL);
    // draw ours here
    break;
```

xample

e also: DialogEx, DN_DRAWDIALOG

N_DRAWDLGITEM

ents | <u>Dialog API</u>

The **DN_DRAWDLGITEM** event is sent to the dialog callback function before a dialog item is drawn.

aram1

ID of the dialog item that is about to be drawn.

aram2

Pointer to the <u>FarDialogItem</u> structure for the item to be drawn.

eturn

If the dialog callback function returns FALSE, the item will not be drawn.

ontrols

Control	Description
All	All dialog items

emarks

A dialog item of the **DI_USERCONTROL** type, but not any other dialog item, can be prepared in advance in a <u>virtual buffer</u>. FAR Manager does not copy the contents of this buffer to its internal structures, so as a response to the **DN_DRAWDLGITEM** event it is sufficient to change only the state of the virtual buffer:

```
case DN_DRAWDLGITEM:
  CHAR_INFO *VBuf=((struct FarDialogItem*)Param2)->Pa
  for (I=0; I < 256; I++)
   {
     VBuf[I].Char.AsciiChar = DecodeTable[I];
     VBuf[I].Attributes = Color;
   }
  return TRUE;
```

xample

e also: <u>DialogEx</u>

N_EDITCHANGE

<u>ents | Dialog API</u>

The **DN_EDITCHANGE** event is sent to the dialog callback function when an edit item has changed (for example, a character has been entered).

aram1

The ID of the dialog item that has changed.

aram2

Pointer to the <u>FarDialogItem</u> structure for the item that has changed.

eturn

In Dialog API 1.0, the return value is ignored, but to ensure compatibility with future versions of Dialog API, it is recommended to return:

- TRUE to confirm the changes
- FALSE to cancel the changes

ontrols

Control	Description
DI COMBOBOX	Combined list
<u>DI EDIT</u>	Text input line
<u>DI FIXEDIT</u>	Fixed size text input line
DI PSWEDIT	Password input line

emarks

- 1. The ability to change the state of the dialog item by changing the elements of the <u>FarDialogItem</u> structure is missing in Dialog API 1.0.
- 2. The **DN_EDITCHANGE** event will not be sent in a response to the following messages: <u>DM_SETTEXTPTR</u> and <u>DM_SETTEXT</u>.

xample

```
// show a copy of the entered data only for numbers
if (Msg == DN_EDITCHANGE && Param1 == 2)
```

{
 FarDialogItem *Item=(FarDialogItem *)Param2;
 int Len=strlen(Item->Data);
 for(int I=0; I < Len; ++I)
 if(!isdigit(Item->Data[I]))
 break;
 if(I == Len)
 return FALSE;
 Info.SendDlgMessage(hDlg, DM_SETTEXTPTR, 3, (LONG_F
 return TRUE;
}
....

N_ENTERIDLE

ents | Dialog API

The **DN_ENTERIDLE** event is sent to the dialog callback function when the dialog enters the idle state.

• Attention! This event has nothing to do with the timer. The dialog enters the idle state when there are no events in the system (no keystrokes, no mouse moves).

aram1

0

aram2

0

eturn

0

xample

```
// while in Idle state let the computer calculate the
case DN_ENTERIDLE:
  Timer(hDlg);
  return 0;
```

N_GOTFOCUS

ents | Dialog API

The **DN_GOTFOCUS** event is sent to the dialog callback function after a dialog item has received keyboard focus.

aram1

ID of the dialog item that has received the focus.

aram2

0

eturn

0

ontrols

Control	Description
All	All elements that can receive <u>keyboard focus</u> .

emarks

The event is also sent to the focused element immediately after <u>DN_INITDIALOG</u>.

xample

e also: <u>DialogEx</u>

N_HELP

ents | Dialog API

The **DN_HELP** event is sent to the dialog callback function before a help topic is displayed.

aram1

ID of the dialog item that has the keyboard focus.

aram2

Pointer to a string containing the <u>name of a help topic</u> that the dialog callback function wants to display.

eturn

Pointer to a string containing the name of a help topic that the dialog callback function should display. If NULL is returned, no help topic will be displayed.

ontrols

Control	Description
All	All elements that can receive keyboard focus.

emarks

This event allows to control which help topic will be displayed based on the current dialog item.

xample

```
// based on the state of the game select which help t
// in this case the current dialog item is if no matt
case DN_HELP:
{
    static char *HelpTopics[3]={"Contents", "Rule", "Recc
    int i=1;
    if(GAME[0].NumPl1==2 && GAME[0].NumPl2 == 2)
        i=0;
    else if(GAME[0].NumPl1+GAME[0].NumPl2 > 16)
        i=2;
    return (LONG_PTR)(&HelpTopics[i][0]);
}
```

e also: <u>DialogEx</u>

N_HOTKEY

ents | Dialog API

The **DN_HOTKEY** event is sent to the dialog callback function when the user presses a hotkey (Alt-letter).

aram1

The ID of the dialog item that will receive the <u>keyboard focus</u> after the hotkey is processed.

aram2

Internal FAR key code.

eturn

TRUE - allow the hot key to be processed FALSE - cancel the hot key.

ontrols

Control	Description
<u>DI DOUBLEBOX</u>	double line frame; if it has a hotkey (the <u>DIF_SHOWAMPERSAND</u> flag is not set) then the focus is transfered to the nearest control that can have <u>keyboard focus</u> .
<u>DI SINGLEBOX</u>	single line frame; if it has a hotkey (the <u>DIF_SHOWAMPERSAND</u> flag is not set) then the focus is transfered to the nearest control that can have <u>keyboard focus</u> .
<u>DI_TEXT</u>	text label; if it has a hotkey (the <u>DIF_SHOWAMPERSAND</u> flag is not set) then the focus is transfered to the nearest control that can have <u>keyboard focus</u> .
<u>DI VTEXT</u>	vertical text label; if it has a hotkey (the <u>DIF_SHOWAMPERSAND</u> flag is not set) then the focus is transfered to the nearest control that can have <u>keyboard focus</u> .
<u>DI BUTTON</u>	Push Button; if the <u>DIF_SHOWAMPERSAND</u> flag is not set the element itself responds to the Alt-Letter combination.
DI CHECKBOX	Check Box; if the <u>DIF_SHOWAMPERSAND</u> flag is not set the element itself responds to the Alt-Letter combination.

	1
DI RADIOBUTTON	Radio Button; if the <u>DIF_SHOWAMPERSAND</u> flag is
	not set the element itself responds to the Alt-Letter
	combination.

xample

N_INITDIALOG

<u>ents | Dialog API</u>

The **DN_INITDIALOG** event is sent to the dialog callback function after all dialog items are initialized, but before they are displayed.

aram1

The ID of the dialog item that will initially receive the keyboard focus.

aram2

Programmer specific data that was passed to the <u>DialogEx</u> - in the Param parameter.

eturn

TRUE - if dialog items were changed. FALSE - no changes were made.

emarks

In response to this event the dialog callback function initializes each dialog element to some correct initial state. For example, it can fill a list item with elements that the user can later view...

If changes were made but the dialog callback function returned FALSE, the changes will be discarded.

xample

```
case DN_INITDIALOG:
   struct FarDialogItem DialogItem;
   Info.SendDlgMessage(hDlg,DM_GETDLGITEM,75,(LONG_PTF
   if(DialogItem.ListItems->Items[0].Flags&LIF_SELECTE
        Computer=Pl2;
   else
   ...
   return TRUE;
```

N_KEY

ents | Dialog API

The **DN_KEY** event is sent to the dialog callback function after the user presses a key in the dialog.

aram1

The ID of the dialog item receiving the event (usually it is the item that has the keyboard focus).

aram2

Internal key code.

eturn

TRUE - the key was processed internally.

FALSE - the key should be processed by the internal handler of the Dialog API kernel.

emarks

By default the **DN_KEY** event is not sent for an open <u>DI_COMBOBOX</u>. Use the <u>DM_SETCOMBOBOXEVENT</u> message to enable the sending of **DN_KEY** events.

xample

Example of processing the event:

```
// Center the dialog if the user has pressed Ctrl-App
case DN_KEY:
    if(Param2 == KEY_CTRLAPPS)
    {
        COORD c={-1,-1};
        Info.SendDlgMessage(hDlg,DM_MOVEDIALOG,TRUE,(LONG
        return TRUE;
    }
```

e also: <u>DialogEx</u>, <u>DM_KEY</u>

N_KILLFOCUS

ents | Dialog API

The **DN_KILLFOCUS** event is sent to the dialog callback function before a dialog item loses the focus.

aram1

The ID of the dialog item that loses the focus.

aram2

0

eturn

-1 - allow the dialog item to lose focus>=0 - ID of the dialog item you want to pass the focus to.

ontrols

Control	Description
All	All elements that can receive the <u>keyboard focus</u> .

emarks

xample

e also: <u>DialogEx</u>

N_LISTCHANGE

ents | <u>Dialog API</u>

The **DN_LISTCHANGE** event is sent to the dialog callback function to notify of changes that occurred to an item in the list of the <u>DI_COMBOBOX</u>, <u>DI_LISTBOX</u> or <u>DIF_HISTORY</u> types.

aram1

Dialog item ID.

aram2

Current position in the list.

eturn

TRUE - allow the changes FALSE - discard changes.

ontrols

Control	Description
DI COMBOBOX	Combined list
<u>DI LISTBOX</u>	List box
<u>DI EDIT</u>	Text input control (with the <u>DIF_HISTORY</u> flag set)
<u>DI FIXEDIT</u>	Fixed size text input control (with the <u>DIF HISTORY</u> flag set)

emarks

xample

N_LISTHOTKEY

ents | Dialog API

The **DN_LISTHOTKEY** event is sent to the dialog callback function to notify that the user used a hotkey to change to an item in the list (<u>DI_LISTBOX</u>).

aram1

Dialog item ID <u>DI_LISTBOX</u>.

aram2

Position of the selected item in the list.

eturn

FALSE - allow the change, TRUE - the plugin has processed the event by itself.

ontrols

Control	Description
DI_LISTBOX	List box
N_MOUSECLICK

<u>ents | Dialog API</u>

The **DN_MOUSECLICK** event is sent to the dialog callback function after the user clicks the mouse on one of the dialog items or outside the dialog.

aram1

Dialog item ID or -1 if the user clicked outside the dialog.

aram2

Pointer to a <u>MOUSE_EVENT_RECORD</u> structure.

eturn

TRUE - the dialog callback function has handled the message, no further processing by the kernel is needed.

FALSE - continue processing the message by the kernel.

ontrols

Control	Description
All	All dialog elements excluding hidden and disabled

emarks

- 1. For the <u>DI_USERCONTROL</u> control the mouse cursor coordinates are relative to the upper left corner of the control. For all other elements the coordinates are relative to the upper left corner of the screen.
- 2. The dialog callback function can handle the DOUBLE_CLICK event. For this you should check the

MouseEvent.dwEventFlags==DOUBLE_CLICK flag:

- 3. If the user clicks outside the dialog, then:
 - the left mouse button is treated as if **<Esc>** was pressed;
 - the right mouse button is treated as if <Enter> was pressed;
 - other mouse buttons are ignored (for a multiple button mice).
 - If the dialog callback function doesn't handle the message with ID=-1, the dialog will be closed

xample

e also: <u>DialogEx</u>

N_MOUSEEVENT

<u>ents | Dialog API</u>

The **DN_MOUSEEVENT** event is sent to the dialog callback function before a mouse message is handled by the dialog kernel.

aram1

0

aram2

Pointer to a <u>MOUSE_EVENT_RECORD</u> structure.

eturn

TRUE - enable further processing of the event by the dialog kernel. FALSE - the event has been handled by the plugin.

ontrols

Control	Description
Dialog	the event applies only to the dialog

emarks

- 1. By default, the **DN_MOUSEEVENT** event is not sent to the dialog callback function. To enable/disable sending of this event, use the <u>DM_SETMOUSEEVENTNOTIFY</u> message.
- 2. Do not enable this event unless you really need it: mouse events are sent very frequently!
- 3. A plugin may modify the fields of the <u>MOUSE_EVENT_RECORD</u> structure passed to it.
- 4. By default the **DN_MOUSEEVENT** event is not sent for an open <u>DI_COMBOBOX</u>. Use the <u>DM_SETCOMBOBOXEVENT</u> message to enable the sending of **DN_MOUSEEVENT** events.

xample

e also: <u>DialogEx</u> <u>DM SETMOUSEEVENTNOTIFY</u>

N_RESIZECONSOLE

ents | <u>Dialog API</u>

The **DN_RESIZECONSOLE** event is sent to the dialog callback function after the console window size has changed (for example, when the user presses Alt - F9).

aram1

0

aram2

Pointer to a <u>COORD</u> structure containing the current size of the console.

eturn

TRUE.

ontrols

Control	Description
Dialog	This event apllies only to the dialog

emarks

The **DN_RESIZECONSOLE** message is useful in the case when a dialog can adapt to the current size of the console. An example of such functionality is the Find files dialog (Alt-F7).

xample

Adaptation of Find File's (Alt-F7) dialog size when console size changes:

```
case DN_RESIZECONSOLE:
{
   COORD coord=(*(COORD*)Param2);
   SMALL_RECT rect;
   int IncY=coord.Y-DlgHeight-4;
   ...
   Info.SendDlgMessage(hDlg,DM_ENABLEREDRAW,FALSE,0);
```

```
Info.SendDlgMessage(hDlg,DM_GETDLGRECT,0,(LONG_PTR)
coord.X=rect.Right-rect.Left+1;
DlgHeight+=IncY;
coord.Y=DlgHeight;
if (IncY > 0)
  Info.SendDlgMessage(hDlg,DM_RESIZEDIALOG,0,(LONG_
for (I=0; I < 2; I++)
{
  Info.SendDlgMessage(hDlg,DM_GETITEMPOSITION,I,(LC
  rect.Bottom+=(short)IncY;
  Info.SendDlgMessage(hDlg,DM_SETITEMPOSITION,I,(LC
}
for (I=2; I < 10; I++)
{
  Info.SendDlgMessage(hDlg,DM_GETITEMPOSITION,I,(LC
  if (I==2)
    rect.Left=-1;
  rect.Top+=(short)IncY;
  Info.SendDlgMessage(hDlg,DM_SETITEMPOSITION,I,(LC
}
if (IncY \leq 0)
  Info.SendDlgMessage(hDlg,DM_RESIZEDIALOG,0,(LONG_
Info.SendDlgMessage(hDlg,DM_ENABLEREDRAW,TRUE,0);
. . .
return TRUE;
```

```
e also:
DialogEx
```

}

put focus

in | Dialog API | Dialog items

For keyboard messages distribution the so-called keyboard input focus conception is used. Input focus is an attribute which applies to a dialog element. If the element has the focus it means that it receives all (nearly :)) keyboard messages from FAR Manager system queue.

The Dialog Manager can move input focus from one element to another. When you press Tab, Shift-Tab, or Alt-Symbol, the input focus moves to the next/previous dialog element, or to the element that have the corresponding hot-key defined, respectively.

The dialog callback function can track getting/losing the input focus of dialog controls. When an element gets the input focus, the callback function receives the <u>DN_GOTFOCUS</u> event. When an element loses the input focus, the callback function receives the <u>DN_KILLFOCUS</u> event. In responce to the <u>DN_KILLFOCUS</u> event the callback function can disallow focus loss by the element, by returning the value -1. The <u>DN_GOTFOCUS</u> event has only an informative meaning, i.e. you can't undo/prevent this event from happening.

FAR Manager Dialog API programming interface contains two messages which allow to get or change the control having the input focus. These messages are <u>DM_GETFOCUS</u> and <u>DM_SETFOCUS</u> correspondingly.

Listed below are dialog controls which can receive keyboard input focus (provided that there're no <u>DIF_NOFOCUS</u> and/or <u>DIF_DISABLE</u> flags set for them):

Item	Description
<u>DI BUTTON</u>	Button.
DI CHECKBOX	Check Box.
DI COMBOBOX	Dropdown List (ComboBox).
<u>DI EDIT</u>	Edit Box.
<u>DI FIXEDIT</u>	Fixed-size Edit Box.
<u>DI LISTBOX</u>	List Box.
<u>DI PSWEDIT</u>	Password Input Box.
DI RADIOBUTTON	Radio Button.
DI USERCONTROL	Custom control element defined by a programmer.

e also:

DefDlgProc, DialogEx, SendDlgMessage

ervice functions - Panel

in

Function	Description
<u>Control</u>	allows to request different information about the FAR panels and perform various control actions.
<u>FreeDirList</u>	releases the memory allocated for files list by GetDirList and GetPluginDirList functions.
<u>GetDirList</u>	returns the list of files in the specified directory.
GetPluginDirList	returns list of files in the specified directory in the file system emulated by a plugin.

e also:

Exported functions, <u>Structures</u>, <u>Archive support</u>, <u>Addons</u>, <u>Win32</u> <u>structures and function</u>

anel plugin structures

in | structures

Structure	Description
InfoPanelLine	One line in the info panel
<u>KeyBarTitles</u>	Overrides function key labels in the key bar
PanelInfo	Information about a Far panel
PanelMode	Describes a panel view mode
PanelRedrawInfo	Used to redraw a panel
<u>PluginPanelItem</u>	Information about an item in the emulated file system

e also:

Exported functions, Service functions, Dialog API, Archive support, Addons, Delphi structures, Win32 structures

ddons - General purpose functions

in | addons

Function	Description
<u>InitDialogItems</u>	Translates <u>InitDialogItem</u> structure to <u>FarDialogItem</u> structure
<u>InitMenuItems</u>	Translates <u>InitMenuItem</u> structure to <u>FarMenuItem</u> structure
<u>LocMsg</u>	Returns a pointer to a string according to language settings of FAR Manager

e also:

Custom API Exported functions Structures Archive support

nitDialogItems

in | <u>Addons</u>

The **InitDialogItems** function translates an array of <u>InitDialogItem</u> structures to an array of <u>FarDialogItem</u> structures.

* Attention! The InitDialogItems function is not part of the standard API.

```
void InitDialogItems(
       const struct InitDialogItem *Init,
       struct FarDialogItem *Item,
       int ItemsNumber
)
{
  int I;
  const struct InitDialogItem *PInit=Init;
  struct FarDialogItem *PItem=Item;
  for (I=0; I < ItemsNumber; I++,PItem++,PInit++)</pre>
  {
    PItem->Type=PInit->Type;
    PItem->X1=PInit->X1;
    PItem->Y1=PInit->Y1;
    PItem->X2=PInit->X2;
    PItem->Y2=PInit->Y2;
    PItem->Focus=PInit->Focus;
    PItem->Reserved=PInit->Selected;
    PItem->Flags=PInit->Flags;
    PItem->DefaultButton=PInit->DefaultButton;
    if ((unsigned int)PInit->Data < 2000)
      strcpy(PItem->Data,GetMsg((unsigned int)PInit->D
    else
      strcpy(PItem->Data,PInit->Data);
  }
}
```

arameters

Init

Pointer to an array of InitDialogItem structures. Each array item defines one

dialog item.

Item

Pointer to an array of <u>FarDialogItem</u> structures to initialize. Each array item will define one dialog item.

ItemsNumber

Number of elements in *Init*.

eturn value

None.

emarks

Use of the InitDialogItems function is reasoned by the following: *FarDialogItem.Data* variable is large, that's why direct initalization of an array of **FarDialogItem** structures may significantly increase memory usage.

The example bellow shows how to use the InitDialogItems function and the <u>InitDialogItem</u> structure to create a dialog.

xample

TempPanel plugin configuration dialog sample:

```
if (ExitCode != 7)
    return(FALSE);
. . .
}
```

e also: Dialog | InitDialogItem | FarDialogItem

iitMenultems

in | <u>Addons</u>

The **InitMenuItems** function translates an array of <u>InitMenuItem</u> structures to an array of <u>FarMenuItem</u> structures.

• Attention! The InitMenuItems function is not part of the standard API.

```
void InitMenuItems(
         const struct InitMenuItem *Init,
         struct FarMenuItem *Item,
         int ItemsNumber)
{
  int I;
  struct FarMenuItem *PItem=Item;
  const struct InitMenuItem *PInit=Init;
  for (I=0; I < ItemsNumber; I++,PItem++,PInit++)</pre>
  {
    PItem->Selected=PInit->Selected;
    PItem->Checked=PInit->Checked;
    PItem->Separator=PInit->Separator;
    if ((unsigned int)PInit->Text < 2000)
      strcpy(PItem->Text,GetMsg((unsigned int)PInit->T
    else
      strcpy(PItem->Text,PInit->Text);
  }
}
```

arameters

Init

Pointer to an array of <u>InitMenuItem</u> structures. Each array item defines one menu item.

Item

Pointer to an array of <u>FarMenuItem</u> structures to initialize. Each array item defines one menu item.

ItemsNumber

Number of elements in *Init*.

eturn value

None.

emarks

Use of the **InitMenuItems** function is reasoned by the following: *FarMenuItem.Text* variable is large, that's why direct initalization of an array of **FarMenuItem** structures may significantly increase memory usage.

Example bellow shows how to use the InitMenuItems function and the <u>InitMenuItem</u> structure to create a menu.

e also: <u>Menu | InitMenuItem | FarMenuItem</u>

ocMsg

in | Addons

The **LocMsg** function returns a pointer to a string according to the language settings of FAR Manager. It is an analogue of the standard GetMsg function.

* Attention! The LocMsg function function is not part of the standard API. The source code is available here.

```
char *LocMsg(
  struct PluginStartupInfo *psi,
  char *MsgName,
  char *Var,
  int Len
);
```

arameters

psi

Pointer to a <u>PluginStartupInfo</u> structure; it is used to get RootKey and ModuleNumber variables.

MsqName

Name of the requested message.

Var

If not NULL, points to a buffer that receives the requested string.

Len

If zero, all the data is copied.

eturn value

Pointer to the requested string.

emarks

- 1. In contrast to the <u>GetMsg</u>, this function does not cache data. This allows to change data dynamically (without restarting FAR Manager, etc.).
- 2. The LocMsg function may be used to "localize" "second level" plugins.
- 3. It considers the language setting of the current user.

4. A file containing the messages should be located in the same folder, as the plugin, and should be named as following: **PluginName.msg**. It is a standard ini-file with the following structure:

```
[Default]
//This section defines the default language section
//if the requested section is not found.
//It is an optional section, if it is absent, fir:
//default.
Language=Russian
//Section names SHOULD be taken from the first lin
//the corresponding .lng file:
//.Language=Russian,Russian (Русский)
             \Lambda \Lambda \Lambda \Lambda \Lambda \Lambda \Lambda
//
             This should be the the name of the sec
//
[Russian]
String1=Текст первого сообщения
String2=Текст второго сообщения
IsSkipNoWord=1
[English]
String1=Text of the first message
String2=Text of the second message
IsSkipNoWord=1
//etc.
```

xample

```
BOOL IsSkipNoWord;
p=LocMsg("IsSkipNoWord",NULL,1);
IsSkipNoWord=(*p != '0')?TRUE:FALSE;
```

```
phf=strrchr(LocMsg(temp, "String1", 956), ".");
```

e also: <u>GetMsg | Language files</u>

ddons - Structures

in | Addons

Structure	Description
<u>InitDialogItem</u>	Defines the dialog item
<u>InitMenuItem</u>	Defines the menu item

e also:

Custom API Exported functions Structures Archive support

nitDialogItem

in | structures | Addons

The **InitDialogItem** structure defines one dialog item. This structure is nonstandard - it is the analogue of the <u>FarDialogItem</u> structure. The only difference is the *Data* member.

Attention! The **InitDialogItem** structure is not a part of the standard API.

```
struct InitDialogItem
{
    int Type;
    int X1;
    int Y1;
    int X2;
    int Y2;
    int Focus;
    DWORD_PTR Selected;
    unsigned int Flags;
    int DefaultButton;
    char *Data;
};
```

emarks

Use of the InitDialogItem structure instead of the <u>FarDialogItem</u> structure is reasoned by the following:

The *FarDialogItem.Data* variable is large, that's why direct initalization of an array of **FarDialogItem** structures may significantly increase memory usage. You can use the <u>InitDialogItems</u> function for translation of InitDialogItem structures to FarDialogItem structures .

e also:

Structures | FarDialogItem | TFarDialogItem | InitDialogItems

nitMenultem

in | structures | Addons

The **InitMenuItem** structure defines one menu item. This structure is nonstandard - it is the analogue of the <u>FarMenuItem</u> structure. The only difference is the *Text* member.

* Attention! The InitMenuItem structure is not a part of the standard API.

```
struct InitMenuItem
{
    char* Text;
    int Selected;
    int Checked;
    int Separator;
};
```

emarks

Use of the InitMenuItem structure instead of the <u>FarMenuItem</u> structure is reasoned by the following:

FarMenuItem.Text variable is large, that's why direct initialization of an array of **FarMenuItem** structures may significantly increase memory usage.

You can use the <u>InitMenuItems</u> function for translation of InitMenuItem structures to FarMenuItem structures.

e also: <u>Structures | FarMenuItem | TFarMenuItem</u>

MT-modules exported functions

in | Custom API | archive support

🗹 Note

ote This information is valid only for MultiArc plugin which comes with the standard distribution of FAR Manager!

Function	Description
<u>CloseArchive</u>	Close an archive.
<u>GetArcItem</u>	Get the information about the next archive element.
<u>GetDefaultCommands</u>	Get archiver command list used by default
<u>GetFormatName</u>	Get archive format name.
<u>GetSFXPos</u>	Get archive beginning position.
<u>IsArchive</u>	Check whether the specified file is an archive.
LoadFormatModule	Called when a sublugin is being loaded.
<u>OpenArchive</u>	Open an archive and prepare it for reading.
<u>SetFarInfo</u>	Pass global settings to a plugin.

e also:

Exported functions, Service functions, structures, Addons, custom.ini file format

oadFormatModule

```
in | Custom API | archive support
```

It's called when a subplugin is finished loading. This function is optional and can be omitted.

```
DWORD WINAPI LoadFormatModule(
    const char *ModuleName
);
```

arameters

ModuleName

subplugin module name (FMT file full path name)

eturn value

Must be 0. May be used in the future for returning information about a subplugin.

emarks

This function is called only once when MultiArc loads a subplugin (*see MultiArc sources* - ARCPLG.CPP file, ArcPlugins::LoadFmtModules function).

Archive

in | Custom API | archive support

Check whether the specified file is an archive.

```
BOOL WINAPI IsArchive(
   const char *Name,
   const unsigned char *Data,
   int DataSize
);
```

arameters

Name

Archive name

Data

archive file data (pointer to data of DataSize size relative to the beginning of the file)

DataSize

the size of data passed to the function

eturn value

TRUE if the archive type is supported by plugin.

xample

An example of ZIP archive definition (ZIP.CPP)



etSFXPos

in | Custom API | archive support

Returns the position of the beginning of the archive in the data stream passed to the function.

DWORD WINAPI GetSFXPos(void);

arameters

None.

eturn value

Position of the beginning of the archive relative to the beginning of file.

emarks

GetSFXPos function is called right after the successful archive recognition by the <u>IsArchive</u> function. MultiArc selects the FMT-module which returned the least value (closer to the beginning of the file). This fixes the issue when, for example, there's an ARJ archive placed without compression inside the ZIP archive with the name, say, "N3gk8v1t.106". Previous versions of MultiArc would show the contents of the ARJ archive instead of the ZIP one.

penArchive

```
in | Custom API | archive support
```

Opens an archive and prepares it for reading. Called after a successful execution of the <u>IsArchive</u> function.

```
BOOL WINAPI OpenArchive(
   const char *Name,
   int *Type
);
```

arameters

Name

archive name

Туре

if the plugin supports several archive types then set it to the archive type else set it to 0.

eturn value

TRUE if success, FALSE in case of error.

emarks

This function should be used for archive file opening, getting the necessary information about an archive and all that is needed for successfull filling of FAR's virtual file system.

etArcItem

in | Custom API | archive support

Get the information about archive's next element. Executed in a cycle after the <u>OpenArchive</u> function call while the module returns **GETARC_SUCCESS** value.

```
int WINAPI GetArcItem(
   struct PluginPanelItem *Item,
   struct ArcItemInfo *Info
);
```

arameters

Item

This structure must be initialized (see <u>PluginPanelItem</u> for detailed info about structure fields)

Info

Additional item information, which should be filled if possible (see <u>ArcItemInfo</u>).

eturn value

Return value	Description
GETARC_EOF	End of archive reached.
GETARC_SUCCESS	Item received successfully.
GETARC_BROKEN	Broken (damaged) archive
GETARC_UNEXPEOF	Unexpected end of archive
GETARC_READERROR	Read error

e also:

archive support

loseArchive

in | Custom API | archive support

Close the archive. Launched after the last call of the <u>GetArcItem</u> function.

```
BOOL WINAPI CloseArchive(
   struct ArcInfo *Info
);
```

arameters

Info

Extended archive information which should be filled if there's an opportunity (see <u>ArcInfo</u>).

eturn value

TRUE if the function succeeds.

e also:

archive support, ArcInfo, GetArcItem

etFormatName

in | Custom API | archive support

Get archive format name.

```
BOOL WINAPI GetFormatName(
    int Type,
    char *FormatName,
    char *DefaultExt
);
```

arameters

Туре

archive type

FormatName

format name; it's used for saving parameters in the registry and for selecting the desired archive format.

DefaultExt

default file extension for the format (without '.'); used for accelerating format recognition.

eturn value

TRUE in case of success. FALSE when the specified archive type is not supported.

xample

An example getting ZIP-file format name (ZIP.CPP)

```
BOOL WINAPI _export GetFormatName(int Type, char *Form
{
    if (Type==0)
    {
        strcpy(FormatName, "ZIP");
        strcpy(DefaultExt, "zip");
        return(TRUE);
    }
    return(FALSE);
```

e also: archive support

}

etDefaultCommands

in | Custom API | archive support

Get archiver command strings used by default

```
BOOL WINAPI GetDefaultCommands(
    int Type,
    int Command,
    char *Dest
);
```

arameters

Туре

Archive type

Command

Archiver command number:

Number	Description
0	extract
1	extract without path
2	test
3	delete
4	comment archive
5	comment files
6	convert to SFX
7	lock
8	add recovery record
9	recover
10	add files
11	move files
12	add files and folders
13	move files and folders
14	mask to select all files

This is how it looks like in MultiArc plugin settings:

Etract	rar x {-p2%P} -y -kb %%A @%%LNM
Extract without aths	rar e $-av = \langle -pxxP \rangle -y -c = \langle xxS \rangle xxA @xxLNM$
est	rar t -y {-pxxP} xxa @xxLNM
lete	rar d -y (-wxxW) xxA @xxLNM
Comment archive	rar c $-y \left\{-wxxW\right\} xxA$
Comment Jiles	rar of -y {-wxxW} xxA @xxLNM
Convert to FX	rar s -y %%A
lock archive	rar k -y XXA
Pr_tect archive	rar rr -y %%A
ecover archive	rar r -y 22A
dd files	rar a -y {-pxxP} {-apxxR} {-wxxW} xxA @xxL
ove files	rar m -y {-pxxP} {-apxxR} {-wxxW} xxA @xxL
Add files and folders	rar a -r0 -y (-pxxP) (-apxxR) (-wxxW) xxA
Mo e files and folders	rar m -r0 -y (-pxxP) (-apxxR) (-wxxW) xxA
"All files" mas	*.*
Archive extensio	rar
[Ok	1 [Cancel 1 [Reset 1

Dest

Buffer for copying commands. MultiArc expects string of no more than 512 bytes.

eturn value

TRUE in case of success. FALSE if the specified archive type is unsupported. If the type is supported but the required command is missing return TRUE and set **Dest** to an empty string.

etFarInfo

in | Custom API | archive support

Called when a subplugin module is loaded and MultiArc passes a <u>PluginStartupInfo</u> structure pointer to the plugin. This function is optional and can be omitted.

```
void WINAPI SetFarInfo(
  const struct PluginStartupInfo *Info
);
```

arameters

Info

a pointer to a **<u>PluginStartupInfo</u>** structure

eturn value

None.

emarks

- 1. Function is called only once immediately after <u>LoadFormatModule</u> function call, after which a subplugin knows about FAR Manager as much as MultiArc plugin itself (plus some additional info from MultiArc).
- 2. **Info** pointer is valid only within the function scope (only inside the function) so the structure must be copied to subplugin's internal variable for further use:

```
static struct PluginStartupInfo Info;
...
void WINAPI _export SetStartupInfo(const struct Pi
{
    ::Info=*Info;
    ...
}
```

3. If there're "standard functions" (from <u>FarStandardFunctions</u> structure) used in the plugin then <u>PluginStartupInfo.FSF</u> member must be saved to subplugin's local scope as well:

```
static struct PluginStartupInfo Info;
```

```
static struct FarStandardFunctions FSF;
void _export SetStartupInfo(struct PluginStartup:
{
    Info=*psInfo;
    FSF=*psInfo->FSF;
    Info.FSF=&FSF; // correct the address in the I
    ...
}
```

e also: archive support, PluginStartupInfo

tructures used by FMT-modules

in | Custom API | archive support

🗹 Note

This information is valid only for MultiArc plugin which comes with the standard distribution of FAR Manager!

Structure	Description
ArcInfo	Common information about an archive
ArcItemInfo	Information about a specific archive element

e also:

Exported functions, Service functions, structures, Addons, custom.ini file format

rciteminfo

```
in | Custom API | archive support
```

This structure is passed to the <u>GetArcItem</u> function (all fields are already initialized by 0's).

```
struct ArcItemInfo
{
    char HostOS[32];
    char Description[256];
    int Solid;
    int Comment;
    int Encrypted;
    int DictSize;
    int UnpVer;
    int Chapter;
};
```

embers

HostOS

Host operating system or empty string if it's unknown

Description

Element description or an empty string

Solid

Flag that indicates that the archive is solid.

Comment

Set this flag if there's a comment in the archive

Encrypted

Set this flag if the archive is encrypted

DictSize

Dictionary size or 0 if it's unknown

UnpVer

Archiver version necessary for unpacking (HighNumber*256+LowNumber) or 0 if it's unknown
Chapter

archive chapter where the file belongs.

e also:

archive support structures

rcinfo

in | Custom API | archive support

This structure is passed to the <u>CloseArchive</u> function.

