TRUEyesnono&About&PrintyesyesyesyesWinBatch Help FileWinBatchyes24/10/94

WinBatch
User's Guide

Table of Contents

INTRODUCTION

WinBatch brings the power of batch language programming to the Windows environment. Although WinBatch can do everything that the familiar DOS batch language can do, the capabilities of WinBatch begin where the DOS batch language leaves off.

With almost two hundred and fifty functions and commands, WinBatch 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.

Whether you are creating batch files for others, or looking for a way to automate your own work and eliminate the drudgery of repetitive tasks, you will find WinBatch to be a powerful, versatile, and easy-to-use tool.

About This Manual

WinBatch 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 WinBatch).

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

Note: WinBatch is a **batch file** based implementation of WIL.

System Requirements

WinBatch requires an IBM PC or compatible running Microsoft Windows version 3.1 or higher.

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.

Acknowledgments

WinBatch software developed by Morrie Wilson.

Documentation written by Richard Merit and Tina Browning.

GETTING STARTED

WinBatch is quite easy to install. You will find an appropriate diskette in your WinBatch package. Take the diskette and insert it into your floppy drive. The WinBatch installation program is itself a Windows application, so make sure Windows is running.

Insert your disk into your A: or B: disk drive. From the **File/Run** menu in **Program Manager** or your favorite shell, type A:\WSETUP or B:\WSETUP, depending on which floppy drive contains the WinBatch diskette. Follow whatever instructions WSETUP gives you. WSETUP will create the necessary files in a directory of your choice.

The first time you launch WinBatch you will be asked to enter your license number. The license numbers can be found in the back of your WinBatch User's guide.

USING WINBATCH

Creating WinBatch files

Command-Line Parameters

Passing Parameters

WinBatch Functions

Creating WinBatch files

WinBatch is a batch file interpreter. Therefore, before you can do anything useful with WinBatch, you must create at least one WinBatch file to run. WinBatch files are plain text files, which you can create with **Notepad** or a comparable text editor. Each WIL statement appears on a separate line, and can be a maximum of 255 characters long (refer to the **WIL Reference Manual** for information on the commands you can use in WinBatch). Indentation does not matter.

You should give each WinBatch file a name which has an extension of **WBT** (eg. TEST.WBT). We'll use the terms **WinBatch files** and **WBT files** interchangeably.

Command-Line Parameters

WinBatch is run with the following command line:

WINBATCH filename.wbt p1 p2 ... p9

"filename.wbt" is any valid WBT file, and is a required parameter.

"p1 p2 ... p9" are optional parameters (maximum of nine) to be passed to the WBT file on startup, delimited by spaces.

You can launch WinBatch using any method that is convenient for you. The WinBatch setup program associates WBT files with WINBATCH.EXE, so you can just double-click on a WBT file in **File Manager** or a similar program (however, see the Note below). Or, you can create a **Program Manager** icon which executes the appropriate command line (see your Windows manual for further details).

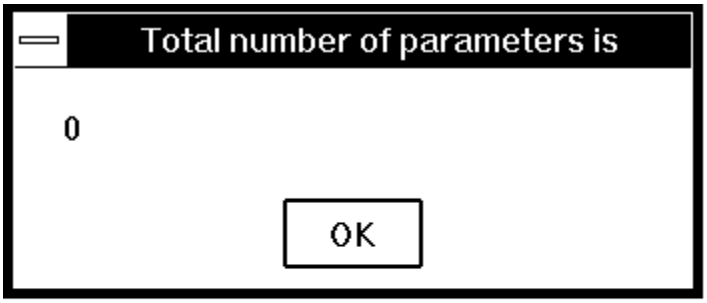
Parameters passed to a WBT file are automatically parsed into variables named **param1**, **param2**, etc. An additional variable, **param0**, is the total number of command-line parameters.

To display the total number of command line parameters, use **param0** as a variable in a message box.

Example:

```
Message("Total number of parameters is", "%param0%")
```

would produce:



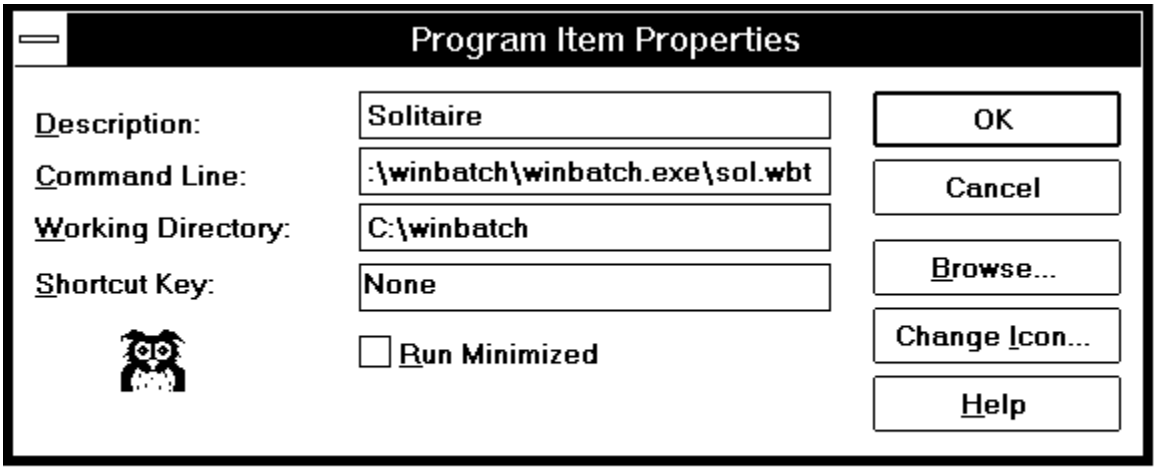
Passing Parameters

Associated Files

If you "run" a WBT file directly, using the file association capability of Windows, it will **not** receive any command line parameters. In order to pass parameters to a WinBatch file, you **must** run WINBATCH.EXE, followed by the name of the WBT file and any other desired parameters.

WBT files which are launched from the Program Manager as icons must have the complete path in the Properties dialog box in order for command line parameters to be received. The command line for "**SOL.WBT**" generally reads, "C:\WINBATCH\SOL.WBT". However, it is necessary to add WINBATCH.EXE to complete the path.

The program item Properties box should look like the following:



Between WBT files

To pass command line parameters from one WBT to a called WBT place % signs around the variables as in %variable%.

For Example:

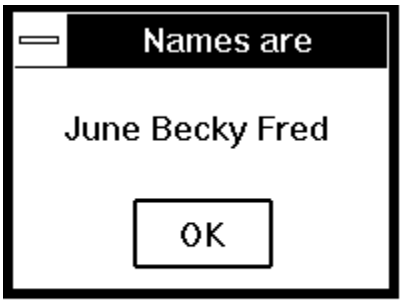
The first WBT calls a second WBT then passes three parameters.

```
Call("test.wbt", "Fred Becky June")
```

TEST.WBT contains the following line:

```
Message("Names are", "%param3% %param2% %param1%")
```

which produces:


Names are
June Becky Fred
OK

WINBATCH FUNCTIONS

Introduction

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

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

BoxOpen (title**,** text**)**
: Opens a WinBatch message box.

BoxShut ( )
: Closes the WinBatch message box.

BoxText (text**)**
: Changes the text in the WinBatch message box.

BoxTitle (title**)**
: Changes the title of the WinBatch message box.

CallExt (filename**,** parameters**)**
: Calls another WBT file as a separate subprogram.

BoxOpen

Opens a WinBatch message box.

Syntax:

BoxOpen (title, text)

Parameters:

(s) title	title of the message box.
(s) text	text to display in the message box.

Returns:

(i)	always 1.

This function opens a message box with the specified title and text. The message box stays in the foreground while the WIL program continues to process. The title of an existing message box can be changed with the **BoxTitle** function, and the text inside the box can be changed with the **BoxText** function. Use **BoxShut** to close the message box.

Note: In our shorthand method for indicating syntax the **(s)** in front of a parameter indicates that it is a string. An **(i)** indicates that it is an integer and a **(f)** indicates a floating point number parameter.

Example:

```
BoxOpen("Processing", "Be patient")
Delay(2)
BoxTitle("Still processing")
Delay(2)
BoxText("Almost done")
Delay(2)
BoxShut()
```

See Also:

BoxShut, BoxText, BoxTitle, Display, Message *(both found in main WIL documentation)*

BoxShut

Closes the WinBatch message box.

Syntax:

BoxShut ( )

Parameters:

(none)

Returns:

(i) always 1.

This function closes the message box that was opened with **BoxOpen**.

Example:

```
BoxOpen("Processing", "Be patient")
Delay(2)
BoxTitle("Still processing")
Delay(2)
BoxText("Almost done")
Delay(2)
BoxShut()
```

See Also:

BoxOpen, BoxText, BoxTitle

BoxText

Changes the text in the WinBatch message box.

Syntax:

BoxText (text)

Parameters:

(s) text	text to display in the message box.

Returns:

(i)	always 1.

This function changes the text in the message box opened with **BoxOpen**.

Example:

```
BoxOpen("Processing", "Be patient")
Delay(2)
BoxTitle("Still processing")
Delay(2)
BoxText("Almost done")
Delay(2)
BoxShut()
```

See Also:

BoxOpen, BoxShut, BoxTitle

BoxTitle

Changes the title of the WinBatch message box.

Syntax:

BoxTitle (title)

Parameters:

(s) title title of the message box.

Returns:

(i) always 1.

This function changes the title of the message box opened with **BoxOpen**.

