File Commander User's Guide

Wilson WindowWare, Inc.1

2701 California Ave SW   ste 212

Seattle, WA 98116

Orders:	**(800) 762-8383**
Support:	**(206) 937-9335**
Fax:	**(206) 935-7129**

File Commander User's Guide

CONTENTS

INTRODUCTION

File Commander is an add-on for the Windows File Manager, which allows you to add your own custom menu items to the normal File Manager menu.

With almost two hundred functions and commands, File Commander can:

- Run Windows and DOS programs.
- Send keystrokes directly to applications.
- Rearrange, resize, hide, and close windows.
- Run programs either concurrently or sequentially.
- Display information to the user in various formats.
- Prompt the user for any needed input.
- Present scrollable file and directory lists.
- Copy, move, delete, and rename files.
- Read and write files directly.
- Copy text to and from the Clipboard.
- Perform string and arithmetic operations.
- Make branching decisions based upon numerous factors.

And much, much more.

System Requirements

File Commander requires an IBM PC or compatible with a minimum of 640K memory running Microsoft Windows version 3.1 or higher.

About This Manual

File Commander is an application which uses our Windows Interface Language (WIL). Please refer to the **WIL Reference Manual** for an introduction to WIL, as well as for complete documentation of the many functions available in WIL (and therefore, in File Commander).

This User's Guide includes only topics and functions which are exclusive to File Commander or which behave differently in File Commander, as well as additions and changes that have been made since the WIL Reference Manual went to press.

Note: File Commander is a **menu file** based implementation of WIL.

Notational Conventions

Throughout this manual, we use the following conventions to distinguish elements of text:

ALL-CAPS
Used for filenames.

Boldface

Used for important points, programs, function names, and parts of syntax that must appear as shown.

`system`

Used for items in menus and dialogs, as they appear to the user.

`Small fixed-width`

Used for WIL sample code.

Italics

Used for emphasis, and to liven up the documentation just a bit.

Acknowledgements

File Commander software developed by Morrie Wilson.

Documentation written by Richard Merit.

Help System Engineering by James Stiles, The Stiles Group, (206) 937-3141 CIS   73240,3131 Internet 73240.3131@compuserve.com

GETTING STARTED

File Commander is easy to install. You will find an installation diskette in your File Commander package. You will need to put it into a floppy drive. The File Commander installation program is itself a Windows application, so make sure Windows is running and the Windows File Manager is loaded.

Click on the directory window of the drive with the installation diskette in it. A directory tree will appear for the File Commander diskette. You should see a root directory icon. Double-click on this icon and a list of filenames will appear. Find the filename WSETUP.EXE and double-click on it. Follow the instructions WSETUP will display.

WSETUP will copy or create its files in a directory of your choice.

USING FILE COMMANDER

Installation

File Commander is not run as a normal executable program. Rather, as an extension to File Manager, it is loaded automatically when File Manager starts up. The WSETUP program, which is used to install File Commander, adds or modifies two sections to your WINFILE.INI file: [AddOns] and [FileCmdr]. The settings in these sections determine the start-up defaults for File Commander.

Add-On Limitations

File Commander allows you to attach from one to four top-level menu items to File Manager, each of which is considered to be a separate File Manager **add-on** (and each of which is represented by a combination of a DLL file and a menu file).

By default, WSETUP configures File Commander to add four add-ons. However, you may want to use a smaller number, since File Manager allows a maximum of only four or five add-ons, and you may have other add-ons which you want to use in addition to File Commander.

You can reduce the number of add-ons which File Commander adds on by editing the [AddOns] section of WINFILE.INI. The default installation adds four add-ons, as follows:

```
[AddOns]
WWWFC1=WWWFC1.DLL
WWWFC2=WWWFC2.DLL
WWWFC3=WWWFC3.DLL
WWWFC4=WWWFC4.DLL
```

(Your [AddOns] section will contain additional lines if you have add-ons from any other products already installed). To reduce the number of File Commander add-ons to three, simply delete the reference to add-on #4 (WWWFC4.DLL):

```
[AddOns]
WWWFC1=WWWFC1.DLL
WWWFC2=WWWFC2.DLL
WWWFC3=WWWFC3.DLL
```

You can likewise remove add-on #3 (WWWFC3.DLL) and #2 (WWWFC2.DLL).

Note: Early versions of Windows 3.1 only support four add-on DLL's. It is expected that future versions will support five add-on DLL's. Please refer to the File Commander README.TXT file for any updated information on this.

Menu Files

Each File Commander add-on DLL has an associated menu file, which defines the menu structure of that add-on. The names of these menu files are specified in the [FileCmdr] section of WINFILE.INI, as follows:

```
MenuFile1=WWWFC1.MNU
MenuFile2=WWWFC2.MNU
MenuFile3=WWWFC3.MNU
```

```
MenuFile4=WWWFC4.MNU
```

Each of these menu files is independent, and is edited individually (refer to the **WIL Reference Manual** for information on menu file structure). Each menu file can contain a maximum of 99 menu items.

Menu Titles

The titles of each of the top-level menu items are determined by the following settings in the [FileCmdr] section of WINFILE.INI:

```
MenuTitle1=
MenuTitle2=
MenuTitle3=
MenuTitle4=
```

"MenuTitle1" is the title of the menu defined by "MenuFile1", etc.

Restrictions

File Commander does not support the use of menu hotkeys, as discussed in the **WIL Reference Manual**.