```
struct ArcInfo
{
    int SFXSize;
    int Volume;
    int Comment;
    int Recovery;
    int Lock;
    DWORD Flags;
    DWORD Reserved;
    int Chapters;
};
```

embers

SFXSize

SFX module size

Volume

Volume flag

Comment

Archive comment is present

Recovery

Recovery record is present

Lock

Archive is locked

Flags

Additional archive information flags. The *Flags* field can be a combination of the following values:

Structure	Description
AF_AVPRESENT	There's a digital signature (archive authenticity information).

AF_IGNOREERRORS	Archive command exit codes are ignored for this archive.
AF_HDRENCRYPTED	Archive header is encrypted.

Reserved

reserved for future use.

Chapters

Number of chapters in an archive.

emarks

ArcInfo structure, initialized by 0's, is passed to the <u>CloseArchive</u> function.

e also: archive support, structures

rchivers' command line arguments

in | Custom API | archive support

The following variables can be used in external archivers' command-lines for different archive handling operations:

Variables	Description			
%%A	Archive name.			
%%a	Archive short name.			
%%W	Folder for temporary files, including trailing slash symbol.			
%%P	Password.			
%%R	Current folder in the archive.			
%%L	List of file names (file list). It's a file containing file names to be processed by the external archiver.			
%%l	List of short file names.			
%%F	Name(s) of the file(s) to process. If all the file names don't fit in the command line then the archiver will be launched several times until all file names are processed. By default the maximum command line length is 127 symbols, but one can define it explicitly right after %%F (i.e. %%F512). The %%F variable should be used only in case the archiver doesn't support file lists.			
%%f	Name of a single file to process. Archiver will be launched several times until all the file names are processed. This variable should be used only in case the archiver doesn't support file lists or several file names in its command line.			
%%E «exit code»	Maximum allowable exit code. For example, %%E2 shows that exit codes of 0, 1 and 2 are allowed. The variable can be used at any place in the command line. If it's missing the only available exit code is 0.			
%%S	Place for additional command line keys defined in the second line of the "Add" dialog. When the variable is missing keys are added to the end of the command line.			

Modifiers for the variables Q, S, M, N, *, W can be used for more accurate definitions of file name formats.

Modifier	Description
----------	-------------

'Q'	enclose file names with spaces in quotes;				
'q'	enclose all file names in quotes;				
'S'	use '/' instead of '\' in full paths;				
'M'	pass folder names to archivers as 'name*.*';				
'N'	pass folder names to archivers as 'name';				
1*1	use '*' mask to select all files. This modifier overrides "All files Mask" parameter for any single command of the archiver;				
'W'	use name without path;				
'P'	use path without name;				
'A'	use ANSI encoding.				

These modifiers can be used right after the %%F, %%f, %%L and %%l variables without spaces. 'P' and 'A' can also be specified after %%A and %%a. It's allowed to use several modifiers at once, for example, %%LMN or %%FQSN128. If there're no 'M' or 'N' modifiers the archiver receives both 'name' and 'name*.*'.

If some part of the command line is enclosed in curly braces {...} it will be added to the resulting command line only in case it contains at least one not empty variable described above.

If an external archiver doesn't support some command simply leave the corresponding line empty.

e also:

archive support, GetDefaultCommands

ustom.ini file format

in | Custom API | archive support

In the standard distribution of MultiArc there's a limited number of FMTmodules supporting most commonly used archivers. Among them there's the CUSTOM.FMT module which allows to define "missing" archivers by defining the archiver's output listing format. For these descriptions the CUSTOM.INI file is used.

Each new archive type is defined within individual **[Archive Name]** sections used for archive type name. The following parameters are used in this section:

TypeName (optional parameter)

overrides format name defined in the section name (left for compatibility reasons)

ID (optional parameter)

archive identifier - a sequence of space delimited two-digit hex numbers (i.e. the "ID=1a 4a 61 72 1b" sequence defines JAR archive identifier)

IDOnly (optional parameter)

"archive identifier only". If it's set to "1" file extension is ignored and archive type is determined by the ID field only. If it's "0" then both file extension and *ID* field (if it's defined in this section) are used to determine archive type.

IDPos (optional parameter)

position from which CUSTOM.FMT begins to look for the archive identifier (*ID*). If the field is not defined then the ID is searched from the beginning of the file. By default FAR "gives" to the plugin the first 128K of data.

Extension

```
archive extension (without '.').
```

List

external archiver command for file listing. Archive name is added after this string. For example, for List="jar32 v -y" field the jar32 v -y archive_name command will be composed.

Errorlevel (optional parameter)

listing analysis is performed only if archiver's exit code is less than the one defined in the **Errorlevel** parameter.

IgnoreErrors (optional parameter)

if it's not 0 then archiver's exit code is ignored. Use with caution!

Start (optional parameter)

substring which defines the beginning of the listing analysis. If the string begins with the '^' symbol then the search for substring is performed starting from the second symbol of the current line in the listing.

End (optional parameter)

substring which defines the end of the listing analysis. If the string begins with the '^' symbol then search for substring is performed starting from the second symbol the current line in the listing.

Format0 ... FormatN

parameters describing the record for a single archive element. Each parameter describes an ordinary record string, allowing the following codes:

Code	Description
n	file name
	remove trailing spaces from file name and append name with dot
Z	file size
с	file description
р	packed file size
d	day
t	month
TTT	three letters of month name (Jan, Feb, Mar,)
у	year
h	hours
Н	hours modifier letter (a - a.m. time, p - p.m. time)
m	minutes
s	seconds
a	file attributes
*	skip until first space or end of line
b	day of last access
v	month of last access
e	year of last access

Х	hours of last access
1	minutes of last access
k	seconds of last access
j	day of creation
g	month of creation
f	year of creation
0	hours of creation
i	minutes of creation
u	seconds of creation
r	CRC (hex)
С	Chapter descriptions field
(<number>)</number>	optional digits, some or all digits may be missing

If the string begins with the '^' symbol then search for substring is performed starting from the second symbol of the current line in the listing.

IgnoreString0 ... IgnoreStringN

parameters defining strings which are ignored by the module in the listing analysis. If the string begins with the '^' symbol then search for substring is performed starting from the second symbol of current line in the listing.

Extract ... MoveRecurse

parameters describing archiver commands.

Command	Description			
Extract	unpack			
ExtractWithoutPath	unpack without paths			
Test	test archive			
Delete	delete files from archive			
Comment	add archive comment			
CommentFiles	add files comment			
SFX	convert to SFX			
Lock	lock archive			
Protect	add recovery record			
Recover	recover damaged archive			
Add	add files			

Move	move files to archive		
AddRecurse	add files with recursive folder tree traversal		
MoveRecurse	move files with recursive folder tree traversal		

Pseudo-variables described <u>here</u> can be used in external archivers' command lines for different archive processing purposes.

AllFilesMask

"All Files" mask

egular expressions

Regular expressions (Perl5 syntax) can be used in the following parameters: *FormatX, IgnoreStringX, Start, End*. To specify that a pattern should be matched as a regular expression, enclose it in slashes ('/'). The following regexp flags can be used (if corresponding characters are placed after closing slash):

Flag	Description			
i	ignore case			
x	extended (use explaining spaces)			

The following open source library is used to process regular expressions: <u>PCRE</u> (by Philip Hazel, copyright by the University of Cambridge, England).

When using regexps in the *FormatX* parameter, named strings (of the form (? P<name>\w+)) are used to extract information from listing lines. The following field names are available:

Field	Description			
name	file name with relative path (leading and trailing spaces are trimmed)			
description	file description			
size	unpacked file size in bytes (non-digit characters are ignored in this and all other numeric fields)			
packedSize	packed file size in bytes			
attr	file attributes (A - archive, R - readonly, S - system, C - compressed, H - hidden, D - directory)			
cYear, cDay, cMonth, cHour, cMin, cSec	creation time			
mYear, mDay, mMonth, mHour, mMin, mSec	modification time			
mAMPM, mMonthA	extended time fields for modification time			
aYear, aDay, aMonth, aHour, aMin, aSec	access time			
CRC	32 bit file Cyclic Redundancy Checksum (CRC)			

For example to process the following listing:

Volume in drive D is work Volume Serial Number is xxxx-xxxx Directory of D:\works\MultiArc\final\Formats 15.11.2004 07:15<DIR> 15.11.2004 07:15 <DIR> 14.11.2004 5 632 Ace.fmt 00:46 14.11.2004 4 608 Arc.fmt 00:46 6 656 Arj.fmt 14.11.2004 00:46 14.11.2004 6 144 Cab.fmt 00:46 56 832 Custom.fmt 15.11.2004 06:07 15.11.2004 07:14 1 551 custom.ini-re 15.11.2004 07:11 54 344 Formats.jkr 4 608 Ha.fmt 14.11.2004 00:46 14.11.2004 6 144 Lzh.fmt 00:46 15.11.2004 0 out 07:1514.11.2004 00:46 6 144 Rar.fmt 9 216 TarGz.fmt 14.11.2004 00:46 6 656 Zip.fmt 14.11.2004 00:46 13 File(s) 168 535 bytes 2 Dir(s) 37 840 883 712 bytes free

the following settings are used:

```
Start="/^ Directory of/"
End="/File\(s\)/"
Format0="/^(?P<cDay>\d+).(?P<cMonth>\d+).(?P<cYear>\c
```

The following settings:

Start="/^..reading directory/"
End="/^.listing completed/"
Format0="/^[^\|]*\|(?P<size>[^\\]+)\|(?P<attr>[^\\]+)
IgnoreString0="/^----

can be used to process this listing:

jkRes utility version 002

pack/unpack tool to work with jkRes resource bundle ..listing files from resource bundle .. reading directory...12 items found _ _ _ zlib | 5632 Ace.fmt | A.... zlib | Arc.fmt 4608 Ι Α.... Arj.fmt zlib | 6656 Ι Α.... zlib Cab.fmt 6144 | A.... zlib | Custom.fmt 56832 | A.... zlib 4608 | A.... Ha.fmt zlib | Lzh.fmt 6144 | A.... Rar.fmt zlib | 6144 | A.... zlib TarGz.fmt 9216 | A.... Zip.fmt zlib 6656 Α.... zlib 13012 Custom.ini Α.... zlib custom.ini-re 1072 Α.... _ _ _ _ _ _ _ _ ..listing completed ..done

e also: archive support

RegExp

in | <u>Custom API</u> | <u>regexp</u>

A regular expression object initialization and destruction.

Class methods:

```
CRegExp();
CRegExp(char *Text);
~CRegExp();
```

DLL interface:

```
PRegExp WINAPI reCreate();
PRegExp WINAPI reCreateCompile(char *text);
BOOL WINAPI reDestroy(PRegExp re);
```

embers

re

Pointer to an object to destroy.

Text

pointer to the string that contains the regular expression to compile.

eturn value

The reCreate and reCreateCompile functions return a pointer to the created object, which will be used in every function call, or NULL in case of failure. reDestroy returns the result of regular expression object destruction.

emarks

Constructor with an initialization parameter compiles the expression. Result is available through <u>isok</u> method or through <u>geterror</u> method.

e also:

<u>isok</u>

RegExp.isok

in | Custom API | regexp

Returns the result of the last compilation.

Class method:

bool isok();

DLL interface:

```
BOOL WINAPI relsok(PRegExp re);
```

embers

re

a pointer to an object

eturn value

Returns TRUE or FALSE depending on success or failure of the last compilation.

emarks

It's possible to use <u>geterror</u> extended method to get more detailed information about an error.

e also:

<u>geterror</u> <u>CRegExp</u>

RegExp.geterror

in | Custom API | regexp

Extended information about a compilation error.

Class method:

```
EError geterror();
```

DLL interface:

```
EError WINAPI reGetError(PRegExp re);
```

embers

re

a pointer to an object

eturn value

EError enumeration type - error type:

```
enum EError
{
   EOK = 0, EERROR, ESYNTAX, EBRACKETS, EOP
};
```

all goes without saying.

e also:

RegExp.SetNoMoves

in | <u>Custom API</u> | regexp

Allows/disallows moving inside the target string.

Class method:

bool SetNoMoves(bool Moves);

DLL interface:

BOOL WINAPI reSetNoMoves(PRegExp re, BOOL Moves);

embers

re

a pointer to an object

Moves

If TRUE, the scanning function doesn't move inside the target string. Otherwise an attempt is taken to find the match for every position in the string.

eturn value

Returns success/failure of setting a parameter.

emarks

With movement along the target string enabled in fact there's a scanning function call for every position in the string from the first through the last, and for every position there's an attempt to find a match. If you don't need this feature, don't forget to call SetNoMoves(TRUE);

e also: <u>CRegExp</u>

RegExp.SetBkTrace

in | Custom API | regexp

Set references for \N operator

Class method:

bool SetBkTrace(char *Str, PMatches Trace);

DLL interface:

BOOL WINAPI reSetBkTrace(PRegExp re, char *str, PMatch

embers

re

a pointer to an object

Str

string for setting a relation

trace

set of bracket matches for the previous regexp

eturn value

Returns TRUE or FALSE depending of operation success/failure.

emarks

This function is used for setting a relation between different regexps (it's used in the colorer plugin, for example) - by means of \N operator, where N - bracket number in the first regexp for reference.

e also: <u>CRegExp</u>

RegExp.SetExpr

<u>in | Custom API | regexp</u>

Sets and compiles a regexp.

Class method:

bool SetExpr(char *Expr);

DLL interface:

```
BOOL WINAPI reSetExpr(PRegExp re, char *expr)
```

embers

re

a pointer to an object

expr

String with a regexp written by following colorer rules. If you perform the search in non-OEM codepage then you should first pass the string converted to your codepage to this method, and call the <u>SetCodePage</u> function with the address of the transliteration table from your codepage to OEM.

eturn value

Returns TRUE or FALSE depending on operation success/failure.

emarks

It's possible to use extended <u>geterror</u> method for getting more detailed information about an error.

e also: SetCodePage geterror CRegExp

RegExp.SetCodePage

in | <u>Custom API</u> | <u>regexp</u>

Sets codepage for working with texts in non-OEM codepage.

Class method:

void SetCodePage(char *Table);

DLL interface:

void WINAPI reSetCodePage(PRegExp re, char *Table);

embers

re

a pointer to an object

Table

Table for transliteration from the target code page to OEM

eturn value

None

emarks

For regexps to be full-fledged in another codepage it's necessary to set the regexp itself to the target codepage. You can recompile the class so that is uses non-OEM codepage by default. In this case setcodepage method will also work but the transliteration tables would have to be set relative to the codepage the cRegExp class has been compiled with.

e also: <u>CRegExp</u>

RegExp.Parse

in | <u>Custom API</u> | regexp

Parses the compiled regexp against a target string.

Class method:

bool Parse(char *Str, PMatches Mtch);

DLL interface:

BOOL WINAPI reParse(PRegExp re, char *str, PMatches mt

embers

re

a pointer to an object

str

a target string to process

mtch

Pointer to a structure for saving bracket matches

eturn value

Returns TRUE or FALSE depending on parsing success/failure.

emarks

When one uses this method some things go by default. The beginning of the string corresponds to the real begginning, and the end of the string corresponds to... er... the real end. If you want to use the extended capabilities - use a more feature-full <u>Parse</u> variant.

e also: <u>Parse</u> <u>CRegExp</u>

RegExp.Parse

in | <u>Custom API</u> | <u>regexp</u>

Parses the compiled regexp against a target string using extended parameters.

Class method:

bool Parse(char *Str, char *Sol, char *Eol, PMatches M

DLL interface:

```
BOOL WINAPI reParseParam(PRegExp re, char *str, char *
```

embers

re

a pointer to an object

str

a target string to process

Sol

the beginning of a string

Eol

the end of a string

mtch

Pointer to a structure for saving bracket matches

moves

-1 by default, it means NoMoves state isn't changing. It's possible to set another value - see <u>SetNoMoves</u> function.

eturn value

Returns TRUE or FALSE depending on parsing success/failure.

emarks

If you see it more convenient to leave most parameters in their default values then use the simplified <u>Parse</u> variant.

e also: <u>SetNoMoves</u> <u>Parse</u> <u>CRegExp</u>

Matches

in | <u>Custom API</u> | <u>regexp</u>

Returns the last compilation result.

```
#define MATCHESNUM 0x10
typedef struct SMatches
{
    int s[MATCHESNUM];
    int e[MATCHESNUM];
    int CurMatch;
} *PMatches;
```

embers

S

array of matched opening bracket offsets

е

array of matched closing bracket offsets

CurMatch

Total number of brackets in the regexp

emarks

If there's no match for a bracket (it was skipped) then both s and e are -1. Mind that CurMatch is not necessary equals to the number of brackets matched - the latter can occur in any order.

e also: CRegExp

reas of execution

in | Macros | General background

FAR can create independent macro commands (macros with the same shortcut keystroke) for different areas of execution.

ATTENTION! Area to which the new macro command will be assigned is defined by the place where its recording **had been started**.

Current version supports the following independent areas of execution:

- file panels;
- internal file viewer;
- internal file editor:
- dialog boxes;
- quick file search;
- drive selection menu;
- main menu;
- other menus:
- help system;
- informational panel;
- quick view panel;
- folder tree panel;
- folder search panel;
- user menu;
- screen capturing mode.

It is not allowed to create different macro commands with the same name (keyboard shortcut) in the same area of execution (there's some work-around though using the <u>primitive macro-language</u>). Existing macro command will be overwritten in attempt to create the macro command with the same keyboard shortcut.

Thus each area of execution can contain only one macro command with the same keyboard shortcut.

There's also the **Common** area of execution. Macro command assigned to this area can be executed in every other area of execution but this area has the lower priority then native ones so if some area already contains macro with the same shortcut then common macro will be overridden. Macro commands cannot be

<u>created</u> interactively in the common area, but can be <u>deleted</u>.

ecording a macro

in | Macros | Using macros

Macro command should be created using the following actions:

1. Start recording a macro. Hotkey Ctrl-<.> (Ctrl and 'dot') to record a macro in general mode and Ctrl-Shift-<.> (Ctrl, Shift and 'dot'), to record a macro in special mode.

In the top left corner of the screen you should see **R** symbol indicating that FAR Manager has began to record a new macro.

2. Macro command text.

FAR records every key the user presses with the following exceptions: the only keystrokes will be stored that are processed by FAR Manager. I.e. if any application is launched from FAR Manager console than FAR will save only the keystrokes made before the launching and after closing this application.

3. Stop recording a macro.

Since the parameters of the macro command can be additionally adjusted there're two different commands that the user can use to complete the recording of a macro: Ctrl-<.> (Ctrl and 'dot' key) and Ctrl-Shift-<.> (Ctrl, Shift and 'dot' key). First can be used to complete the macro with default execution parameters, the second one brings up the <u>'execution</u> <u>parameters</u>' dialog box upon completion the recording of the macro.

4. Assigning the shortcut key combination.

Shortcut assignment dialog box will appear immediately after completion of a recording of a macro. You can use it to assign a specific shortcut key combination that can be used to call the macro execution.

■ **Notes:** All existing macro commands will be deactivated while the user attempts to record a new macro command. Thus it is impossible to create a 'nested' macro command that could use existing macro commands in its command sequence.

eleting a macro

in | Macros | Using macros

To delete a previously recorded macro you should record an empty (with no actions) macro and assign it the same shortcut key combination as in the existing one that you would like to remove.

You can do it as following:

- 1. Start recording a macro (Ctrl-<.>)
- 2. Stop recording a macro (Ctrl-<.>)
- 3. Enter or select in the dropdown box the shortcut key combination value of the macro command you want to remove.

▲ **ATTENTION!** Shortcut key combination will perform its original functions if any as soon as you delete the macro command. I.e. if some macro command overrides some function performed by a plugin or FAR manager then you will return this function back to a plugin module or FAR manager by deleting this macro.

ssigning macro commands

in | Macros | Using macros

Macro command can be assigned to:

- 1. any key;
- 2. any key combination with Ctrl, Alt and Shift modifiers;
- 3. any key combination with two modifiers. FAR is capable to recognize the following double modification keys: Ctrl-Shift-<key>, Ctrl-Alt-<key> and Alt-Shift-<key>

Macro command **cannot** be assigned to the following key combinations: Alt-Ins, Ctrl-<.>, Ctrl-Shift-<.>, Ctrl-Alt, Ctrl-Shift, Shift-Alt, Shift-<character>, Ctrl-Alt-Shift.

Macro command **cannot** be assigned to the following system key combinations: Alt-Space, Alt-Shift-Space, Alt-Enter, Alt-Shift-Enter.

There are several key combinations (particularly Enter, Esc, F1, Ctrl-F5, Ctrl-Down, MsWheelUp and MsWheelDown with all Ctrl, Shift and Alt combinations) that cannot be entered in shortcut assignment dialog box directly because these combinations perform some special functions in FAR. These key combinations should be selected using the drop-down selection box instead.

e also: <u>Record a macro, Delete a macro</u>

lacro command parameters

in | Macros | Using macros

Macro command parameters define the conditions and the way the macro command is going to be executed.

To set up the additional parameters for the macro command created complete the recording using Ctrl-Shift-. instead of Ctrl-. and choose its parameter values using the dialog box:

Allow screen output while executing the macro

Setting this parameter to 0 will disable screen refreshing operations for the time necessary to complete the macro command execution. All changes made to screen content will be complete after the command execution.

Execute after FAR Manager startup

Allows to execute the macro command immediately after FAR Manager startup.

The following conditions are possible for the active and inactive panels activating the macro command:

Panel type

```
[x] - execute if the panel is used by a plugin
[ ] - execute if the panel is used by FAR file list
[?] - ignore the panel type
```

Execute for the folders

[X]	-	execute if	there's	а	fold	er und	er t	he d	curso	or d	or
[]	-	execute if	there's	а	file	under	the	cu	rsor	on	t
[?]	-	disregard	the type	01	f an	elemen	t un	der	the	cui	٢S

Selection is present

[X]	-	execute	if	there	are	fil	Les	or	fol	Lders	sel	Lected
[]	-	execute	if	there	are	no	fil	Les	or	folde	ers	select

[?] - ignore the file or folder selection state

Miscellaneous activation conditions:

Command line is empty

[x] - execute only if the command line is empty

[] - execute only if the command line is not empty

[?] - ignore the command line state

Text block is selected

```
[x] - execute only if there's a selected text block i
```

```
[] - execute if there's no selected text block
```

```
[?] - ignore the block selection state
```

Notes: * FAR Manager checks all above conditions before executing the macro command.

* There are several key combinations (particularly Enter, Esc, F1, Ctrl-F5, Ctrl-Down, MsWheelUp and MsWheelDown with all Ctrl, Shift and Alt combinations) that cannot be entered in shortcut assignment dialog box directly because these combinations perform some special functions in FAR. To assign one of these key comdinations select it from the drop-down selection box instead.

e also:

Record a macro, Delete a macro

lacro-commands

in | Macros | Macro-language

acro-commands that can be used in the macro:

Macro-command	Description		
<pre>\$Date ["format"]</pre>	Insert the current date/time. See " <u>\$Date</u> " for details.		
\$Text "string"	Insert arbitrary text. See " <u>\$Text</u> " for details.		
\$IClip	Working with the clipboard. See " <u>\$IClip</u> " for details.		
\$MMode 1	Changes the screen refresh mode during the macro command execution. See " <u>\$MMode</u> " for details.		
\$Exit	Stops macro command execution. See " <u>\$Exit</u> " for details.		
\$XLat	Invokes the text transliteration function. See " <u>\$XLat</u> " for details.		
\$KbdSwitch	Cyclic switching of the keyboard layouts. See " <u>\$KbdSwitch</u> " for details.		
\$If (Expr) TruePart [\$Else FalsePart] \$End	Implements a conditional execution operator. See " <u>\$If</u> " for details.		
\$While (Expr) Sequence \$End	Implements an iterative process - "Execute the Sequence while condition Expr is true". See " <u>\$While</u> " for details.		
\$Rep (Expr) Sequence \$End	Implements the iterative process - "Execute the Sequence Expr number of times". See " <u>\$Rep</u> " for details.		
\$AKey	The hotkey that was used to call the macro. See " <u>\$AKey</u> " for details.		
\$SelWord	Select the word under the cursor. See " <u>\$SelWord</u> " for details.		
%var=value; %%var=value;	Using variables. See " <u>variables</u> " for details.		

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only

by editing the registry manually or by using special applications or FAR plugins.

AKey

<u>in | Macros | Macro-language | Macro-commands</u>

The **\$AKey** macro-command inserts the name of the hotkey that was used to call the current macro.

\$AKey

rguments

None.

emarks

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

xample

A macro for switching between screens. 10 similar AltNumber macros should be entered in the Common area:

```
$If ( Dialog || MainMenu || Menu || UserMenu || Searc
    $AKey
$Else
    %s = AKey ( 1 );
    %c = substr ( %s, len ( %s ) - 1, 1 );
    F12
    $If ( CheckHotkey ( %c ) )
        eval ( %c )
    $Else
        Esc
        $AKey
    $End
$End
```

e also: akey, Examples

Date

in | Macros | Macro-language | Macro-commands

The **\$Date** macro-command inserts the current date/time into the current object editor.

\$Date ["format"]

rguments

format

The current date/time is converted to a string according to the "format" parameter. Its meaning is similar to the Unix 'date' function formatting parameter. **\$Date** supports the following format specification (similar to the format used by the strftime function):

Format	Description		
%%	Percent sign (%)		
%a	Abbreviated day of week name (Sun)		
%A	Full day of week name (Sunday)		
%b	Abbreviated month name (Jan)		
%B	Full month name (January)		
%с	Date and time representation in the format: WDay Mnt Day HH:MM:SS yyyy (Mon Jan 10 04:11:54 2000)		
%C	Century as a decimal number (00 - 99). 1992 => 19		
%d	Day of the month as decimal number (01 - 31)		
%D	Same as %x		
%е	Similar to %d, but the leading zero is replaced with a space.		
%h	Same as %b		
%Н	Hour in 24-hour format (00 - 23)		
%I	Hour in 12-hour format (01 - 12)		
%ј	Day of year as decimal number (001 - 366)		
%k	Similar to %H, but the leading zero is replaced with a space.		
%l	Similar to %I, but the leading zero is replaced with a space		
%L	Use the localized set of names of months and days rather then English. FAR Manager requests two sets of names from the system during startup:		

	1) English names of months and weekdays			
	(LOCALE_USER_DEFAULT).			
	By default the English names are used.			
%m	Month as decimal number (1 - 12)			
%m0	Similar to %m, but with a leading zero (01 - 12)			
%mh	Current month as hexadecimal number (1 - C)			
%M	Minute as decimal number (00 - 59)			
%n	Line break `\\n' ATTENTION! Using this specifier in the editor macros will switch the auto indentation off.			
%р	AM or PM			
%r	Same as %I:%M:%S %p			
%R	Same as %H:%M			
%S	Seconds as decimal number (00 - 59)			
%t	Tab character `\\t'			
%Т	Same as %X			
%U	Week of year as decimal number, with Sunday as first day of week (00 - 53)			
%v	Date in the format dd-mmm-yyyy (mmm - the month name is in upper case)			
%V	Week of year according to ISO 8601: "If the week number containing January 1st has >= 4 days in the new year, then it is week 1; otherwise, it is the last numbered week of the previous year (52 or 53)"			
%w	Day of week as decimal number (0 - 6, Sunday is 0)			
%W	Week of year as decimal number, with Monday as first day of week (00 - 53)			
%x	Date in the format dd <sep>mm<sep>yyyy (the separator <sep> and the order of day, month and year are determined from the regional settings)</sep></sep></sep>			
%X	Time in the format HH <sep>MM<sep>SS the separator <sep> is determined from the regional settings)</sep></sep></sep>			
%у	Year without century, as decimal number (00 - 99)			
%Y	Year with century, as decimal number (19yy-20yy)			
%Z	Time-zone name or abbreviation, nothing if time zone is unknown			

Format string must be encoded using the OEM codepage. The following escape sequences are allowed:

Format	Description

\"	The ''' character	
Υ.	The ''' character	
\\	The '\' character	
\n	New line `\n'	
\t	Tab `\t'	
∖a	bell	
\b	`/ b '	
\f	`\f'	
\v	`\ v '	
\N \NN \NNN	Octal characters code for the OEM code page	
∖xNN	Hexadecimal characters code for the OEM code page	

otes

- 1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.
- 2. If the "format" argument is not specified FAR uses the "%a %b %d
 %H:%M:%S %Z %Y" format string.

xample

REGEDIT

```
; insertion of current date
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Common\Ctrl
"Sequence"="$If (Editor || Dialog || Search) $Date \"
"DisableOutput"=dword:00000001
```

e also: Examples, date()
Text

in | Macros | Macro-language | Macro-commands

The **\$Text** macro-command inserts the text into the current object editor.

\$Text "string"

lements

string

The text to be inserted 'string' must be encoded using OEM codepage. The following escape-sequences can be used in this macro-command:

Format	Description
\"	The "" character
\'	The " character
\\	The '\' character
∖n	New line `\n'
\t	Tab `\t'
∖a	Bell
\b	`\b'
\f	`\f'
\v	`\ v '
\N \NN \NNN	Octal characters code for the OEM code page
∖xNN	Hexadecimal characters code for the OEM code page

otes

- 1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.
- 2. This macro command allows to minimize the usage of memory allocated for the macro command sequence and make the sequence more comprehensive.
- 3. Keyword "\$Text" can be omitted i.e. FAR will automatically transform any

quoted text to \$Text "text" except the situation when the " character is followed by the 'space' character.

xample

```
REGEDIT
```

```
;create the new folder named by the current date
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlS
"Sequence"="%Folder=date(\"%d.%m0.%Y\"); $If (!panel.
"DisableOutput"=dword:00000001
"NoPluginPanels"=dword:00000001
```

e also:

Examples

IClip

in | Macros | Macro-language | Macro-commands

The **\$IClip** macro-command helps to organize the clipboard usage.

FAR Manager uses the standard Microsoft Windows clipboard to perform copy/paste operations by default. This macro-command can switch between the usage of the internal FAR clipboard or the standard Windows clipboard within a single macro.

Works as a trigger.

\$IClip

rguments

None.

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

xample

REGEDIT4

```
;quick find the file in the passive panel with the sa
;sequence F5 ShiftEnter CtrlIns is used to get the fi
;of the state of the command line and number of selec
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlA
"Sequence"="$IClip $If (((Bof && APanel.Root) || !Bof
$If (fexist(PPanel.Path+\"\\\\\"+APanel.Current) == 1
CtrlIns Esc Tab Home Alt< ShiftIns Esc $End $End"
"DisableOutput"=dword:00000001
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlA
"Sequence"="$IClip $If (((Bof && APanel.Root) || !Bof
$If (fexist(PPanel.Path+\"\\\\\"+APanel.Current) == 1
```

Esc Tab Home Alt< ShiftIns Esc \$End \$End" "DisableOutput"=dword:0000001</pre>

e also:

Examples, Clip, IClip

MMode

<u>in | Macros | Macro-language | Macro-commands</u>

The **\$MMode** macro-command toggles the screen refreshing mode during the execution of a macro sequence. Works as a trigger.

\$MMode 1

rguments

1

Should always be 1.

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

xample

REGEDIT4

```
; insert the current time
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Common\Ctrl
"Sequence"="$If (Editor || Dialog) $Date \"%H:%M:%S\"
$Else $MMode 1 CtrlT $End"
"DisableOutput"=dword:00000001
```

```
e also:
Examples
```

Exit

in | Macros | Macro-language | Macro-commands

The **\$Exit** macro-command stops the execution of a macro sequence.

\$Exit

rguments

None.

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

xample

REGEDIT4

```
;remove the selected characters from the command line
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Common\Shif
"Sequence"="$If (Qview || Shell)
$If (!APanel.Visible) ShiftDel $Exit $End Ctrl0 Shift
$Else ShiftDel $End"
"DisableOutput"=dword:00000001
```

```
"NotEmptyCommandLine"=dword:0000001
```

e also: Examples

XLat

in | Macros | Macro-language | Macro-commands

The **\$XLat** macro-command transliterates the text selected in the current object.

\$XLat

rguments

None.

otes

- 1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.
- 2. This macro-command ignores the hotkeys assigned using the TechInfo#10.

xample

REGEDIT4

```
;transliterates the last word using XLAT
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\Shift
"Sequence"="CtrlShiftLeft $XLat CtrlRight"
"DisableOutput"=dword:0000001
"NotEmptyCommandLine"=dword:00000001
```

e also: Examples

KbdSwitch

in | Macros | Macro-language | Macro-commands

The **\$KbdSwitch** macro-command toggles the keyboard layout in a cyclic manner.

\$KbdSwitch

arameters

None.

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

e also:

Examples

SelWord

in | Macros | Macro-language | Macro-commands

The **\$SelWord** macro-command allows to select the word under the cursor.

\$SelWord

rguments

None.

emarks

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

e also:

Examples

lf

in | Macros | Macro-language | Macro-commands

The **\$If** macro-command implements conditional execution of the macro sequence fragments.

\$If (Expr) TruePart [\$Else FalsePart] \$End

emnts

Expr

Conditional expression.

TruePart

Execute if **Expr** is true.

FalsePart

Execute if **Expr** is false.

otes

- 1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.
- 2. The **\$Else** element can be omitted.

xample

REGEDIT4

```
;more convenient Alt-navigation
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Common\Altu
"Sequence"="$If (Viewer || Qview) Up $Else AltUp $Enc
"DisableOutput"=dword:00000001
```

e also: Examples

While

in | Macros | Macro-language | Macro-commands

The **\$While** loop macro-command macro-command executes the Sequence of the macro-commands while the Expr is true.

\$While (Expr) Sequence \$End

lements

Expr

Loop conditional expression.

Sequence

Sequence to be executed while **Expr** is true.

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

xample

REGEDIT

```
;set the cursor to the nearest file in the panel
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\AltAc
"Sequence"="$While (APanel.Folder && !APanel.Eof) Dow
"DisableOutput"=dword:00000001
```

e also:

Examples

Rep

in | Macros | Macro-language | Macro-commands

The **\$Rep** loop macro-command executes the Sequence of the macro-commands Expr number of times.

\$Rep (Expr) Sequence \$End

lements

Expr

Expression, evaluated once on loop initialization.

Sequence

Sequence to be executed **Expr** times.

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

xample

REGEDIT4

;exit FAR Manager

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Common\Altx
"Sequence"="$If (Editor && (Editor.State & 0x8)) F2 $
$Rep (2) $If (!Shell) Esc $End $End F10"
"DisableOutput"=dword:00000001
```

e also: Examples, \$While

oolean operators

in | Macros | Macro-language

You can use the following operators in a conditional expression **Expr** within the **<u>\$If</u>** and **<u>\$While</u>** macro-commands:

Operator	Description
==	equal
!=	not equal
<	less
>	greater
<=	less or equal
>=	greater or equal

ATTENTION! Comparison operations for string arguments are case sensitive.