Example:

```
BoxOpen("Processing", "Be patient")
Delay(2)
BoxTitle("Still processing")
Delay(2)
BoxText("Almost done")
Delay(2)
BoxShut()
```

See Also:

BoxOpen, BoxShut, BoxText, WinTitle *(found in main WIL documentation)*

CallExt

Calls another WBT file as a separate subprogram.

Syntax:

CallExt (filename.wbt, parameters)

Parameters:

(s) filename.wbt	the WBT file you are calling (the extension is required).
(s) parameters	the parameters to pass to the file, if any, in the form "p1 p2 p3 ... pn".

Returns:

(i)	always 0.

This function is used to pass control temporarily to a secondary WBT file. The main WBT file can optionally pass parameters to the secondary WBT file. All variables are exclusive (**local**) to their respective files, so that neither WBT file "knows about" variables being used by the other. The secondary WBT file should end with a **Return** statement, to pass control back to the main WBT file.

If a string of parameters is passed to the secondary WBT file, it will automatically be parsed into individual variables with the names **param1**, **param2**, etc. (maximum of nine parameters). The variable **param0** will be a count of the total number of parameters in the string.

Example:

MAIN.WBT

```
old = AskLine("RENAME", "File to rename", "")
If !FileExist(old) Then Exit
new = AskLine("RENAME", "New name for %old%", "")
If FileExist(new)
    Message("Rename aborted", "%new% already exists")
    Exit
Endif
CallExt("rename.wbt", "%old% %new%")
Exit
```

RENAME.WBT

```
old = param1
new = param2
FileRename(old, new)
Message("New Filename", new)
Return
```

See Also:

Call, ParseData, Return *(found in main WIL documentation)*

NETWORK and Other EXTENDERS

Network and Other extenders are documented fully in the on-line help files. For more extensive information look there, for a brief overview, continue.

Introduction

Novell 3.x Network Extender

Novell 4.x Network Extender

Basic Win3.1 Network Extender

Multinet, WinForWrkGrp, Win4 Network Extender

Introduction

WIL extender Dlls are special Dlls designed to extend the built-in function set of the WIL processor. These Dlls typically add functions not provided in the basic WIL set, such as network commands for particular networks (Novell, Windows for WorkGroups, LAN Manager and others), MAPI, TAPI, and other important Application Program Interface functions as may be defined by the various players in the computer industry from time to time. These Dlls may also include custom built function libraries either by the original authors, or by independent third party developers. (An Extender SDK is available). Custom extender Dlls may add nearly any sort of function to the WIL language, from the mundane network math or database extensions, to items that can control fancy peripherals, including laboratory or manufacturing equipment.

WIL extenders must be installed separately. Up to 10 extender Dlls may be added. The total number of added items may not exceed 100 functions and constants. The **AddExtender** function must be executed before attempting to use any functions in the extender library. The **AddExtender** function should be only executed once in each WIL script that requires it.

To use a WIL extender, at the top of each script in which you use network commands add the appropriate extender with the AddExtender command.

```
AddExtender(extender filename)
```

Remember you can add up to 10 extender Dlls or a combined total of 100 functions.

Novell 3.x Network Extender

This extender provides standard support for Novell 3.x networks. It may be used in addition with other extenders, such as the Windows for WorkGroups Multinet extender.

Note: If you want to use any of the following commands you need to add the following line to the top of your script.

```
AddExtender("wwn3x16i.dll")
```

Other required DLL's: NWCALLS.DLL

This particular Dll, wwn3x16i.dll, is for use on 16-bit versions of Windows on Intel 386, 486, and 586 type processors. Your system may require the use of a different Dll. See Filename Appendix B for more information on Dll filenames.

Function Reference

LastError ( )
Returns the most-recent error encountered during the current WIL program.

n3Attach(server-name, user-name, password)
Attaches to a network file server.

n3Detach(server-name)
Detaches from a network file server.

n3DrivePath(local-name)
Returns the network resource associated with the local-name.

n3DriveStatus(local-name)
Returns a status code number indicating the type of connection associated with a local-name.

n3GetUser(server-name)
Determines the currently logged on user name on the specified server.

n3Map(server-name, net-path, local-name)
Maps a drive to a resource on the specified server.

n3MapDelete(server-name, local-name)
Removes a drive mapping.

n3MemberDel(server-name, group-name ,user-name)
Deletes the specified user from the specified group on the specified server.

n3MemberGet(server-name, group-name ,user-name)
Determines if the specified user is a member of the specified group on the specified server.

n3MemberSet(server-name, group-name , user-name)
Sets the specified user as a member of the specified group on the specified server.

n3MsgSend(server-name, message , user-name)
Sends a message (max 56 characters) to the specified user.

n3MsgSendAll(server-name, message)
Sends a message (max 56 characters) to all logged on users.

Novell 4.x Network Extender

This extender provides standard support for Novell 4.x networks. It may be used in addition with other extenders, such as the Windows for WorkGroups Multinet extender, and the Novell 3.x extender.

Note: There are certain differences in using Novell 4 over Novell 3. In Novell 4, you must login to the network before attaching to a File Server.

Note: If you want to use any of the following commands you need to add the following line to the top of your script.

```
AddExtender("wwn4x16i.dll")
```

Other required DLL's: NWCALLS.DLL, NWNET.DLL, NWLOCALE.DLL

This particular Dll, wwn4x16i.dll, is for use on 16-bit versions of Windows on Intel 386, 486, and 586 type processors. Your system may require the use of a different Dll. See Filename Appendix B for more information on Dll filenames.

Function Reference

LastError ( )
Returns the most-recent error encountered during the current WIL program.

n4Attach(server-name)
Attaches to - opens a communication channel with - a network file server.

n4Detach(server-name)
Detaches from a network file server.

n4DrivePath(local-name)
Returns the network resource associated with the local-name.

n4DriveStatus(local-name)
Returns a status code number indicating the type of connection associated with a local-name.

n4GetUser(server-name)
Determines the currently logged on user name on the specified server.

n4Login(user-name, password)
Performs a login to the Novell network.

n4Logout()
Performs a network logout.

n4Map(server-name, net-path, local-name)
Maps a drive to a resource on the specified server.

n4MapDelete(server-name, local-name)
Removes a drive mapping.

n4MemberDel(server-name, group-name ,user-name)
Deletes the specified user from the specified group on the specified server.

n4MemberGet(server-name, group-name ,user-name)
Determines if the specified user is a member of the specified group on the specified server.

n4MemberSet(server-name, group-name , user-name)
Sets the specified user as a member of the specified group on the specified server.

n4MsgSend(server-name, message , user-name)
Sends a message (max 56 characters) to the specified user.

n4MsgSendAll(server-name, message)
Sends a message (max 56 characters) to all logged on users.

Basic Win3.1 Network Extender

This extender provides basic links to networks (like Novell) for the Windows 3.1 environment. It is not designed for Windows for WorkGroups, Chicago, or for other versions of Windows. There are alternate extenders available for those products. In addition, some networks, like Novell, have better, more fully featured extenders available.

Additional Dlls required: NONE

Note: If you want to use any of the following commands you need to add the following line to the top of your script.

```
AddExtender("www3a16i.dll")
```

This particular Dll, wwn3a16i.dll, is for use on 16-bit versions of Windows on Intel 386, 486, and 586 type processors. Your system may require the use of a different Dll. See Filename Appendix B for more information on Dll filenames.

Function Reference

LastError ( )
Returns the most-recent error encountered during the current WIL program.

w3AddCon(net-path, password, local-name)
Connects (maps) network resources to redirected local disk drives or printer ports.

w3CancelCon(local-name/net-path, forceflag)
Disconnects the specified local drive or network path.

w3DirBrowse( )
Displays a network dialog box allowing the user to select a network resource.

w3GetCaps(request code #)
Returns information on network capabilities.

w3GetCon(local-name/net-path)
Returns the name of a connected network resource.

w3NetDialog( )
Brings up a network dependent information box.

w3NetGetUser( )
Returns the name of the user currently logged into the network.

w3PrtBrowse( )
Displays a network printer browsing dialog box allowing the user to select a network printer.

Multinet, WinForWrkGrp, Win4 Network Extender

This extender is designed for versions of Windows containing the Microsoft MultiNet network driver support. This includes Windows for WorkGroups and newer versions of Windows. The commands in this package handle the Windows and Microsoft networks. It is designed to work in conjunction with other extenders for other networks, such as extenders for Novell networks.

Additional Dlls required: NONE

Note: If you want to use any of the following commands you need to add the following line to the top of your script.

```
AddExtender("wwwn16i.dll")
```

This particular Dll, wwwn16i.dll, is for use on 16-bit versions of Windows on Intel 386, 486, and 586 type processors. Your system may require the use of a different Dll. See Filename Appendix B for more information on Dll filenames.