FILE COMMANDER FUNCTIONS

Introduction

This section includes only those additional File Commander functions which do not appear in the **WIL Reference Manual.** The **WIL Reference Manual** is your primary reference to the functions available in File Commander.

Note: The functions listed under the **See Also** headings may be documented either in this User's Guide or in the **WIL Reference Manual**.

Function List

CurrentPath **( )**
Returns path of the selected filename.

DirExist **(**[d**:**]path**)**
Determines if a directory exists.

MsgTextGet **(**window-name**)**
Returns the contents of a Windows message box.

Refresh **(**request#**,** p1**,** p2**,** p3**,** p4**)**
Updates file window display.

CurrentPath

Returns path of the selected filename.

Syntax:

CurrentPath ( )

Parameters:

(none)

Returns:

(s) path of currently-selected file.

When a WIL menu shell displays the files in the current directory, one of them may be "selected." This function returns the drive and path of that file, if any.

This is different than a "highlighted" file. When a file is highlighted, it shows up in inverse video (usually white-on-black). To find the filenames that are highlighted, see **FileItemize**.

Example:

```
myfile = StrCat (CurrentPath(), CurrentFile())
DirChange ("c:\word")
Run("winword.exe", myfile)
```

See Also:

CurrentFile, DirGet, FilePath

DirExist

Determines if a directory exists.

Syntax:

DirExist ([d:]path)

Parameters:

(s) [d:]path	directory name, with optional drive.

Returns:

(i)	**@TRUE** if the directory exists;
@FALSE if it doesn't exist.

You can use this function to determine whether a specified drive is valid by checking for the existence of the root directory on that drive.

Examples:

```
wpdir = "c:\wp"
If DirExist(wpdir) == @FALSE Then DirMake(wpdir)
DirChange(wpdir)

:top
drive = AskLine("Run Excel", "Enter a drive letter", "")
Ifdrive == "" ThenExit
drive = StrSub(drive, 1, 1)
If DirExist("%drive%:\") == @FALSE Then Goto top
NetAddCon("\\userapps\excel", "", drive)
```

See Also:

DirChange, DirMake, DirRemove, DirRename, FileExist

MsgTextGet

Returns the contents of a Windows message box.

Syntax:

MsgTextGet (window-name)

Parameters:

(s) window-name	full title of the message box window.

Returns:

(s)	contents of the message box.

This function returns the text contents of a standard Windows message box. "Window-name" must be the full title of the message box window, and is case-sensitive.

Note: This function will not work with the types of message boxes created by most WIL functions, since they are not standard Windows message boxes.

Example:

```
msg = MsgTextGet("Microsoft Word")
If msg == "Search text not found" Then SendKey("~")
```

Refresh

Updates file window display.

Syntax:

Refresh (request#)

Parameters:

(i) request#	see below.

Returns:

(i)	always 1.

This function updates the files being displayed in the File Manager window, as well as the drive icons that are displayed. It is useful after running an application (such as a DOS program) which creates, deletes, or renames a file in the directory being viewed, or connects to (or disconnects from) a network server, and you wish the changes to be reflected in the File Manager window.

Specifying a request# of 0 causes only the active window to be updated, and specifying a request# of 1 causes all File Manager windows to be updated.

Note: This command does not take effect until the WIL program has completed, regardless of where the command may appear in the program.

Example:

```
Run("pkunzip.exe", CurrentFile())
Refresh(0)
```

UTILITIES

Dialog Editor

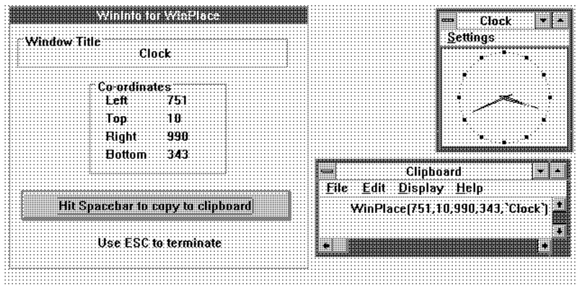
The WIL Dialog Editor (WWWDLGED.EXE) provides a convenient method of creating dialog box templates for use with the **Dialog** function.   It displays a graphical representation of a dialog box, and allows you to create, modify, and move individual controls which appear in the dialog box.   After you have defined your dialog box, the Dialog Editor will generate the appropriate WIL code, which you can save to a file or copy to the Clipboard for pasting into your WIL program.

The WIL Dialog Editor comes with an online help file (WWWDGEDT.HLP).   Simply select the **<u>H</u>elp** function in the Dialog Editor for detailed instructions on using the program.

If the file WWWDLGED.EXE is located in the FIL-CMDR.HLP directory , you can try it by clicking the button here.

WinInfo

The **WinInfo** utility (WININFO.EXE) lets you take an open window that is sized and positioned the way you like it, and automatically create the proper **WinPlace** statement for you.   It puts the text into the Clipboard, from which you can paste it into your WIL program:



Window Title: Clock

Co-ordinates

Left	751
Top	10
Right	990
Bottom	343

```
WinPlace(751,10,990,343,`Clock`)
```

You'll need a mouse to use **WinInfo**.   While **WinInfo** is the active window, place the mouse over the window you wish to create the **WinPlace** statement for, and press the spacebar.   The new statement will be placed into the Clipboard.   Then press the **Esc** key to close **WinInfo**.
If the file WININFO.EXE is located in the FIL-CMDR.HLP directory, you can try it by clicking the button here.