Boolean conditional expression **Expr** can be complex. You can use the following operators to combine the result of several simple expressions:

Operator	Description
&&	Logical AND
	Logical OR

You can use the braces to change the condition evaluation precedence.

otes

1. It is impossible to use macro-language elements while recording a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

oolean object states

in | Macros | Macro-language

It is allowed to use the following keywords within the <u>\$If</u> and <u>\$While</u> conditional expression:

State	Description			
detection of the current area of execution of the macro command				
Shell	file panels			
Viewer	internal viewer			
Editor	internal editor			
Dialog	dialog boxes			
Search	quick file search			
Disks	drive selection menu			
MainMenu	main menu			
Menu	other menus			
Help	help system			
Info	informational panel			
QView	quick view panel			
Tree	folder tree panel			
FindFolder	folder search panel			
UserMenu	user menu			
Other	screen capturing mode			
panel object states				
APanel.Bof PPanel.Bof	true if the current panel item is the first			
APanel.Eof PPanel.Eof	true if the current panel item is the last			
APanel.Empty PPanel.Empty	true if the panel is empty			
APanel.LFN PPanel.LFN	true if LFN (long file names) is enabled for the panel			
APanel.Selected PPanel.Selected	true if there are any files or folders selected on the panel			
APanel.Root PPanel.Root	true if the panel folder is root			

APanel.Visible PPanel.Visible	true if the panel is visible			
APanel.Plugin PPanel.Plugin	true if the panel is a plugin panel			
APanel.Folder PPanel.Folder	true if the current element is a folder			
APanel.Left PPanel.Left	true if the panel is a left panel			
APanel.FilePanel PPanel.FilePanel	true if the panel is a file panel			
command line object states				
CmdLine.Bof	true if the cursor is at the beginning of the command line			
CmdLine.Eof	true if the cursor is at the end of the command line			
CmdLine.Empty	true if the command line is empty			
CmdLine.Selected	true if there is a selected block of text within the command line			
execution context dependent object states				
Bof	beginning of file is reached in the editor, viewer; in QView/Info/Tree panels - beginning of file is reached in the viewing area, in other "panels" - cursor is at the beginning of the command line; in dialog input lines - cursor is at the beginning of the line; in lists/menus - cursor is set on the first item.			
Eof	end of file is reached in the editor, viewer; in QView/Info/Tree panels - end of file is reached in the viewing area, in other "panels" - cursor is at the end of the command line; in dialog input lines - cursor is at the end of the line; in lists/menus - cursor is set on the last item.			
Empty	in QView/Info/Tree panels - viewing area is empty, in other panels - command line is empty; dialog input line is empty; in viewer/editor - the file is empty; menu/list is empty.			
Selected	a block is selected in dialog input lines, in the editor or viewer, in the viewing are of QView/Info/Tree panels, in other panels - a block is selected in the command line (current area is Shell)			
IClip	true if FAR Manager is working with the internal clipboard (See '\$IClip')			
Windowed	true if FAR Manager is in the windowed mode			

state of macro command parameters		
DisableOutput	true if the screen output is disabled	

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

unctions

in | Macros | Macro-language

unction list

You can use the following functions in the macro sequence:

Mathematical functions				
Function	Description			
N=abs(N)	Returns the absolute value of a numeric argument <i>N</i>			
N=max(N1,N2)	Returns the maximal value of two numeric argumen			
N= min (N1,N2)	Returns the minimal value of two numeric argumer			
N=mod(N1,N2)	Division of <i>N1</i> by the modulus <i>N2</i> .			
N=int(V)	Evaluates the <i>V</i> argument to a number.			
String functions				
Function	Description			
N=index(S1,S2)	Returns the position of the first occurrence of the su			
N=rindex(S1,S2)	Returns the position of the last occurrence of the su			
Result= replace (Str,Find,Replace[,Cnt])	Replace substing <i>Find</i> in <i>Str</i> with <i>Replace</i> , <i>Cnt</i> time Returns the resulting string.			
N=asc(S)	Returns the code of character <i>S</i> .			
S=chr(N)	Returns the character for the code <i>N</i> .			
S= itoa (N[,Radix])	Converts numeric value <i>N</i> to a string. <i>Radix</i> parame conversion. The valid <i>Radix</i> values are from 2 to 36			
N= atoi (S[,Radix])	Converts string representation of a number <i>N</i> into a <i>S</i> . The valid <i>Radix</i> values are from 2 to 36. If <i>Radix</i> is omitted, function tries to determine the c			
N=len(V)	Returns the length of the expression <i>V</i> . This functio			
S=string(V)	Converts the value of the expression <i>V</i> to a string.			
S= substr (V,N1[,N2])	Returns the substring of the expression <i>V</i> beginning If <i>N2</i> parameter is omitted or equal to -1, function re If <i>N2</i> parameter is equal to 0, function returns empty			
S=trim(V[,N])	Removes whitespace (spaces, tabs, line feeds, and c leading (N =1), or only trailing (N =2).			
S=lcase(S)	Transforms the string <i>S</i> to lower-case.			
S=ucase(S)	Transforms the string <i>S</i> to upper-case.			

S= date (format)	Returns the current date/time according to the forma This function is similar to \$Date macro-command b
S=xlat(S)	Returns the transliterated string. This function is similar to \$XLat macro command b

Function	Descriptio	Description	
S= fsplit (path,N)	Splits the J	path <i>path</i> into the components and return	
	Value	Description	
	0x0000001	Current drive in "C:" representation. For network shares - "\\server\share"	
	0x0000002	Program Files\Far\" representation	
	0x0000004	Name	
	0x0000008	Extension in ".EXT" representation or empty strin	
N=fexist(S) N=fattr(S)	Checks if t Usage of the the file or the This funct Returns th	he file of folder S exists and returns 1 in he '*' and '?' mask symbols is allowed in folder. ion does not work with the panels.	
	Attribute	Description	
	0x0000001	FILE_ATTRIBUTE_READONLY. The file or fol case of a folder, applications cannot delete it.	
	0x0000002	FILE_ATTRIBUTE_HIDDEN. The file or folder	
	0x0000002	FILE_ATTRIBUTE_SYSTEM. The file or folder	
	0x00000010	FILE_ATTRIBUTE_DIRECTORY. This is a fold	
	0x00000020	FILE_ATTRIBUTE_ARCHIVE. The file or fold	
	0x0000080	FILE_ATTRIBUTE_NORMAL. The file or fold	
	0x00000100	FILE_ATTRIBUTE_TEMPORARY. The file is the sufficient cache memory is available, because oftic case, the system can entirely avoid writing the dates of the system can entirely avoid writing t	
	0x00000200	FILE_ATTRIBUTE_SPARSE_FILE. The file is	
	0x00000400	FILE_ATTRIBUTE_REPARSE_POINT. The file	
	0x0000800	FILE_ATTRIBUTE_COMPRESSED. The file o For a folder, this means that compression is the d	
	0x00001000	 FILE_ATTRIBUTE_OFFLINE. The data of the physically moved to offline storage. This attribut should not arbitrarily change this attribute. 	
	0x00001000	 FILE_ATTRIBUTE_OFFLINE. The data of the physically moved to offline storage. This attribut should not arbitrarily change this attribute. FILE_ATTRIBUTE_NOT_CONTENT_INDEXE 	
	0x00001000 0x00002000 0x00004000	 FILE_ATTRIBUTE_OFFLINE. The data of the physically moved to offline storage. This attribut should not arbitrarily change this attribute. FILE_ATTRIBUTE_NOT_CONTENT_INDEX. FILE_ATTRIBUTE_ENCRYPTED. The file or this means that encryption is the default for newleting the storage. 	

		Usage of the '*' and '?' mask symbols is allowed in the file or folder. This function does not work with the panels.			
Panel					
Function		Description			
N= panel.fexist (panelType,S)		Checks if the file or folder <i>S</i> exists in the active (<i>p</i> otherwise). Usage of the '*' and '?' mask symbols is allowed in the file or folder. This function works only with the panels.			
N= panel.fattr (panelType,S)		Returns the file system attributes of the file or folde			
		Attribute	Description		
		0x00000001	FILE_ATTRI case of a fold	BUTE_READONLY. The file or folder, applications cannot delete it.	
		0x00000002	FILE_ATTRIBUTE_HIDDEN. The file or fold		
		0x00000004	FILE_ATTRI	BUTE_SYSTEM. The file or folder is	
		0x00000010	FILE_ATTRI	BUTE_DIRECTORY. This is a folder.	
		0x00000020	FILE_ATTRIBUTE_ARCHIVE. The file or folder		
		0x0000080	FILE_ATTRIBUTE_NORMAL. The file or fold		
		0x00000100	FILE_ATTRIBUTE_TEMPORARY. The file is sufficient cache memory is available, because o case, the system can entirely avoid writing the c		
		0x00000200	FILE_ATTRI	BUTE_SPARSE_FILE. The file is a s	
		0x00000400	FILE_ATTRI	BUTE_REPARSE_POINT. The file o	
		0x00000800	FILE_ATTRI For a folder, t	BUTE_COMPRESSED. The file or for his means that compression is the defation of	
		0x00001000	FILE_ATTRIBUTE_OFFLINE. The data of the f physically moved to offline storage. This attribute should not arbitrarily change this attribute.		
		0x00002000	FILE_ATTRI	BUTE_NOT_CONTENT_INDEXED	
		0x00004000	FILE_ATTRI this means that	BUTE_ENCRYPTED. The file or fole at encryption is the default for newly c	
		0x00010000	FILE_ATTRI	BUTE_VIRTUAL. A file is a virtual f	
		if file system Usage of the the file or fo This functio	n object doe e '*' and '?' r older. on works onl	s not exist this function retu nask symbols is allowed in t y with the panels.	
V= panelitem (panelType,itemIndex,propIndex)		Queries the <i>propIndex</i> c	information an be one of	for the panel element <i>itemI</i> the following values:	
		Value	Туре	Description	
		0	String	File or folder name	
		1	String	File or folder short name	

	2	Number	File attributes
	3	String	Creation date/time
	4	String	Last access date/time
	5	String	Last modification date/time
	6	Number	Size
	7	Number	Packed size
	8	Number	"Is selected?"
	9	Number	Number of the hardlinks
	10	Number	SortGroup
	11	String	Diz-text
	12	String	Owner
	13	Number	CRC32
	14	Number	Position of the element while reading
	Date/time i the regiona	s returned in l settings. To	the DD.MM.YYYY HH:M receive information on the
N= panel.setpos (panelType,fileName)	Positions the cursor on the element with name <i>fileN</i> Returns the position of the element or 0 if there is no For example, the following macro creates a director cursor on this directory if it already exists.		
	%dt=da \$If (1 pane \$Else F7 \$ \$End	ate("%Y%m ⁼ exist(%d el.SetPos SText %dt	10%d"); lt)) s(0,%dt) : Enter
N= panel.setpath (panelType,pathName[,fileName])	In the activ name fileNa Returns 1 if For exampl the active of \$if(AF panel. panel.	e (panelType ame (if fileN f folder was e, in the acti ne and set co Panel.Lef SetPath(SetPath(<pre>2=0) or passive (panelType= ame parameter was specified changed successfully and 0 ve panel change folder to "C ursor to "FAR" folder: t) CtrlU \$End 1, "C:\\WINDOWS") 0, "C:\\Program File</pre>
N= panel.setposidx (panelType,indexItem)	Positions th Returns ele	e cursor on ment positio	the element with index <i>index</i> n or 0 if such element does
Control functions			
Function	Description	n	
V= iif (Cond,True,False)	Conditiona	l function. If	condition <i>Cond</i> is true then
Execution environment			

Function	Description			
V=akey(N)	Returns the name or code of the key combination the set of the set			
S=env(V)	Returns the value of an environm			
	<pre>\$if (Env("FARLANG") == "English"</pre>			
B=msave(S)	Stores the value of a global variable with the nam Example: %%GlobalVars="Foo Bar"; %a=msav			
V= clip (Cmd[,S])	Allows to p	perform miscellaneous manipulations on		
	Cmd	Description		
	0	Returns data from the Clipboard; the <i>S</i> parameter is		
	1	Put the string <i>S</i> to the Clipboard; in case of error ret		
	2	Add the string <i>S</i> to the Clipboard; in case of error re		
	3	Copy the Windows Clipboard to the internal clipboa		
	4	Copy the internal clipboard buffer to the Windows C		
	For example, put to the Clipboard a list of selected 1			
	Clip(: Home \$While \$If (\$End \$End	Clip(1,"Name;Size\r\n") Home \$While(!APanel.Eof) \$If(!APanel.Folder && PanelIter Clip(2,PanelItem(0,0,0)+";" \$End Down \$End		
	See <u>\$IClip</u> , <u>IClip</u> .			
V= flock (Nkey,NState)	Toggles Lock keys state (NumLock, CapsLock, and Key <i>Nkey</i> :			
	Nkey	Description		
	0	NumLock		
	1	CapsLock		
	2	2 ScrollLock		
	State <i>NState</i> :			
	NState	Description		
	-1	get key state		
	0	toggle key to off		
	1	toggle key to on		

	2 toggle key to reverse state	
	In Windows 95/98/Me the function can change only no such restriction. The function is unstable (computer may freeze) and (switching between windows) should be wrapped i	
	<pre>CtrlShiftTab %a=flock(1,-1)&1; \$while((flock(1,-1)&1)==%a)</pre>	
N=sleep(N)	Pause macro execution for <i>N</i> milliseconds. Negativ	
V=waitkey([N, [T]])	Waits until a key is pressed and returns its name (T an empty string (T =0) or 0 (T =1). If N is omitted or	
S= key (V)	Converts parameter <i>V</i> into the string equivalent of t If <i>V</i> is a string, its correctness is checked and return name. Function returns empty string in case of any errors.	
N= eval (S[,T])	"Play back" (<i>T</i> =0) or check (<i>T</i> =1) macro sequence played. FAR compiles sequence <i>S</i> . For playback mode (<i>T</i> =0): if there are no errors, sta playing <i>S</i> , suspended macro continues its work. If the sequence <i>S</i> compilation was successful, funct	
	Error Description	
	1 Unrecognized keyword 'keyword'	
	2 Unrecognized function 'function'	
	3 Incorrect number of arguments for function 'function'	
	4 Unexpected \$Else	
	5 Unexpected \$End	
	b Unexpected end of source string 7 Fracted line conditions	
	/ Expected 'keyword'	
	o Bad Hex Control Char 0 Bad Castral Char	
	9 Bad Control Char	
	10 Valiable Expected Valiable	

```
contents should be saved.
We set 2 string variables, in the Vars area, containin
  %%CmdSave=$If (!CmdLine.Empty) %F
             %CmdCurPos=CmdLine.Iten
             %CmdVal=CmdLine.Value;
  %%CmdRestore=$If (%Flg_Cmd==1) $1
             %Flg_Cmd=0; %Num=%CmdCu
             $While (%Num!=0) %Num=9
then we set the following macro to F10:
  $If (APanel.Visible && (!APanel.F
    Eval(%%CmdSave)
    $If (APanel.Current == "..")
      ShiftEnter
    $Else
      "%SystemRoot%\\explorer.exe /
      $If (Apanel.LFN)
        CtrlN
      $End
      CtrlEnter
      $If (!Apanel.LFN)
        CtrlN
      $End
      Enter
    $End
    Eval(%%CmdRestore)
  $End
```

User interaction

Function	Description	L
N= msgbox ([Title[,Text[,Flags]]])	Shows up th <i>Flags</i> can be	e message box with the <i>Text</i> and the <i>Ti</i> e a combination of the following values
	Flags	Description
	0x00000001	Use the 'Warning' color scheme (usually white text
	0x0000008	Show the message lower by two lines.
	0x00000010	Use the left alignment for the message strings (alig
	0x00010000	Show the <ok> button.</ok>
	0x00020000	Show the <ok> and <cancel> buttons.</cancel></ok>
	0x00030000	Show the <abort>, <retry> and <ignore> buttons.</ignore></retry></abort>
	0x00040000	Show the <yes> and <no> buttons.</no></yes>
	0x00050000	Show the <yes>, <no> and <cancel> buttons.</cancel></no></yes>
	0x00060000	Show the <retry> and <cancel> buttons.</cancel></retry>

	If the <i>Flags</i> argument is 0 (or c The return value is the number the result will be 0.	omitted) then standard of the button the use
S= prompt ("Title"[,"Prompt"[,flags[, "Src"[, "History"]]]])	Function allows to enter one te Parameters: <i>Title</i> - dialog caption <i>Flags</i> can be the set of these variables	xt string. on, <i>Prompt</i> - prompt llues (similar to <u>FIB</u>
	Flag	Desc
	0x0000001	the f
	0x0000002	enter
	0x0000004	after by th conta
	0x0000008	if Sro histo
	0x0000010	displ grow
	0x0000020	the a defir
	Function returns a string entere For example, ask for password	ed by user. and show it:
	%s=prompt("Password	l","Input passw

Function	Description	
N= checkhotkey (S[,Pos])	Function checks and gets the position of the eleme function will return the position of the element wit Returns: -1 - if called from the wrong area (MainM menu/dialog/list contains hot key <i>S</i> . For example, in the plugin menu (F11), hot key '7'	
	F11 \$if (checkhotkey("7")) 7 1 \$	
	will call the plugin if hot key is defined, or standar	
S=gethotkey(N)	Returns the hot key <i>S</i> for item <i>N</i> , if the item has a N - item position beginning with 1 (0 - current item	
N= Menu.Select (S[,N])	In the menu, places cursor to the first item that cor	
	Mode De	
	0 str	
	1 str	
	2 str	
	3 str	

	Search is Returns: - position.	case-insensiti 1=Error - if c	ve. If item is called not from	not found, does 1 the menu, 0=I
	F11 \$	sif(Menu.	Select("A	dvanced com
	If N paran	neter is omitt	ed, full string	coincidence is
V= Dlg.GetValue (ID,Type)	Get misc. <i>ID</i> - dialo, <i>Type</i> - typ are allowe	values of the g element nu e of the value d:	current dialo; mber (elemen e to get, deper	g. ts are numberec ading on the typ
	Туре	Var type	Description	
	0	Number	Number of eler	nents in the dialog
	2	Number	X1	
	3	Number	Y1	
	4	Number	X2	
	5	Number	Y2	
	6	Number	ID of the curre	nt element that has th
	For dialog	elements the	e following va	llues are allowe
	Туре	Var type	Element	Description
	0	Number	CheckBox	State of the check
	0	Number	RadioButtons	State of the radio h
	0	String	ListBox	Value of the curre
	0	String	ComboBox	Value of the currer
	0	String	Other	Textual value (the
	1	Number	All	Element type
	2	Number	All	X1 relative to the
	3	Number	All	Y1 relative to the
	4	Number	All	X2 relative to the
				170 1
	5	Number	All	Y2 relative to the
	5 6	Number Number	All	does the element h
	5 6 7	Number Number Number	All All CheckBox	does the element h
	5 6 7 7	Number Number Number Number	All All CheckBox RadioButtons	does the element h State of the checkl State of the radio h
	5 6 7 7 7	Number Number Number Number String	All All CheckBox RadioButtons ListBox	42 relative to the of does the element h State of the checkl State of the radio h Current position in
	5 6 7 7 7 7 7	Number Number Number Number String String	All All CheckBox RadioButtons ListBox ComboBox	does the element h State of the checkl State of the radio h Current position in Current position in
	5 6 7 7 7 7 7 7 7	Number Number Number Number String String Number	All All CheckBox RadioButtons ListBox ComboBox Other	42 relative to the does the element h State of the checkl State of the radio h Current position in Current position in 0
	5 6 7 7 7 7 7 7 8	NumberNumberNumberNumberStringStringNumberNumberNumber	All All CheckBox RadioButtons ListBox ComboBox Other All	42 relative to the of does the element h State of the checkl State of the radio h Current position in Current position in 0 Element flags
	5 6 7 7 7 7 7 7 8 9	NumberNumberNumberNumberStringStringNumberNumberNumberNumberNumber	All All CheckBox RadioButtons ListBox ComboBox Other All All	42 relative to the of does the element h State of the checkt State of the radio h Current position in Current position in 0 <u>Element flags</u> A flag that specifie

V=**Editor.Set**(Idx,Var)

Change settings of the current editor instance.

Index	Var Type	Description
0	Number	Tab size
1	Number	Expand tabs
2	Number	Persistent blocks
3	Number	Del removes blocks
4	Number	Autoindent
5	Number	Autodetect character table
6	Number	Create new files in WIN encoding
7	Number	Cursor beyond end of line
8	Number	BackSpace behaviour (TechInfo #13
9	Number	Representation of the character und
10	Number	Save file position
11	Number	Save bookmarks
12	String	Word delimiters (TechInfo #1)
13	Number	EditorF7Rules (TechInfo #18)
14	Number	AllowEmptySpaceAfterEof (TechIn

The value of *Var* can be -1, in this case the function Returns the previous value of the option or -1 if wrc For example, a macro that moves the selected block

\$IClip CtrlX %a=Editor.Set(2,1);

V=**Editor.Sel**(Action[,Opt])

Function allows manipulations with blocks in the te

Action	Description		
0	get parameters of the current block		
	Opt	Description	
	0	return the string nun	
	1	return the position n	
	2	return the string nun	
	3	return the position n	
	4	return the type of highighted, 1 - regu (alternative way is E	
1	place cursor	within current block (block state is not	

	Opt	Description	
	0	place cursor to the s	
	1	place cursor to the e	
2	set block pos	itions	
	Opt	Description	
	0	set block start positi	
	1	set block end positic	
	Позицией начала/конца блока считается текущая (<i>Opt</i> =1), с ранее выделенного блока пометка сниравносильна снятию пометки блока.		
3	установка позиций вертикального блока		
	Opt	Description	
	0	отметить позицию	
	1	отметить позицию	
	Позицией начала/конца блока считается текущая (<i>Opt</i> =1), с ранее выделенного блока пометка снигравносильна снятию пометки блока.		
4	снять выделение с блока Параметр <i>Орt</i> игнорируется. Всегда возвращается 1.		

В случае ошибки (в т.ч. функция вызвана не из " возвращается запрошенное значение. Например, повторим некоторые сочетания клави

```
%k=waitkey(0,1);
$if(%k==B || %k==b || %k==CtrlB)
  editor.sel(2,0)
$else
  $if(%k==K || %k==k || %k==Ctrlł
    editor.sel(2,1)
  $else
    $if((%k==C || %k==C || %k==C1
      CtrlC
    $else
      $if((%k==Y || %k==y || %k==
        CtrlD
      $else
        eval("F1")
      $end
    $end
  $end
```

	\$end	
Bookmarks		
Function	Description	
N= BM.Add ()	Добавить текущие параметры закладки и обреза	
N= BM.Clear ()	Clear all bookmarks	
N= BM.Del ([Idx])	Удалить закладку с указанным индексом <i>Idx</i> (ну удаляется текущая закладка.	
N= BM.Get (Idx,M)	Получить параметры закладки с индексом Idx (н M parameter can have these values:	
	Type Des	
	0 коор	
	1 коор	
	2 пози	
	З	
N= BM.Next ()	Jump to the next bookmark	
N= BM.Prev ()	Jump to the previous bookmark	
N= BM.Stat ([M])	Get bookmark information. <i>M</i> parameter can have these values:	
	Type Des	
	0 curr	
Bookmark handling example:	I	

otes

- 1. Bookmark functions BM.XXX() are available only within the editor.
- 2. Function arguments given in square brackets may be omitted. See default values in function description.
- 3. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

perations

in | Macros | Macro-language

You can use the following operations on the variables:

Operation	Description
+	Numeric addition or string concatenation
-	Numeric subtraction or unary negation
*	Numeric multiplication
/	Numeric division. FAR interrupts macro execution if division by 0 occurs.
&	Bitwise AND
٨	Bitwise XOR
	Bitwise OR
<<	Logical left shift
>>	Logical right shift
!	Logical NOT
~	Bitwise NOT

Operator precedence:

Operation	Description	
!~-	Unary (R->L)	
*/	Arithmetic	
+ -	Arithmetic	
<< >>	Logical shift	
< <= > =>	Logical comparison	
== !=	Logical comparison	
&	Bitwise (and)	
٨	Bitwise (xor)	
	Bitwise (or)	
&&	Logical	
	Logical	

otes

- 1. Named keys (e.g. CtrlK) can be present in any expressions; in this case they are treated as numbers.
- 2. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

bject states in | Macros | Macro-language

Panels				
Operator	Туре	Descrip	tion	
APanel.ItemCount PPanel.ItemCount	Number	stores th	stores the number of the elements on the panel	
APanel.SelCount PPanel.SelCount	Number	stores th	e number of the selected files on the panel	
APanel.CurPos PPanel.CurPos	Number	stores th	e index of an element on the panel	
APanel.Current PPanel.Current	String	stores th	e name of an element under the cursor	
APanel.Path PPanel.Path	String	stores th	e path for the panel (without trailing '\')	
APanel.UNCPath PPanel.UNCPath	String	stores th	e UNC path for the panel (without trailing '\')	
APanel.Width PPanel.Width	Number	stores th	stores the width of the panel	
APanel.Type PPanel Type	Number	stores the type of the panel:		
r r unei. rype		Value	Description	
		0	File panel	
		1	Tree panel	
		2	Quick view panel	
		3	Informational panel	
APanel.DriveType	Number	er specifies the drive type of the panel:		
i i anei.Diive iype		Value	Description	
		-1	plugin panel	
		0	Drive type couldn't be detected	
		2	Removable drive	
		3	Hard disk	
		4	Mapped network share	
		5	CDROM	
		6	Virtual drive	
		15	SUBST-disk	
APanel.OPIFlags	Number	plugin p	anel flags, can be a combination of the following values (if the	

		Value	Description	
		0x0000000	01 there's a filter applied to the panel	
		0x0000000	22 there're a sort groups used on the panel	
		0x000000	04 the highlighting is used on the panel	
		0x0000001	10 folder selection mode does not depend on the FAR Manager settings	
		0x0000002	the standard FAR file processing mode is used, if the requested opera plugin; if this flag is set then panel element names are the real file name	
		0x0000004	40 file names without paths are shown	
		0x0000008	30 file names are aligned to the right	
		0x0000010	00 the original case is used to display the file names (despite of FAR Ma	
APanel.ColumnCount PPanel.ColumnCount	Number	Number o	of panel columns.	
Dialogs	-			
Dlg.ItemCount	Number	number o	f elements in a dialog box	
Dlg.CurPos	Number	the numb	er of the item of a dialog box currently in focus	
Dlg.ItemType	Number	type of the element currently in focus		
		Value	Description	
		-1	error, for example Dlg.ItemType was called outside of the dialog	
		4	Text input box	
		5	Password input box	
		6	Fixed width input box	
		7	Push Button	
		8	Check Box	
		9	Radio Button	
		10	Combo box	
		11	List box	
		255	Custom control	
		0x8004	Text input box history	
		0x800A	Combo box list	
Command Line				
CmdLine.ItemCount	Number	number o	f characters in the command line	
CmdLine.CurPos	Number	current co	ommand line cursor position	
CmdLine.Value	String	command	l line content	
Editor				

Editor.FileName	String	full name of	the file being edited	
Editor.CurLine	Number	current line	in the editor (first line is 1)	
Editor.CurPos	Number	current curse	or position in the current line in the editor (first column is	
Editor.RealPos	Number	current curse size	or position in the current line in the editor (first column is	
Editor.Value	Value	contents of t For example	he current line in the editor (under the cursor). e, to show the character under the cursor:	
		MsgBox	(substr(Editor.Value,Editor.CurPos-1,1	
Editor.Lines	Number	number of li	nes in the editor	
Editor.State	Number	state of the current internal file editor - bit flag set:		
		Value	Description	
		0x00000001	file is new or already deleted	
		0x00000002	can be switched to the viewer by F6	
		0x00000004	remove the file after closing the editor	
		0x00000008	file was modified in the editor (there is a '*' sign in the editor status line)	
		0x00000010	there is a stream selection box (alternative is <u>Editor.Sel(0,4</u>))	
		0x00000020	there is a vertical selection block (alternative is <u>Editor.Sel(0,4)</u>)	
		0x00000040	file was modified during the whole editing session	
		0x00000080	the cursor is in replace mode	
		0x00000100	cursor position was modified by the plugin	
		0x00000200	the editor is locked (ReadOnly)	
		0x00000400	permanent blocks are used	
		0x00000800	modal editor	
		0x08000000	FAR is started with /e	
		Example:		
		\$If (E	ditor.State & 0x8)do something if a	
Viewer				
Viewer.FileName	String	full name of	the file being viewed	
Viewer.State Number		state of the current internal file viewer - bit flag set:		
		Value	Description	
		0x00000001	codepage autodetection is on	
		0x00000002	text and codepage are not in ANSI encoding	
		0x00000004	Unicode mode	
		0x0000008	line wrapping is on	

		0x00000020	hexadecimal mode is used	
		0x00000800	modal viewer	
		0x08000000	FAR is started with /v	
		Example,		
		\$If (Vi	lewer.State & 0x20)do something if	
Drive menu				
Drv.ShowPos	Number	drive menu is	s shown for the left panel (AltF1, value 1) or the right par	
Drv.ShowMode	Number	drive menu representation flags; bitmask:		
		Value	Description	
		0x00000001	disk type displaying is enabled	
		0x00000002	network name (and the path associated with a SUBST drive under NT) displa	
		0x00000004	disk label displaying is enabled	
		0x0000008	file system type displaying is enabled	
		0x00000010	total and free disk size displaying is enabled	
		0x00000020	removable disk parameters displaying is enabled	
		0x00000040	plugin items displaying is enabled	
		0x00000080	CD parameters displaying is enabled	
		0x00000100	notwork parameters displaying is enabled	
		0x0000200	network parameters uspraying is enabled	
Other				
MacroArea	String	name of the o	current macro area	
ItemCount	Number	number of elements in the current object; delimiters are also counted		
CurPos	Number	position in th	e current object	
Title	String	title of the cu	urrent object	
Far.Width	Number	FAR Manager console width		
Far.Height	Number	FAR Manager console height		
Far.Title	String	current title of FAR console window		
Help.FileName	String	full path to the opened help file; for the list of plugins help topics (Sh		
Help.Topic	String	ID of the current help topic (without the leading '@' symbol)		
Help.SelTopic	String	ID of the selected help topic (without the leading '@' symbol)		
MsX	Number	Horizontal of	ffset of mouse cursor since the last mouse event was trigg	
MsY	Number	Vertical offse	et of mouse cursor since the last mouse event was triggere	
MsButton	Number	Indicates the status of the mouse buttons. The least significant bit cor mouse button. The next least significant bit corresponds to the rightm		

		indicates buttons. A The follo	the next-to-leftmost mouse button. The bits then correspond A bit is 1 if the button was pressed. wing constants are defined for the first five mouse buttons:
		Value	Description
		0x0001	FROM_LEFT_1ST_BUTTON_PRESSED
		0x0002	RIGHTMOST_BUTTON_PRESSED
		0x0004	FROM_LEFT_2ND_BUTTON_PRESSED
		0x0008	FROM_LEFT_3RD_BUTTON_PRESSED
		0x0010	FROM_LEFT_4TH_BUTTON_PRESSED
MsCtrlState	itate Number	Indicates	the state of the control keys. This member can be one or mor
		Value	Description
		Value 0x0001	Description RIGHT_ALT_PRESSED
		Value 0x0001 0x0002	Description RIGHT_ALT_PRESSED LEFT_ALT_PRESSED
		Value 0x0001 0x0002 0x0004	Description RIGHT_ALT_PRESSED LEFT_ALT_PRESSED RIGHT_CTRL_PRESSED
		Value 0x0001 0x0002 0x0004	DescriptionRIGHT_ALT_PRESSEDLEFT_ALT_PRESSEDRIGHT_CTRL_PRESSEDLEFT_CTRL_PRESSED
		Value 0x0001 0x0002 0x0004 0x0008 0x0010	DescriptionRIGHT_ALT_PRESSEDLEFT_ALT_PRESSEDRIGHT_CTRL_PRESSEDLEFT_CTRL_PRESSEDSHIFT_PRESSED
		Value 0x0001 0x0002 0x0004 0x0008 0x0010 0x0020	DescriptionRIGHT_ALT_PRESSEDLEFT_ALT_PRESSEDRIGHT_CTRL_PRESSEDLEFT_CTRL_PRESSEDSHIFT_PRESSEDNUMLOCK_ON
		Value 0x0001 0x0002 0x0004 0x0008 0x0010 0x0020 0x0040	DescriptionRIGHT_ALT_PRESSEDLEFT_ALT_PRESSEDRIGHT_CTRL_PRESSEDLEFT_CTRL_PRESSEDSHIFT_PRESSEDNUMLOCK_ONSCROLLLOCK_ON
		Value 0x0001 0x0002 0x0004 0x0008 0x0010 0x0020 0x0040 0x0080	DescriptionRIGHT_ALT_PRESSEDLEFT_ALT_PRESSEDRIGHT_CTRL_PRESSEDLEFT_CTRL_PRESSEDSHIFT_PRESSEDNUMLOCK_ONSCROLLLOCK_ONCAPSLOCK_ON

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

ariables

in | Macros | Macro-language

You can use the variables to store some values associated with the names and use them later in the macro sequences. Both global and local variables can be used in the macro sequence.

aming

Local variable name begins with the '%' sign followed by the alpha characters, numbers or '_' characters.

Global variable name begins with the '%%' signs followed by the alpha characters, numbers or '_' characters.

Variable value assignment should be ended with ';'

Variable names are not case sensitive thus '%myStr' and '%MYstr' is the same variable.

/pes

Variables can be either string or integer.

Integer constants can be represented by: NNN - decimal constant, 0NNN - octal constant, 0xNNN - hexadecimal constant.

Integers are of 64 bit width.

cope of action

The scope for the local variables is the current macro sequence.

The scope for the global variables is a current FAR Manager session. Global variables can be <u>stored</u> in the **Vars** execution area. Every time FAR starts up it restores the global variables stored in this area.

epresentation in registry

In the registry, global variables are stored in the special key [HKEY_CURRENT_USER\Software\Far\ [Users\USERNAME\]KeyMacros\Vars].

Every global variable has a name and can be of three types REG_SZ (for string variables) and REG_DWORD or REG_QDWORD (for integer variables). If a variable had REG_DWORD type initially, it changes type to REG_QDWORD
during saving with <u>msave</u> function.

otes

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

e also:

msave()

echnical details

in | <u>Macros</u>

toring macro commands

FAR macro commands are located in Windows registry under
[HKEY_CURRENT_USER\Software\Far\
[Users\USERNAME\]KeyMacros] registry key in a text form.

Macro commands are divided into 16 <u>areas of execution</u> and 2 areas to store global variables and constants. Area name corresponds to the registry key:

Area	Purpose	
"Shell"	File panels	
"Viewer"	Internal file viewer	
"Editor"	Internal file editor	
"Dialog"	Dialog boxes	
"Search"	Quick file search	
"Disks"	Drive selection menu	
"MainMenu"	Main menu	
"Menu"	Other menus	
"Help"	Help system	
"Info"	Informational panel	
"QView"	Quick view panel	
"Tree"	Folder tree panel	
"FindFolder"	Folder search panel	
"UserMenu"	User menu	
"Other"	Screen capturing mode	
"Common"	Common area. Macros created in this area can be used everywhere. This area has the lowest priority while processing macros.	
"Vars"	Global variables area read by FAR during startup. Variables can be string (REG_SZ) and numeric (REG_QWORD). One can write to this area using msave() function.	
"Consts"	Global constants area read by FAR during startup. Constants can be string (REG_SZ) and numeric	

Every macro command is stored in a separate sub key of the certain area of execution, which name is assigned to a new shortcut name and contains a set of values that define the macro command behaviour.

Name	Туре	Default	Description
Sequence	REG_SZ REG_MULTI_SZ		Contains the sequense of the keystrokes.
DisableOutput	REG_DWORD	0	Disable screen output while executing the macro. When user creates new macro FAR sets this parameter to 1 (disable screen output). If this parameter is omitted FAR uses its default value and enables screen output.
RunAfterFARStart	REG_DWORD	0	Execute macro command after FAR startup. This flag applies only to macros that start in the Shell area.
EmptyCommandLine	REG_DWORD	0	Execute this macro only if command line is empty.
NotEmptyCommandLine	REG_DWORD	0	Execute this macro only if command line is not empty.
NoFilePanels	REG_DWORD	0	Do not execute this macro for a file panel.
NoFilePPanels	REG_DWORD	0	Do not execute this macro for a passive file panel.
NoPluginPanels	REG_DWORD	0	Do not execute this macro for a plugin panel.
NoPluginPPanels	REG_DWORD	0	Do not execute this macro for a passive plugin panel.
NoFolders	REG_DWORD	0	Do not execute this macro if the current element is a folder.
NoPFolders	REG_DWORD	0	Do not execute this macro if the current element is a folder for a passive panel.
NoFiles	REG_DWORD	0	Do not execute this macro if the current element is a file.
NoPFiles	REG_DWORD	0	Do not execute this macro if the current element is a file for an inactive panel.