Function Reference

LastError ( )
Returns the most-recent error encountered during the current WIL program.

wnAddCon(net-path, password, local-name)
Connects (maps) network resources to redirected local disk drives or printer ports.

wnCancelCon(local-name/net-path, forceflag)
Disconnects the specified local drive or network path.

wnCmptrInfo(network name, request code #)
Returns information about the local computer.

wnDialog(network name)
Brings up a network dependent dialog box.

wnDlgBrowse(network name, select mode)
Displays a drive, printer or server selection dialog box.

wnDlgCon(network name, select mode)
Displays a type 1(default) connection dialog box.

wnDlgCon2(network name, select mode)
Displays a type 2(alternate) connection dialog box.

wnDlgCon3(network name, select mode)
Displays a type 3(network dependent) connection dialog box.

wnDlgCon4(network name, select mode)
Displays a type 4(network dependent alternate) connection dialog box.

wnDlgDiscon(network name, select mode, drive letter)
Brings up a disconnection dialog box.

wnDlgNoShare(network name, mode, directory share)
Brings up dialog box that allows termination of a directory share.

wnDlgShare(network name, mode, directory share)
Brings up a dialog box to allow sharing of specified directory.

wnGetCaps(network name, request code #)
Returns information on network capabilities.

wnGetCon(local-name/net-path)
Returns the name of a connected network resource.

wnGetUser(network name)
Returns the name of the user currently logged into the specified network.

wnNetNames( )
Returns a space delimited list of available networks.

wnRestore(local-name/net-path)

Restores connections on permanently attached devices or directories.

wnServers(workgroup name,requestcode#,commentcode#)

Generates a TAB delimited list of the network servers/workstations.

wnShareCnt(network name, select mode)

Returns number of shared printers or drives for the specified network on the local machine.

wnShareName(network name, path-name)

Returns the share name of a given path on a local computer.

wnSharePath(network name, share name)

Returns the path name of a given share name on a local computer.

wnShares(computer name, request code#, comment code#)

Generates a TAB delimited list shared network resources.

wnWrkGroups( )

Returns a list of workgroups on the windows network.

Dialog Editor

The WIL Dialog Editor (see Filename Appendix B for filename) 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.

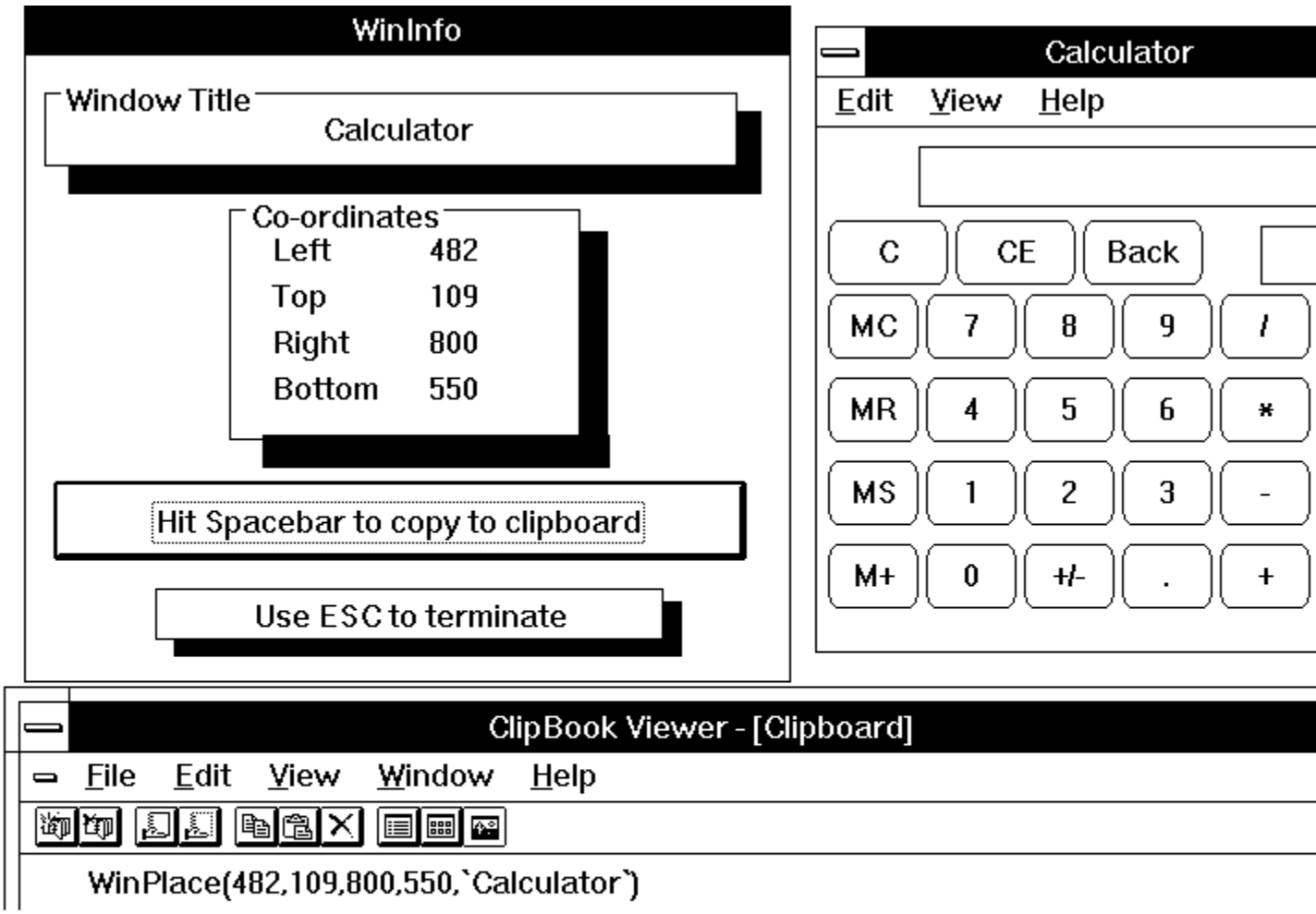
The WIL Dialog Editor comes with an on-line help file (see Filename Appendix B for filename), as well as detailed instructions in the next section. Simply select the **Help** function in the Dialog Editor for detailed instructions on using the program.

See DIALOG EDITOR help section for more information.

WinInfo

The **WinInfo** utility (see Filename Appendix B for filename) 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:

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**.



DIALOG EDITOR

Menu Commands

There are three standard menus in this program; FILE, EDIT, and HELP.

File
Edit
Help

User Interface

Control Attribute Specifics

Some of the Controls require extra knowledge or special handling.

COMPILER APPENDIX A

NOTE: This section is applicable only if you own the WinBatch Compiler. The Compiler is a separate product and is not included in the purchase of WinBatch, the single-user version. If you would like additional information on the Compiler and its capabilities, please call Customer Service.

The purpose of the WinBatch Compiler is transform a WinBatch WBT file into any one of the following:

A small EXE file.
A standalone EXE file.
An encoded WBT file.
An encrypted WBT file.

No royalties of any kind are required for distribution of any file created by this compiler.

COMPILER INSTALLATION

You will find an appropriate diskette in your WinBatch Compiler package. Take the diskette and insert it into your floppy drive. The installation program is itself a Windows application, so make sure Windows is running.

Insert your disk into your A: or B: disk drive. From the **File/Run** menu in **Program Manager** or your favorite shell, type A:\WSETUP or B:\WSETUP, depending on which floppy drive contains the Ccmpiler diskette. Follow whatever instructions WSETUP gives you. WSETUP will create the necessary files in a directory of your choice.

The first time you launch the Compiler you will be asked to enter your license number. The license numbers can be found in the back of your WinBatch User's guide.

COMPILER USAGE

The compiler may be run in either interactive or batch mode. In interactive mode, the user is prompted to provide all necessary information via a popup dialog box. In batch mode, all required parameters are supplied via command line arguments.

Before you can do anything useful with the Compiler, you must use the batch file interpreter to create a WinBatch file. Each WinBatch file should have a file extension of .WBT, .WBM, or .WIL.

Running the Compiler in Interactive Mode

Running the Compiler in Interactive Mode

Start the compiler by double-clicking the compiler icon or the Compiler.EXE file name. (or by choosing the appropriate item in any menu system you may be using).

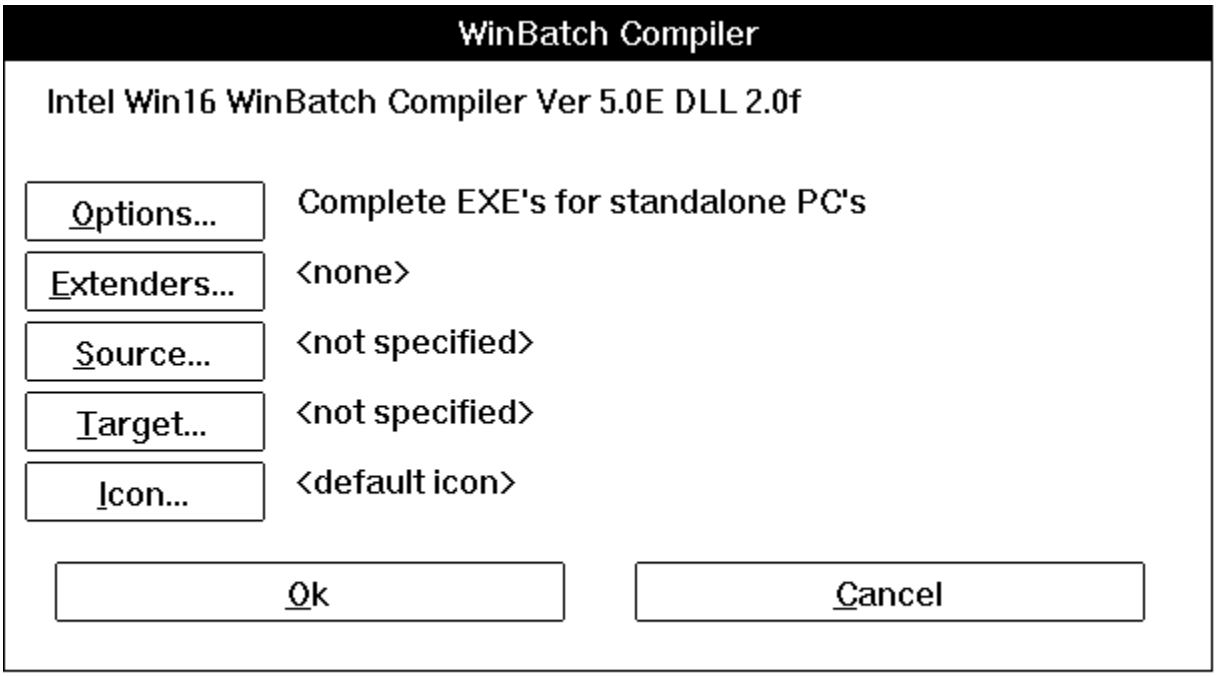
A dialog box will be displayed asking for input. Select the type of compile desired (large EXE, small EXE, encoded or encrypted), choose the source .WBT file, and supply an output file name. If you wish, choose an icon along with any necessary extenders. Press the OK button.