Selection	REG_DWORD	0	Execute only if there are any selected elements on the panel.
PSelection	REG_DWORD	0	Execute only if there are any selected elements on the passive panel.
NoSelection	REG_DWORD	0	Execute only if there are no selected elements on a panel.
NoPSelection	REG_DWORD	0	Execute only if there are no selected elements on a passive panel.
EVSelection	REG_DWORD	0	Execute if there is a selected text block in the viewer, editor or dialog text boxes.
NoEVSelection	REG_DWORD	0	Execute if there is no selected text block in the viewer, editor or dialog text boxes.
NoSendKeysToPlugins	REG_DWORD	0	Special mode: don't send keystrokes to the plugins during recording and executing.
Description	REG_SZ		Macro command description.

There's only one mandatory parameter "Sequence", other parameters can be omitted and they will have the default values according to their specifications.

The "Sequence" parameter contains literal representation of the command sequence. Every keystroke is represented by its string equivalent delimited by space or tab character.

During startup FAR compiles command sequence for every macro command defined into internal byte code representation. If there is any error during compilation of the macro command then the macro will be ignored.

xecution conditions

Before executing the macro FAR checks the execution conditions in the following order:

- 1. NoSendKeysToPlugins
- 2. EmptyCommandLine + NotEmptyCommandLine
- 3. NoFilePanels + NoPluginPanels
- 4. NoFilePPanels + NoPluginPPanels

- 5. NoFolders + NoFiles
- 6. NoPFolders + NoPFiles
- 7. Selection + NoSelection (outside Viewer, Editor and Dialog)
- 8. PSelection + NoPSelection (outside Viewer, Editor and Dialog)
- 9. EVSelection + NoEVSelection (inside Viewer, Editor and Dialog)

e also:

General background, Using macros, Macro-language, Examples

xamples

in

Description of FAR plugins examples delivered with FAR Manager. Source files are located in %FAR%\PlugDoc\Examples\.

ditor Plugins:

Align

Performs block align in FAR editor.

A simple plug-in related to "Activated - working - done" category. It gives an example of using <u>EditorControl</u> service function.

Auto Wrap

Enables auto wrap in FAR editor.

This is an example of "hooking" the input stream in FAR Manager internal editor.

When enabled plugin monitors all keyboard and mouse events using ProcessEditorInput function:

```
int WINAPI _export ProcessEditorInput(const INPUT_R
{
    // KEY_EVENT processing code
}
```

Brackets

Searches and highlights the paired brackets in FAR editor. This is an example of a "Opened, made something, finished" plugin. Plugin uses <u>ECTL_GETSTRING</u> command to search brackets, <u>ECTL_SETPOSITION</u> command to place cursor on the paired bracket found and <u>ECTL_SELECT</u> command to highlight block.

DrawLine

Enables user to draw a pseudo graphics lines and tables in the FAR editor. The "hooking" version of plugin related to "Activated - working - done" category. This plugin uses an infinite cycle to process data using <u>ECTL_READINPUT</u> and <u>ECTL_PROCESSINPUT</u> commands:

```
HANDLE WINAPI _export OpenPlugin(int OpenFrom,INT_P
{
    ...
    while (!Done)
```

```
{
    Info.EditorControl(ECTL_READINPUT,&rec);
    ...
    Info.EditorControl(ECTL_PROCESSINPUT,&rec);
    ...
}
```

EditCase

Enables FAR editor to change the case of the selected block or a word under cursor.

This is an example of a "Opened, made something, finished" plugin. This plugin gets a string using <u>ECTL_GETSTRING</u> command then transforms it and puts it back into the document being edited using <u>ECTL_SETSTRING</u> command.

HlfViewer

HlfViewer adds to the FAR editor an ability to view HLF help files. HlfViewer plugin is supposed to be a useful tool for developers who create or modify HLF-files or for those who just want to view an HLF help file (processes F1 key and shows up HLF-file being edited in a window, processes "hlf:" prefix).

ther Plug-ins:

HelloWorld

"Hello, World!" is a simple plugin helping beginners to understand the common plugin structure and the way it interacts with FAR environment.

FAR Commands

FAR Commands plug-in (FARCmd) brings additional functionality to the user menu, file associations and command line usage. There are several samples of prefixes handling.

FileCase

This command enables user to change the file name case for selected files according to one of case of the case change rules specified. This plugin is quite simple so you can use it as a template to create new commands for FAR Manager.

MultiArc

This plugin extends FAR Manager to be able to read the contents of archives,

compress, extract and process files using <u>archives</u>. MultiArc represents archive and everything within as a directory structure transparent to user with some limitations though. FAR passes your commands to external archivers to process your requests. This plugin is quite complex. It supports second-level plugins so you can add support for new archive formats by writing small modules and without having to recompile the MultiArc itself.

Network

Using Network plugin you can browse the network and network shared resources, mount and dismount them as local drives. You can press F5 to map the chosen shared resource to the next available letters or F6 if you want to choose the letter yourself. F8 disconnects already mapped drive.

TmpPanel

Temporary panel adds to FAR Manager the ability to maintain virtual file lists similar to the file panels without directory structure. Temporary panels enable processing of multiple files from different folders. Up to 10 temporary panels with different file lists can be used.

ow to make a FAR plug-in using Visual C++ tep by step

in | articles

Phoenix aka Ruslan Ilgasov <u>?subject=Articles"></u> <u>phoenixbird@hotmail.ru</u>

It's difficult to find a man who doesn't know about or who doesn't use FAR -IMHO the best NC clone for Windows. It is really very good file-manager, moreover, there are a lot of plug-in modules for it. Plug-in module is a DLL file that uses FAR functions instead of using standard Windows functions in order to work with monitor, keyboard, etc. FAR supports all functions necessary for working in the text mode. You can install the plug-in module easily - just copy DLL file and data files to the folder under Far\Plugins and restart FAR.

FAR is distributed along with the full set of files required for writing a plug-in using any Windows-based C compiler. This article guides how to make a FAR plug-in using Visual C++ (I used Visual C++ 5.0). After FAR is installed, there's a PlugDoc.rar file in its folder that contains examples of plug-ins and header file (*note: beginning from FAR 1.70 beta 5 examples are installed into the separate PlugDoc folder*). All the examples are used in FAR. Also, the VCReadme . txt is included there, in which the details of Visual C++ workflow are described. You'll investigate examples later.

We'll write the plug-in that gets the list of open windows and can be used as a prototype for your own plug-ins. As a matter of fact, you have only to start, the things are not so complicated as you can think. So, let's go:

- 1. Start VC, and make a new project named SimpleFP from the "Win32 Dynamic-Link Library" template. Create simplefp.cpp file we'll actually write there. Copy plugin.hpp header file from the PlugDoc.rar archive to the SimpleFP folder. (*note: the samples are installed into the PlugDoc folder, beginning from FAR 1.70 beta 5*).
- 2. We should make a .def file now the file where the functions called from external modules are described. We should describe the FAR functions that we'll use in our module. So, make a simplefp.def text file, in which:

LIBRARY EXPORTS GetPluginInfo=_GetPluginInfo@4

```
OpenPlugin=_OpenPlugin@8
SetStartupInfo=_SetStartupInfo@4
```

Here we describe the 3 functions we'll need later. Now add the simpledef.def to the project files (Project - Add to project - Files - simplefp.def).

3. We're writing the plug-in now - working with simplefp.cpp file. I decided the source text to be commented, so you can copy it into C++ and begin to play with it. But let's begin from fundamentals.

Far works using the same principles as Windows uses - in your program you call any functions you want if they are already exist in the system. Far provides functions for handling form views in the console application mode. When starting a plug-in, FAR starts OpenPlugin function, we'll treat it as similar to main() or WinMain(). But we still have to pass our plug-in data to FAR. The GetPluginInfo function does that.

```
/*
* SimpleFP - a simple FAR plug-in. (C) 2000 Phoenix,
*/
#include <stdio.h> // for sprintf calling
#include <windows.h> // for Windows functions
#include "plugin.hpp" // for FAR functions
#define PLUGIN_NAME "Open windows" // Plug-in na
#define WINDOW HEAD "Open windows list" // Our menu t
11
// Here the FAR functions we are working with are des
11
extern "C"
{
void WINAPI __export SetStartupInfo(struct PluginStart
HANDLE WINAPI __export OpenPlugin(int OpenFrom, int Ite
void WINAPI __export GetPluginInfo(struct PluginInfo *
};
static struct PluginStartupInfo Info; // Our plug-in
```

```
11
// Module information is defined in the Info structur
11
void WINAPI __export SetStartupInfo(struct PluginStart
{
  ::Info=*Info;
}
// This function is called to get the plug-in informa
// We must fill the Info structure fields.
11
void WINAPI __export GetPluginInfo(struct PluginInfo *
{
  Info->StructSize=sizeof(*Info); // Info structure s
  Info->Flags=0; // It's useless for us
  Info->DiskMenuStringsNumber=0; // It's also useless
  // Determine a string with module name
  static char *PluginMenuStrings[1];
  PluginMenuStrings[0]= PLUGIN_NAME;
  // Determine a plug-in module name
  Info->PluginMenuStrings=PluginMenuStrings;
  Info->PluginMenuStringsNumber=
    sizeof(PluginMenuStrings)/sizeof(PluginMenuString
  Info->PluginConfigStringsNumber=0; // It's useless
}
// This function is called when starting the plug-in
11
HANDLE WINAPI __export OpenPlugin(int OpenFrom, int Ite
{
  HWND hwnd; // Use it to get the handle
  char p[128], o[128]; // Use it to create a menu str
  int i=0; // Counter
```

```
struct FarMenuItem MenuItems[64]; // Description of
memset(MenuItems, 0, sizeof(MenuItems)); // Initializ
MenuItems[0].Selected=TRUE;
hwnd = GetDesktopWindow(); // Get desktop handle
hwnd = GetWindow(hwnd, GW_CHILD); // Get its handle
while (hwnd !=0) // While it is not last
{
  hwnd = GetWindow(hwnd, GW_HWNDNEXT); // Get windc
  GetWindowText(hwnd, p, 128); // and its caption
  if (strlen(p)>0) // if caption exists
  {
    sprintf(o,"%0.8xld %s", hwnd, p); // create a s
    strcpy(MenuItems[i++].Text, o); // copy this st
  }
}
// Call the menu we created just now, get the selec
//
int MenuCode=Info.Menu(Info.ModuleNumber,
                 -1, -1, 0,
                FMENU_AUTOHIGHLIGHT | FMENU_WRAPMODE,
                WINDOW_HEAD,
                 NULL,
                "Menu content",
                 NULL,
                 NULL,
                MenuItems,
                i);
return(INVALID_HANDLE_VALUE);
```

Then, compile the project, copy to Far\Plugins folder, and restart FAR. When in FAR, press F11 - this is the list of plug-in modules. "Open windows" string must be there. Look at the result. You can now develop it, for example, process the MenuCode data, and then pass the WM_CLOSE message to the selected window, or do something more peculiar. Plug-in modules creation for FAR is

}

13.05.2000

to the top

CTL_GETSTRING works very slowly

in | articles

```
Stanislav V. Mekhanoshin <u>?subject=Articles"></u>
<u>rampitec@tu.spb.ru</u>
```

Let's suppose a plugin is to scan a large number of strings in the editor in sequence. In my case, the <u>Incremental Search</u> plugin searched for a substring in the editor. The first idea was to perform a sequential search for strings in this way:

```
{
    struct EditorGetString egs;
    struct EditorSetPosition esp;
    struct EditorInfo ei;
    Info.EditorControl(ECTL_GETINFO,&ei;);
    for( egs.StringNumber=ei.CurLine;
         eqs.StringNumber < ei.TotalLines;</pre>
         egs.StringNumber++ )
    {
        Info.EditorControl(ECTL_GETSTRING, ≥);
        if( process( egs.StringText, egs.StringLength
            esp.CurLine=egs.StringNumber;
            esp.CurPos=-1;
            esp.CurTabPos=-1;
            esp.TopScreenLine=-1;
            esp.LeftPos=-1;
            esp.OverType=-1;
            Info.EditorControl(ECTL_SETPOSITION, &esp;
            return TRUE; // Success, the string is se
        }
    return FALSE; // Fail, just return back. There ar
                  // no changes in the editor.
}
```

However, having code written in this way, I discovered that string processing code (essentially the *process()* function) worked considerably faster than the

whole iteration. In other words, the procedure that returned the string by its number took ~99% of time.

The code was rewritten according to the ER's advice (Andrew Tretyakov did the same in the <u>EditCompletion</u> plugin). Essentially, the advice is to obtain a current string (-1) always, without using its real number. In other words, to substitute <u>ECTL_GETSTRING</u> with the string number for twain <u>ECTL_SETPOSITION</u> with the string number and ECTL_GETSTRING with -1.

Need to mention that you must store the current cursor position in the editor and restore it when doing rollback in order to use this method. But the matter is worthy of it. So, you must rewrite the code mentioned above in this way:

```
{
    struct EditorGetString egs;
    struct EditorSetPosition esp;
    struct EditorInfo ei;
    Info.EditorControl(ECTL_GETINFO,&ei;);
    egs.StringNumber=-1;
    for( esp.CurLine=ei.CurLine;
         esp.CurLine<ei.TotalLines;</pre>
         esp.CurLine++ )
    {
        Info.EditorControl(ECTL_SETPOSITION,&esp;);
        Info.EditorControl(ECTL_GETSTRING, ≥);
        if( process( egs.StringText, egs.StringLength
            return TRUE; // Success, the string is se
    }
    // Restore the old position:
    esp.CurLine=ei.CurLine;
    esp.CurPos=ei.CurPos;
    esp.TopScreenLine=ei.TopScreenLine;
    esp.LeftPos=ei.LeftPos;
    esp.CurTabPos=-1;
    esp.OverType=-1;
    Info.EditorControl(ECTL_SETPOSITION,&esp;);
```

return FALSE;

}

{

By the way, FAR doesn't redraw changes immediately, so the screen won't flicker.

And the most pleasant: time metering performed on my computer showed that we get the string (only get, without processing - the raw time) 63 times faster in the second case than in the first case. The effect is stable for both relatively small files and files with size more than half of my RAM. Andrew Tretyakov has almost the same results - he has ratio of 1/65. In other words, the figures are rather close.

For metering, I used the Watcom C 11.0 run-time profiler based on the rdtsc Pentium profiling instruction. IMHO it's the best profiler. But speed-up is **highly** noticeable even without any tools. All tests were performed using IP-240, 96Mb RAM, Windows NT 4.0 SP6. Andrew Tretyakov used 486-dx4-100 for metering.

Minor warning: When setting the position, FAR may change the LeftPos, TopScreenLine, and CurPos values even if you set them to -1 already. For example, if the cursor can't move beyond the end of the line, but the line is shorter than CurPos you try to store by setting it to -1, then CurPos will change despite of that. Such behaviour is acceptable for most users. However, user doesn't see intermediate moves through the text when searching strings sequentially within the iteration mentioned above. He will be surprised seeing the position he doesn't expect (from his point of view) when moving from the 1st to the 10th string. Such problem can't appear in the first example since only one move is actually performed. But you should modify the second example in order not to face with problem like that.

There are many ways to modify it. For example, this problem gets eliminated if your plugin doesn't change the current position at all (i.e. always restores it). If the plugin computes TopScreenLine, LeftPos, or CurPos values according to the its own concept (perhaps not related to their previous state), it just calls this code after the iteration is finished. In my case I always restore the stored position, and then use the <u>ECTL_SETPOSITION</u> by passing the required string and -1 for other parameters there. Here's the example of the modified code:

```
struct EditorGetString egs;
struct EditorSetPosition esp;
struct EditorInfo ei;
```

```
nFound=-1; // number of the found string
int
Info.EditorControl(ECTL_GETINFO,&ei;);
egs.StringNumber=-1;
for( esp.CurLine=ei.CurLine;
     esp.CurLine < ei.TotalLines;</pre>
     esp.CurLine++ )
{
    Info.EditorControl(ECTL_SETPOSITION,&esp;);
    Info.EditorControl(ECTL_GETSTRING, \geq);
    if( process( egs.StringText, egs.StringLength
        nFound=esp.CurLine; // Success
        break;
    }
}
// Restore the old position:
esp.CurLine=ei.CurLine;
esp.CurPos=ei.CurPos;
esp.TopScreenLine=ei.TopScreenLine;
esp.LeftPos=ei.LeftPos;
esp.CurTabPos=-1;
esp.OverType=-1;
Info.EditorControl(ECTL_SETPOSITION,&esp;);
if( nFound \geq = 0 )
                // Now set again to the found pos
{
    esp.CurLine=nFound;
                           // Despite these fiel
    esp.CurPos=-1;
    esp.TopScreenLine=-1; // already, they must
                            // It's not the same!
    esp.LeftPos=-1;
    esp.CurTabPos=-1; // unconditional. -1
    esp.OverType=-1;
                           // the old value, if
    Info.EditorControl(ECTL_SETPOSITION, &esp;);
```

```
}
```



28.11.1999 Rev. 26.06.2000

to the top

dditional topics

<u>in</u>

- Additional structures and functions
- <u>Examples</u>
- <u>'Rules to set the right tone'</u>
- <u>Overview of plugin capabilities</u>
- Lyrical introduction to plugins
- FAR plugins API History
- How to setup the Encyclopedia
- <u>FAQ</u>
- <u>Authors</u>

ow to setup the Encyclopedia

in

icrosoft Visual C++ - MSDN

It is possible to incorporate the Encyclopedia into the MSDN library if you have one installed on your computer to be able to use Encyclopedia along with MSDN help. Serg Bormant (2:5027/12.80@fidonet) sent us a «GuideLine»; here it is (*with my remarks*, ruiv).

```
While looking through the encyclopedia (FarEncycl
topic about its setup.
. .
    Separate .chm and .chi are good only if you keep
while having index (.chi) set up. But it's meaningles
into account its small size. A "monolithic" .chm also
easy to link it.
    To incorporate it into the MSDN collection you mu
the collection definition (logical structure) on the
the physical file locations are kept in the hhcolreg.
    Here's the method of FarEncyclopedia.en.chm incor
correct :), but it works well...
    0. Input data...
Windows 2000 English, Visual Studio 6 SP3, MSDN Jan 2
Encyclopedia:
D:\Program Files\Far\PlugDoc\
  FarEncyclopedia.en.chm
Collection (this path you can find in the MSDN desktc
D:\Program Files\mvs\MSDN\2000JAN\1033\
  MSDN000.COL
Collection registration:
D:\Documents and Settings\All Users\Application Data\
  hhcolreg.dat
```

```
1. Do as following:
```

1.1. Unload MSDN Library...

1.2. Find all *.col in the MSDN folder (it's msdr (same for every other file we change). Then before Fc the following (added strings are marked with ">"):

```
< XML >
 <HTMLHelpCollection>
 <collectionnum value=10003/>
 . . .
 <Folders>
 <Folder>
   <TitleString value="MSDN Library - January 2000"/>
   <FolderOrder value=1/>
 </Folder>
>
><Folder>
> <TitleString value="Far PlugRinG Help Project"/>
><FolderOrder value=2/>
> <Folder>
     <TitleString value="=pluginsr"/>
>
     <FolderOrder value=1/>
>
     <LangId value=1033/>
>
   </Folder>
>
></Folder>
>
 </Folders>
 </HTMLHelpCollection>
 </XML>
```

Pay attention to the tags <collectionnum.../> (cc <FolderOrder.../> (number of folder within the level)

(You will need collectionnum when changing next f depends on the location where you've inserted the fra greater than the one specified in the previous Folder

1.3. Find collection registration file (there car

```
we need the one where Encyclopedia collection is defi
section), and add the file information to the <DocCom
(This file (hhcolreg.dat) was in the d:\winnt\help\ a
 <XML>
 <HTMLHelpDocInfo>
 . . .
 <Collections>
 . . .
 <Collection>
   <ColNum value=10003/>
   <ColName value="D:\Program Files\MVS\MSDN\2000JAN\
 </Collection>
 </Collections>
 <Locations>
 . . .
 </Locations>
 <DocCompilations>
 <DocCompilation>
 </DocCompilation>
><DocCompilation>
><DocCompId value="pluginsr"/>
   <DocCompLanguage value=1033/>
>
   <LocationHistorv>
>
     <ColNum value=10003/>
>
     <TitleLocation value="D:\Program Files\Far\PlugD
>
     <IndexLocation value="D:\Program Files\Far\PlugD</pre>
>
     <OueryLocation value=""/>
>
     <LocationRef value=""/>
>
>
     <Version value=0/>
     <LastPromptedVersion value=0/>
>
     <TitleSampleLocation value=""/>
>
     <TitleQueryLocation value=""/>
>
     <SupportsMerge value=0/>
>
   </LocationHistory>
>
></DocCompilation>
>
```

```
</DocCompilations>
</HTMLHelpDocInfo>
</XML>
```

The collection number here is <ColNum.../> (This noted while changing collection, see 1.2)

1.4. Load MSDN. You should see a new topic under wait until the new index is created (it's a long proc by typing any keyword, e.g. AF_AVPRESENT.

2. If you didn't succeed, then you made a mistake to the first step and repeat from the beginning :) Serg Bormant, 2:5027/12.80@fidonet, bormant@chat.ru, Yours faithfully, Serg Bormant --- GoldED+/W32 1.1.4.5 * Origin: To be, to be, two beer: without questions.

The only drawback of this method is that MSDN indexes should be rebuilt every time the Encyclopedia is changed - i.e. after a new version has been installed (it takes 3-5 minutes). But now you have an integrated FAR Manager Plug-in API help along with MSDN help system.

Note that incorporated Encyclopedia supports cross-links to the MSDN topics.

The latest amendment:

```
Date: Wed, 26 Sep 2001 21:40:15 +0400
From: <u>Dennis Trachuk <dennis.trachuk@nm.ru></u>
Subject: MSDN Integration
Hello All!
Article about HTML-help files integration into MSDN
```

http://codeproject.com/winhelp/msdnintegrator.asp
(local copy of this article is available)

and the best thing, a utility for doing that is avail <a href="http://codeproject.com/winhelp/MSDNIntegrator/MS

AR Manager

If you want to write your programs using FAR built-in editor, take advantage of the <u>"Active-Help"</u> plug-in. It extends the editor's capability to context-sensitive help invocation...

It works with HLP as well as with CHM files. Also, you can specify your MSDN collection (MSDNVS98.COL) path in plug-in settings and it will use MSDN collection!

yrical introduction to plugins

in

Tell me, what has the power? Money, you say. My brother thinks so as well. But actually... plugins hold the power.

Plugins might seem clumsy and overwhelming, but in fact, there's over a hundred plugins of all kinds for FAR Manager now - ranging from Tetris games to e-mail clients. This phenomenon is strange at first sight, but it is understandable - everyone wants what he needs and doesn't wait for someone to do that.

Plugin technology is definitely not new, but few other application can claim such an extensive use of plugins as FAR. The reason is, most likely, the fairly flexible structure of FAR itself, along with the simple approach of shell interaction with plugins. Essentially, this encyclopedia is dedicated to this approach. We won't praise, analyze or rave about the advantages of plugin interface. Just start study it and you will understand that in FAR you can do anything you want.

The main idea behind the concept of plugins is customizability. Install only those plugins that provide the functionality you want, and discard anything unneeded. That way, your tool will end up exactly the way you want it, without any unnecessary sag -- and that's the main advantage of FAR Manager over the integrated shells. Besides, it can always learn new tricks and do something that it has never done before.

So how do you take advantage of that? Alas, it's not entirely magic; you will need certain programming experience. However, from this point there's very few requirements; almost any programming language would do, given that you have a compiler capable of producing Windows DLLs. Best documentation and examples are available for C/C++, Delphi and assembly language; there are plugins known to be written in Modula, Ada and C--. In some cases you'll be faced with lots of extra details. You can read about them in <u>special topics</u>.

During initialization, FAR scans its **Plugins** folder and its respective subfolders, and considers all files with .DLL extension found there to be its plugins. Therefore, if your plugin uses its own DLLs, they must have a different extension. In fact, plugin is an ordinary library that runs in the console process environment, so it functions just like any console Win32 application. However,

due to the compatibility issues plugin shouldn't directly write text to the screen; <u>Text</u> function from <u>FAR Manager service</u> should be used instead.

e also:

<u>Overview of plugin capabilities, Exported functions, Service</u> <u>functions, Structures, Archive support</u>

verview of plugin capabilities

<u>in</u> | <u>Internet</u>

FAR Manager is so tightly integrated with its plugins that it is simply meaningless to talk about FAR and not to mention the plugins. Plugins present an almost limitless expansion of the features of FAR.

Without going into details, some of the capabilities can be noted:

- printers control, both connected to PC and network.
- syntax highlighting in program source texts.
- working with FTP-servers (including access through proxy, automatic download resume, etc.).
- search and replace in many files at the same time, using regular expressions.
- renaming groups of files with support for complex compound masks consisting of substitution symbols and templates.
- NNTP/SMTP/POP3/IMAP4 clients and sending messages to a pager.
- working with non-standard text screen resolutions.
- conversion of texts from one national code page to another.
- manipulating the contents of the Recycle Bin.
- Process priority control on local or network PC.
- Words autocomplete in editor and working with templates.
- Windows system registry editing.
- Creating and modifying Windows shortcuts.
- File and text operations making it more comfortable to use FidoNet.
- Files UU-encode and UU-decode.
- WinAmp control and MP3-tags modifying.
- Quake PAK-files processing.
- Connection and debugging of queries to ODBC-compatible databases.
- RAS service control.
- External programs executing (compilers, converters etc.) while editing text in FAR editor.
- Windows help files contents displaying (.hlp and .chm)
- Calculators with different possibilities.
- Several games :-)
- Spell checker functions while editing text in FAR editor.
- Removable drives catalog preparation and much more...

It is useless to list all the functions provided by FAR and its plugins, because this list is constantly growing. As an information source, which can be used to search for specific plugins, one can recommend:

- 1. PlugRinG site <u>http://plugring.farmanager.com</u>
- 2. Online forum <u>http://forum.farmanager.com</u>
- 3. USENET echo conference <u>news:fido7.far.support</u> (<u>at the Google Groups</u>) <u>news:fido7.far.development</u> (<u>at the Google Groups</u>)
- 4. FidoNet echo conferences far.support far.development
- 5. Mailing lists http://groups.google.com/group/fardeven http://groups.yahoo.com/group/plugringenglish http://groups.yahoo.com/group/farpluginsapi http://groups.yahoo.com/group/plugring_announce
- 6. Use the <u>PlugRinG viewer</u> plugin and you will be able to view and download all the new plugins directly from FAR.

lugin parameters...

in

..or "Professional ethics". Recommendations. Last revised n 30.04.2001

This document is an attempt of creating the FAR manager plugins specification (or, rather, recommendations). All materials are in plain text for now (will be divided into topics later).

This document is for programmers who write plugins for FAR manager.

You are encouraged to add, adjust, recommend, etc.

- 1. Plugin files MUST be packed into ZIP archive, since some servers treat RAR archives as text/plain, which results in the so-called "broken" archive.
- 2. The plugin archive MUST be supplied with a file_id.diz file, which should contain the plugin name, version, release date, short description of features (in English and Russian) and how to contact the developer. This file must be <u>in the root of the archive</u>.
- 3. The plugin archive should be supplied with a whatsnew.txt file (or history.txt), which should contain decription of changes (indicating version and date).
- 4. The plugin name should be the same in Plugin commands menu and in Plugins configuration menu:

If plugin adds "Search and replace" string to the menu, then the string with the same beginning must be in the Plugins configuration menu. Strings like "This is Search and replace configuration" or "Settings" are not suitable. In this case user will find the settings for your plugin much faster.

5. If you supply the plugin with REG-file macros, do not forget to fill the "Description" value.

6. The plugin should not add multiple strings to the menu. It's better to show your own menu with all needed items instead:

Do not add "Search", "Replace", "Search in highlighted" topics. Your plugin is just one among others, there are many other strings in the menu already. You'd better add a single "Search and replace" item, and then show a menu with "Search", "Replace" strings, etc.

- 7. The plugin MUST delete all temporary files or folders upon completion of its job.
- 8. If the plugin supports processing groups of files (using wildcards), the user should be able to specify these wildcards manually.

Example.

Trucer deletes trailing spaces from the file in the editor. It permits you to specify the exclusion mask. It's good. It would be bad if it didn't. Conclusion: if we don't want to handle all the files, we must provide the file mask support; and it is desirable to permit the user to adjust this setting.

9. If the plugin supports several actions for each of these files, you should provide the possibility to access all these actions from the commandline, user menu, and file associations.

Example.

"Shell link..." plugin (it's about version 1.20!) permits editing the link properties and calling the original file from the Plugins commands menu. It's bad. It would be better if Oscar (the author) added the commandline prefixes support. In that case it would be possible to make the *.LNK associations. For example, F4 - edit properties, Enter - go to the link target. This is more natural.

10. All messages the plugin shows on the screen (in dialogs or in menu), must

be in <u>LNG-files</u>. Alternatively, you can use the <u>LocMsg.cpp</u> localization module from the Encyclopedia bonus (it is recommended to use the LocMsg.cpp in the second-level plugins or in cases when it is necessary for your plugin to respond to changes in the message file without restarting FAR. It helps you to make the plugin language-independent.

- 11. The user should have a possibility to interrupt the process during timeconsuming operations. Unambiguously, the plugin MUST have this feature. Even such as Colorer. FAR should not crash because Colorer can't parse a 200KB string.
- 12. For help files, it is recommended:
 - to keep all the necessary information: "Why should I keep any readmes on my hard disk? I have them, but they're archived. Why should I search the archive when I need some additional info? IMHO, all that I need to work with plugin can be in HLF, but common information should be separated from advanced info, and links like "detailed", "advanced" should be available... So I don't have to close the plugin inappropriately and search for that separate text file. All that is "IMHO", of course."
 - At the main page
 - 1. show the plugin purpose and its version
 - 2. show links to other topics
 - It is desirable to have an "Alphabetical list" topic, where all other topics are listed.
 - There should be a link to the main page in any help topic.
 - It is necessary to check how any of the help pages appear in the 80x25 console window. Many people forget about that, and as a result only the author can see the help as he wanted it to be; others see messed-up strings. HlfViewer plugin can help in that case.
- 13. Plugin in any case should be visible while pressing F11 on any file. Also, the user shoud have a possibility to disable the <u>OpenFilePlugin</u> functionality in the plugin settings.

- 14. Some words about calling from commandline :-). It would be nice, if the prefix can be adjusted from the plugin configuration in order not to interfere with other plugins' prefixes.
- 15. Plugin may have no feature to change the menu items order, but if it has such feature, it should make use of Ctrl-Up/Down key combinations.
- 16. Try to use the <u>FMENU_WRAPMODE</u> flag while showing menus, otherwise they will be inconvenient for users.
- 17. About menus.
 - 1. If a menu item has an ellipsis ("...") a dialog box should appear
 - 2. If a menu item has a right-aligned ">" sign a submenu should appear
 - 3. In other cases, the selected command should be executed

Example:

Just now I wanted to print a text file, so I decided for the first time to do it from FAR using its Print Manager. I placed the cursor on the file, pressed F11 / Print Manager. In the menu I saw the "Print selected files" item, but I didn't know what would happen when I choose that item - immediate printing or some more settings invoked. The same is in the printers list. As a result it is extremely inconvenient.

18. If your plugin uses components (DLL, OCX, etc.), which are not a part of Windows, specify that in the "Installation" topic of your readme.txt.

It will be perfect if you test your plugin installation on plain, "clean" Windows 95 installed using minimal configuration.

19. When handling keyboard events in the editor, do not forget that the cursor movement keys (Left, Right, Up, Down, PgUp, PgDn, Home, End) are not the only keys used to position the cursor - mind the Ctrl-N, Ctrl-E and Ctrl-S key combinations.

20. If you supplement your plugin (not being of multi-purpose type) with macros to provide the fast and convenient operation, and these macros are usable only when editing C++ sources or only with text files, you shouldn't use the ordinary macros that are kept in the Windows Registry since they affect all editors and occupy the keys at the expense of files of other types. You may offer the macros for [ESC] plugin instead (here in brackets should be the author's name, direct reference to the plugin or to the author's home page, since the plugin and URL can change) or similar plugin (there are no such plugins at the moment). They are actually the same as FAR macros, but don't have their drawbacks (they affect only user-specified file types and aren't kept in the Windows Registry).

AR Plugins API History

in

The key moments of FAR Manager Plugins API history are noted here.

AR 1.75 01.04.2009

- + LIF_HIDDEN, LIF_GRAYED
- + MIF_HIDDEN, MIF_GRAYED
- + COL_MENUGRAYTEXT
- + COL_MENUSELECTEDGRAYTEXT
- + COL_DIALOGCOMBOGRAY
- + COL_DIALOGCOMBOSELECTEDGRAYTEXT
- + COL_DIALOGLISTGRAY
- + COL_DIALOGLISTSELECTEDGRAYTEXT
- + COL_WARNDIALOGCOMBOGRAY
- + COL_WARNDIALOGCOMBOSELECTEDGRAYTEXT
- + COL_WARNDIALOGLISTGRAY
- + COL_WARNDIALOGLISTSELECTEDGRAYTEXT

AR 1.71 26.11.2008

+ COL_EDITORSCROLLBAR

AR 1.71 05.11.2008

+ KEY_SLEEP

AR 1.71 24.09.2008

- * FLINK_SYMLINK renamed to FLINK_JUNCTION
- + FLINK_SYMLINKFILE
- + FLINK_SYMLINKDIR

AR 1.71 11.08.2008

+ COL_COMMANDLINEUSERSCREEN

AR 1.71 17.06.2008

+ VE_GOTFOCUS, VE_KILLFOCUS

AR 1.71 15.05.2008

- + FDIS_DELREMOVESBLOCKS
- + FDIS_MOUSECLICKOUTSIDECLOSESDIALOG

AR 1.71 30.03.2008

- + ECTL_ADDSTACKBOOKMARK
- + ECTL_CLEARSTACKBOOKMARKS
- + ECTL_DELETESTACKBOOKMARK
- + ECTL_GETSTACKBOOKMARKS
- + ECTL_NEXTSTACKBOOKMARK
- + ECTL_PREVSTACKBOOKMARK

AR 1.71 29.03.2008

+ PFLAGS_PANELLEFT

AR 1.71 17.03.2008

- + COL_DIALOGLISTARROWS,
- + COL_DIALOGLISTARROWSDISABLED,
- + COL_DIALOGLISTARROWSSELECTED,
- + COL_DIALOGCOMBOARROWS,
- + COL_DIALOGCOMBOARROWSDISABLED,
- + COL_DIALOGCOMBOARROWSSELECTED,
- + COL_WARNDIALOGLISTARROWS,
- + COL_WARNDIALOGLISTARROWSDISABLED,
- + COL_WARNDIALOGLISTARROWSSELECTED,
- + COL_WARNDIALOGCOMBOARROWS,
- + COL_WARNDIALOGCOMBOARROWSDISABLED,
- + COL_WARNDIALOGCOMBOARROWSSELECTED,
- + COL_MENUARROWS,
- + COL_MENUARROWSDISABLED,
- + COL_MENUARROWSSELECTED,

AR 1.71 05.01.2008

+ KEY_MSWHEEL_LEFT, KEY_MSWHEEL_RIGHT

AR 1.71 31.12.2007

+ ACTL_REDRAWALL

AR 1.71 23.12.2007

- + ProcessDialogEvent
- + FarDialogEvent
- + OpenDlgPluginData
- + PF_DIALOG
- + OPEN_DIALOG
- + DIALOG_EVENTS

AR 1.71 14.12.2007

+ MCMD_GETSTATE

AR 1.71 06.12.2007

- + EE_GOTFOCUS, EE_KILLFOCUS
- + FE_GOTFOCUS, FE_KILLFOCUS

AR 1.71 04.12.2007

+ DIF_NOAUTOCOMPLETE

AR 1.71 06.08.2007

- _FAR_USE_FARFINDDATA
- + _FAR_USE_WIN32_FIND_DATA

AR 1.71 20.02.2007

+ DIF_NOTCVTUSERCONTROL

AR 1.71 30.01.2007
+ FCTL_GETUSERSCREEN

AR 1.71 07.12.2006

+ VIEWER_SETMODEFLAGS_TYPES

AR 1.71 01.12.2006

- * ViewerMode.TypeWrap renamed to ViewerMode.WordWrap
- + VCTL_SETMODE
- + enum VIEWER_SETMODE_TYPES
- + struct ViewerSetMode

AR 1.70 29.03.2006

- ! restrictions to keys received by the ProcessKey() f
- + PKF_PREPROCESS
- + ACTL_GETSHORTWINDOWINFO
- + ECF_TAB1
- ! EOPT_EXPANDTABS -> EOPT_EXPANDALLTABS
- + EOPT_EXPANDONLYNEWTABS, EXPAND_TABS
- + FARINT64
- + ViewerAPI: ViewerInfo, ViewerMode, ViewerSelect, Vi ViewerControl, ProcessViewerEvent,
- * All enumerations in plugin.hpp are now named.
- + FARMACRO_KEY_EVENT
- + DM_GETSELECTION, DM_SETSELECTION
- + DN_LISTHOTKEY
- ! Now plugins receive keyboard events through Process when recording and when playing back macros. Previc events were received only when recording macros.
- * Removed some of the limitations on the keys sent to ProcessEditorInput.

```
Keys which are not sent to plugins: Ctrl-W, F11, Al Ctrl-Shift-Tab, Alt-Ins, Ctrl-Alt-Shift.
```

```
The following key combinations are sent if:
```

```
Alt-F5 - the PrintMan plugin is not installed;
```

- Alt-F11 the editor is modal;
- F6 switching to the viewer is disabled

- + DN DRAWDIALOGDONE
- + ACTL_GETPLUGINMAXREADDATA, ACTL_GETWCHARMODE
- + ACTL GETDIALOGSETTINGS
- ! FIS PERSISTENTBLOCKSINEDITCONTROLS -> FDIS PERSISTE
- ! FIS_HISTORYINDIALOGEDITCONTROLS -> FDIS_HISTORYI
- ! FIS_AUTOCOMPLETEININPUTLINES -> FDIS_AUTOCOMF
- + FRS SCANSYMLINK
- + FSS SCANSYMLINK
- ! Now GetReparsePointInfo returns an error for remote correct information about symbolic link contents ca such case.
- Garbage had been returned in PanelInfo.SelectedItem selected and the cursor was positioned on the "..."
- + MCMD_POSTMACROSTRING
- + ActlKeyMacro.Param
- + KSFLAGS_NOSENDKEYSTOPLUGINS
- + ESPT_SETWORDDIV, ESPT_GETWORDDIV
- + FMENU CHANGECONSOLETITLE
- ! DM_LISTSETMOUSEREACTION (behaviour changed)
- + LMRT *
- + FCTL_SETNUMERICSORT, FCTL_SETANOTHERNUMERICSORT, PF

AR 1.70 beta 5 09.04.2003

- + OPM OUICKVIEW
- + FCTL_GETPANELSHORTINFO, FCTL_GETANOTHERPANELSHORTIN
- + FIB NOAMPERSAND
- + ESPT LOCKMODE
- + ECTL_TURNOFFMARKINGBLOCK
- ! CONSOLE_* -> FAR_CONSOLE_*
- + MAXSIZE_SHORTCUTDATA
- + FCTL CHECKPANELSEXIST
- + LIF DELETEUSERDATA
- + EF_IMMEDIATERETURN, VF_IMMEDIATERETURN
- + FLINK DONOTUPDATEPANEL
- + ECTL DELETEBLOCK
- + FDLG_SMALLDIALOG, FDLG_SMALLDIALOG
- + ACTL_SETARRAYCOLOR, FarSetColors, FARCOLORFLAGS.FCL
- + FarListColors

MultiArc:

+ ArcInfo.Chapters

+ ArcItemInfo.Chapter

AR 1.70 beta 4 13.03.2002

- + EJECT_LOAD_MEDIA
- + FDLG_WARNING, FDLG_SMALLDIALOG
- + PFLAGS_*
- ! PanelInfo.Flags
- + KEY_MSWHEEL_UP, KEY_MSWHEEL_DOWN
- + DM_LIST*
- + VF_ENABLE_F6, VF_DISABLEHISTORY
- + EF_ENABLE_F6, EF_DISABLEHISTORY, EF_DELETEONCLOSE
- + FarList*
- + DIF_LISTWRAPMODE, DIF_LISTAUTOHIGHLIGHT, DIF_LISTNC DIF_SEPARATOR2
- + DM_GETCHECK, DM_SETCHECK, DM_SET3STATE, DM_SETITEMF DM_GETITEMPOSITION, DM_GETDROPDOWNOPENED, DM_SETDRC DM_SETHISTORY, DM_SETMOUSEEVENTNOTIFY
- + DM_LISTSETMOUSEREACTION
- + LINFO_*
- + BSTATE_*
- + ACTL_GETWINDOWCOUNT, ACTL_SETCURRENTWINDOW, ACTL_CC ACTL_GETFARHWND, ACTL_POSTKEYSEQUENCE
- + WTYPE_*
- + KeySequenceFlags.KSFLAGS_DISABLEOUTPUT
- + KeySequence
- ! Unquote
- ! ExpandEnvironmentStr
- + ECTL_GETBOOKMARKS
- + EditorBookMark
- + DN_LISTCHANGE, DN_MOUSECLICK, DN_DRAGGED, DN_RESIZE DN_MOUSEEVENT
- + LIFIND_EXACTMATCH
- + FMENU_USEEXT
- + MIF_*
- + FarMenuItemEx

- ! FarRecursiveSearch
- ! DI_RADIOBUTTON
- + ESPT_CHARTABLE, ESPT_SAVEFILEPOSITION
- ! FarCharTable
- + FCTL_GETCMDLINESELECTEDTEXT, FCTL_SETCMDLINESELECTI FCTL_GETCMDLINESELECTION
- + PluginPanelItem.CRC32
- + ConvertNameToReal
- + GetReparsePointInfo
- ! AddEndSlash

AR 1.70 beta 3 20.04.2001

- ! In plugin.hpp, the "const" modifier has been added unchangeable parameters of some exported and servic
- + DIF_VAREDIT "breaking of the 512-byte barrier" and DI_COMBOBOX controls.
- ! Changed the FarListItem structure (possibility of w data").
- + Added flag LIF_PTRDATA for working with "long dat
- ! Changed the values of LISTITEMFLAGS (LIF_*) the v LIF_SELECTED, LIF_CHECKED and LIF_SEPARATOR have be lower byte of the upper word, and LIF_DISABLE has t
- ! Changed field names in the FarDialogItemData struct
- + It is now possible to handle a double click event t DN_MOUSECLICK message (MouseEvent.dwEventFlags==DOL
- ! Significant revamping of the internal key codes (fa
- + ACTL_EJECTMEDIA
- + DM_GETTEXTPTR, DM_SETTEXTPTR, DM_SHOWITEM, DM_ADDH]

- ! Ctrl-Break can now be controlled. Earlier the dialc unconditionally, regardless of what the callback fu returned on the DM_CLOSE message.
- + DIF_HIDDEN, DIF_MANUALADDHISTORY
- + The DIF_SHOWAMPERSAND flag can be used for DI_SINGL DI_DOUBLEBOX items.
- + ACTL_KEYMACRO: struct ActlKeyMacro - the interface part for ACTL_k MCMD_LOADALL - load all macros from the registr memory MCMD_SAVEALL - save all macros from the registr memory
- ! Extended the syntax of the Message function
- + FSF.qsortex
- + EF_CREATENEW
- ! FSF.FarRecursiveSearch added parameter void *para
- + FAR knows about 4 predefined help topics in HLF fil Contents - the standard main topic;
 - Config the topic invoked by pressing Shift-F1 i configuration menu;
 - Editor the topic invoked by pressing Shift-F1 i of the editor;
- + FHELP_USECONTENTS
- * ACTL_GETFARVERSION returns the version number, and
- * ACTL_WAITKEY waits for any key if NULL or -1 is pas

Param.

- + ECTL_SETPARAM allows to set editor parameters:
 - Tab size (ESPT_TABSIZE)
 - Expand tabs to spaces (ESPT_EXPANDTABS)
 - Auto indent (ESPT_AUTOINDENT)
 - Cursor beyond end of line (ESPT_CURSORBEYOUNDEOL
 - The format of the current character code (ESPT_C

AR 1.70 beta 2 16.12.2000

- ! Changes in Param2 for messages DM_SETTEXT and DM_GE Now they use pointers to FarDialogItemData structur
- + 6 new colors are added for DISABLED items (in dialc messages). See Headers.c\farcolor.hpp (or Headers.r
- + DIF_3STATE
- ! The DIF_EDITEXPAND flag is no longer processed by C DI_FIXEDIT controls.
- + ACTL_GETCOLOR, ACTL_GETARRAYCOLOR
- ! The AddEndSlash function works with both types of s the existing trailing slash with the type of slashe more often.
- ! The version format changed the constant in plugir has the following format: HIWORD: = NNN - # build LOWORD: HIBYTE = 1 - version Hi LOBYTE = 70 - version Lo

AR 1.70 beta 1 20.11.2000

- + Dialog API 1.0
- ! New/changed header files:

farcolor.hpp - Color attributes indexes. farkeys.hpp - FAR manager internal key codes.

- + "Standard functions" for reducing plugins size the FarStandardFunctions structure.
- + GetMinFarVersion().
- + URL-activators and color attributes in HLF-files.

struct PluginStartupInfo:

- + AdvControl
- + InputBox
- + ShowHelp
- + DialogEx
- + SendDlgMessage
- + DefDlgProc
- + PF_FULLCMDLINE
- + FCTL_*SORTMODE, FCTL_*SORTORDER

struct KeyBarTitles:

- + CtrlShiftTitles
- + AltShiftTitles
- + CtrlAltTitles
- + EF_NONMODAL
- + ECTL_SETKEYBAR
- + DI_COMBOBOX, DI_LISTBOX, DI_USERCONTROL
- + DIF_EDITEXPAND, DIF_DROPDOWNLIST, DIF_USELASTHISTOF DIF_MASKEDIT, DIF_DISABLE
- + Input fields with fixed length can have an input ma
- + Structures: FarListItem, FarList.
- + Header files for Pascal/Delphi

AR 1.64 beta 24.05.2000

- + ECTL_SAVEFILE
- + ECTL_QUIT
- + struct EditorSaveFile

AR 1.63 b 20.08.1999

- + VF_NONMODAL
- + VF_DELETEONCLOSE

AR 1.62 23.05.1999

+ FCTL_GETCMDLINEPOS

AR 1.62 b2 10.05.1999

+ EE_CLOSE

AR 1.62 b 24.04.1999

The "Colorer" plugin "introduced" highlighting in the editor:

- + EE_REDRAW
- + ECTL_ADDCOLOR
- + ECTL_GETCOLOR
- + struct EditorColor
- + FCTL_SETCMDLINEPOS
- + FE_COMMAND

AR 1.61 24.02.1999

- + EE_READ
- + EE_SAVE

AR 1.60 13.10.1998

Rapid progress of the Editor API:

+ FCTL_SETPANELDIR

- + ECTL_*
- + E0PT_*
- + BTYPE_*
- + PF_DISABLEPANELS
- + PF_EDITOR
- + PF_VIEWER
- + OPEN_EDITOR
- + OPEN_VIEWER
- + struct EditorGetString
- + struct EditorSetString
- + struct EditorInfo
- + struct EditorSetPosition
- + struct EditorSelect
- + struct EditorConvertText
- + struct EditorConvertPos

struct PluginStartupInfo:

+ EditorControl

AR 1.52 26.06.1998

- + FCTL_SETUSERSCREEN
- + FCT_DETECT
- + SM_COMPRESSEDSIZE
- + SM_NUMLINKS
- + FE_BREAK
- + OPM_DESCR
- + struct CharTableSet

struct PanelInfo:

- + ShortNames
- + SortMode

struct PluginStartupInfo:

- + CharTable
- + Text

AR 1.50 (release version) 09.02.1998

- + PPIF_USERDATA
- + FCTL_SETVIEWMODE
- + FCTL_SETANOTHERVIEWMODE
- + FCTL_INSERTCMDLINE
- + OPIF_USEATTRHIGHLIGHTING

struct PluginPanelItem:

+ UserData

AR 1.50 beta 03.10.1997

- + DIF_HISTORY
- + FCTL_GETCMDLINE
- + FCTL_SETANOTHERSELECTION
- + FCTL_SETCMDLINE
- + FCTL_SETSELECTION
- + FE_CLOSE
- + FMSG_LEFTALIGN
- + OPEN_COMMANDLINE
- + OPEN_SHORTCUT
- + OPIF_EXTERNALDELETE
- + OPIF_EXTERNALGET
- + OPIF_EXTERNALMKDIR
- + OPIF_EXTERNALPUT
- + OPM_TOPLEVEL
- + PPIF_SELECTED

struct PluginStartupInfo:

- + Viewer
- + Editor
- + CmpName

struct PluginInfo:

+CommandPrefix

struct PanelMode:
 +StatusColumnTypes

```
+StatusColumnWidths
struct KeyBarTitles:
    +ShortcutData
MultiArc:
    Changed a member of the ArcInfo structure:
        - int AuthVer
        + DWORD Flags
Flags added...
    enum ARCINFO_FLAGS
    {
        AF_AVPRESENT=1,
        AF_IGNOREERRORS=2
    };
```

AR 1.40 beta 27.06.1997

First time the FAR Plugins API was mentioned: "External DLL modules (plugins) can be used to implement new FAR commands and emulate file systems..."

ER> "As a matter of fact I started thinking about p ER> asked me to create a version of FAR with Arvid ER> (thank you very much) a good name for it - FARv ER> want to develop such a version and so I decided ER> whomever was interested, the API that will be s ER> capability to FAR without my participation..."

The following items are available in this version:

• Exported functions:

ClosePlugin, Configure, DeleteFiles, ExitFAR, FreeFindData, FreeVirtualFindData, GetFiles, GetFindData, GetOpenPluginInfo, GetPluginInfo, GetVirtualFindData, MakeDirectory, OpenFilePlugin, OpenPlugin, ProcessEvent, ProcessHostFile, ProcessKey, PutFiles, SetDirectory, SetFindList, SetStartupInfo

• Service functions:

Control, Dialog, FreeDirList, GetDirList, GetMsg, GetPluginDirList, Menu, Message, RestoreScreen, SaveScreen

• Structures

FarDialogItem, FarMenuItem, InfoPanelLine, KeyBarTitles, OpenPluginInfo, PanelInfo, PanelMode, PanelRedrawInfo, PluginInfo, PluginPanelItem, PluginStartupInfo

• Language and help files

rogramming FAR plugins - Encyclopedia for evelopers - Long History

Affairs of bygone days, Inheritance of olden times.

"*In a land far, far away, long, long ago...*", or maybe not so far, in any case this is already history... Enjoy :-)

arch 29 2006

• First release of the english Encyclopedia translation.

requently Asked Questions

in

eneral questions

- 1. <u>How can I hide the cursor?</u>
- 2. <u>How can I put it to the required position?</u>
- 3. <u>How should I set the flags for plugin to work in the editor as well as in the panels: Info->Flags=????</u>;
- 4. Is dynamic plugin connection/disconnection possible without FAR restart?
- 5. <u>Are plugins in the editor operable after activation only? Can I make a plugin work in the background mode?</u>
- 6. Can I merge several plugins (or functions) into one DLL module?...
- 7. <u>...and then recognize the user's choice using the Item parameter of</u> <u>OpenPlugin function? Am I right?</u>
- 8. <u>Can I make a plugin that works in the background mode using the current API?</u>
- 9. <u>I have a file that can be processed by a plugin. How can I find that out?...</u>
- 10. How can I get the full plugin module path from within it?
- 11. Can I switch FAR background screens from the plugin?
- 12. <u>How does FAR determine that the block in the Clipboard is a vertical block?</u>
- 13. Is it possible to recognize where the cursor is set on the current panel?...
- 14. <u>Can I redraw the window caption while redrawing any panel?...</u>
- 15. <u>Where does FAR get the procedures from for file copying/deletion/...?</u>
- 16. How can I get the name of the folder from which FAR was started?
- 17. <u>How can I reduce the size of the DLL?...</u>
- 18. <u>There are dupes in the Clipboard, why should FAR put there the same stuff</u> <u>several times?</u>
- 19. How can I determine what is the "symbolic link"?...
- 20. <u>...what '\\?\Volume{...' hides inside?</u>
- 21. Can I get some more useful info about symbolic links?
- 22. Can I get the changed/unchanged flags for the file being edited?...
- 23. <u>How can I get the name of the file loaded in viewer?</u>
- 24. Is it possible in FSF.ProcessName to get TRUE, when comparing...
- 25. <u>How does Windows dump the file name from the Explorer to the console</u> <u>window?</u>
- 26. ProcessKey doesn't work for some reason...
- 27. <u>How can a plugin position the cursor on a specific file?</u>

28. <u>How can a plugin run a program? And show its output under the panels?</u>

ialogs

- 1. <u>Can I somehow overcome the edit window limitation, where the macro</u> <u>sequence is limited to 512 characters?</u>
- 2. If I don't use the Dialog function to work with dialog, how can I handle the data input? Should I write that myself?
- 3. If I use the Dialog function and an input line within it, will the <u>ProcessEditorInput and ProcessEditorEvent functions work for this control</u> <u>element?</u>
- 4. Can I change elements like Static (their captions) dynamically in a dialog?
- 5. The set of control elements handled by FAR isn't sufficient for me...
- 6. Why the ListBox and ComboBox are so crude?...
- 7. <u>Why the buttons for closing the window don't work in the "new-style"</u> <u>dialog?</u>
- 8. <u>Should I close the dialog myself from now on?</u>
- 9. If I use the mouse, should I catch the DN_MOUSECLICK?
- **10.** <u>Should I handle the DefaultButton myself as well?</u>
- 11. In no way the SEPARATOR can reach the frame right edge...
- 12. <u>Can I somehow disable the automatic selection of elements using mouse in lists?...</u>

eneral questions

1. **Q: How can I hide the cursor? A:** Win32 console API:

SetConsoleCursorInfo forget to bring it back. , but don't

2. **Q:** How can I put it to the required position?

A: There: <u>SetConsoleCursorPosition</u>

- 3. Q: How should I set the flags for plugin to work in the editor as well as in the panels: Info->Flags=????;
 A: <u>PF_EDITOR</u>
- 4. **Q:** Is dynamic plugin connection/disconnection possible without FAR restart?

A: Disconnection is impossible. Why do you want to disconnect it? Connections are handled by FAR itself. You can still ask FAR not to cache the configuration of that plugin, but to load it to memory every time FAR starts by specifying the <u>PF_PRELOAD</u> flag. But it will be better for you not to do that since it leads to delays and extra memory consumption.

5. **Q:** Are plugins in the editor operable after activation only? Can I make a plugin work in the background mode?

A: Yes, you can. Export the <u>ProcessEditorInput</u>, and it will intercept all keyboard events in the editor.

6. **Q:** Can I merge several plugins (or functions) into one DLL module? And call each of them as if it was in a separate DLL-file (even if I load that DLL every time)...

A: Yes, you can. In the <u>PluginInfo</u> you can specify several strings that will be added to the Plugins menu. But it violates the rules of <u>professional</u> <u>ethics</u>.

- 7. Q: ...and then recognize the user's choice using the Item parameter of OpenPlugin function? Am I right?
 A: You're right.
- 8. **Q:** Can I make a plugin that works in the background mode using the current API?

A: In the editor only. There's no support for doing that in panels.

9. **Q:** I have a file that can be processed by a plugin. How can I find that out? I think it would be better to send the cd command there and get the answer - was it able to do that or not.

A: You can't do this since there's no API for plugin interaction in FAR. A simple check for file belonging to a plugin is not sufficient.

- Q: How can I get the full plugin module path from within it?
 A: FAR passes the pointer in the <u>PluginStartupInfo</u> structure to the <u>SetStartupInfo</u> function. Full module path is in the ModuleName field of that structure.
- 11. Q: Can I switch FAR background screens from the plugin?A: You can do it beginning from the FAR Manager 1.70 beta 4.
- 12. **Q:** How does FAR determine that the block in the Clipboard is a vertical block?

A: Vertical block has "FAR_VerticalBlock" Clipboard format.

13. Q: Is it possible to recognize where the cursor is set on the current panel? In particular, at the folder or at the file...A:

Control(FCTL_GETANOTHERPANELINFO); Control(FCTL_GETPANELINFO); PanelInfo.PanelItems[PanelInfo.CurrentItem].FindD;

14. **Q:** Can I redraw the window caption while redrawing <u>any</u> panel? I think it would be easier and faster than calling FCTL_REDRAWPANEL.

A: It won't necessarily be faster. If the caption was not changed by the plugin (it's usual practice), there's no need to redraw it.

15. **Q:** Where does FAR get the procedures from for file copying/deletion/...?

A: It gets them from Win32 API.

Copying: CreateFile

+<u>ReadFile</u>

+<u>WriteFile</u>

+<u>CloseHandle</u>

or <u>CopyFile</u>

(Ex), depending on the operating system and the "Use system copy routine" option.

Deletion: DeleteFile

, depending on the

<u>SHFileOperation</u> "Delete to Recycle Bin" option.

16. **Q: How can I get the name of the folder from which FAR was started?** A:

char lpName[_MAX_PATH], lpFullPath[_MAX_PATH]; LPTSTR lpFile; GetModuleFileName

(NULL,lpName,sizeof(lpName));
GetFullPathName

```
(lpName,sizeof(lpFullPath),lpFullPath,&lpFile);
*lpFile='\0';
```

or

17. **Q:** How can I reduce the size of the DLL - the module is so heavy? A:

- 1. There are some notes regarding that in the "<u>Articles</u>" topic.
- The writers of plugins who use Visual C++ and want to reduce the plugin module size are strongly advised to read this: <u>http://msdn.microsoft.com/msdnmag/issues/01/01/hood/default.aspx</u> (local copy is <u>here</u>)

18. **Q:** There are dupes in the Clipboard, why should FAR put there the same stuff several times?

A: If the text in the Clipboard is in any particular format, but the program wants another one, Windows does the conversion itself, and sometimes does this incorrectly. The pseudographics gets corrupted in this case, russian symbols are converted into '?' sometimes. The bugs depend on the Windows version and initial/final formats combination. If all the formats are in the Clipboard together, Windows finds the necessary one and uses it without any conversions.

19. **Q:** How can I determine what is the "symbolic link" - a simple directory link or a mounted volume?

A: The main rule: symbolic links on Win2K are **FOR DIRECTORIES ONLY!** So, we need to know - "*Is THIS a directory*?". We can use FA_DIREC (or FILE_ATTRIBUTE_DIRECTORY) file attribute to do that.

Let's continue.

We know for sure that any symbolic link (or reparse point) in Win2K has a FILE_ATTRIBUTE_REPARSE_POINT attribute - so let's check it. Call

FSF.GetReparsePointInfo(FullFolderName, DestName, s:

This function returns real "DestName" for specified "FullFolderName", specifically:

- 1. "\??\D:\Junc..." directory junction.
- 2. "\\?\Volume{..." mounted volume.

Moreover, the first 4 symbols are irrelevant for us! Therefore it is enough to

check 7 symbols beginning from the 4th:

```
if(!strncmp(JuncName+4,"Volume{",7))
{
    // obviously, this is a mounted volume!
}
else
{
    // an ordinary directory junction.
}
```

20. **Q:** But it's interesting what '\\?\Volume{...' hides inside?

A: It's easy as a pie. Call the

```
FSF.GetPathRoot(JuncName, Root);
```

function that returns the real root directory in one of two forms:

1. "D:\"
2. "\\?\Volume{..."

The second case is turbid :-) - apparently at the system level (from the disk manager) someone has deleted the letter assigned to this disk... One might say, in order not to see it ;-)

21. **Q: Can I get some more useful info about symbolic links?**

A: Of course you can :-) Really, not entirely about symbolic links, but... We can, for example, get a portion of information about the mounted volumes. It's simple - call the standard <u>GetVolumeInformation()</u>

function.

In other words, we know the root already -

"GetPathRoot (JuncName, Root);", it remained only to get the information (e.g., file system extended attributes support - compression, encryption, and file system type):

A typical function to check the ability of hard link creation looks like that:

```
BOOL CanCreateHardLinks(char *TargetFile, char *Hai
{
  char RootTarget[NM], RootHardLink[NM], FSysName[NN
  GetPathRoot(TargetFile,RootTarget);
  GetPathRoot(HardLinkName,RootHardLink);
  if(!strcmp(RootTarget,RootHardLink)) // the same
  {
    // NTFS drive?
    DWORD FileSystemFlags;
    if(GetVolumeInformation(RootTarget, NULL, 0, NULI
                             FSysName, sizeof(FSysName)
    {
      if(!strcmp(FSysName, "NTFS"))
        return TRUE;
    }
  }
  return FALSE;
}
```

22. **Q:** Can I get the changed/unchanged flags for the file being edited? I didn't find that, but FAR knows about that and displays an asterisk in the first line.

A: See the <u>EditorInfo</u> structure description, specifically the EditorInfo.CurState field values.

23. **Q: How can I get the name of the file loaded in viewer?**

A: This code gets the current file name in the viewer:

WindowInfo wi; wi.Pos=-1; Info.AdvControl(Info.ModuleNumber,ACTL_GETWINDOWI

That's all. File name is in wi.Name.

24. **Q:** Is it possible in FSF.ProcessName to get TRUE, when comparing "OUTBOUND\\???????.MO?" mask and "C:\\FILES\\OUTBOUND\\0000ee2c.mod" file, but to get FALSE with "C:\\MUSIC\\assol_1.mod"? PN_SKIPPATH is a wrong approach. When "\\" is in the mask it won't ever return TRUE. A: Compare with the "*\\OUTBOUND\\???????.MO?" mask without using the <u>PN_SKIPPATH</u>.

25. **Q: How does Windows dump the file name from the Explorer to the console window?**

A: Vasily Titsky: "...Briefly, when inserting text into the console application (through the system menu or through the link drag-and-drop) the kind OS does the following: if the next symbol code is not within the current application Keyboard Layout (current language is English, but we need to insert a Russian symbol; or vice versa), system "emulates" the input of this symbol through the Alt+digits. For example, when inserting Russian 'A', the following will be generated: press Alt, press '1', release '1', press '2', release '2', press '8', release '8', release Alt. When inserting symbols with codes below 99 (? - I didn't check this) the pressing of two digits is generated. We only have to detect it correctly and then handle it..."

26. **Q: ProcessKey doesn't work for some reason...** I write in Delphi:

```
function ProcessKey(hPlugin: THandle; Key: Integer
begin
windows.Beep(300,200);
//return False in order to be processed by FAR :
result:=0;
end;
```

Any key pressing should be followed by a beep - but there are no beeps. The same is for any particular key.

A: FAR calls the **ProcessKey** function for the active plugin panel only.

27. **Q:** How can a plugin position the cursor on a specific file? A:

```
{
  struct PanelInfo PInfo;
 Info.Control(INVALID_HANDLE_VALUE, FCTL_GETPANEL]
 // set cursor position on the selectItem panel
  struct PanelRedrawInfo PRI;
  char Name[NM], Dir[NM*5];
 int pathlen;
  strcpy(Name, Info.FSF->PointToName(selectItem));
 pathlen=Info.FSF->PointToName(selectItem)-select
  if(pathlen)
    memcpy(Dir,selectItem,pathlen);
 Dir[pathlen]=0;
 Info.FSF->Trim(Name);
 Info.FSF->Trim(Dir);
  Info.FSF->Unguote(Name);
 Info.FSF->Unguote(Dir);
  if(*Dir)
    Info.Control(INVALID HANDLE VALUE,FCTL SETPANI
 Info.Control(INVALID_HANDLE_VALUE, FCTL_GETPANEL]
```

```
PRI.CurrentItem=PInfo.CurrentItem;
PRI.TopPanelItem=PInfo.TopPanelItem;
for(int J=0; J < PInfo.ItemsNumber; J++)
{
    if(!Info.FSF->LStricmp(Name,Info.FSF->PointTol
      {
      PRI.CurrentItem=J;
      PRI.TopPanelItem=J;
      break;
    }
    Info.Control(INVALID_HANDLE_VALUE,FCTL_REDRAWPAN)
}
```

28. Q: How can a plugin run a program? And show its output under the panels? A:

```
Info.Control(INVALID_HANDLE_VALUE,FCTL_GETUSERSCRF
if (CreateProcess(NULL,"ls.exe",NULL,NULL,TRUE,0,F
{
   WaitForSingleObject( pi.hProcess, INFINITE );
   CloseHandle( pi.hProcess );
   CloseHandle( pi.hThread );
}
```

Info.Control(INVALID_HANDLE_VALUE, FCTL_SETUSERSCRI

<u>To the top</u>

ialogs

1. **Q:** Can I somehow overcome the edit window limitation, where the macro sequence is limited to <u>512 characters</u>?

A: It's the FAR dialog property. In the <u>FarDialogItem</u> structure the Data field size is 512 characters. (the answer is relevant for FAR Manager 1.70 beta 2 and earlier; see the <u>DIF_VAREDIT</u> flag description)

2. **Q:** If I don't use the Dialog function to work with dialog, how can I handle the data input? Should I write that myself?

A: You can write that yourself :-) But you can also use the <u>DialogAPI</u>...

3. **Q:** If I use the Dialog function and an input line within it, will the **ProcessEditorInput and ProcessEditorEvent** functions work for this control element?

A: No, they won't. Use the <u>DialogEx</u> function instead.

4. **Q:** Can I change elements like Static (their captions) dynamically in a dialog?

A: It depends on the dialog type used. After the dialog creation and <u>Dialog</u> function call, you can't. But you can close the dialog, change its caption and recreate the dialog. If you do all that in a sequence it will be rather fast. If you use the <u>DialogEx</u> function, you can send the <u>DM_SETTEXT</u> message to the DialogAPI from the function that handles the dialog.

5. **Q:** OK, I use the DialogAPI, but the set of control elements handled by FAR isn't sufficient for me.

A: Use the <u>DI_USERCONTROL</u> element. It handles the element drawing and controls it - the plugin can do all the stuff!

6. **Q:** Why the ListBox and ComboBox are so crude? I can't deal with them in a usual way - I have to extricate from this problem in order to add or delete them.

A: What do you want from the very first version of the DialogAPI?

7. **Q:** Why the buttons for closing the window don't work in the "newstyle" dialog?

A: "Old-style" ones don't have a handler. It's no wonder that their behavior should be the same as in FAR 1.65 and earlier.

"New-style" ones use a handler. You should handle the dialog closing yourself, except for: Ctrl-Break - always closes, Ctrl-Enter - closes if there

exists at least one button with field DefaultButton=1, then a <u>Close request</u> with Param1 = current element number, Esc and F10 can be handled from the handler - should the dialog close or not.

- Q: Should I close the dialog myself from now on?
 A: If you use a dialog handler YES, you should: catch the DN_BTNCLICK event, check Param1 for match with required number and send the DM_CLOSE message to the dialog core.
- 9. Q: If I use the mouse, should I catch the DN_MOUSECLICK?A: It's to your discretion :-) First, a mouse message comes (ignore it), then the DN_BTNCLICK comes right after it...
- 10. Q: Should I handle the DefaultButton myself as well?
 A: No, you don't have to. For DefaultButton, the sequence is like that:
 [DN MOUSECLICK ->] DN BTNCLICK -> DM CLOSE.
- 11. **Q:** Somehow it turned out that in no way the SEPARATOR could reach the frame right edge, but it got recovered when I rewrote everything from scratch.

A: See the comments for the <u>DIF_SEPARATOR</u> flag.

12. **Q:** Can I somehow disable the automatic selection of elements using mouse in lists (when mouse hovering over the list selects the element, particularly if this list is not active)?

A: Partially, you can. See the DM_LISTSETMOUSEREACTION

<u>To the top</u>

uthors

in

"**Programming FAR plugins - Encyclopedia for Developer**" is a collective creation by a group of authors (and co-authors) - free translation (with additions) of the original **plugins.hlp**

This is a list of all authors and co-authors of this work. All who took part in the creation of this work - by word or by deed,... (except authors from the «<u>Articles</u>» section)

The authors want to thank <u>Eugene Roshal</u> for the support provided in the creation of this work.

uthors of the english version

Valentin Skirdin

vskirdin@mail.ru

«Programming FAR plugins - Encyclopedia for Developers» project coordinator - compilation and release.

Alex Yaroslavsky

Alexey Yatsenko

«translator»

Max Belugin

belugin@mail.ru «translator»

Alexander Kornienko alexfh@mail.ru «translator»

Eugene Mindrov

reaper_man@mail.ru «translator»

WARP ItSelf

WARP_ItSelf@inbox.ru «translator»

Roman Vorobets gembox@yandex.ru «translator»

Alexander Nesterovsky nsky@bigfoot.com «translator»

Max Gorobchuk maxgorobchuk@mail.ru «translator»

Andrey Tsybin andrew.sloven@gmail.com «translator» and «proofreading»

Roman Synyshyn

synyshyn@univ.kiev.ua «translator»

Peter Koves

kovesp@sympatico.ca «proofreading»

Oliver Schneider Borbarad@gmxpro.net «proofreading»

Wesha the Leopard

weshasmail@cmtk.net «proofreading»

Pawel Pawlak morris@elysium.pl

«proofreading»

uthors of the original russian version

Valentin Skirdin

vskirdin@mail.ru http://www.farmanager.com/skirda

«Programming FAR plugins - Encyclopedia for Developers» project coordinator - compilation and release.

Igor Ruskih

cail@nm.ru http://cail.nm.ru/ co-coordinator / design, scripts, adaptation...

Ivan Sintyurin

spinoza@mail.ru http://www.moris.ru/~spinoza

«Head translator» - translation of articles from english to russian...

Kirill Kirichenko

nectokris@mail.ru

«translator» - translation of articles from english to russian...

Artyom Nazarov

<u>">tema@mail.ru</u> «DrWeb» - proofreading, etc...

Vasily Moshninov

vmoshninov@newmail.ru http://proxykit.narod.ru The <u>Delphi</u> section, remarks, etc...