The compiler will process for 5 to 10 seconds, and then report that the file has been compiled. The compiler does not perform error checking. It is assumed the WBT file has been properly debugged with the standard WinBatch product prior to the compile step.

INTERACTIVE MODE

When you launch the Compiler EXE, a dialog box similar to the following will be displayed:

Options
Extenders
Source
Target
Icon



OPTIONS

The OPTIONS button allows you to select which type of executable file you would like to create from your WBT file.

Large EXE for Standalone PC's
Small EXE for Networked PC's
Encode for Call's from EXE files
Encrypted with Password

Large EXE for Standalone PC's

This option creates an EXE designed for Standalone PC's and does not require any extra Dlls. When a Standalone EXE is launched on a PC, the necessary Dlls are written into the executables directory. Subsequent EXE files installed on this same machine can be compiled under the Small EXE option.

If Network commands have been used, you will need to compile the Network Extender Dlls into the EXE. This is explained more specifically in the section, EXTENDERS.

Note: The executables directory. cannot be READ-ONLY.

Small EXE for Networked PC's

This option is suitable for network file server installation, or for distribution with a separate Dll file. Outside Dlls are required in order for it to run.

When a Small EXE is run, it will look in the Windows directory and the directories in the environment PATH variable for the Dlls. Place the WinBatch Dlls, and network extender Dlls on the path or search drive. If you launch this EXE on a PC in which a Standalone EXE has been run previously, it will access the same Dlls the Standalone installed.

Encode for Call's from EXE files

This option creates an encoded WBT file. The standard WinBatch product or a compiled EXE file is needed to access and run the encoded file. Encoded WBT files provide the following:

Source code is protected from unauthorized or accidental modification.
Encoded WBT files may be CALL'ed from compiled files.

If your code has a Call to another WBT file, the called WBT must be compiled with this option. Otherwise, when you run your EXE, you will get an "Encrypted/Encoded Verification Failed" Error.

Note: When you compile your file, your Target filename will have a .WBC extension. It is necessary to have a different filename from the original filename. You cannot compile a file to its own name without corrupting the file. To protect the innocent, the default Target extension is .WBC. After compiling, go into your EXE and change the Call statement to reflect the new filename .WBC. Recompile the EXE.

Encrypted with Password

This option encrypts a WBT file and uses a default Target extension of .WBE. The standard WinBatch product is needed to access the encrypted file. During the compilation a password is provided to the compiler. The same password must be supplied when the WBT file is run. The purpose of an encrypted WBT file is to prevent unauthorized personnel from executing it.

EXTENDERS

The EXTENDERS button displays a list of extenders which can be chosen and compiled into a Standalone EXE option. More than one extender may be chosen. If any of the Network extender functions are used, the corresponding extender must be compiled into the Standalone, or placed in the Windows directory or on the network path for a Small EXE to access. The selected extenders will be displayed in the WinBatch Compiler Dialog box next to the EXTENDERS button.

SOURCE

The SOURCE button displays a File Selection Box. Select your file or type the filename and path into the File Name box and press OK. The path and filename will be displayed in the WinBatch Compiler dialog box next to the SOURCE button.

Note: After you select a SOURCE file, a default TARGET name will be generated and displayed next to the TARGET button. To change the default name, click on the TARGET button.

TARGET

The TARGET button displays a File Selection Box. Select your file or type the filename and path into the File Name box and press OK. The path and filename will be displayed in the WinBatch Compiler dialog box next to the TARGET button.

Note: A default filename and path will generally be generated from the SOURCE filename and path.

ICON

The ICON button displays a File Selection Box which allows you to choose an icon. Select your .ICO file and press OK. The path and icon filename will be displayed in the WinBatch Compiler dialog box next to the ICON button.

BATCH MODE

In batch mode all the information required to compile the program is passed to the compiler when it is initiated. The WinBatch WBT files and various other menu systems, including the program manager, can be configured to pass all required information in one swell foop.

Command Lines

Below are several examples of the basic command line used to run in Batch Mode. These are the command lines which would be specified in the Program Manager command line, or in the File.Run dialog box.

For standalone (large) EXE compiles

Five parameters are required.

```
WBC-xxx.exe 1 Source.wbt Target.exe  none     none
                                   or      or
                                Icon.ico  Ext.dll
                                          or
                              Ext.dll,Ext1.dll,Ext2.dll
```

About the Extenders: If you need to specify more than one Ext.dll, the string is delimited by a comma without spaces. Some extenders require more than one Dll. Interactive Mode will worry about this for you and include any extra Dlls. Look in the corresponding .DAT file for a list of extra Dlls the extender uses.

For network/noDLL (small) EXE compiles

Four parameters are required.

```
WBC-xxx.exe 2 Source.wbt Target.exe Icon.ico
                                     or
                                    NONE
```

For encoded compiles

Three parameters are required.

```
WBC-xxx.exe 3 Source.wbt Target.wbc
```

For encrypted compiles

Four parameters are required.

```
WBC-xxx.exe 4 Source.wbt Target.wbe Password
```

Sample WinBatch code for an EXE compile

For a Standalone EXE compile without a default icon or network extenders.

```
Run("WBC-16i.exe", "1 Source.wbt Target.exe none none")
```

This particular compiler exe, WBC-16i.exe, is for use on 16-bit versions of Windows on Intel 386, 486, and 586 type processors. Your system may require the use of a different EXE. See Filename Appendix B for more information on filenames.

NETWORK CONSIDERATIONS

If you plan to put the compiled files on a network, the following information will be helpful:

1) Set the compiled EXE files to read-only so that multiple users may access the same file.

2) Copy the Dll's from the compiler directory in File Manager to a file server directory in the search path and set the Dll's as read-only. (see Filename Appendix B)

3) Whenever the compiler, or any compiled WBTs with the Standalone option selected, are run, they will search the entire PATH for the required Dlls (see Filename Appendix B). If the Dlls are not found, they will be created in the executables directory. If you skipped item 2 immediately above, you will want to hunt these files down and remove them when you get around to actually doing item 2.

RESTRICTIONS

The CallExt function is not supported in compiled EXE's.

The compiler itself is licensed for a single user. A special license is required to operate the compiler on a network drive. If you need this capability, please call Customer Service.

FILENAME APPENDIX B

There are several different platforms which WinBatch and its utilities may be run on. When a file name is generated, it is made up of four or five characters which specify WHAT the file is, three characters which specify which platform the PC is running under and an .EXE or .DLL file extension.

The following tables show how the filename, minus the extension, is broken down and defined.

The first 4- 5 digits

Filename	Program/Utility
WBAT	WinBatch
WINFO	WinInfo
WWDLG	Dialog Editor
WBC-	WinBatch Compiler

Filename	Network Extender
WWN3X	Novell 3.x extender
WWN4X	Novell 4.x extender
WWW3A	Basic Win 3.1 extender
WWWN	Multinet, WinForWrkGrp, Win4 extender

The second 3 digits-

Filename	Platform
16I	Intel 16-bit version (Windows 3.1)
32I	Intel 32-bit version
32M	Mips 32-bit version
32D	Dec Alpha 32-bit version

If you have Windows 3.1 and ordered the single-user version of WinBatch, the executable files you received are WBAT16I.EXE, WWDLG16I.EXE, and WINFO16I.EXE. You will only have files which are suitable to your platform needs.

Note: Not all of the possible combos above will exist.

WinBatch Dlls

WinBatch has two Dlls necessary to its processing, a WBO Dll and a WBD Dll. If you compile your script under the Standalone option, these Dlls will be added into executable and will be placed in the Windows directory when the EXE is launched. If you compile under the Small option, place these Dlls on the network path, or in the network Windows directory.

The WinBatch Dll names are made up of 3 parts.

WBOxxyyy.dll
WBDxxyyy.dll

The first three characters will tell which Dll it is. This will either be, WBO or WBD. The second part is two characters chosen at random. These will match for both the WBO and the WBD Dll. The last three characters will

specify which platform it is running under.

The first 3 digits

Dllname
WBO
WBD

The second 2 digits-

Dllname
AA
AD
AF
AK

The last 3 digits-

Filename	**Platform**
16I	Intel 16-bit version (Windows 3.1)
32I	Intel 32-bit version
32M	Mips 32-bit version
32D	Dec Alpha 32-bit version

Here is an example of a pair of Dlls for use on 16-bit versions of Windows on Intel 386, 486, and 586 type processors.

WBDAK16I.Dll
WBOAK16I.Dll

ORDERING INFORMATION

Licensing our products brings you wonderful benefits. Some of these are:

- **Gets rid of that pesky reminder window that comes up when you start up the software.**
- **Entitles you to one hour free phone support for 90 days (Your dime).**
- **Insures that you have the latest version of the product.**
- **Encourages the authors of these programs to continue bringing you updated/better versions and new products.**
- **Gets you on our mailing list so you are occassionally notified of spectacular updates and our other Windows products.**
- **And, of course, our 90-day money back guarantee.**

International customers.