Dmitry Jemerov

<u>">yole@yole.ru</u> *«Purify»* - proofreading, etc...

inks

Far PlugRinG site: PlugRinG http://plugring.farmanager.com/ Far official site: http://www.farmanager.com http://www.rarlab.com



AR Manager on the Internet

- in
- <u>http://www.farmanager.com</u> Official site of the FAR Manager.
- <u>http://www.rarlab.com</u> Official site of WinRAR and FAR Manager support.
- <u>http://www.farmanager.com/mantis</u> Registration and handling of problems and wishes related to Far Manager and its standard plug-ins.
- <u>http://api.farmanager.com</u>
 Online version of "Programming FAR plugins Encyclopedia for Developers".
- <u>http://forum.farmanager.com</u> Official online forum for FAR manager users.
- http://plugring.farmanager.com Official site of the FAR Manager user support team, the main site of the Far PlugRinG web-ring - joint community of plugin developers.
- http://farmanager.com/svn Official SVN repository for source code of Far Manager and its standard plug-ins.
- Mailing lists (from 2000 to 2007):
 - <u>http://groups.yahoo.com/group/plugringenglish</u>
 Mailing list: Discussion of FAR Manager plugins takes place in the "PlugRinG Forum" mailing list.
 - http://groups.yahoo.com/group/farpluginsapi
 Mailing list: The "FAR Manager Plugins API" mailing list is intended for FAR Manager plugin developers to discuss questions that may arise while developing plugins and other topics that concerning the "FAR Manager Plugins API" (including discussion of the "Programming FAR plugins - Encyclopedia for Developers" help

topics).

- <u>http://groups.yahoo.com/group/plugring_announce</u> Mailing list: Announcements of new additions to the PlugRinG site.
- Mailing lists (from 2007):
 - <u>http://groups.google.com/group/fardeven</u>
 "Far Manager Development" mailing list is intended for discussion on any issues related to development of Far Manager and its plugins.
 - <u>http://groups.google.com/group/farcommits</u>
 "SVN Official Patches Monitoring" mailing list is intended for getting information about the changes in official source code repository.
- http://farplugins.sourceforge.net/wiki Far Wiki - a place where miscellaneous information about Far Manager and its plugins is collected. A sort of an open encyclopedia!
- <u>news:fido7.far.support</u> USENET FAR.SUPPORT echo conference (see also <u>in Google Groups</u>)
- <u>news:fido7.far.development</u> USENET FAR.DEVELOPMENT echo conference (see also <u>in Google</u> <u>Groups</u>)



ictionary

in

Some materials about the file system properties have been taken from <u>http://www.windowsfaq.ru/</u>. All links to MSDN which are mentioned in the *Dictionary* will be opened in a separate window as search results within <u>msdn.microsoft.com</u>.

lugin

Software component-addon to the FAR Manager, which permits the implementation of additional functions. In fact, plugin is an ordinary library that runs in the console process environment, so it functions like an ordinary console Win32 application. The plugin concept itself allows you to tune the shell as you want by adding necessary and discarding unnecessary stuff.

eparse Points

Most innovations in the Windows 2000 file system become possible due to the introduction of the Reparse Points concept, which allows attaching of the additional data storage subsystems without using the additional programs.

Reparse points are actually file system objects with special attributes that allow using the extended functionality of data storage subsystem. Any file or directory can contain a reparse point. It means that several kinds of extended functionality are available at once when accessing the resource at the same path.

See also: MSDN: Reparse Points

irectory Junctions

Directory Junctions allow you to map any local folder to any other local folder. For example, if you have three folders, C:\folder1, C:\folder2 and C:\documents, you can create directory junctions in such a way that C:\documents will look like a subfolder of two other folders, i.e. folders C:\folder1\documents and C:\folder2\documents will exist.

It was supposed initially that a special utility called linkd.exe will be supplied for creation of directory junctions, but it isn't included in Win2000 and supplied as a part of the Resource Kit. Also, directory junctions can be created using API, but this requires writing own software.
At first sight, Directory Junctions and Distributed File System implement the same functions, since they both can make multiple distributed folders look like a single folder tree. But there are some essential differences between them:

- DFS uses the Active Directory service for storing its data
- Due to the Active Directory using, DFS can provide fault protection and system load balancing, while directory junctions can't, although it isn't necessary when using a local computer
- DFS is intended mainly to the integration of the network resources into the common namespace, while directory junctions join the local resources only
- DFS can operate on several file systems, while directory junctions are based on NTFS 5.0 only
- DFS requires a client application, while directory junctions don't

See also: <u>MSDN: Directory Junctions</u>, <u>MSDN: Inside Win2K NTFS</u>, Part 1.

ount Points

Mount Points are essentially the same as the Directory Junctions, but they only allow mapping of the root folder of one volume to a local folder of another volume. Mount points are created with the help of reparse points and therefore need NTFS 5.0.

Mount Points are useful for increasing the volume size without changing the actual structure of the volumes on the disk. For example, with a volume mount point set the user might see drive D as "C:\Documents" as well as "D:", and the size of drive C will seem to increase.

See also: MSDN: Volume Mount Points

ard Links

Hard Link is the term used when a file has more than one name. For example, a file has names 1.txt and 2.txt. If the user deletes 1.txt, 2.txt remains, and vice versa. In other words, once the file has a new name, the names are equivalent. A file actually is deleted when its last hard link is deleted. Hard links cannot be cross volume boundaries, this means you cannot hardlink, for example, D:\somefile.txt to C:\somefilelink.txt. Keep in mind, that having the same drive letter in the path, files may reside on different volumes (see Mount

Points). You can create a hard link only if the underlying filesystem supports it. See also: <u>MSDN: Hard Links, Q106166 - Windows NT Backup and Hard Links</u>.

ymbolic Links

Much more efficient feature that allows virtual folders creating - just as virtual disks created using SUBST command in DOS. It has a wide range of uses - folder structure simplification, for example. If you don't like the name "Documents and settings\Administrator\Documents", you can link it to the root folder, so the system will still handle the real name of this folder while you are working with much more convenient, shorter name, fully equivalent to the real one.

Keep in mind, that symbolic links created under Windows Vista will not be accessible under previous versions of Windows.

See also: <u>MSDN: Symbolic Links</u>, <u>Windows 2000 Magazine: Inside Win2K</u> <u>NTFS, Part 1</u>.

parse File

NTFS 5 supports sparse files, in which much of the data is zeros. Such file can be marked as sparse file. The system does not allocate hard drive space to a sparse file except in regions where it contains nonzero data. It keeps the information about location of nonzero values only. This method allows optimal data allocation on NTFS volumes when storing sparse files and handling them by applications.

See also: <u>MSDN: Sparse Files</u>, <u>NTFS possibilities</u>.

umeric sort

The sorting algorithm which is used by the operating system to sort file lists was changed in Windows XP. A numeric sort is used instead of string sort. FAR also allows using numeric sort as in Windows XP - in other words, leading zeros in a file name are ignored. The following example shows how the files are sorted:

Numeric sort is on (Windows XP)	Numeric sort is off (Windows 2000)
Ie4_01	Ie4_01
Ie4_128	Ie4_128
Ie5	Ie401sp2
Ie6	Ie5
Ie401sp2	Ie501sp2
Ie501sp2	Ie6
5.txt	11.txt
11.txt	5.txt
88.txt	88.txt

See also: <u>Q319827 - The Sort Order for Files and Folders Whose Names...</u>



rticles

in

This section contains articles and notes on programming and similar topics. We hope that the following articles will be helpful to you in the process of writing Far Manager plugins.

- <u>How to make a FAR plug-in using Visual C++</u> Ruslan Ilgasov. 13.05.2000
- <u>ECTL_GETSTRING works very slowly...</u> Stanislav V. Mekhanoshin. 28.11.1999

If you have some non secret "little secrets" or while working on a plugin you have come across some peculiarities, "hidden features" and have overcame those obstacles then tell us about them, write it in a form of a small article and we will happily publish your material in this Encyclopedia...

e also:

tructures

in

eneral purpose structures:

Structure	Description
<u>ActlEjectMedia</u>	Eject media
<u>ActlKeyMacro</u>	Macro-oriented operations
<u>CharTableSet</u>	Character tables
<u>CmdLineSelect</u>	Command-line text selection/deselection
<u>FarMenuItem</u>	Menu item
<u>FarSetColors</u>	FAR Manager color scheme manipulations
FarStandardFunctions	Useful functions from Far.exe
FARINT64	used to hold a 64 bit integer value.
<u>KeySequence</u>	Description of a key code sequence
<u>OpenPluginInfo</u>	Information about the current plugin instance
PluginInfo	Information about a plugin module
PluginStartupInfo	Various pieces of important plugin information
<u>WindowInfo</u>	Information about the FAR Manager window

anel plugin structures

Structure	Description
<u>InfoPanelLine</u>	One line in the info panel
<u>KeyBarTitles</u>	Overrides function key labels in the key bar
PanelInfo	Information about a Far panel
PanelMode	Describes a panel view mode
PanelRedrawInfo	Is used to redraw a panel
<u>PluginPanelItem</u>	Information about an item in the emulated file system

ditor plugin structures

Structure	Description
EditorBookMarks	Information about bookmarks in the currently edited file

EditorColor	Information about color regions
<u>EditorConvertPos</u>	Conversion between real and screen positions of the cursor
<u>EditorConvertText</u>	Text conversion between the OEM and the internal FAR character set
EditorGetString	Editor line retrieval
<u>EditorInfo</u>	Current Far editor state
<u>EditorSaveFile</u>	Editor file saving
EditorSelect	Text selection/deselection in the editor
<u>EditorSetParameter</u>	Editor parameter control
EditorSetPosition	Position control in the internal FAR editor
EditorSetString	String control in the internal FAR editor

ialog API structures

Structure	Description
<u>FarDialogEvent</u>	Information about dialog event
<u>FarDialogItem</u>	Dialog item
<u>FarDialogItemData</u>	Passing data to a dialog item
<u>FarList</u>	The DI_LISTBOX list
<u>FarListColors</u>	Describes color schemes for DI_COMBOBOX and DI_LISTBOX controls
<u>FarListDelete</u>	Parameters for deletion from DI_COMBOBOX or DI_LISTBOX
<u>FarListFind</u>	Search in a DI_COMBOBOX or DI_LISTBOX
<u>FarListGetItem</u>	Retrieval of one element from a DI_COMBOBOX or DI_LISTBOX
<u>FarListInfo</u>	Retrieval of information about a DI_COMBOBOX or DI_LISTBOX
<u>FarListInsert</u>	Item insertion into a DI_COMBOBOX or DI_LISTBOX
FarListItem	List item
FarListPos	Positioning in the list
<u>FarListItemData</u>	Association of a list item with data
<u>FarListTitles</u>	Set or get list labels
<u>FarListUpdate</u>	List item update data

<u>OpenDlgPluginData</u>	Information about dialog and activated plugin item.

iewer specific structures

Structure	Description
<u>ViewerInfo</u>	Current viewer state
<u>ViewerMode</u>	Information about the current view mode
<u>ViewerSelect</u>	Block selection in the internal viewer
<u>ViewerSetPosition</u>	Position setting in the viewer

e also:

Exported functions, Service functions, Dialog API, Archive support, Addons, Delphi structure, Win32 structure

ervice functions

in

The following functions allow plugins to use FAR menu, dialogs, language support and some other useful functionality. Pointers to these functions are passed to plugins in the <u>SetStartupInfo</u> function.

Attention! Some of the functions listed below are absent in the older FAR versions. Read the remarks to the **PluginStartupInfo** structure and the GetMinFarVersion function for more information.

Function	Description
<u>AdvControl</u>	advanced control functions; can be called from anywhere: panels, viewer or editor.
<u>CharTable</u>	allows to get information about installed character tables.
<u>CmpName</u>	function compares a text string (for example, a file name) with a pattern (mask).
<u>GetMsg</u>	returns a message from the <u>language file</u> .
Menu	shows a menu.
RestoreScreen	restores a screen area previously saved by SaveScreen.
<u>SaveScreen</u>	saves a screen area.
<u>ShowHelp</u>	shows the specified FAR help topic for the specified hlf file.
Text	writes a text string to the screen.

ommon functions

anel specific functions

Function	Description
Control	allows to request different information about the FAR panels and perform various control actions.
<u>FreeDirList</u>	releases the memory allocated for files list by GetDirList and GetPluginDirList functions.
<u>GetDirList</u>	returns the list of files in the specified directory.
<u>GetPluginDirList</u>	returns list of files in the specified directory in the file system emulated by a plugin.

ditor specific functions

Function	Description
Editor	allows to invoke the FAR internal editor.
EditorControl	provides access to low level internal editor API.

ialog API specific functions

Function	Description
<u>DefDlgProc</u>	allows to call the internal dialog callback function.
Dialog	shows a dialog.
<u>DialogEx</u>	shows a dialog that allows to assign for it a callback function.
InputBox	a simple dialog box allowing to enter one line of text.
<u>Message</u>	shows a message.
<u>SendDlgMessage</u>	used to send a message to the dialog callback function.

iewer specific functions

Function	Description
<u>Viewer</u>	allows to invoke the internal viewer.
<u>ViewerControl</u>	allows to query and control the state of the internal viewer

e also:

Exported functions, <u>Structures</u>, <u>Archive support</u>, <u>Addons</u>, <u>Win32</u> <u>structures and function</u>

xported functions

in

FAR Manager communicates with its plugins by a set of functions exported by the plugins. A plugin does not need to export all functions listed below. Implement only those that are required by the functionality of the plugin.

Attention! All file names passed to FAR must be in OEM code page. FAR also passes file names in OEM code page. Before calling plugin functions FAR calls **SetFileApisToOEM**. If plugin uses anywhere SetFileApisToANSI, it must call SetFileApisToOEM again before returning control to FAR.

ommon functions

Function	Description
<u>Configure</u>	plugin configuration
ExitFAR	before closing the FAR Manager
<u>GetMinFarVersion</u>	get mininum FAR Manager version
<u>GetPluginInfo</u>	get plugin information
<u>SetStartupInfo</u>	global settings

ile panel specific functions

Function	Description
<u>ClosePlugin</u>	before closing an open plugin instance.
Compare	overrides sorting algorithm
<u>DeleteFiles</u>	delete files
<u>FreeFindData</u>	frees memory, allocated by GetFindData
<u>FreeVirtualFindData</u>	frees memory, allocated by GetVirtualFindData
<u>GetFiles</u>	get files
<u>GetFindData</u>	get file list
<u>GetOpenPluginInfo</u>	get information about an open plugin instance
<u>GetVirtualFindData</u>	get files
MakeDirectory	make a directory

<u>OpenFilePlugin</u>	open a file
<u>OpenPlugin</u>	create a new plugin instance
<u>ProcessEvent</u>	process events
<u>ProcessHostFile</u>	execute archive commands
ProcessKey	process keyboard events
PutFiles	put files to the emulated file system
<u>SetDirectory</u>	set current directory in the emulated file system
<u>SetFindList</u>	transfers found files from the "Find file" dialog to the emulated file system

ditor specific functions

Function	Description
ProcessEditorInput	process keyboard events
ProcessEditorEvent	process editor events

iewer specific functions

Function	Description
ProcessViewerEvent	process viewer events

ialog specific functions

Function	Description
ProcessDialogEvent	Process dialog events

e also:

Service functions, Structures, Archive support, Addons

ialog API 1.0

in

All Dialog API details are written with taking named unions of the FarDialogItem structure into account. For more details see the description of **FAR NO NAMELESS UNIONS**.

Dialog API represents an individual API beginning from FAR Manager version 1.70. What's in it for plugin developers? The main thing is higher control over the created dialog.

There're two different dialog types:

- Dialogs of so-called "About" style (FAR version 1.65 and below)
- Extended style dialogs those using the dialog callback function.

Regardless of the style used, dialog manager v1.0 supports only so-called **Modal Dialogs**. This dialog represents a window which disables user interaction with any FAR Manager object outside the boundaries of the dialog. The modal dialog cuts off all keyboard/mouse events sent to other FAR Manager objects.

About" style

It's simple - UNCONTROLLABLE DIALOGS! Dialogs of this type are created with either **Dialog** or **DialogEx** function call with the **DlgProc** parameter set to NULL.

This style defines the following dialog behavior:

- any changes in control element state become known to the plugin only after dialog is closed;
- dialog is closed when user presses one of the following keys: Esc, Ctrl-Enter, Enter (pressing Enter closes the dialog regardless of which control element has the focus, with the exception of edit boxes with **<u>DIF</u>** EDITOR flag set), or clicks a mouse button beyond the dialog bounds.
- This style is intended for simple dialogs.

Extended" style

This is the most interesting style. Plugin has ultimate control over the dialog.

- dialog has its own callback function which reacts to a lot of messages sent by the Dialog Manager;
- the callback function communicates with the Dialog Manager by sending <u>messages</u> with the <u>SendDlgMessage</u> function;
- the callback function can delegate control to the Dialog Manager with the <u>DefDlgProc</u> function;
- when user tries to interact with inaccessible FAR Manager objects, clicking mouse buttons outside the dialog, the Dialog Manager warns him/her with beeps.
- the plugin is in control of the dialog session ending.
- this style is intended for dialogs implementing complex user interaction logic.

If one simply delegates control to the kernel in a dialog callback function with the <u>DefDlgProc</u> function call, then one gets simple About-styled dialog:

```
// dialog callback function with minimal code
LONG_PTR WINAPI MyDlgProc(HANDLE hDlg,int Msg,int Par
{
   return Info.DefDlgProc(hDlg,Msg,Param1,Param2);
}
```

Choosing dialog style is simple:

- to create a dialog with one <u>InputBox</u> choose simple About style the lesser the code the simpler the result, reducing possibility of errors;
- to create more complex plugins (game, calculator, querying a database using ODBC, etc.) choose Extended style as it provides complex logic, dynamic controls, advanced user interaction (on which may depend sunsequent actions), etc..

Well then, the Dialog API v1.0:

Functions StructuresMacros and typesDialog elementsDialog element flagsEvents and MessagesInput focusWorking with lists

emarks

To get more familiar with the Dialog API see the dialog.cpp file from "<u>Reversi</u>" plugin sources.

e also:

Exported functions, Service functions, Structures, Archive support, Addons

iewer API

in

xported functions

Function	Description
ProcessViewerEvent	process viewer events

ervice functions

Function	Description
<u>Viewer</u>	allows to invoke the internal viewer.
<u>ViewerControl</u>	allows to query and control the state of the internal viewer

tructures

Structure	Description
FARINT64	used to hold a 64 bit integer value.
<u>ViewerInfo</u>	current viewer state
<u>ViewerMode</u>	information about the current view mode
<u>ViewerSelect</u>	block selection in the internal viewer
<u>ViewerSetMode</u>	set the view mode of the current viewer instance
<u>ViewerSetPosition</u>	position setting in the viewer

e also:

anel API

in

xported functions

Function	Description
<u>ClosePlugin</u>	before closing an open plugin instance.
Compare	overrides sorting algorithm
<u>DeleteFiles</u>	delete files
<u>FreeFindData</u>	frees memory, allocated by GetFindData
<u>FreeVirtualFindData</u>	frees memory, allocated by GetVirtualFindData
GetFiles	get files
<u>GetFindData</u>	get file list
<u>GetOpenPluginInfo</u>	get information about an open plugin instance
<u>GetVirtualFindData</u>	get files
<u>MakeDirectory</u>	make a directory
<u>OpenFilePlugin</u>	open a file
<u>OpenPlugin</u>	create a new plugin instance
<u>ProcessEvent</u>	process events
ProcessHostFile	execute archive commands
<u>ProcessKey</u>	process keyboard events
PutFiles	put files to the emulated file system
SetDirectory	set current directory in the emulated file system
<u>SetFindList</u>	transfers found files from the "Find file" dialog to the emulated file system

ervice functions

Function	Description
Control	allows to request different information about the FAR panels and perform various control actions.
<u>FreeDirList</u>	releases the memory allocated for files list by GetDirList and GetPluginDirList functions.
<u>GetDirList</u>	returns the list of files in the specified directory.
GetPluginDirList	returns list of files in the specified directory in the file

tructures

Structure	Description
<u>InfoPanelLine</u>	One line in the info panel
<u>KeyBarTitles</u>	Overrides function key labels in the key bar
PanelInfo	Information about a Far panel
PanelMode	Describes a panel view mode
PanelRedrawInfo	Used to redraw a panel
<u>PluginPanelItem</u>	Information about an item in the emulated file system

onstants

Constants	Description
<u>NM</u>	size of the buffer needed to receive a full file name.
<u>OPM_*</u>	additional information about function operation mode and place, from which it was called.
<u>SM *</u>	sort modes.

e also:

ditor API

<u>in</u>

xported functions

Function	Description
ProcessEditorInput	process keyboard events
ProcessEditorEvent	process editor events

ervice functions

Function	Description
Editor	allows to invoke the FAR internal editor.
<u>EditorControl</u>	provides access to low level internal editor API.

tructures

Structure	Description
EditorBookMarks	information about bookmarks in the currently edited file
<u>EditorColor</u>	information about color regions
<u>EditorConvertPos</u>	conversion between real and screen positions of the cursor
<u>EditorConvertText</u>	text conversion between OEM and internal FAR character set
<u>EditorGetString</u>	editor line retrieval
<u>EditorInfo</u>	current editor state
<u>EditorSaveFile</u>	editor file saving
<u>EditorSelect</u>	text selection/deselection in the editor
EditorSetParameter	editor parameters setting
EditorSetPosition	position setting in the editor
EditorSetString	change or insert string in the editor

e also:

ar Standard Functions

in | Structures | TFarStandardFunctions

The **FarStandardFunctions** structure contains addresses to standard FAR functions. Using these functions allows the plugin to use standard operations and reduce the size of the plugin's DLL.

```
struct FarStandardFunctions
{
  int StructSize;
  FARSTDATOI
                                atoi;
                                atoi64;
  FARSTDAT0164
  FARSTDITOA
                                itoa;
                                itoa64;
  FARSTDIT0A64
  FARSTDSPRINTF
                                <u>sprintf;</u>
  FARSTDSSCANF
                                sscanf;
  FARSTDQSORT
                                <u>qsort</u>;
  FARSTDBSEARCH
                                bsearch;
  FARSTDQSORT
                                <u>qsortex;</u>
                                snprintf;
  FARSTDSNPRINTF
                                Reserved[8];
  DWORD PTR
  FARSTDLOCALISLOWER
                                LIsLower;
  FARSTDLOCALISUPPER
                                LIsUpper;
                                LIsAlpha;
  FARSTDLOCALISALPHA
                                LIsAlphanum;
  FARSTDLOCALISALPHANUM
  FARSTDLOCALUPPER
                                LUpper;
                                LLower;
  FARSTDLOCALLOWER
  FARSTDLOCALUPPERBUF
                                LUpperBuf;
                                LLowerBuf;
  FARSTDLOCALLOWERBUF
  FARSTDLOCALSTRUPR
                                LStrupr;
  FARSTDLOCALSTRLWR
                                LStrlwr;
  FARSTDLOCALSTRICMP
                                LStricmp;
                                LStrnicmp;
  FARSTDLOCALSTRNICMP
```

FARSTDUNQUOTE Unquote; FARSTDEXPANDENVIRONMENTSTR ExpandEnvironmentStr; FARSTDLTRIM FARSTDRTRIM FARSTDTRIM FARSTDTRUNCSTR FARSTDTRUNCPATHSTR FARSTDQUOTESPACEONLY FARSTDPOINTTONAME FARSTDGETPATHROOT FARSTDADDENDSLASH FARSTDCOPYTOCLIPBOARD FARSTDPASTEFROMCLIPBOARD FARSTDKEYTOKEYNAME FARSTDKEYNAMETOKEY FARSTDINPUTRECORDTOKEY FARSTDXLAT FARSTDGETFILEOWNER FARSTDGETNUMBEROFLINKS FARSTDRECURSIVESEARCH FARSTDMKTEMP FARSTDDELETEBUFFER FARSTDPROCESSNAME FARSTDMKLINK FARCONVERTNAMETOREAL FARGETREPARSEPOINTINFO

<u>LTrim;</u> RTrim; Trim; TruncStr; TruncPathStr; OuoteSpaceOnly; PointToName; GetPathRoot; AddEndSlash; CopyToClipboard; PasteFromClipboard; FarKeyToName; FarNameToKey; FarInputRecordToKey; <u>XLat;</u> GetFileOwner; <u>GetNumberOfLinks;</u> FarRecurseSearch; <u>MkTemp;</u> DeleteBuffer; ProcessName; MkLink; ConvertNameToReal; GetReparsePointInfo;

};

embers

Function	Description
StructSize	Structure size. If the structure will change in the future, this field will allow to determine it.
atoi	converts a string to a 32-bit integer.
<u>atoi64</u>	converts a string to a 64-bit integer (int64).
<u>itoa</u>	converts a 32-bit integer value into a string.
<u>itoa64</u>	converts a 64-bit integer (int64) value into a string.
<u>sprintf</u>	allows to write formatted output to a string.
sscanf	allows to read formatted data from a string.

<u>qsort</u>	allows to sort an array of any type of data using the QuickSort algorithm.
<u>bsearch</u>	allows to perform a binary search of a sorted array.
<u>qsortex</u>	allows to sort an array of any type of data using the QuickSort algorithm.
<u>snprintf</u>	allows to write formatted output to a string.
LIsLower	tests whether the given character is in lower case.
<u>LIsUpper</u>	tests whether the given character is in upper case.
<u>LIsAlpha</u>	tests whether the given character is a letter.
<u>LIsAlphanum</u>	tests whether the given character is a number or a letter.
<u>LUpper</u>	converts a character to upper case.
LLower	converts a character to lower case.
<u>LUpperBuf</u>	converts an array of characters, including null ones, to upper case.
<u>LLowerBuf</u>	converts an array of characters, including null ones, to lower case.
LStrupr	converts a null-terminated string to upper case.
LStrlwr	converts a null-terminated string to lower case.
LStricmp	compares two strings without case sensitivity.
<u>LStrnicmp</u>	compares portions of two strings without case sensitivity.
<u>Unquote</u>	removes all quotes from a null-terminated string.
ExpandEnvironmentStr	used to expand environment variables in a string to their values.
LTrim	removes leading whitespace from a string.
RTrim	removes trailing whitespace from a string.
Trim	removes all leading and trailing whitespace from a string.
TruncStr	truncates a given string to the specified length and, if needed, inserts in its beginning an ellipsis instead of the truncated part.
<u>TruncPathStr</u>	truncates a given path to specified length and, if needed, inserts an ellipsis to indicate the place of truncation.
QuoteSpaceOnly	quotes the input string if it contains at least one space inside.

PointToName	used to get the file name from a given file path.
<u>GetPathRoot</u>	used to get the root directory from a given file path.
<u>AddEndSlash</u>	used to add a trailing backslash or a slash to a path.
<u>CopyToClipboard</u>	copies a text string to the Windows clipboard.
PasteFromClipboard	used to get data from the Windows clipboard.
<u>FarKeyToName</u>	used to convert an internal FAR key code to a string.
<u>FarNameToKey</u>	used to convert a literal key name to an internal FAR key code.
<u>FarInputRecordToKey</u>	used to convert a key code from an <u>INPUT_RECORD</u> structure to an internal FAR key code.
XLat	used to transliterate a string portion from one character set (for example Russian) to another character set (for example Latin).
<u>GetFileOwner</u>	used to determine the owner of the given file.
<u>GetNumberOfLinks</u>	returns the number of <u>hard links</u> to the specified file.
FarRecursiveSearch	used to find a file in a directory tree with name matching the given mask.
<u>MkTemp</u>	used to create a temporary file name with the path based on a specified template.
<u>DeleteBuffer</u>	used to free an allocated buffer returned by the PasteFromClipboard function.
ProcessName	allows to perform various actions on a file name: compare with a mask, with a list of masks or to generate new file name using a mask.
MkLink	supports creating hard and symbolic links and mounting local drives to the file system.
<u>ConvertNameToReal</u>	converts a relative name of a file object to its full pathname and expands symbolic links (reparse points).
<u>GetReparsePointInfo</u>	allows to determine the target (path to the target drive and directory) of a symbolic link (reparse point).

emarks

Before you can start using standard functions you have to store structure contents locally:

```
static struct PluginStartupInfo Info;
static struct FarStandardFunctions FSF;
```

```
void __export SetStartupInfo(struct PluginStartupInfc
{
    Info=*psInfo;
    FSF=*psInfo->FSF;
    Info.FSF=&FSF; // now Info.FSF will point to the
    ....
}
```

e also: <u>Structures</u> | <u>PluginStartupInfo</u>

anguage and Help files

in

Using the information given below will allow you to easily adjust the language of your plugins (within sensible limits) - just create a few files and you're done.

All * . lng and * . hlf files in the FAR directory, Plugins directory and all subdirectories of the Plugins directory are considered to be language and help files, respectively. The <u>Language</u> control statement defines the language of the file.

Format	Description
Control statements	description of control statements in language and help files
Language files	description of language files syntax
<u>Help files</u>	description of help files syntax

e also:

ustom API

<u>in</u>

This topic presents "third-party" APIs, which aid in writing subplugins for several well-known FAR plugins.

API	Description
<u>Archive support API</u>	FMT-modules development API for the MultiArc plugin.
Regular expressions interface for the colorer library	Regular expressions description for colorer , both sources and external DLL interface.
Search and Replace plugins programming	SRP-modules development API for the Search and Replace plugin.
<u>Mr.Ripper API</u>	RIP-modules development API for the Mr.Ripper plugin.

lacros

in

Macro command or 'macro' in FAR is a recorded sequence of keystrokes that can be assigned to a shortcut command. Macro can be used to repeatedly perform a certain sequence of user actions at a time by pressing a single hotkey.

Generally macros can be used to:

- 1. Assign an often used set of actions to a shortcut to use it repeatedly;
- 2. Perform special functions which can be specified in a macro command text by using special commands;
- 3. Redefine the standard FAR hotkeys.

Mostly macros are used to define shortcut access to the external plugin modules and redefine hotkeys for standard FAR functions.

See also:

General background Using macros Macro-language Technical details Examples

ddons

<u>in</u>

The following fuctions and structures are not included in the standart API, but often used when writing plugins.

This section will also contain structures and functions, which may help in writing plugins. If you think, that your structure or function is worthy of being here, write to <u>authors</u> of this encyclopedia.

eneral purpose functions

Function	Description
<u>InitDialogItems</u>	Translates <u>InitDialogItem</u> structure to <u>FarDialogItem</u> structure
<u>InitMenuItems</u>	Translates <u>InitMenuItem</u> structure to <u>FarMenuItem</u> structure
LocMsg	Returns a pointer to a string according to language settings of FAR Manager

tructures

Structure	Description
InitDialogItem	Defines the dialog item
<u>InitMenuItem</u>	Defines the menu item

e also: <u>Custom API Exported functions</u> <u>Structures</u> <u>Archive support</u>

onstants

in | Macros | Macro-language

You can use the constants to store values associated with the names and use them later in the macro sequences.

aming

Names of the constants should comply with these rules:

- 1. Name should begin from the english letter, and can contain in arbitrary order english letters, digits, and '_' symbol.
- 2. Name should not coincide with predefined <u>boolean states</u>, <u>macro functions</u> <u>names</u>, names of the keys, and other <u>macro states</u>.

/pes

Constants can be either string or integer.

Integer constants can be represented by: NNN - decimal constant, 0NNN - octal constant, 0xNNN - hexadecimal constant.

Integers are of 64 bit width.

rea of effect

The scope for the constants is a current FAR Manager session.

epresentation in registry

In the registry, constants are stored in the special key
[HKEY_CURRENT_USER\Software\Far\
[Users\USERNAME\]KeyMacros\Consts].

Every constant has a name and can be of three types REG_SZ (for string constants) and REG_DWORD or REG_QDWORD (for integer constants).

emarks

1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.

xample

Example of using FIB_PASSWORD constant (0x0000002) for prompt function

REGEDIT

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Consts]
"FIB_PASSWORD"=dword:00000002
```

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Editor\Ctrl
"Sequence"="%s=prompt(\"Password\",\"Input password:\
"DisableOutput"=dword:00000001
```

e also:

AR_FIND_DATA

in | structures | win32 structures

The **FAR_FIND_DATA** structure describes a file object located in the FAR file panel.

```
typedef struct _FAR_FIND_DATA {
   DWORD dwFileAttributes;
   FILETIME ftCreationTime;
   FILETIME ftLastAccessTime;
   FILETIME ftLastWriteTime;
   DWORD nFileSizeHigh;
   DWORD nFileSizeLow;
   DWORD dwReserved0;
   DWORD dwReserved1;
   TCHAR cFileName[ MAX_PATH ];
   TCHAR cAlternateFileName[ 14 ];
} FAR_FIND_DATA;
```

embers

dwFileAttributes

File attributes of the file found. This member can be one or more of the following values:

Attribute	Description
FILE_ATTRIBUTE_ARCHIVE	The file or directory is an archive file or directory. Applications use this attribute to mark files for backup or removal.
FILE_ATTRIBUTE_COMPRESSED	The file or directory is compressed. For a file, this means that all of the data in the file is compressed. For a directory, this means that compression is the default for newly created files and subdirectories.
FILE_ATTRIBUTE_DIRECTORY	Identifies a directory.
FILE_ATTRIBUTE_ENCRYPTED	The file or directory is encrypted. For a file, this means that all data in the file is encrypted. For a directory, this means that encryption is the

	default for newly created files and subdirectories.
FILE_ATTRIBUTE_HIDDEN	The file or directory is hidden. It is not included in an ordinary directory listing.
FILE_ATTRIBUTE_NORMAL	The file or directory has no other attributes set. This attribute is valid only if used alone.
FILE_ATTRIBUTE_OFFLINE	The file data is not immediately available. This attribute indicates that the file data has been physically moved to offline storage.
FILE_ATTRIBUTE_READONLY	The file or directory is read-only. Applications can read the file but cannot write to it or delete it. In the case of a directory, applications cannot delete it.
FILE_ATTRIBUTE_REPARSE_POINT	The file or directory has an associated reparse point.
FILE_ATTRIBUTE_SPARSE_FILE	The file is a sparse file.
FILE_ATTRIBUTE_SYSTEM	The file or directory is part of the operating system or is used exclusively by the operating system.
FILE_ATTRIBUTE_TEMPORARY	The file is being used for temporary storage. File systems attempt to keep all of the data in memory for quicker access, rather than flushing it back to mass storage. A temporary file should be deleted by the application as soon as it is no longer needed.
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	The file or directory is not be indexed by the content indexing service.
FILE_ATTRIBUTE_VIRTUAL	A file is a virtual file.

ftCreationTime

A **<u>FILETIME</u>** structure that specifies when the file or directory was created.

The **<u>FindFirstFile</u>**

and <u>FindNextFile</u>

functions return file time in the UTC format. Those functions set FILETIME to 0 if the underlying file system does not support creation time. You can use the <u>FileTimeToLocalFileTime</u>

functions to transform the file time from UTC format to local time format. And then use the <u>FileTimeToSystemTime</u>

function to transform the local time to a <u>SYSTEMTIME</u> structure which has different members for month, day, year, etc.

ftLastAccessTime

A **FILETIME** structure. For a file, the structure specifies when the file was last read from or written to. For a directory, the structure specifies when the directory was created. For both files and directories, the specified date will be correct, but the time of day will always be set to midnight. If the underlying file system does not support last access time, this member is zero. File time is

returned in the UTC format.

ftLastWriteTime

A **FILETIME** structure. For a file, the structure specifies when the file was last written to. For a directory, the structure specifies when the directory was created. If the underlying file system does not support last write time, this member is zero. File time is returned in the UTC format.

nFileSizeHigh

High-order DWORD value of the file size, in bytes. This value is zero unless the file size is greater than MAXDWORD. The size of the file is equal to (nFileSizeHigh * (MAXDWORD+1)) + nFileSizeLow.

nFileSizeLow

Low-order DWORD value of the file size, in bytes.

dwReserved0

If the *dwFileAttributes* member includes the

FILE_ATTRIBUTE_REPARSE_POINT attribute, this member specifies the reparse tag. Otherwise, this value is undefined and should not be used.

dwReserved1

Reserved for future use.

cFileName

A null-terminated string that specifies the name of the file.

cAlternateFileName

A null-terminated string that specifies an alternative name for the file. This name is in the classic 8.3 (filename.ext) file name format.

emarks

1. If a file has a long file name, the complete name appears in the *cFileName* member, and the 8.3 format truncated version of the name appears in the *cAlternateFileName* member. Otherwise, *cAlternateFileName* is empty. As an alternative, you can use the <u>GetShortPathName</u>

function to find the 8.3 format

version of a file name.

2. Not all file systems can record creation and last access time and not all file systems record them in the same manner. For example, on NT FAT, create time has a resolution of 10 milliseconds, write time has a resolution of 2 seconds, and access time has a resolution of 1 day (really, the access date). On NTFS, access time has a resolution of 1 hour.

e also:

FAR USE WIN32 FIND DATA | WIN32 FIND DATA | FILETIME | TWin32FindData

FAR_USE_WIN32_FIND_DATA

in | types and definitions

The macro _FAR_USE_WIN32_FIND_DATA controls whether the PluginPanelItem structure uses the WIN32 FIND DATA structure defined in the Windows API headers or the the FAR FIND DATA that has the same layout but is defined in plugin.hpp. As some compilers (for instance, Borland C++ 5.5) force 8-byte alignment for structures defined in the Windows API headers, it is not possible to achieve the correct 2-byte alignment for the entire PluginPanelItem structure when the WIN32 FIND DATA structure is used.

If the _FAR_USE_WIN32_FIND_DATA macro is defined, the

PluginPanelItem structure will be compatible with FAR 1.65 and FAR 1.70 headers. But correct compilation will not be possible with Borland C++ 5.5 without modifying Windows API header files.

If the macro **_FAR_USE_WIN32_FIND_DATA** is not defined, the FAR FIND DATA structure will be used. This way the code would compile correctly with any compiler, but some plugins that depend on the usage of WIN32 FIND DATA would not compile.

Attention! In FAR 1.70 release <u>WIN32_FIND_DATA</u> is used by default (**FAR USE WIN32 FIND DATA** is defined). In FAR 1.71 FAR FIND DATA will be used by default.

e also:

FAR FIND DATA | WIN32 FIND DATA
M <u>in | types and definitions</u>

The constant **NM** deffines the size for a buffer containing a full file name.

olumn types

in | exported functions

The following column types are allowed:

N - file name, allowed modifiers (e.g. "*NMR*"):

- M show selection marks;
- O show names without paths (intended mainly for plugins);
- R right aligned names.
- S file size;
 - Allowed modifiers for file sizes:
 - C format file size with commas;

T - use 1000 instead of 1024 as divider, if column width is too small to show full size.

F - file size if shown as a float number as in Windows Explorer (i.e. 999 bytes are shown as 999, however 1000 bytes are shown as 0.97 KB);

E - economic mode, no space is shown between the size and the suffix

P - packed file size.

Allowed modifiers for file sizes:

C - format file size with commas;

T - use 1000 instead of 1024 as divider, if column width is too small to show full size.

F - file size if shown as a float number as in Windows Explorer (i.e. 999 bytes are shown as 999, however 1000 bytes are shown as 0.97 KB);

E - economic mode, no space is shown between the size and the suffix

D - file modification date

T - file modification time

DM - file modification date and time

Allowed modifiers:

B - brief (Unix style) file time format;

M - use text month names;

DC - file creation date and time

Allowed modifiers:

B - brief (Unix style) file time format;

M - use text month names;

DA - file last access date and time

Allowed modifiers:

B - brief (Unix style) file time format;

M - use text month names;

- A file attributes
- Z file description
- O file owner
- *LN* hard links number

C0..C9 - user defined column types.

If the column types description contains more than one file name column, the file panel will be displayed in multicolumn form.

e also: PanelMode

/in32 structures, functions and definitions

<u>in</u>

This chapter contains the articles on WinAPI structures and functions commonly used.

/in32 functions:

Function	Description
<u>GetFileTime</u>	retrieves the date and time that a file was created, last accessed, and last modified
PeekConsoleInput	reads data from the specified console input buffer without removing it from the buffer
<u>ReadConsoleInput</u>	reads data from a console input buffer and removes it from the buffer
<u>SetFileApisToAnsi</u>	causes the file I/O functions to use the ANSI character set code page
<u>SetFileApisToOem</u>	causes the file I/O functions to use the OEM character set code page
<u>SetFileTime</u>	sets the date and time that a file was created, last accessed, or last modified
<u>WriteConsoleInput</u>	writes data directly to the console input buffer

/in32 structures:

Structure	Description
CHAR_INFO	specifies the character and its attributes
CONSOLE CURSOR INFO	contains information about the console cursor
COORD	defines the coordinates of a character cell in a console screen buffer
FILETIME	the 64-bit number of 100-nanosecond intervals since January 1, 1601 (UTC)
FOCUS EVENT RECORD	reports focus events in a console <u>INPUT_RECORD</u> structure
INPUT_RECORD	reports input events in the console input buffer
KEY EVENT RECORD	reports keyboard input events in a console <u>INPUT_RECORD</u> structure
MENU EVENT RECORD	reports menu events in a console <u>INPUT_RECORD</u> structure

MOUSE EVENT RECORD	reports mouse input events in a console <u>INPUT_RECORD</u> structure
RECT	defines the coordinates of the upper-left and lower- right corners of a rectangle
<u>SMALL RECT</u>	defines the coordinates of the upper-left and lower- right corners of a rectangle
<u>SYSTEMTIME</u>	represents a date and time using individual members for the month, day, year, weekday, hour, minute, second, and millisecond
WIN32 FIND DATA	describes a file found by the FindFirstFile , FindFirstFileEx , or FindNextFile function
WINDOW BUFFER SIZE RECORD	reports changes in the size of the screen buffer in a console <u>INPUT_RECORD</u> structure

/in32 definitions:

Constant	Description
<u>Virtualkeycodes</u>	virtual key codes

e also:

Exported functions Service functions Addons

ditorSaveFile

in | structures

The **EditorSaveFile** structure is used in the <u>EditorControl</u> function to save the currently edited file (the <u>ECTL_SAVEFILE</u> command).

```
struct EditorSaveFile
{
    char FileName[NM];
    char *FileEOL;
};
```

emnts

FileName

Optional new name for the file (full path required). Set the first byte of the **FileName** field to zero to retain the current file name.

FileEOL

End-of-line sequence. Can be NULL (leave current EOL-sequence), $\n n$

e also:

structures | TEditorSaveFile

ialog API Events & Messages

in | <u>Dialog API</u>

The Dialog API has many <u>messages</u> and <u>events</u> that allow plugins to work with <u>extended dialogs</u>.

e also: Events | Messages



Below are known to Far errors that can be returned by <u>GetLastError</u>

function (see FMSG_ERRORTYPE):

Constants from winerror.h	Text from Far*.lng
ERROR_INVALID_FUNCTION	"Incorrect function"
ERROR_BAD_COMMAND, ERROR_CALL_NOT_IMPLEMENTED	"Command not recognized"
ERROR_FILE_NOT_FOUND	"File not found"
ERROR_PATH_NOT_FOUND	"Path not found"
ERROR_TOO_MANY_OPEN_FILES	"Too many open files"
ERROR_ACCESS_DENIED	"Access denied"
ERROR_NOT_ENOUGH_MEMORY, ERROR_OUTOFMEMORY	"Not enough memory"
ERROR_WRITE_PROTECT	"Cannot write to write protected disk"
ERROR_NOT_READY	"The device is not ready"
ERROR_NOT_DOS_DISK	"Disk cannot be accessed"
ERROR_SECTOR_NOT_FOUND	"Sector not found"
ERROR_OUT_OF_PAPER	"The printer is out of paper"
ERROR_WRITE_FAULT	"Write fault error"
ERROR_READ_FAULT	"Read fault error"
ERROR_GEN_FAILURE	"Device general failure"
ERROR_SHARING_VIOLATION, ERROR_LOCK_VIOLATION	"File sharing violation"
ERROR_BAD_NETPATH	"The network path was not found"
ERROR_NETWORK_BUSY	"The network is busy"
ERROR_NETWORK_ACCESS_DENIED	"Network access is denied"
ERROR_NET_WRITE_FAULT	"A write fault occurred on the network"
ERROR_DRIVE_LOCKED	"The disk is in use or locked by another process"
ERROR_ALREADY_EXISTS	"File or folder already exists"
ERROR_BAD_PATHNAME, ERROR_INVALID_NAME	"The specified name is invalid"
ERROR_DISK_FULL, ERROR_HANDLE_DISK_FULL	"Insufficient disk space"

ERROR_DIR_NOT_EMPTY	"The folder is not empty"
ERROR_INTERNET_INCORRECT_USER_NAME	"Incorrect user name"
ERROR_INTERNET_INCORRECT_PASSWORD	"Incorrect password"
ERROR_INTERNET_LOGIN_FAILURE	"Login failure"
ERROR_INTERNET_CONNECTION_ABORTED	"Connection aborted"
ERROR_CANCELLED	"Operation cancelled"
ERROR_NO_NETWORK	"No network present"
ERROR_DEVICE_IN_USE	"Device is in use and cannot be disconnected"
ERROR_OPEN_FILES	"This network connection has open files"
ERROR_ALREADY_ASSIGNED	"The local device name is already in use"
ERROR_DEVICE_ALREADY_REMEMBERED	"The local device is already in the user profile"
ERROR_NOT_LOGGED_ON	"User has not logged on to the network"
ERROR_INVALID_PASSWORD	"The user password is invalid"
ERROR_NO_RECOVERY_POLICY	"There is no valid encryption recovery policy configured for this system"
ERROR_ENCRYPTION_FAILED	"The specified file could not be encrypted"
ERROR_DECRYPTION_FAILED	"The specified file could not be decrypted"
ERROR_FILE_NOT_ENCRYPTED	"The specified file is not encrypted"

e also:

<u>Message</u>

RSUSERFUNC

in | types and definitions

The **FRSUSERFUNC** type describes a callback function for the <u>FSF.FarRecursiveSearch</u> function that is called for each found file.

```
typedef int (WINAPI *FRSUSERFUNC)(
  const WIN32 FIND DATA *FData,
  const char *FullName,
  void *Param
);
```

arameters

FData

Points to the <u>WIN32_FIND_DATA</u> structure of the found file.

FullName

Full path to the found file.

Param

Points to user data (the *Param* parameter of the <u>FSF.FarRecursiveSearch</u> function).

eturn value

The function should return TRUE to continue searching the file system, or FALSE for the search process to be interrupted.

emarks

FData and *FullName* are defined only in the context of this function, the plugin should not save those pointers. To save the data, the plugin must copy those values to internal structures. *Param* - the data that was passed to the <u>FSF.FarRecursiveSearch</u> function.

e also: <u>FSF.FarRecursiveSearch</u>

elphi structures

in | structures

eneral purpose structures

Structure	Description
<u>TActlEjectMedia</u>	Eject media
<u>TActlKeyMacro</u>	Macro-oriented operations
<u>TCharTableSet</u>	Character tables
TCmdLineSelect	Command-line text selection/deselection
<u>TFarMenuItem</u>	Menu item
<u>TFarSetColors</u>	FAR Manager color scheme manipulations
TFarStandardFunctions	Useful functions from Far.exe
<u>TInfoPanelLine</u>	One line in the info panel
<u>TKeySequence</u>	Description of key code sequence
<u>TKeyBarTitles</u>	Overrides function key labels in the key bar
<u>TOpenPluginInfo</u>	Information about the current plugin instance
<u>TPanelInfo</u>	Information about a Far panel
<u>TPanelMode</u>	Describes a panel view mode
TPanelRedrawInfo	Is used to redraw a panel
<u>TPluginInfo</u>	Information about a plugin module
<u>TPluginPanelItem</u>	Information about one item in the emulated file system
<u>TPluginStartupInfo</u>	Various pieces of important plugin information
<u>TWindowInfo</u>	Information about the FAR Manager window

ditor specific structures

Structure	Description
<u>TEditorBookMarks</u>	Information about bookmarks in the currently edited file
<u>TEditorColor</u>	Information about color regions
<u>TEditorConvertPos</u>	Conversion between real and screen positions of the cursor
TEditorConvertText	Text conversion between the OEM and the internal FAR character set

TEditorGetString	Editor line retrieval
<u>TEditorInfo</u>	Current Far editor state
<u>TEditorSaveFile</u>	Editor file saving
<u>TEditorSelect</u>	Text selection/deselection in the editor
<u>TEditorSetParameter</u>	Editor parameter control
TEditorSetPosition	Position control in the internal FAR editor
TEditorSetString	String control in the internal FAR editor

ialog API specific structures

Structure	Description
<u>TFarDialogItem</u>	Dialog item
<u>TFarDialogItemData</u>	Passing data to a dialog item
<u>TFarList</u>	The DI_LISTBOX list
<u>TFarListColors</u>	Describes color scheme for DI_COMBOBOX and DI_LISTBOX controls
<u>TFarListDelete</u>	Parameters for deletion from DI_COMBOBOX or DI_LISTBOX
<u>TFarListFind</u>	Search in a DI_COMBOBOX or DI_LISTBOX
<u>TFarListGetItem</u>	Retrieval of one element from a DI_COMBOBOX or DI_LISTBOX
<u>TFarListInfo</u>	Retrieval of information about a DI_COMBOBOX or DI_LISTBOX
<u>TFarListInsert</u>	Item insertion into a DI_COMBOBOX or DI_LISTBOX
<u>TFarListItem</u>	List item
<u>FarListPos</u>	Positioning in the list
<u>TFarListTitles</u>	Set or get list labels
<u>TFarListItemData</u>	Association of a list item with data
<u>FarListUpdate</u>	List item update data

iewer API specific structures

Structure	Description
TFarInt64	used to hold a 64 bit integer value.
<u>TViewerInfo</u>	current viewer state

<u>TViewerMode</u>	information about the current view mode
<u>TViewerSelect</u>	block selection in the internal viewer
<u>TViewerSetMode</u>	set the view mode of the current viewer instance
TViewerSetPosition	position setting in the viewer

e also:

Exported functions, Service functions, Archive support, Addons

olumn width

in | structures

The **ColumnWidths** member of the <u>PanelInfo</u> and <u>PanelMode</u> structures describes the width of the panel columns (e.g. "**0**, **8**, **0**, **5**").

The format is simple - a string of numbers (representing the column width) delimited by commas.

If width is 0, the default value will be used. If the width of the name, description or owner column is 0, the it will be automatically calculated depending on panel width. At least one of the column widths should be set to 0 for correct representation in any panel width.

If 12-hour format is used, the standard width of the file time or the file time and date column should be increased by one. Increasing by a bigger value will show seconds and milliseconds.

To show the year in a 4-digit format you should increase the width of the date column by 2.

e also: structures | PanelInfo | PanelMode

ialog API controls

in | Dialog API | FarDialogItem

Dialog API supports a set of controls. Each one is described by a <u>FarDialogItem</u> structure. An array of these structures is passed to the <u>Dialog</u> and <u>DialogEx</u> functions to show a dialog.

Dialog item	Value	Description
DI BUTTON	7	Button control
DI CHECKBOX	8	Checkbox control
DI COMBOBOX	10	Combobox control
DI DOUBLEBOX	3	Double line frame
<u>DI EDIT</u>	4	Edit control
DI FIXEDIT	6	Edit control with a fixed text length
DI LISTBOX	11	Listbox control
DI PSWEDIT	5	Password edit control
DI RADIOBUTTON	9	Radiobutton
DI SINGLEBOX	2	Single line frame
DI TEXT	0	Text string
DI USERCONTROL	255	User-defined control
DI_VTEXT	1	Vertical text string

e also:

Dialog item flags

ialog item flags

in | Dialog API | Dialog items

It is possible to control DialogAPI behavior and appearance using flags. To know which flags affect a specific control, read in the control item's description.

Flag	Description
DIF 3STATE	A <u>DI CHECKBOX</u> control will have 3 states.
DIF BOXCOLOR	Control text will have frame color.
DIF_BTNNOCLOSE	For button do not close dialog.
DIF_CENTERGROUP	Centering a group of controls.
DIF CENTERTEXT	Text centering in static controls.
DIF COLORMASK	Mask for color attributes selection.
DIF_DISABLE	Disabling a control.
DIF_DROPDOWNLIST	Defines read-only list style.
DIF EDITEXPAND	Environment variables "expansion" in edit boxes.
DIF EDITOR	Sequentially defined edit boxes.
DIF_GROUP	Grouping of radio buttons.
DIF_HISTORY	Allows to keep history for edit boxes.
DIF HIDDEN	Hides a control.
DIF LEFTTEXT	Left-align title of a frame.
DIF LISTAUTOHIGHLIGHT	Automatic assignment of list hotkeys.
DIF_LISTNOAMPERSAND	Allows showing ampersands in the listbox.
DIF_LISTNOBOX	Display listbox without a frame.
DIF_LISTNOCLOSE	Do not close the dialog upon list item selection.
DIF_LISTWRAPMODE	"Wraps" list upon navigation.
DIF MANUALADDHISTORY	The dialog handler itself will be adding lines to the history.
DIF MASKEDIT	Defines mask in edit boxes.
DIF MOVESELECT	Change selection in radio buttons group upon input focus change.
DIF NOAUTOCOMPLETE	Disable autocompletion for input lines.
DIF_NOBRACKETS	Displays button caption without brackets.
DIF_NOFOCUS	Control item can't receive input focus.

DIF_NOTCVTUSERCONTROL	do not convert characters (CHAR_INFO::Char) while writing the virtual buffer to the screen.
DIF READONLY	Makes edit boxes "read-only".
DIF SELECTONENTRY	Edit box contents will be selected upon receiving input focus.
DIF SEPARATOR	Text string is displayed as a separator (single horizontal line).
DIF SEPARATOR2	Text string is displayed as a separator (double horizontal line).
DIF_SETCOLOR	Element color definition.
DIF SHOWAMPERSAND	Show ampersand character not using it for the hot key definition.
DIF USELASTHISTORY	Initial value is taken from the history list.
DIF_VAREDIT	Edit box without size limit.

e also:

ialog API structures

alog API

Dialog API 1.0 supports the following structures.

Structure	Description
<u>FarDialogItem</u>	Dialog item
<u>FarDialogItemData</u>	Passing data to a dialog item
<u>FarList</u>	The DI_LISTBOX list
<u>FarListColors</u>	Describes a color scheme for DI_COMBOBOX and DI_LISTBOX controls
<u>FarListDelete</u>	Parameters for deletion from DI_COMBOBOX or DI_LISTBOX
<u>FarListFind</u>	Search in a DI_COMBOBOX or DI_LISTBOX
<u>FarListGetItem</u>	Retrieval of one element from a DI_COMBOBOX or DI_LISTBOX
<u>FarListInfo</u>	Retrieval of information about a DI_COMBOBOX or DI_LISTBOX
<u>FarListInsert</u>	Item insertion into a DI_COMBOBOX or DI_LISTBOX
<u>FarListItem</u>	List item
<u>FarListPos</u>	Positioning in the list
<u>FarListItemData</u>	Association of a list item with data
<u>FarListTitles</u>	Set or get list labels
<u>FarListUpdate</u>	List item update data

e also:

FAR API structures

FarListUpdate

in | structures | FarListUpdate

The **FarListUpdate** stucture for Delphi:

```
TFarListUpdate = packed record
  Index: integer;
  Item: TFarListItem;
end;
PFarListUpdate = ^TFarListUpdate;
```

FarStandardFunctions

in | structures | FarStandardFunctions

The **FarStandardFunctions** stucture for Delphi:

```
TFarStandardFunctions = packed record
  StructSize: integer;
  atoi: TFarStdAToI;
  atoi64: TFarStdAToI64;
  itoa: TFarStdIToA;
  itoa64: TFarStdIToA64;
  sprintf: pointer;
  sscanf:
           pointer;
  qsort: TFarStdQSort;
  bsearch: TFarStdBSearch;
  qsortex: TFarStdQSortEx;
  Reserved: packed array[0..8] of DWORD;
  LISLower:
               TFarStdLocalIsLower;
  LIsUpper:
               TFarStdLocalIsUpper;
               TFarStdLocalIsAlpha;
  LIsAlpha:
  LISAlphanum: TFarStdLocalIsAlphaNum;
               TFarStdLocalUpper;
  LUpper:
               TFarStdLocalLower;
  LLower:
  LUpperBuf:
               TFarStdLocalUpperBuf;
  LLowerBuf:
               TFarStdLocalLowerBuf;
  LStrupr:
               TFarStdLocalStrUpr;
  LStrlwr:
               TFarStdLocalStrLwr;
  LStricmp:
               TFarStdLocalStrICmp;
  LStrnicmp:
               TFarStdLocalStrNICmp;
  Unquote: TFarStdUnquote;
  ExpandEnvironmentStr: TFarStdExpandEnvironmentStr;
  LTrim: TFarStdLTrim;
  RTrim: TFarStdRTrim;
  Trim: TFarStdTrim;
  TruncStr: TFarStdTruncStr;
```

```
TruncPathStr: TFarStdTruncPathStr;
  QuoteSpaceOnly: TFarStdQuoteSpaceOnly;
  PointToName: TFarStdPointToName;
 GetPathRoot: TFarStdGetPathRoot;
 AddEndSlash: TFarStdAddEndSlash;
  CopyToClipboard: TFarStdCopyToClipboard;
 PasteFromClipboard: TFarStdPasteFromClipboard;
 FarKeyToName: TFarStdKeyToKeyName;
 FarNameToKey: TFarStdKeyNameToKey;
 FarInputRecordToKey: TFarStdInputRecordToKey;
  XLAT: TFarStdXLAT;
 GetFileOwner: TFarStdGetFileOwner;
 GetNumberOfLinks: TFarStdGetNumberOfLinks;
  FarRecurseSearch: TFarRecursiveSearch;
  MkTemp: TFarStdMkTemp;
 DeleteBuffer: TFarStdDeleteBuffer;
  ProcessName: TFarStdProcessName;
  MkLink: TFarStdMkLink;
 ConvertNameToReal: TFarStdConvertNameToReal;
 GetReparsePointInfo: TFarStdGetReparsePointInfo;
end;
PFarStandardFunctions = ^TFarStandardFunctions;
```

Where:

```
TFarStdatoi = function(
 S: PChar): integer; stdcall;
TFarStdAToI64 = function(
 S: PChar): int64; stdcall;
TFarStdIToA = function(
 Value: integer;
 Str: PChar;
 Radix: integer): PChar; stdcall;
TFarStdIToA64 = function(
 Value: int64;
 Str: PChar;
```

```
Radix: integer): PChar; stdcall;
  TFarStdQSortFunc = function(
    Param1: pointer;
    Param2: pointer): integer; cdecl;
TFarStdQSort = procedure(
  Base: pointer;
  NElem: size t;
  Width: size t;
  fcmp: TFarStdQSortFunc); stdcall;
TFarStdBSearch = procedure(
  Key: pointer;
  Base: pointer;
  NElem: size_t;
 Width: size_t;
  fcmp: TFarStdQSortFunc); stdcall;
TFarStdLocalIsLower = function(
  Ch: integer): integer; stdcall;
TFarStdLocalIsUpper = function(
  Ch: integer): integer; stdcall;
TFarStdLocalIsAlpha = function(
  Ch: integer): integer; stdcall;
TFarStdLocalIsAlphaNum = function(
  Ch: integer): integer; stdcall;
TFarStdLocalUpper = function(
  LowerChar: integer): integer; stdcall;
TFarStdLocalLower = function(
  UpperChar: integer): integer; stdcall;
TFarStdLocalUpperBuf = procedure(
  Buf: PChar;
```

```
Length: integer); stdcall;
TFarStdLocalLowerBuf = procedure(
  Buf: PChar;
  Length: integer); stdcall;
TFarStdLocalStrUpr = procedure(
  s1: PChar); stdcall;
TFarStdLocalStrLwr= procedure(
  s1: PChar); stdcall;
TFarStdLocalStrICmp = function(
  s1: PChar;
  s2: PChar): integer; stdcall;
TFarStdLocalStrNICmp = function(
  s1: PChar;
  s2: PChar;
  n: integer): integer; stdcall;
TFarStdUnguote = procedure(
  Str: PChar); stdcall;
TFarStdExpandEnvironmentStr = function(
  Src: PChar;
  Dst: PChar;
  Size: DWORD): DWORD; stdcall;
TFarStdLTrim = function(
  Str: PChar): PChar; stdcall;
TFarStdRTrim = function(
  Str: PChar): PChar; stdcall;
TFarStdTrim = function(
  Str: PChar): PChar; stdcall;
TFarStdTruncStr = function(
```

```
Str: PChar;
  MaxLength: integer): PChar; stdcall;
TFarStdTruncPathStr = function(
  Str: PChar;
  MaxLength: integer): PChar; stdcall;
TFarStdQuoteSpaceOnly = function(
  Str: PChar): PChar; stdcall;
TFarStdPointToName = function(
  Path: PChar): PChar; stdcall;
TFarStdGetPathRoot = procedure(
  Path: PChar;
  Root: PChar); stdcall;
TFarStdAddEndSlash = function(
  Path: PChar): integer; stdcall;
TFarStdCopyToClipboard = function(
  Data: PChar): integer; stdcall;
TFarStdPasteFromClipboard = function: PChar; stdcall;
TFarStdKeyToKeyName = function(
  Key: integer; KeyName: PChar; Size: integer): BOOL;
TFarStdKeyNameToKey = function(
  Name: PChar): integer; stdcall;
TFarStdInputRecordToKey = function(
  var R: INPUT RECORD): integer; stdcall;
TFarStdXLAT = function(
  Line: PChar;
  StartPos: integer;
  EndPos: integer;
  TableSet: PCharTableSet;
```

```
Flags: DWORD): PChar; stdcall;
TFarStdGetFileOwner = function(
  Computer: PChar;
  Name: PChar;
  Owner: PChar): integer; stdcall;
TFarStdGetNumberOfLinks = function(
  Name: PChar): integer; stdcall;
TFRSFunction = function(
  var FindData: TWin32FindDataEx;
  FullName: PChar): integer; stdcall;
TFarRecursiveSearch = procedure(
  InitDir: PChar;
  Mask: PChar;
  Func: TFRSFunction;
  Flags: DWORD); stdcall;
TFarStdMkTemp = function(
  Dest: PChar;
  Template: PChar): PChar; stdcall;
TFarStdDeleteBuffer = procedure(
  Buffer: PChar); stdcall;
TFarStdProcessName = function(
  Param1: PChar;
  Param2: PChar;
  Flags: DWORD): integer;
TFarStdMkLink = function(
  const Src: PChar;
  const Dest: PChar;
  Flags: DWORD): integer;
TFarStdConvertNameToReal = function(
  const Src: PChar;
```

```
Dest: PChar;
DestSize: integer): integer;
TFarStdGetReparsePointInfo = function(
const Src: PChar;
Dest: PChar;
DestSize: integer): integer;
```

ypes and definitions

in

/pes

Туре	Description
<u>FARWINDOWPROC</u>	Dialog window callback function.
<u>FRSUSERFUNC</u>	Callback function for the <u>FSF.FarRecursiveSearch</u> function.

acros

Масто	Description
MAKEFARVERSION	generates the needed FAR Manager version.

onstants

Constant	Description
<u>COL *</u>	color indexes in the FAR color scheme (see farcolor.hpp).
FARMACRO KEY EVENT	the type of input event passed while "playing" a macro sequence.
FARMANAGERVERSION	FAR Manager version number.
FAR NO NAMELESS UNIONS	"Pure C".
<u>KEY *</u>	key codes used in FAR Manager (see farkeys.hpp).
<u>NM</u>	size of the buffer needed to receive a full file name.
<u>OPM *</u>	additional information about function operation mode and place, from which it was called.
<u>FPS *</u>	information about the panel settings.
<u>FSS *</u>	information about the system settings.
<u>FDS *</u>	information about the file description settings.
<u>FCS *</u>	information about the confirmation settings.
<u>FIS</u> *	information about the interface settings.
<u>FDIS</u> *	information about the dialog settings.
<u>PKF *</u>	state of the shift keys of an event sent to the <u>ProcessKey</u> function
d	1

<u>SM</u>	*			
-----------	---	--	--	--

e also:

Exported functions, Service functions, Structures, Archive support, Addons

lacros and types

/pes

Туре	Description
FARWINDOWPROC	Dialog window callback function.

acros

Масто	Description
Dlg_GetDlgData(Info,hDlg)	Retrieve the 32-bit data value associated with the dialog (<u>DM_GETDLGDATA</u>).
Dlg_GetDlgItemData(Info,hDlg,ID)	Retrieve the 32-bit data value associated with a dialog item (<u>DM_GETITEMDATA</u>).
Dlg_RedrawDialog(Info,hDlg)	Redraw the entire dialog window (<u>DM_REDRAW</u>).
Dlg_SetDlgData(Info,hDlg,Data)	Set the 32-bit data value associated with the dialog (<u>DM_SETDLGDATA</u>).
Dlg_SetDlgItemData(Info,hDlg,ID,Data)	Set the 32-bit data value associated with a dialog item (<u>DM_SETITEMDATA</u>).
DlgEdit_AddHistory(Info,hDlg,ID,Str)	Add an item to the history of an input line (<u>DM_ADDHISTORY</u>).
DlgItem_Disable(Info,hDlg,ID)	Disable dialog item (<u>DM_ENABLE</u>).
DlgItem_Enable(Info,hDlg,ID)	Enable dialog item (<u>DM_ENABLE</u>).
DlgItem_GetCheck(Info,hDlg,ID)	Retrieve the state of <u>DI_CHECKBOX</u> and <u>DI_RADIOBUTTON</u> dialog items (<u>DM_GETCHECK</u>).
DlgItem_GetFocus(Info,hDlg)	Retrieve the ID of the dialog item that has the keyboard focus (<u>DM_GETFOCUS</u>).
DlgItem_IsEnable(Info,hDlg,ID)	Check if a dialog item is disabled (<u>DM_ENABLE</u>).
DlgItem_SetCheck(Info,hDlg,ID,State)	Set the state of <u>DI CHECKBOX</u> and <u>DI RADIOBUTTON</u> dialog items to one of three allowed states (<u>DM SETCHECK</u>).

DlgItem_SetFocus(Info,hDlg,ID)	Set keyboard focus to the given dialog item (<u>DM_SETFOCUS</u>).
DlgItem_SetText(Info,hDlg,ID,Str)	Set new string value for an input line or new caption for an item (<u>DM_SETTEXTPTR</u>).
DlgList_AddString(Info,hDlg,ID,Str)	Add a string to a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list (<u>DM_LISTADDSTR</u>).
DlgList_ClearList(Info,hDlg,ID)	Clear a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list (<u>DM_LISTDELETE</u>).
DlgList_DeleteItem(Info,hDlg,ID,Index)	Delete an item from a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list (<u>DM_LISTDELETE</u>).
DlgList_GetCurPos(Info,hDlg,ID)	Retrieve the current cursor position in a <u>DI COMBOBOX</u> or <u>DI LISTBOX</u> list (<u>DM LISTGETCURPOS</u>).
DlgList_GetItemData(Info,hDlg,ID,Index)	Retrieve data associated with an item in a <u>DI_COMBOBOX</u> or <u>DI_LISTBOX</u> list (<u>DM_LISTGETDATA</u>).
DlgList_SetCurPos(Info,hDlg,ID,NewPos)	Set new cursor position in a <u>DI COMBOBOX</u> or <u>DI LISTBOX</u> list (<u>DM LISTSETCURPOS</u>).
DlgList_SetItemStrAsData(Info,hDlg,ID,Index,Str)	Associate a string with an item of a <u>DI COMBOBOX</u> or <u>DI LISTBOX</u> list (<u>DM LISTSETDATA</u>).
DlgList_SortDown(Info,hDlg,ID)	Sort the items of a <u>DI COMBOBOX</u> or <u>DI LISTBOX</u> list in descending order (<u>DM LISTSORT</u>).
DlgList_SortUp(Info,hDlg,ID)	Sort the items of a <u>DI COMBOBOX</u> or <u>DI LISTBOX</u> list in ascending order (<u>DM LISTSORT</u>).

Here:

Info

Reference to the <u>PluginStartupInfo</u> structure.

hDlg

Dialog handle.

ID

The index of the dialog item in the <u>FarDialogItem</u> structure.

Data

32-bit value (numeric value of DWORD type or a pointer).

Str

Null-terminated string.

State

Item state (one of the BSTATE_* constants, described here <u>DM_SETCHECK</u>).

Index

Position in a list.

NewPos

New position in a list.

e also:

Types and definitions

rchive support

in | Custom API

🗹 Note

This information is valid only for MultiArc plugin which comes with the standard distribution of FAR Manager!

For specific archive formats support, MultiArc plugin uses second-level plugins (subplugins) - the so called FMT-modules. These are all the * . FMT files contained within Formats subfolder of MultiArc (but not in subfolders of Formats).

Function	Description
<u>CloseArchive</u>	Close archive.
<u>GetArcItem</u>	Get the information about the next archive element.
<u>GetDefaultCommands</u>	Get archiver command strings used by default
<u>GetFormatName</u>	Get archive format name.
<u>GetSFXPos</u>	Get the position of the beginning of the archive.
<u>IsArchive</u>	Check whether the specified file is an archive.
<u>LoadFormatModule</u>	Called when a subplugin is loaded.
<u>OpenArchive</u>	Open an archive and prepare it for reading.
<u>SetFarInfo</u>	Passes global settings to the plugin.

unctions exported by FMT-modules

tructures used by FMT-modules

Structure	Description
ArcInfo	Common information about an archive
ArcItemInfo	Information about a specific archive element

emarks

- 1. You can use subplugin technology in your own plugins. Examples of such plugins:
 - Expression Calculator (Alexey Torgashin)
 - Search and Replace (Ivan Sintyurin)
 - Address Book (Pavel Kostromitinov)

• Mr.Ripper (Vladimir Kubyshev)

e also:

Exported functions, Service functions, structures, Addons, custom.ini file format

olorer library regular expressions interface

in | Custom API

Regular expressions are funny things and if needed they can be quite easily used in your program/plugin. You can use colorer regexps in the most native way they come in C++ sources - as a standalone class. But then, if you use different languages or there's some other reason, you can always use them in precompiled DLL binary form.

The sources themselves are found in the colorer distribution (beginning with the "freecraze" version), and the DLL can be compiled in **icolorer** directory, which is also found in the colorer distribution. In case you're using the sources you've got to work with the CRegExp class and its methods, and if you're using the DLL you simply export caller functions for these methods.

It's good to mention my regexps are slightly incorrect by formal approach - it means they don't follow perl standards when applied to some complex situations. Plus they are not fully compatible with the latter in ways of syntax of some complex operators. But, all was made with speed and colorer in mind - so it pays when it comes to simplicity.

Method	Description
CRegExp	Regexp object constructor and destructor.
isok	Latest expression compilation result.
<u>geterror</u>	Extended error information.
<u>SetNoMoves</u>	Allow/dissallow moving inside the target string.
<u>SetBkTrace</u>	Set references for \yN operator.
<u>SetExpr</u>	Expression compilation.
<u>SetCodePage</u>	Set transliteration table for texts in non-OEM codepage.
Parse	Parse regexp against target string.
Parse	Parse regexp against target string, with extended settings.

ethods

tructures:

Structure	Description

e also:

Addons

eneral background

in | Macros

Every macro command has:

- keyboard shortcut used to execute the recorded sequence;
- additional <u>parameter set</u> affecting the way and specifying the conditions under which the execution will take place.