Although we do prefer payment by Credit Card we can accept non-US-bank checks under certain conditions. The check MUST be in your currency -- NOT IN US$ -- Just look in your newspaper for the current exchange rates, make out your check and send mail it to us. We will take care of the rest. No Eurocheques please.

Send to: Wilson WindowWare, Inc.
2701 California Ave SW #212
Seattle, WA 98116
USA

or call: (800) 762-8383 (USA orders only)
(206) 938-1740 (customer service)
(206) 937-9335 (tech support)
(206) 935-7129 (fax)

(Please allow 2 to 3 weeks for delivery)

Order Form

Click below.
WILSON WINDOWWARE ORDER FORM

yesTRUEBasic Windows 3.1 Network Extender Help Filenono&About&Printyesyesyesyesbasic 31yes14/09/94

Basic Windows 3.1 Network Extender

Basic Windows 3.1 Network Extender

For 16-bit Intel processors Extender DLL Name

AddExtender("www3a16i.dll")
Additional Dlls required: NONE

This extender provides basic links to networks (Like Novell) for the Windows 3.1 environment. It is not designed for Windows for Workgroups, Chicago, or or other versions of Windows. There are alternate extenders available for those products. In addition, some networks, like Novell, have better, more fully featured extenders available.

Table of Contents

Help file produced by **HELLLP!** v2.3a , a product of Guy Software, on 09/14/1994 for WILSON WINDOWWARE, INC..

The above table of contents will be automatically completed and will also provide an excellent cross-reference for context strings and topic titles. You may leave it as your main table of contents for your help file, or you may create your own and cause it to be displayed instead by using the I button on the toolbar. This page will not be displayed as a topic. It is given a context string of _._ and a HelpContextID property of 32517, but these are not presented for jump selection.

HINT: If you do not wish some of your topics to appear in the table of contents as displayed to your users (you may want them ONLY as PopUps), move the lines with their titles and contexts to below this point. If you do this remember to move the whole line, not part. As an alternative, you may wish to set up your own table of contents, see Help under The Structure of a Help File.

Do not delete any codes in the area above the Table of Contents title, they are used internally by HELLLP!

Introduction

WIL extender Dlls are special Dlls designed to extend the built-in function set of the WIL processor. These Dlls typically add functions not provided in the basic WIL set, such as network commands for particular networks (Novell, Windows for WorkGroups, LAN Manager and others), MAPI, TAPI, and other important Application Program Interface functions as may be defined by the various players in the computer industry from time to time. These Dlls may also include custom built function libraries either by the original authors, or by independent third party developers. (An Extender SDK is available). Custom extender Dlls may add nearly any sort of function to the WIL language, from the mundane network math or database extensions, to items that can control fancy peripherals, including laboratory or manufacturing equipment.

WIL extenders must be installed separately. Up to 10 extender Dlls may be added. The total number of added items may not exceed 100 functions and constants. The **AddExtender** function must be executed before attempting to use any functions in the extender library. The **AddExtender** function should be only executed once in each WIL script that requires it.

INSTALLATION - Using a Dll.

To use a WIL extender, at the top of each script in which you use network commands add the appropriate extender with the AddExtender command.

```
AddExtender(extender filename)
```

Remember you can add up to 10 extender Dlls or a combined total of 100 functions.

About this Help File

This extender adds certain network capability to the Windows Interface Language (WIL) processing engine. 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 the programs that use it. This help file includes only topics and functions which are exclusive to this particular WIL Network Extender.

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.

Acknowledgments

WinBatch software developed by Morrie Wilson.

Documentation written by Tina Browning.

Contact Information

Wilson WindowWare, Inc.
2701 California Ave SW ste 212
Seattle, WA 98116

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

Installation - Using a Dll

To use a WIL extender, at the top of each script in which you use network commands add the appropriate extender with the AddExtender command.

```
AddExtender(extender filename)
```

Remember you can add up to 10 extender Dlls or a combined total of 100 functions.

AddExtender

Installs a WIL extender Dll.

Syntax:

AddExtender(filename)

Parameters:

(s) filename WIL extender Dll filename

Returns:

(i) **@TRUE** if function succeeded
@FALSE if function failed.

WIL extender Dlls are special Dlls designed to extend the built-in function set of the WIL processor. These Dlls typically add functions not provided in the basic WIL set, such as network commands for particular networks (Novell, Windows for WorkGroups, LAN Manager and others), MAPI, TAPI, and other important Application Program Interface functions as may be defined by the various players in the computer industry from time to time. These Dlls may also include custom built function libraries either by the original authors, or by independent third party developers. (An Extender SDK is available). Custom extender Dlls may add nearly any sort of function to the WIL language, from the mundane network, math or database extensions, to items that can control fancy peripherals, including laboratory or manufacturing equipment.

Use this function to install extender Dlls as required. Up to 10 extender Dlls may be added. The total number of added items may not exceed 100 functions and constants. The **AddExtender** function must be executed before attempting to use any functions in the extender library. The **AddExtender** function should be only executed once in each WIL script that requires it.

The documentation for the functions added are supplied either in a separate manual or disk file that accompanies the extender Dll.

Example:

```
; Add vehicle radar processing dll controlling billboard visible to
; motorists, and link to enforcement computers.
; The WIL Extender SPEED.DLL adds functions to read a radar speed
; detector(GetRadarSpeed) , put a message on a billboard visible to
; the motorist (BillBoard), take a video of the vehicle (Camera), and
; send a message to alert enforcement personnel (Alert) that a
; motorist in  violation along with a picture id number to help
; identify the offending vehicle and the speed which it was going.
;
AddExtender("SPEED.DLL")
BillBoard("Drive Safely")
While @TRUE
   ; Wait for next vehicle
   while GetRadarSpeed()<5 ; if low, then just radar noise
      Yield        ; wait a bit, then look again
   endwhile
   speed=GetRadarSpeed()    ; Something is moving out there
   if speed < 58
      BillBoard("Drive Safely")  ; Not too fast.
   else
```

```
        if speed < 63
          BillBoard("Watch your Speed")   ; Hmmm a hot one
        else
          if speed < 66
            BillBoard("Slow Down")  ; Tooooo fast
          else
            BillBoard("Violation  Pull Over")
            pictnum = Camera() ; Take Video Snapshot
            Alert(pictnum, speed); Pull this one over
          endif
        endif
      endif
    endwhile
```

See Also:

DllCall *(found in main WIL documentation)*

LastError

Returns the most-recent error encountered during the current WIL program.

Syntax:

LastError ( )

Parameters:

(none)

Returns:

(i) most-recent WIL error code encountered.

In addition to the normal behavior of the LastError function documented in the WIL Reference Guide, if the most recent error occurred in a WIL Extender, then a number assigned by the Extender will be returned. The numbers are documented in the appendix of this Extender document.

It may be possible to obtain error numbers not documented. The "Notes" section of the WIL manual has been provided to allow you to keep records of undocumented error codes.

Example:

```
AddExtender("wwwn16i.dll")
ErrorMode(@OFF)
username=wnGetUser("BADNETNAME")
ErrorMode(@CANCEL)
err=LastError()
If err != 0
   Message("Error", "Function returned error code %err%")
endif
```

See Also:

Debug, ErrorMode *(both found in main WIL documentation)*

Net101

All network functionality for WIL is performed via "WIL Extenders", add-on Dlls for WIL, which contain Network commands for assorted networks.

NetInfo is the only WIL network function. It returns the types of the networks currently active on the local machine, and can be used to help determine which network extenders should be loaded in multi-network environments.

Documentation for the various network extenders are found either in a manual for a particular extender or in an associated disk file.

See Also:

NetInfo, AddExtender, DllCall (*found in main WIL documentation)*

NetInfo(requestcode)

Determines network(s) installed.

Syntax:

NetInfo(requestcode)

Parameters:

(i) requestcode 0 for primary network name
1 for secondary subnet list

Returns:

(s) Primary network name for request code 0, or
Secondary network list for request code 1.

Use this function to determine the network type(s) running on a workstation. When running in a mixed network environment, it may be important to be able to determine the types of networks running on a workstation so as to be able to load the appropriate network extender Dlls and issue the corresponding commands.

NetInfo(0) will return the name of the primary network, or will return "**MULTINET**" , which indicates the Windows multinet driver is active and the secondary subnet list should be queried. **NetInfo**(0) will return one of the following strings:

NetInfo(0) return values:

NONE	No network installed
MULTINET	Multinet driver installed, see subnet codes.
MSNET	Microsoft Network
LANMAN	LAN Manager
NETWARE	Novell NetWare
VINES	Banyan Vines
10NET	10 Net
LOCUS	Locus
SUNPCNFS	SUN PC NFS
LANSTEP	LAN Step
9TILES	9 Tiles
LANTASTIC	Lantastic
AS400	IBM AS/400
FTPNFS	FTP NFS
PATHWORK	DEC PathWorks
OTHER1	Other (code 1)
OTHER2	Other(code 2)
UNKNOWN	Other (unknown)

If **NetInfo**(0) returned "**MULTINET**" then **NetInfo**(1) will return one or more of the following in a space delimited list:

NetInfo(1) return values:

NONE	No networks active
MSNET	Microsoft Network
LANMAN	LAN Manager
WINNET	Windows Network (Windows for Workgroups, etc)
NETWARE	Novell Netware
VINES	Banyan Vines
OTHER2	Other (code 0x20)
OTHER4	Other (code 0x40)
OTHER8	Other (code 0x80)

Example:

```
a=NetInfo(0)
if a=="MULTINET"
   b=NetInfo(1)
   count=ItemCount(b," ")
   Message("Multinet supporting  %count% networks", b)
else
   Message("Installed Network", a)
endif
```

See Also:

Net101, AddExtender, DllCall *(found in main WIL documentation)*

w3AddCon

Connects (maps) network resources to redirected local disk drives or printer ports.