ATTENTION! Please, keep in mind that macro commands are divided by <u>areas of</u> execution, i.e. context of the FAR application from which the macro command is going to be executed.

Macros can contain the <u>special commands</u>, providing some service functions and operators including conditional and iterative sequence execution etc.

lease note:

- FAR ignores macros that have error in the macro name or sequence parameter "Sequence".
- Macro commands are not case sensitive.
- During startup FAR Manager loads all macro commands defined in the macro command storage in the registry. Any macro sequence recorded is going to be put into the registry immediately (if macro command autosave option is on).
- Macro command is considered to be inactive if its name begins with '~' character.
- FAR Manager translates mouse wheel events into the pseudo keystrokes: - scroll the wheel up - "MsWheelUp"
 - scroll thw wheel down "MsWheelDown"
 - You can use these pseudo keys with Ctrl, Shift and Alt modifier keys.
- If you trying to enter a character using Alt-number combination FAR saves the combination, not the resulting character.

For instance, Alt-151 will be saved by FAR as "Alt00151".

- Special keys that have a virtual code other than 0xFF, but names of which are not known to FAR, will be stored as "OemXXXXX" (here XXXXX is the decimal virtual key code with leading zeroes).
- Special keys that have a virtual code of 0xFF, will be stored as "SpecXXXXX" (here XXXXX is the decimal scan code of the key with
leading zeroes).

- We recommend using special plugin "Macro Browser" by Konstantin Melnikov to edit and organize macro commands. This FAR module provides the functionality for viewing, editing, copying and moving macro commands between areas of execution, copying and moving macros between different FAR Manager settings, exporting, creating and removing existing FAR Manager macro commands.
- The value of the "NoSendKeysToPlugins" macro command parameter depends on the way the <u>macro command recording</u> had started.
- CtrlBreak breaks macro execution. Macro assigned to CtrlBreak does not trigger in this case..

For more convenient and comfortable management of the macro commands you can use the special plugin <u>MacroView Manager</u>.

e also:

Using macros, Macro-language, Technical details, Examples

lacro-language

in | <u>Macros</u>

FAR implements a simple macro-language that brings logic into the keystroke sequence and raises macro commands to the higher level turning the macros in combination with the plugins into a powerful tool that extends FAR Manager functionality. Macro-language is highly specialized and thus cannot be considered as an universal language.

Macro-language functionality:

- <u>Macro-commands</u>
- Conditional operators
- <u>Boolean object states</u>
- <u>Functions</u>
- **Operations**
- <u>Object states</u>
- <u>Variables</u>
- <u>Constants</u>

otes

- 1. It is impossible to use macro-language elements while <u>recording</u> a macro in a usual way. Macro-language elements can be added to the sequence only by editing the registry manually or by using special applications or FAR plugins.
- 2. Named keys (e.g. CtrlK) can be present in any expressions; in this case they are treated as numbers.

e also: <u>General background, Using macros, Technical details, Examples</u>

sing macros

<u>in | Macros</u>

FAR provides two modes for recording and execution of the macro commands:

- 1. General mode: all keystrokes **will** be passed to the plugins during macro recording and execution.
- 2. Special mode: all keystrokes **won't** be passed to the plugins intercepting the editor events during recording and execution.

Let's imagine we have a plugin handling Ctrl+A key combination. Using special mode we can disable this plugin receiving this notification and thus prevent it from performing some action it should perform normally handling this keystroke.

So what can we do with the macros:

Record Delete

You can execute macro command by pressing the key combination <u>assigned</u> by the user. Execution <u>parameters</u> can be specified while recording the macro command and the only way to change them is to record certain macro over again.

e also:

<u>General background</u>, <u>Macro-language</u>, <u>Techniacl details</u>, <u>Examples</u>

xamples

in | <u>Macros</u>

REGEDIT4

;open the disks menu in the passive file panel ;make the panel for which the menu is going to be $\mathsf{op}\varepsilon$

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Disks\Tab] "Sequence"="Esc \$If (!PPanel.Visible) \$If (APanel.Lef "DisableOutput"=dword:00000001

REGEDIT4

;select/unselect a single word under the text cursor

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Editor\RAlt "Sequence"="RCtrl9 CtrlRight CtrlLeft \$If (!Selected) "DisableOutput"=dword:00000001

REGEDIT4

;create a new folder with name consisting of the curr

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlS
"Sequence"="%folder=date(\"%d.%m0.%Y\"); $If (!panel.
"DisableOutput"=dword:0000001
"NoPluginPanels"=dword:0000001
```

REGEDIT4

;quick find the file in the passive panel with the sa ;in the active panel ;sequence F5 ShiftEnter CtrlIns is used to get the fi ;of the state of the command line and number of selec

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlA
"Sequence"="$IClip $If (((Bof && APanel.Root) || !Bof
$If (fexist(PPanel.Path+\"\\\\\"+APanel.Current) == 1
```

CtrlIns Esc Tab Home Alt< ShiftIns Esc \$End \$End" "DisableOutput"=dword:00000001

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlA "Sequence"="\$IClip \$If (((Bof && APanel.Root) || !Bof \$If (fexist(PPanel.Path+\"\\\\\"+APanel.Current) == 1 Esc Tab Home Alt< ShiftIns Esc \$End \$End" "DisableOutput"=dword:00000001

REGEDIT4

;select 30 next/previous files or folders

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlC "Sequence"="\$If (!Eof) \$Rep (30) \$If (Eof) Ins \$Exit "DisableOutput"=dword:00000001

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlL "Sequence"="Up \$If (Bof) \$If (!APanel.Root) Down \$Enc \$If (Eof) Ins \$Else Ins Up \$End \$Rep (29) Up Ins Up \$ \$If (!APanel.Root) Down \$End \$Exit \$End \$End Up" "DisableOutput"=dword:00000001

REGEDIT4

;capture the whole screen to the text file far-screer ;this macro works only in the Dialog area of executic

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Dialog\AltF "Sequence"="\$IClip AltIns CtrlHome ShiftEnd ShiftPgDr \$If (!Shell) Esc \$End \$If (Shell && !APanel.Plugin) S \"far-screen.out\" Enter ShiftIns Enter Enter F2 Esc "DisableOutput"=dword:00000001

REGEDIT4

;delete the file or folder using Del hotkey ;if the cursor was on the .. element then try to dele ;if command line cursor is not in the end of the comm

```
[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\Del]
"Sequence"="$If (!CmdLine.Eof) Del $Exit $Else
$If ((APanel.Bof && !APanel.Selected) &&
(!APanel.Root || APanel.Plugin)) CtrlPgUp $End F8 $En
"DisableOutput"=dword:00000001
```

REGEDIT4

;maximize the passive panel before showing the quickv ;restore the panel size before closing the quickview

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlQ "Sequence"="Tab \$If (QView) Tab Ctrl2 Tab CtrlClear \$ \$If (APanel.Left) \$Rep (APanel.width) CtrlRight \$End \$Rep (APanel.width) CtrlLeft \$End \$End \$End Tab CtrlQ "DisableOutput"=dword:00000001

This macro is intended to highlight HTML file/folder pairs:

REGEDIT4

[HKEY_CURRENT_USER\Software\Far\KeyMacros\Shell\CtrlS "Sequence"="%Current=APanel.Current; %Ext=substr(%Cur \$If (%Ext==\"_files\") %Name=substr(%Current,0,len(%C %Ext=fsplit(%Current,FSPLIT_EXT); %Name=fsplit(%Curre \$if (%Ext==\".htm\" || %Ext==\".html\") panel.setpos(panel.setpos(ACTIVEPANEL,%Name+\"_files\") \$end \$if (panel.setpos(ACTIVEPANEL,%Name+\".htm\") panel.setpos \$if (APanel.Current!=%Current) \$If(!panelitem(ACTIVEF panel.setpos(ACTIVEPANEL,%Current) \$If(!panelitem(ACTIVEF panelitem(ACTIVEF) panelsetpos(ACTIVEPANEL,%Current) \$If(!panelitem(ACTIVEF) panelsetpos(ACTIVEPANEL,%Current) \$If(!panelitem(ACTIVEF) panelsetpos(ACTIVEPANEL,%Current) \$If(!panelitem(ACTIVEF) panelsetpos(ACTIVEPANEL,%Current) \$If(!panelitem(ACTIVEF)) panelsetpos(ACTIVEPANEL) \$If(!panelitem(ACTIVEF)) panelsetpos(ACTIV

;Macro is not called when command line is not empty, ;action of CtrlShiftIns combination - Copy names of s "EmptyCommandLine"=dword:00000001

educe EXE and DLL Size with LIBCTINY.LIB

<u>in | articles</u>

Matt Pietrek Original: <u>http://msdn.microsoft.com/msdnmag/issues/01/01/hood/de</u>

Download the code for this article: <u>Hood0101.exe (45KB)</u>

Way back in my October 1996 column in *MSJ*, I addressed a question concerning the size of executable files. Back then, a simple Hello World program compiled to a 32KB executable. Two compiler versions later, the problem is only slightly better. The same program with the Visual C++ 6.0 compiler is now 28KB.

In that column, I provided a replacement runtime library that lets you create very small executable programs. There were some restrictions on what situations it was useful for, but for a large number of my own programs it worked well. After living with these restrictions for quite a while, I decided it was time to fix some of them. Making these modifications also happens to provide a great opportunity to describe a little-known linker option that can be used to further reduce program size.

XE and DLL Size

Before jumping into the code for my replacement runtime library, it's worth taking the time to review why simple EXEs and DLLs are bigger than you might expect. Consider the canonical Hello World program:

```
#include <stdio.h>
void main()
{
    printf ("Hello World!\n" );
}
```

Let's compile this program for size, and generate a map file. Using the command-line Visual C++ compiler, the syntax would be:

```
Cl /O1 Hello.CPP /link /MAP
```

First, look at the .MAP file; a trimmed down version is shown in Figure 1. From

looking at the addresses of main (0001:0000000) and of printf (0001:000000C), you can infer that function main's code is only 0xC bytes in length. Looking at the last line of the file, the _____chkstk function at address 0001:00003B10, you can also infer that there's at least 0x3B10 bytes of code in the executable. That's over 14KB of code to send Hello World to the screen.

Now, start looking through some of the other .MAP file lines. Some items make sense, for example, the ____initstdio function. After all, printf writes its output to a file, so some amount of underlying runtime library support routines for stdio makes sense. Likewise, it's reasonable to expect that the printf code might call strlen, so its inclusion isn't a surprise.

However, take a look at some of the other functions, for instance ___sbh_heap_init. This is the initialization function for the runtime library's small block heap. The Win32-based operating systems offer up their own heap in the form of the HeapAlloc family of functions. Potential performance gains notwithstanding, the Visual C++ library could choose to use the Win32 heap APIs, but doesn't. Thus, you end up with more code than necessary in your executable.

While some people might not care that the runtime library implements its own heap, there are other less defensible examples. Consider the __crtMessageBoxA function near the bottom of the map file. This function allows the runtime library to call the MessageBox API without forcing the executable to link against USER32.DLL. For a simple Hello World program, it's hard to anticipate the need to call MessageBox.

Consider another example: the __crtLCMapStringA function, which does localedependent transformations of strings. While Microsoft is somewhat obligated to provide locale support, it's not really needed for a large number of programs. Why make programs that don't use locales pay the cost for those that do?

I could continue with other examples of unneeded code, but I've made my point. A typical small program contains lots of little nuggets of code that aren't used. By themselves, they don't contribute much to the code size, but add up all the cases and you're into serious amounts of code!

hat About the C++ Runtime Library DLL?

Alert readers might say, "Hey Matt! Why don't you just use the DLL version of the runtime library?" In the past, I could make the argument that there was no consistently named version of the C++ runtime library DLL available on Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, and so forth.

Luckily, we've moved past those days, and in most cases you can rely on MSVCRT.DLL being available on your target machines.

Making this switch and recompiling Hello.CPP, the resulting executable is now only 16KB. Not bad, but you can do better. More importantly, you're just shifting all of this unneeded code to someplace else (that is, to MSVCRT.DLL). In addition, when your program starts up, another DLL will have to be loaded and initialized. This initialization includes items like locale support, which you may not care about. If MSVCRT.DLL suits your needs, then by all means use it. However, I believe that using a stripped-down, statically linked runtime library still has merit.

I may be tilting at windmills here, but my e-mail conversations with readers show that I'm not alone. There are people out there who want the leanest possible code. In this day of writeable CDs, DVDs, and fast Internet connections, it's easy not to worry about code size. However, the best Internet connection I can get at home is only 24Kbps. I hate wasting time downloading bloated controls for a Web page.

As a matter of principle, I want my code to have as small a footprint as possible. I don't want to load any extra DLLs that I don't really need. Even if I might need a DLL, I'll try to delayload it so that I don't incur the cost of loading it until I use the DLL. Delayloading is a topic I've described in previous columns, and I strongly encourage you to become familiar with it. See <u>Under the Hood</u> in the December 1998 issue of *MSJ* for starters.

igging Deeper

Now that I've beaten up the unneeded code within the program, let's turn to the executable file itself. If you were to run DUMPBIN /HEADERS on my Hello.EXE, you'd see the following two lines in the output:

```
1000 section alignment
1000 file alignment
```

The second line is interesting. It says that every code and data section in the executable is aligned on a 4KB (0x1000) byte boundary. Because sections are stored contiguously in a file, it's not hard to see the potential for wasting up to 4KB between the end of one section and the start of the next.

If I had linked the program with a version of the linker that came before Visual C++ 6.0, I would have seen something different, as you see here:

```
1000 section alignment
```

200 file alignment

The key difference is that the alignment between sections is only 512 bytes (0x200). There's much less space available to waste. In Visual C++ 6.0, the linker defaults were changed to make the file alignment of sections equal to the alignment in memory. This provides a slight load-time performance improvement on Windows 9x, but makes executables bigger.

Luckily, the Visual C++ linker has a way to go back to the previous behavior. The magic switch is /OPT:NOWIN98. Rebuilding Hello.CPP as before, but with the addition of this linker switch gets the executable file down to 21KB a savings of 7KB. If I switch to linking with MSVCRT.DLL and using /OPT:NOWIN98, the executable size drops to 2560 bytes!

BCTINY: A Minimal Runtime Library

Now that you understand the problem of why simple EXEs and DLLs are so large, it's time to introduce my new and improved replacement runtime library. In the October 1996 column (mentioned earlier), I created a small static .LIB file designed to replace or augment the Microsoft LIBC.LIB and LIBCMT.LIB libraries. I called this replacement runtime library LIBCTINY.LIB, since it was a very stripped-down version of Microsoft's own runtime library sources.

LIBCTINY.LIB is intended for simple applications that don't require a huge amount of runtime library support. Thus, it's not suitable for MFC applications or other complicated scenarios that make extensive use of the C++ runtime. LIBCTINY's ideal target is small programs or DLLs that call some Win32 APIs and perhaps display some simple output.

There are two guiding principles behind LIBCTINY.LIB. First, it replaces the standard Visual C++ startup routines with much simpler code. This simpler code doesn't refer to any of the more esoteric runtime library functions like

___crtLCMapStringA. Because of this, much less extraneous code is linked into your binary. As I'll show shortly, the LIBCTINY routines perform a bare minimum of tasks before calling your WinMain, main, or DllMain routines.

The second guiding principle of LIBCTINY.LIB is to implement relatively large functions like malloc or printf with code that's already in the Win32 system DLLs. Beyond the minimal startup code, most of the other LIBCTINY source files are simple implementations of standard C++ runtime library functions such as malloc, free, new, delete, printf, strupr, strlwr, and so on. Take a look at the implementation of printf in printf.cpp (see Figure 2) to get an idea of what I'm talking about.

In my original version of LIBCTINY.LIB there were two restrictions that annoyed me. First, the original version did not support DLLs. You could make tiny console and GUI executable programs, but if you wanted to create a tiny DLL, you were out of luck.

Second, the original LIBCTINY did not support static C++ constructors and destructors. By this, I mean constructors and destructors declared at global scope. In the new version, I've added the basic code that implements this support. Along the way, I learned quite a bit about how the compiler and runtime library play a complicated game to make static constructors and destructors work.

he Dark Underbelly of Constructors

When the compiler processes a source file that has a static constructor, it generates two things. The first is a small blob of code with a name like \$E2 that calls the constructor. The second thing the compiler emits is a pointer to this blob of code. This pointer is written to a specially named section in the .OBJ called .CRT\$XCU.

Why the funny section name? It's a bit complicated. Let me throw another piece of data at you to help explain. If you examine the Visual C++ runtime library sources (for instance, CINITEXE.C), you'll find the following:

```
#pragma data_seg(".CRT$XCA")
_PVFV __xc_a[] = { NULL };
#pragma data_seg(".CRT$XCZ")
_PVFV __xc_z[] = { NULL };
```

The previous lines of code create two data segments, .CRT\$XCA and .CRT\$XCZ. In each segment it places a variable (__xc_a and __xc_z, respectively). Note that the segment names are very similar to the .CRT\$XCU segment to which the compiler emits the constructor code pointer.

At this point, a little linker theory is needed. When processing all of the segments to create the final portable executable (PE) file, the linker concatenates all the data from identically named segments. Thus, if A.OBJ has a section called .data, and B.OBJ also has a .data section, all the data from A.OBJ and B.OBJ will be written contiguously into a single .data section in the PE file.

The use of a \$ in a segment name puts a new twist on things. When encountering segment names with a \$ in them, the linker treats the portion of the name

preceding the \$ as the final segment name. Thus, the .CRT\$XCA, .CRT\$XCU, and .CRT\$XCZ segments all end up together in the final executable in a segment called .CRT.

What about the part of the segment name following the \$? When combining these types of sections, the linker writes out the segments in the order dictated by the string following the \$. The ordering is alphabetical, so all the data from .CRT\$XCA goes first, followed by all of the data from .CRT\$XCU, and finally all of the data from .CRT\$XCZ. This is a crucial point to understand.

What's going on here is that the runtime library code has no idea how many static constructor calls are needed for a given EXE or DLL. However, it does know that only pointers to constructor code blobs will be in the .CRT\$XCU segment. When the linker concatenates all the .CRT\$XCU sections, it has the net effect of creating a function pointer array. By defining .CRT\$XCA and .CRT\$XCZ segments along with the __xc_a and __xc_z symbols, the runtime library can reliably locate the beginning and end of the function pointer array.

As you might expect, calling all the static constructors in a module is a simple matter of enumerating through the function pointer array, calling each pointer in turn. The routine that does this is _initterm, shown in Figure 3. This routine is identical to the version from the Visual C++ runtime library sources.

All things considered, getting static constructors to work in LIBCTINY was relatively easy. It was mostly a matter of defining the right data segments (specifically, .CRT\$XCA and .CRT\$XCZ), and calling __initterm from the correct spot in the startup code. Getting static destructors to work was a bit trickier.

Unlike the function pointer array that the compiler and linker conspire to create for static constructors, the list of static destructors to call is built at runtime. To build this list, the compiler generates calls to the atexit function, which is part of the Visual C++ runtime. The atexit function takes a function pointer and adds the pointer to a first-in, last-out list. When the EXE or DLL unloads, the runtime library iterates through the list and calls each function pointer.

LIBCTINY's implementation of the atexit functionality is significantly simpler than what the Visual C++ runtime library does. There are three functions and a handful of static variables for this support, which is also in initterm.cpp. The _atexit_init function simply allocates an array to hold 32 function pointers, and stores the pointer in the pf_atexitlist static variable.

The atexit function checks to see if there's room in the array, and if so, adds the pointer to the end of the list. A more robust version of this code would reallocate the array to a larger size if necessary. Finally, the _DoExit function uses your

friend, __initterm, to iterate through the array and call each function pointer. In an ideal world, _DoExit would iterate through the array in reverse order, mimicking the behavior of the Visual C++ runtime library implementation. However, the whole purpose of LIBCTINY is to be simple and small, rather than striving for perfect compatibility.

BCTINY's Minimal Startup Routines

Now let's take a look at LIBCTINY's new support for small DLLs. As with EXEs, the trick is to make the DLL's entry point code as small as possible and omit calls to unneeded routines that bring in lots of other code. Figure 4 shows the minimal DLL startup code. When your DLL is loaded, it is this code, not your DllMain routine, that executes first.

The _DllMainCRTStartup is the very first place execution begins in your DLL. In LIBCTINY's implementation, it first checks to see if the DLL is in its DLL_PROCESS_ATTACH call. If so, the code calls _atexit_init (described earlier), and _initterm to invoke any static constructors. The heart of the function is the call to DllMain, which is the routine you supply as part of your DLL's code. This DllMain call is made for all four notification types (process attach/detach, and thread attach/detach).

The last thing DllMainCRTStartup does is to check if the DLL is in its DLL_PROCESS_DETACH code. If so, the code calls _DoExit. As described earlier, this causes any static destructors to be called. If you're curious about the startup code for console and GUI mode EXEs, be sure to check out CRT0TCON.CPP and CRT0TWIN.CPP, respectively. (These modules accompany the code download, found at the link at the top of this article.)

One other thing worth checking out in DLLCRTO.CPP (see <u>Figure 4</u>) is this line near the top:

```
#pragma comment(linker, "/OPT:NOWIN98")
```

This puts a linker directive into the DLLCRT0.OBJ file that tells the linker to use the /OPT:NOWIN98 switch. The benefit is that you don't have to manually add /OPT:NOWIN98 to your make files or project files by hand. I figure if you're using LIBCTINY, you'd probably want to use /OPT:NOWIN98 as well.

sing LIBCTINY.LIB

Using LIBCTINY is very simple. All you have to do is add LIBCTINY.LIB to the linker's list of .LIB files to search. If you're using the Visual Studio IDE, this

would be in the Projects | Settings | Link tab. It doesn't matter what type of binary you're building (console EXE, GUI EXE, or DLL), since LIBCTINY.LIB contains appropriate entry point routines for each of them.

Take a look at TEST.CPP in Figure 5. This program simply exercises a few of the routines that LIBCTINY.LIB implements, and includes a static constructor and destructor invocation. When I compile it normally with Visual C++ 6.0,

CL /01 TEST.CPP

the resulting executable is 32768 bytes. By simply adding LIBCTINY.LIB to the command line

CL /01 TEST.CPP LIBCTINY.LIB

the resulting executable shrinks to 3072 bytes.

You might be wondering about the runtime library routines that LIBCTINY doesn't implement. For instance, in TEST.CPP, there's a call to strrchr. There's no problem here because that function exists in the regular LIBC.LIB or LIBCMT.LIB that Visual C++ provides. Both LIBCTINY.LIB and LIBC.LIB implement a variety of routines. LIBCTINY's list is obviously smaller than what LIBC.LIB provides. The important thing for your purposes is that the linker finds the LIBCTINY routines first when resolving function calls, and so LIBCTINY's routines are what's used. If something isn't implemented in LIBCTINY, the linker finds it in LIBC.LIB instead.

Finally, it's worth repeating that LIBCTINY isn't suitable for all purposes. For example, if your code makes use of multiple threads and relies on the runtime library's per-thread data support, then LIBCTINY isn't for you. What I do is try LIBCTINY with a prospective program. If it works, great! If not, I simply use the normal runtime library.

etadata Article Correction

In my October 2000 *MSDN Magazine* article "Avoiding DLL Hell: Introducing Application Metadata in the Microsoft .NET Framework," I said that using the Visual C++ 6.0 #import directive causes the compiler to read in a COM type library and generate ATL-ready header files for all the interfaces contained within. While header files are generated by #import, it turns out they don't use ATL.

Richard Grimes, author of Professional ATL COM Programming (Wrox Press,

1998), kindly pointed out to me that #import generates what Microsoft calls "compiler COM support classes," which are supported by the COMDEF.H header. Richard goes on to say, "There are many differences between the COM compiler support classes and the equivalent in ATL. The most important is that ATL does not use C++ exceptions. In fact, the ATL classes are more lightweight than the COM compiler support classes and so I would have preferred if Microsoft had decided to generate ATL code."

I have to confess that I should have studied this more before I wrote it. My experience with ATL is limited to the wizards in Visual C++, and tweaking the resulting code. I have used #import on a few occasions, but not enough to have made the connection that the resulting code wasn't ATL. Thanks to Richard for pointing this out to me, and for giving me even more incentive to verify everything before I write about it.

Matt Pietrek does advanced research for the NuMega Labs of Compuware Corporation, and is the author of several books. His Web site, at <u>http://www.wheaty.net/</u>, has a FAQ page and information on previous columns and articles.

From the January 2001 issue of MSDN Magazine.

Iorking with lists

in | Dialog API | Dialog items

FAR Manager 1.70 beta 4 offers the following scheme for working with DI_COMBOBOX and DI_LISTBOX lists:

- Dialog[Ex] start FarDialogItem.Param.ListItems - only for first initialization.
- DM_LISTADD Adds list item only, without associated data.
- DM_LISTINSERT Inserts list item only, without associated data.
- DM_LISTUPDATE Deletes UserData for the list item to be updated. Updates list item only, without associated data.
- DM_LISTSETDATA Associates new data with the list item, old data is deleted (if memory was allocated for it).
- DM_LISTGETDATA Returns value previously set by DM_LISTSETDATA.
- DM_LISTGETDATASIZE Returns size of associated data.
- DM_LISTGETITEM Returns list item only, without associated data.
- DM_GETDLGITEM Sets FarDialogItem.Param.ListPos variable.
- Dialog[Ex] completion
 Setting FarDialogItem.Param.ListPos variable.
 Deleting associated data (if memory was allocated for it).

e also:

DefDlgProc, DialogEx, SendDlgMessage

earch and Replace plugins programming

stom API

Search and Replace (S&R) is a FAR plugin that allows to search and replace information both in single file and in multiple ones. It also works in the viewer and the editor.

SRP-modules API information (help file srplugins.chm is needed) is available from the plugin's author: Ivan Sinturin <u>http://www.moris.ru/~spinoza</u>. Also the on-line version is available here:

http://www.moris.ru/~spinoza/download/s_and_r/srplugins

Ir.Ripper API

stom API

Mr.Ripper - is a **FAR** plugin, allowing to rip different files from other files' bowels. It understands more than 20 file formats, among them WAV, AVI, BMP, JPG, PNG, GIF, DJVU, MP3, MOD, XM, IT, RTF and others. Also it supports more than 15 package file formats of different game engines. Format supporting is made through subplugins, both for standalone formats and game packages. To develop RIP-modules you will need the ripapi.chm help file which describes the API. The help file is available from the plugin author: Vladimir

Kubyshev http://vovan.dankov.net/mrripper/doc_ripapi.html.

EditorSaveFile

in | structures

The **EditorSaveFile** stucture for Delphi.

```
TEditorSaveFile = packed record
FileName: packed array[0..Pred(NM)] of char;
FileEOL: PChar;
end;
PEditorSaveFile = ^TEditorSaveFile;
```

e also: structures | EditorSaveFile

ARMACRO_KEY_EVENT

in | types and definitions

The **FARMACRO_KEY_EVENT** constant defines the type of input event in the **INPUT_RECORD** structure, which is sent to the **ProcessEditorInput** function while "playing" a macro sequence. The Event parameter contains the KEY EVENT RECORD structure with information on the event.

Attention! This message is defined for FAR Manager starting with build 1663 and is recieved while "playing" a macro sequence.

Figure 1 Hello World .MAP File

Publics by Value	Rva+Base
main	00401000
printf	0040100c
	0040103c
amsg exit	0040111c
stbuf	00401165
ftbuf	004011f2
output	0040122f
initstdio	00401a39
endstdio	00401ade
cinit	00401af2
_exit	00401b1f
exit	00401b30
XcptFilter	00401bf4
setenvp	00401d78
setargv	00401e31
crtGetEnvironmentStringsA	0040207e
ioinit	004021bC
heap_init	0040235k
global_unwind2	00402398
local_unwind2	004023da
NLG_Return2	00402432
abnormal_termination	00402442
NLG_Notify1	00402465
NLG_Notify	0040246e
NLG_Dispatch	00402481
except_handler3	00402490
seh_longjmp_unwind@4	0040254c
FF_MSGBANNER	00402568
NMSG_WRITE	004025a1
_malloc	004026f4
nh_malloc	00402706
heap_alloc	00402732
isatty	00402768
_fflush	0040278e
flush	004027c9
	<pre>Publics by Value _main _printf _mainCRTStartup _ amsg_exit _stbuf _ftbuf _output initstdio endstdio endstdio cinit _exit _exit _xcptFilter _setenvp _setargv crtGetEnvironmentStringsA _ioinit _heap_init _global_unwind2 _local_unwind2 _local_unwind2 _local_unwind2 _local_unwind2 _NLG_Return2 _abnormal_termination _NLG_Notify1 _NLG_Dispatch _except_handler3 _seh_longjmp_unwind@4 _FF_MSGBANNER _NMSG_WRITE _malloc nh_malloc _heap_alloc isatty _fflush flush</pre>

0001:00001825	flushall	00402825
0001:000018a0	_strlen	004028a0
0001:0000191b	_wctomb	0040291t
0001:00001990	aulldiv	00402990
0001:00001a00	aullrem	00402a00
0001:00001a75	flsbuf	00402a75
0001:00001b8a	_calloc	00402b8a
0001:00001c07	fcloseall	00402c07
0001:00001c5f	_free	00402c5f
0001:00001c90	_strcpy	00402c90
0001:00001ca0	_strcat	00402ca0
0001:00001d80	setmbcp	00402d80
0001:00002144	initmbctable	00403144
0001:00002160	_memcpy	00403160
0001:00002495	sbh_heap_init	00403495
0001:000024d3	sbh_find_block	004034d3
0001:000024fe	sbh_free_block	004034fe
0001:00002829	sbh_alloc_block	00403829
0001:00002b32	sbh_alloc_new_region	00403b32
0001:00002be3	sbh_alloc_new_group	00403be3
0001:00002cde	crtMessageBoxA	00403cde
0001:00002d70	_strncpy	00403d70
0001:00002e6e	callnewh	00403e6e
0001:00002e89	commit	00403e89
0001:00002ee0	write	00403ee0
0001:0000308d	fptrap	0040408c
0001:00003096	lseek	00404096
0001:00003130	getbuf	00404130
0001:00003180	_memset	00404180
0001:000031d8	_fclose	004041d8
0001:0000322e	crtLCMapStringA	0040422e
0001:0000347d	crtGetStringTypeA	0040447c
0001:000035d0	_memmove	004045dC
0001:00003905	free_osfhnd	00404905
0001:0000397f	get_osfhandle	0040497f
0001:000039bc	dosmaperr	004049bc
0001:00003a23	close	00404a23
0001:00003ad6	freebuf	00404ad6
0001:00003b10	alloca_probe	00404b10

```
// LIBCTINY - Matt Pietrek 2001
// MSDN Magazine, January 2001
#include <windows.h>
#include <stdio.h>
#include <stdarg.h>
// Force the linker to include USER32.LIB
#pragma comment(linker, "/defaultlib:user32.lib")
extern "C" int __cdecl printf(const char * format, ..
{
 char szBuff[1024];
 int retValue;
 DWORD cbWritten;
 va_list argptr;
 va_start( argptr, format );
 retValue = wvsprintf( szBuff, format, argptr );
 va_end( argptr );
            GetStdHandle(STD_OUTPUT_HANDLE), szBuff
 WriteFile(
            &cbWritten, 0 );
 return retValue;
}
```

Figure 3 INITTERM

```
#include <malloc.h>
#include "initterm.h"
#pragma data_seg(".CRT$XCA")
PVFV \_xc_a[] = \{ NULL \};
#pragma data_seg(".CRT$XCZ")
_PVFV ___xc_z[] = { NULL };
#pragma data_seg() /* reset */
#pragma comment(linker, "/merge:.CRT=.data")
typedef void (__cdecl *_PVFV)(void);
void __cdecl _initterm (
        _PVFV * pfbegin,
        _PVFV * pfend
        )
{
  /*
   * walk the table of function pointers from the bot
   * the end is encountered. Do not skip the first \epsilon
   * value of pfbegin points to the first valid entry
   * execute what pfend points to. Only entries befc
   * valid.
   */
  while ( pfbegin < pfend )</pre>
  {
    // if current table entry is non-NULL, call thru
    if ( *pfbegin != NULL )
      (**pfbegin)();
    ++pfbegin;
 }
}
static _PVFV * pf_atexitlist = 0;
static unsigned max_atexitlist_entries = 0;
```

```
static unsigned cur_atexitlist_entries = 0;
void __cdecl _atexit_init(void)
ł
  max_atexitlist_entries = 32;
  pf_atexitlist = (_PVFV *)calloc( max_atexitlist_ent
                                     sizeof(_PVFV*) );
}
int __cdecl atexit (_PVFV func )
ł
  if ( cur_atexitlist_entries < max_atexitlist_entrie</pre>
  {
    pf_atexitlist[cur_atexitlist_entries++] = func;
    return 0;
  }
  return -1;
}
void __cdecl _DoExit( void )
{
  if ( cur_atexitlist_entries )
  {
    _initterm( pf_atexitlist,
               // Use ptr math to find the end of the
               pf_atexitlist + cur_atexitlist_entries
 }
}
```

Figure 4 DLLCRTO.CPP

// Force the linker to include KERNEL32.LIB #pragma comment(linker, "/defaultlib:kernel32.lib") // Force 512 byte section alignment in the PE file #pragma comment(linker, "/OPT:NOWIN98") // #pragma comment(linker, "/nodefaultlib:libc.lib") // #pragma comment(linker, "/nodefaultlib:libcmt.lib" // User routine DllMain is called on all notificatior extern BOOL WINAPI DllMain(HANDLE hDllHandle, DWORD dwReason, LPVOID lpreserved); 11 // Modified version of the Visual C++ startup code. // make it easier to read. Only supports ANSI progra 11 extern "C" BOOL WINAPI DllMainCRTStartup(HANDLE hDllHandle, DWORD dwReason, LPVOID lpreserved) { if (dwReason == DLL_PROCESS_ATTACH) { // set up our minimal cheezy atexit table atexit init(); // Call C++ constructors _initterm(__xc_a, __xc_z); } BOOL retcode = DllMain(hDllHandle, dwReason, lprese

```
if ( dwReason == DLL_PROCESS_DETACH )
{
    __DoExit();
}
return retcode ;
}
```

Figure 5 TEST.CPP

```
// Small test program to exercise TINYCRT. Does noth
//
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main( int argc, char *argv[] )
{
  int i;
  for (i = 0; i < argc; i++)
  {
    printf( "argc: %u \'%s\'\n", i, argv[i] );
  }
  char * p = new char[10];
  lstrcpy( p, "Hello" );
  delete p;
  printf( "%s\n", strlwr( "MyLowerCaseString" ) );
  printf ( "strcmpi: %u\n", strcmpi( "Abc", "abc" ) )
  strrchr( "foo", 'o' );
  return 0;
}
// Declare a simple C++ class with a constructor
```

```
class TestClass
{
public:
    TestClass(void)
    {
        printf( "In TestClass constructor\n" );
    }
    ~TestClass(void)
    {
        printf( "In TestClass destructor\n" );
    }
};
// Create a global instance of the class
TestClass g_TestClassInstance;
```