Syntax:

w3AddCon(net-path, password, local-name)

Parameters:

(s) net-path	net resource or string returned by w3DirBrowse.
(s) password	password required to access resource, or "".
(s) local-name	local drive name.

Returns:

(i)	@**TRUE** if successful; @**FALSE** if unsuccessful.

You can use **w3AddCon** to connect a local drive to a network directory, in which case "local-name" will be a drive name (eg, "Z:"). You can also connect a local printer port to a network print queue, in which case, "local name" wil be the name of the printer port. (eg "LPT1").

Use the **w3DirBrowse** function to obtain a value for "net-path".

If no password is required, use a null string ("") for the "password" parameter.

Example:

```
AddExtender("www3a16i.dll")
;Example of a simple hard-coded mapping
w3AddCon("\\SERVER\PUBLIC","","P:")
;;;
;Example of allowing the user to choose a netpath
; to be hard-codes to a drive
netpath=w3DirBrowse()
w3AddCon(netpath,"","Q:")
;;;
;A more complete example
availdrive = DiskScan(0)
drvlen = StrLen(availdrive)
If drvlen == 0 Then Goto nomore
availdrive = StrSub(availdrive, drvlen - 2, 2)
netpath = w3DirBrowse()
pswd = AskPassword("Enter password for", netpath)
w3AddCon(netpath, pswd, availdrive)
Exit
:nomore
Message("Connect Drive to Net", "No drives avail for assignment")
exit
;;;
;Example of adding a printer port
netprinter=w3PrtBrowse()
w3AddCon(netpath,"","LPT2")
```

See Also:

w3DirBrowse, w3CancelCon, w3GetCon

w3CancelCon

Disconnects the specified local drive or network path.

Syntax:

w3CancelCon(local-name, forceflag)

Parameters:

(s) local-name	local name.
(s) force flag	see below.

Returns:

(i)	**@TRUE** if successful; **@FALSE** if unsuccessful.

Disconnects the specified local drive. If the forceflag=@FALSE, this commands will be ignored if any files are open on the specified device. If forceflag=@TRUE, the connection will be terminated regardless of possible open files - unless the application opened the files with a parameter that overrides the forceflag=@TRUE request.

Example:

```
AddExtender("www3a16i.dll")
;Simple map delete of a drive
a=w3CancelCon("P:",0)
if a==0 then Message("Error","Map delete failed")

;Map delete of a UNC path
w3CancelCon("\\SERVER\PUBLIC",0)

;Map delete of a printer
w3CancelCon("LPT2",0)
```

See Also:

w3AddCon, w3GetCon

w3DirBrowse

Displays a network dialog box allowing the user to select a network resource.

Syntax:

w3DirBrowse( )

Parameters:

none

Returns:

(s) a string that can be used by **w3AddCon** to add a connection.

Brings up the network's directory browsing dialog box then returns the netpath selected by the user.

Example:

```
AddExtender("www3a16i.dll")
netpath = w3DirBrowse()
w3AddCon(netpath,AskPassword("Enter Password for %netpath%"),"Q:")
Message("Mapped to Q:",netpath)
```

See Also:

w3PrtBrowse, w3AddCon

w3GetCaps

Returns information on network capabilities.

Syntax:

w3GetCaps(request code #)

Parameters:

(s) request code # see below.

Returns:

(i) see below.

w3GetCaps returns 0 if no network is installed.

Req# **Return value**

1 Network driver specification number
This returns the major and minor version numbers of the network driver specification to which the driver conforms. The high and low bytes of the return value contain the major and minor version numbers, respectively. For Windows 3.1, it returns 0x030A.

2 Type of network installed:
The return value of this function is one of two complex bitmasks. In order to understand the return value, use the following procedure, and also refer to the example code for this function.

1) Obtain value for this request type. Goto Step 2

```
Caps2 = w3GetCaps(2)
```

2) If the return value (Caps2) is equal to or greater than 32768, proceed to Step 3

a) Separate top and bottom bytes of the number. The top byte is the network code and the bottom byte is the network subtype.

```
NetCode =   Caps2 & 65280     ; 65280 is hex 0xFF00
SubType =   Caps2 & 255       ; 255 is hex  0x00FF
```

b) Lookup network code in following table: Use Subtype for reference

NetCodes	NetWork
0	None
256	MSNet
512	LanMan
768	NetWare
1024	Vines
1280	10NET
1536	Locus
1792	Sun PC NFS
2048	LANstep
2304	9TILES
2560	LANtastic

3) If the return value (Caps2) is equal to or greater than 32768, then a Multinet network driver is installed. Multinet network driver support additional network drives on the system.

a) Obtain the bottom byte.

```
MultinetCode =    Caps2 & 255 ;255 is hex 0x00FF
```

b) The Multinet code is the sum of the codes for the installed networks. Use the following table to figure out the installed networks.

MultinetCode	**Installed Networks**
0	NONE
1	MSNet
2	LanMan
4	WinWorkgroups
8	NetWare
16	Other 1
32	Other 2
64	Other 3
128	Other 4

3 Network driver version number

4 Returns 1 if any network is installed

6 Bitmask indicating whether the network driver supports the following connect functions: It can be a combination of the following.

1	AddConnection
2	CancelConnection
4	GetConnections
8	AutoConnect
16	BrowseDialog
32	RestoreConnection

For example: 63 is the sum of all of the above, indicating that all are enabled.

7 Bitmask indicating whether the network driver supports the following print functions: It can be a combination of the following.

2	OpenJob
4	CloseJob
16	HoldJob
32	ReleaseJob
64	CancelJob
128	SetJobCopies
256	WatchQueue
512	UnwatchQueue
1024	LockQueueData
2048	UnlockQueueData
4096	ChangeMsg
8192	AbortJob
16384	NoArbitraryLock
32768	WriteJob

A list of supported functions is arrived at by subtracting the highest possible number from the sum repeatedly until zero is reached. If 16,246 is the returned sum, then the following is supported.

AbortJob (8192)
ChangeMsg (4096)
UnlockQueueData (2048)
LockQueueData (1024)
UnwatchQueue (512)

WatchQueue (256)
CancelJob (64)
ReleaseJob (32)
HoldJob (16)
CloseJob (4)
OpenJob (2)

This is the only possible combination of numbers which will make up the sum of 16,246.

8 Bitmask indicating which dialog functions are available. It can be a combination of the following values:

1	DeviceMode
2	BrowseDialog
4	ConnectDialog
8	DisconnectDialog
16	ViewQueueDialog
32	PropertyDialog
64	ConnectionDialog
128	PrinterConnectDialog
256	SharesDialog
512	ShareAsDialog

A list of supported functions is arrived at by subtracting the highest possible number from the sum repeatedly until zero is reached. If 751 is the returned sum, then the following is supported.

ShareAsDialog (512)
PrinterConnectDialog (128)
ConnectionDialog (64)
PropertyDialog (32)
DisconnectDialog (8)
ConnectDialog (4)
BrowseDialog (2)

This is the only possible combination of numbers which will make up the sum of 751.

9 Bitmask indicating which administrative functions are available. It can be a combination of the following values:

1	GetDirectoryType
2	DirectoryNotify
4	LongNames
8	SetDefaultDrive

A list of supported functions is arrived at by subtracting the highest possible number from the sum repeatedly until zero is reached. If 11 is the returned sum, then the following is supported:

SetDefaultDrive (8)
DirectoryNotify (2)
GetDirectoryType (1)

This is the only possible combination of numbers which will make up the sum of 11.

10 Bitmask indicating which error functions are available. It can be a combination of the following values:

1	GetError
2	GetErrorText

For example, if 3 is returned then both functions are supported.

Example:

```
AddExtender("www3a16i.dll")

Caps2=w3GetCaps(2)

if Caps2 >= 32768
   ; Multinet driver installed
   MultinetCode=Caps2 & 255
   NetNames=""
   if MultinetCode==0
           NetNames="None"
   else
      if (MultinetCode & 1  ) then NetNames=strcat(NetNames,@crlf,"MSNet")
      if (MultinetCode & 2  ) then NetNames=strcat(NetNames,@crlf,"LanMan")
      if (MultinetCode & 4  ) then NetNames=strcat(NetNames,@crlf,"WinWorkgroups")
      if (MultinetCode & 8  ) then NetNames=strcat(NetNames,@crlf,"NetWare")
      if (MultinetCode & 16 ) then NetNames=strcat(NetNames,@crlf,"Other 1")
      if (MultinetCode & 32 ) then NetNames=strcat(NetNames,@crlf,"Other 2")
      if (MultinetCode & 64 ) then NetNames=strcat(NetNames,@crlf,"Other 3")
      if (MultinetCode & 128) then NetNames=strcat(NetNames,@crlf,"Other 4")
   endif
  title= "Windows Multinet driver installed"
  Message(Title,strcat("Avail Networks are:",@crlf,NetNames))

else
   ; Single network driver installed
   NetCode = Caps2 & 65280
   SubType = Caps2 & 255
   NetName="Unknown"
   if NetCode==0      then NetName="None"
   if NetCode==256    then NetName="MSNet"
   if NetCode==512    then NetName="LanMan"
   if NetCode==768    then NetName="NetWare"
   if NetCode==1024   then NetName="Vines"
   if NetCode==1280   then NetName="10NET"
   if NetCode==1536   then NetName="Locus"
   if NetCode==1792   then NetName="Sun PC NFS'
   if NetCode==2048   then NetName="LANstep"
   if NetCode==2304   then NetName="9TILES"
   if NetCode==2560   then NetName="LANtastic"

   Title="Network codes are"
   Text="NetCode=%netcode% %@crlf%NetName=%NetName%%@crlf%SubType=%SubType%"
   Message(Title,Text)

endif
```

See Also:

w3NetGetUser, NetInfo

w3GetCon

Returns the name of a connected network resource.

Syntax:

w3GetCon(local-name/net-path)

Parameters:

(s) local-name/net-path	network resource name or local name.

Returns:

(s)	name of a network resource.

The **w3GetCon** function returns the name of the shared resource associated with the specified redirected local drive or device.

Example:

```
AddExtender("www3a16i.dll")
;Checking what net resource a local drive is connected to
netpath = w3GetCon("Q:")
Message("Local drive Q: is connected to",netpath)

;Checking to see which local drive a net resource is connected to
local = w3GetCon("\\SERVER\PUBLIC")
Message("\\SERVER\PUBLIC is mapped to", local)
```

See Also:

w3AddCon, w3CancelCon

w3NetDialog

Brings up a network dependent information box.

Syntax:

w3NetDialog( )

Parameters:

none

Returns:

(i) @**TRUE** if successful;
@**FALSE** if unsuccessful.

A network driver's dialog box displays copyright information, and may allow access to the network, depending on the particular network driver. The WIL program will wait until the network dialog terminates before continuing.

Example:

```
AddExtender("www3a16i.dll")
;Try it and see what it does on your network.  Oftentimes it
;does something quite useful
w3NetDialog()
```

See Also:

w3DirBrowse,

w3NetGetUser

Returns the name of the user currently logged into the network.

Syntax:

w3NetGetUser( )

Parameters:

none

Returns:

(s)	name of current user.

Returns username currently logged into network.

Note: This function will not work if Windows for Workgroups is installed. In which case, the Windows for Workgroups Multinet extender should be used instead of this one.

Example:

```
AddExtender("www3a16i.dll")
username=w3NetGetUser( )
Message("Logged on user is",unsername)
```

See Also:

w3GetCaps

w3PrtBrowse

Displays a network printer browsing dialog box allowing the user to select a network printer.

Syntax:

w3PrtBrowse( )

Parameters:

none

Returns:

(s) a string that can be used by w3AddCon to add a connection.

Brings up the network's printer browsing dialog box then returns the network printer path selected by the user.

Example:

```
AddExtender("www3a16i.dll")
;Example of mapping a printer port
netprinter=w3PrtBrowse()
w3AddCon(netpath,"","LPT2")
```

See Also:

w3DirBrowse, w3AddCon

w3Version( )

Returns the version of this Extender DLL.

Syntax:

w3Version( )

Parameters:

none

Returns:

(i) the version of number of this extender Dll.

This function is used to check the version number of this Dll in cases where older DLL's exist and alternate processing is desirable. Version numbers of newer versions will be larger than that of older versions.

Example:

```
AddExtender("www3a16i.dll")
a=w3Version()
Message("Dll Version",a)
```

Error Appendix

1000 "Win3.1 Basic Network Extender"
"173: No network found"
"174: Security violation"
"175: Function not supported"
"176: Out of memory"
"177: Network error"
"178: Windows function failed"
"179: Invalid type of request"
"180: Invalid pointer"
"181: Cancelled at users request"
"182: Unknown user / Not logged in"
"183: Buffer too small - Internal error"
"184: Invalid network name (Try uppercase?)"
"185: Invalid local name"
"186: Invalid password"
"187: Local device already connected"
"188: Not a valid local device or network name"
"189: Not a redirected local device or current net name"
"190: Files were open with FORCE=FALSE"
"191: Function busy"
"192: Unknown network error"

TRUEyesnono&About&PrintyesyesyesyesDialog Editor Help dialogyes29/08/94

DIALOG EDITOR

Table of Contents

Introduction

The WIL Dialog Editor allows you to create dialog box templates for WIL using the WDL format. The Dialog Editor will write the WIL script statements necessary to create and display the dialog. You can visually design your dialog box on the screen and then save the template either to a .WDL file or the Windows Clipboard.

You can include the dialog template code directly in your batch code, or you can use the batch language "Call" command to execute the dialog template. For example:

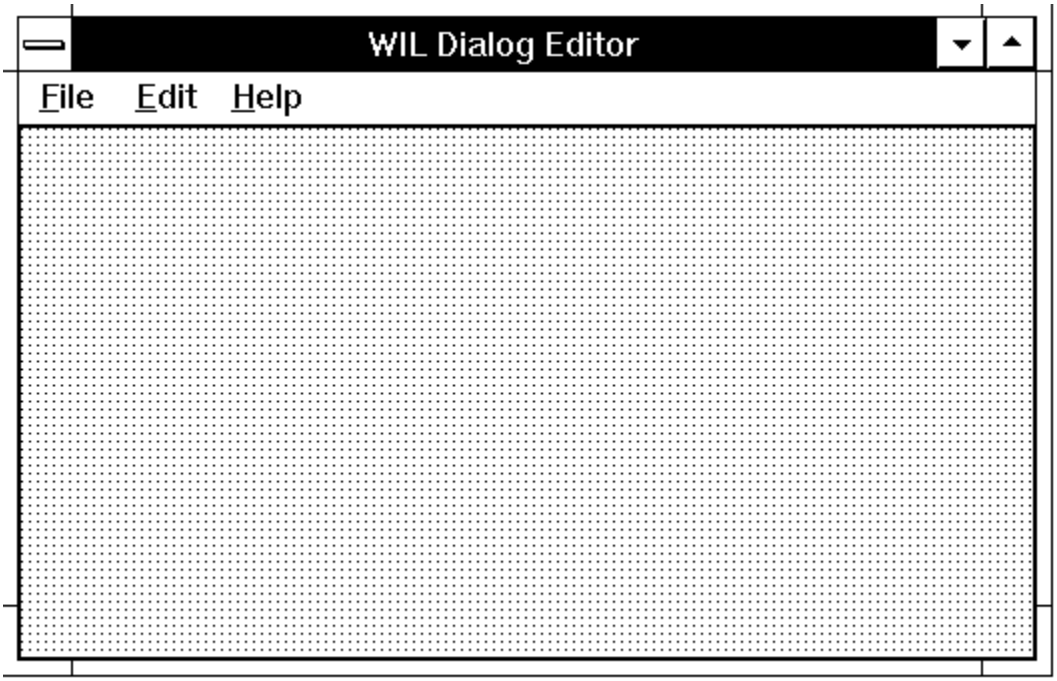
```
Call("Sample.WDL", "")
```

Getting Started

Launch the dialog editor executable, (see Filename Appendix B for filename).

To control the size of your dialog box, resize the WIL Dialog Editor. Your dialog will be the same size as this editor's window.

The editor will look like the following:



Menu Commands

There are three standard menus in this program; FILE, EDIT, and HELP.

File
Edit
Help

File

New
When you select New, any currently loaded template will be discarded and the slate will be clean for a new dialog. You will be prompted to enter the caption (title) for your dialog box, and a WIL variable name used to refer to the dialog box in the WIL scripts.

Load
Loads a dialog template from a file.

Save
Saves a dialog template to the current file.

Save As
Saves the dialog template to a file using a different filename.

Load from Clipboard
Loads a dialog template from the Windows Clipboard.

Save to Clipboard
Saves the dialog template to the Windows Clipboard.

Edit

Change Caption/Name

Allows you to change the Dialog caption (title) and/or the variable name used to refer to the dialog.

Note: Left Mouse double-clicking the dialog box background will also execute this menu item.

Add Control

Adds a new control to your dialog template.
Note: Right Mouse double-clicking has the same effect.

Delete Control

Surprisingly enough, Delete Control does not actually delete a control. It just reminds you how to do it. To delete a control, position the mouse cursor over the control and press the delete key.

Show Script

Displays the WIL script generated during the dialog edit session. Once you learn how the dialog scripts operate, viewing the script is a quick way to scan for errors. You will notice that some script lines cannot be viewed in their entirety, in which case simply double click it to view the entire line.

Help

Index
Displays the Index of the On-line help information.

Menu Commands
Displays information about the WIL Dialog Editor menu commands.

How to use Help
Activates the Microsoft Windows Index to Using Help.

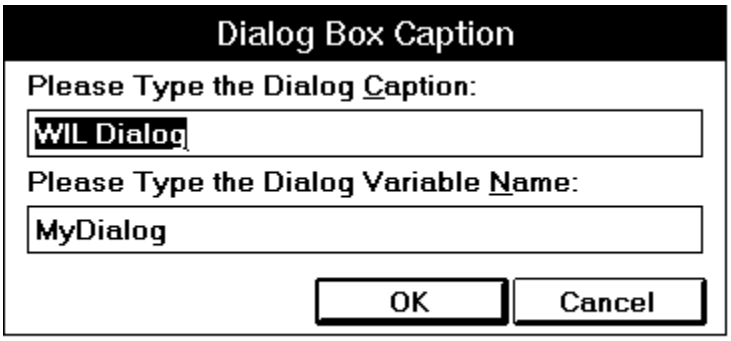
About
Displays the WIL Dialog Editor About dialog which includes the version number of the program.

Caption Box

Double click with the left mouse button on the workspace background to display the caption box.

The **Dialog Caption** is the title of the dialog box as it appears in the title bar. The **variable name** is the name of the dialog as seen in the script.

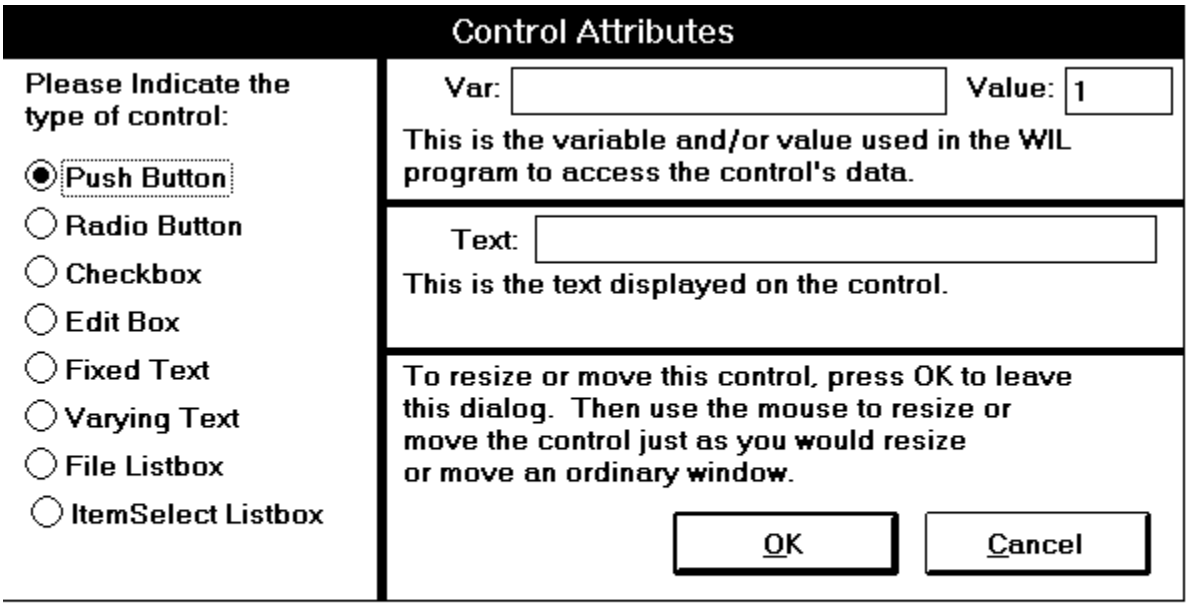
This information can be entered or changed at any time. However, we suggest filling it whenever you start a new dialog box. To change the caption double click on the workspace, (not on a control) with the left mouse button.



Control Attributes

To add a control, double click with the right mouse button where you want the control. Fill in the information in resulting dialog box about the control.

Choose the control on the left and fill in the appropriate attributes on the right. The control may need a **Variable** name, a **Value** or **Text**. Not all information will be needed for each control. Fill in only the items which are not grayed out.



Control Quick Reference

The following table is a quick reference of what attributes are required for each control.

Control	Variable	Value	Text
Push Button		✓	✓
Radio Button	✓	✓	✓
Check Box	✓	✓	✓
Edit Box	✓		✓
Fixed Text			✓
Varying Text	✓		✓
File Listbox	✓		
ItemSelect Listbox	✓		

Altering Controls

To **MOVE** the control, click on it and drag it to a new position with the left mouse button.

To **SIZE** a control, click on the edge and drag with the left mouse button.

To **DELETE** a control, position the mouse over the control and press the delete key.

Save

Once you are happy with your work, choose **"Save"** or **"SaveAs"** from the **File** menu to save your work to a file. Choose **"Save to Clipboard"** to put the work into the clipboard so that it can be easily pasted into one of your WIL scripts.

View the Script

Take a peek at the resulting script with the **File "ShowScript"** command to begin to get used to what WIL Dialog Scripts look like.

See: Decipher the Script

ShowScript

Here is an example of what a WIL Dialog Editor script looks like. For information on what it all means, see Decipher the Script.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
                                              your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

Decipher the Script

The Dialog Editor follows a specific format when creating your script. For example, here is a dialog box script we created.

The first line sets the format and specifies the version of the Dialog Editor being used.

```
ExampleFormat=`WWWDLGED,5.0`
```

The next section establishes the caption which will appear in the title bar of the dialog box along with the coordinates, size and number of controls in the dialog box.

```
ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12
```

The third section contains the code for the actual controls. Each line has specific information.

```
Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
```

When the first line in the example above is broken down, the parts are as follows.

Code	**Definition**
Example	Dialog Variable Name
01	Control Number
27,113,76,DEFAULT	Coordinates of the control
PUSHBUTTON	Control Type
"DEFAULT",	Variable name
OK	Text
1	Value

Each Dialog script will end with the following line, making it easy to test the PushButton return values.

```
ButtonPushed=Dialog("Example")
```

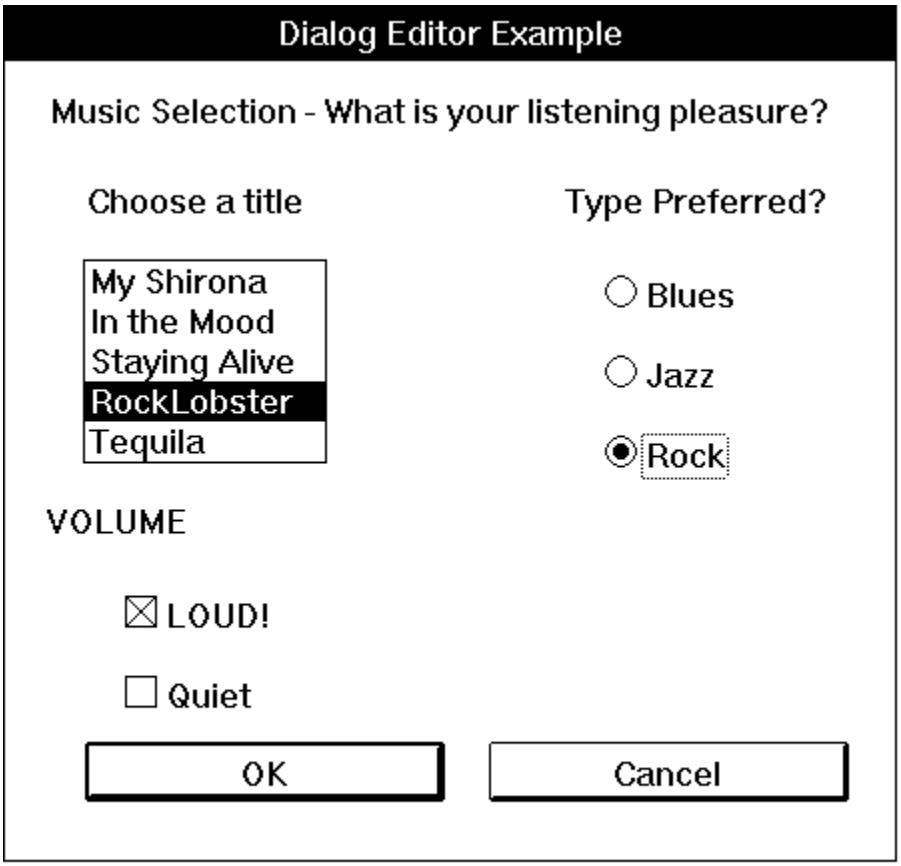
Put all the parts together and the completed script looks like the following.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
                                              your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
```

```
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

Note:

Here is the completed dialog box.



Control Attribute Specifics

Some of the Controls require extra knowledge or special handling.

Setting Variables

Any information which is needed by the Dialog Box Controls should be set up in the script prior to the dialog code. By setting the variables, you can pass lists, files, and set which options are chosen by default.

Push Button

When creating **Push Buttons**, it is a good idea to assign the value of 1 to your "OK" button equivalent and 0 to your "Cancel" button equivalent. Each button will have a separate value. The Dialog Editor adds a line to the end of your script which helps to test return values.

```
Buttonpushed=Dialog"MyDialog"
```

To test a return value do the following:

```
If Buttonpushed == 1 then goto label
```

"Cancel" or the value 0 will generally look for a label **:cancel**. If not found, it will exit. For more information, see **Things to Know** in the **WIL Reference Manual**.

Radio Button

Used in situations to choose one item over another. You can have 9 choices per variable.
In using a **Radio Button**, the variable assigned is the same for each of the choices but the value is different.
For example, the script in a Dialog may look like:

```
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
```

The variable "noise" is the same on both lines. The text and the values are different on each line.

Note: **Radio Button** cannot have a value of 0.

Check Box

Offers a choice of options. Any number may be marked or left unmarked. Each Check Box has its own specific information. Variable, Value and Text are different, allowing the user to choose more than one.

Edit Box

Use this control to create a box in which a choice can be entered by default and then altered by the user.

Fixed Text

Use Fixed Text to display labels, descriptions, explanations, or instructions. The Control Attribute box will let you type an endless amount of information into the text box. However, only about 60 characters will be displayed.

Varying Text

Use Varying Text to grab data which may change, like a date or a password, from somewhere else.

File Listbox

Use File Listbox to allow the user to choose a file from a list box. Set your variable to display a directory path and filemask or the result of **FileItemize**.

```
wbtfiles="C:\WINBATCH\*.WBT"
wbtfiles=FileItemize("*.bak")
```

This box can be tied with the variable to an Edit Box or to Fixed Text. When the user chooses a file, it will be displayed in the Edit Box or in the place of Fixed Text if the variable is the same.

Note: When File Listbox is used, the dialog editor assumes that a file must be chosen before it proceeds. Add the following WIL command to the top of your script if you wish to allow the dialog to proceed without a file selection.

```
IntControl(4, 0,0,0,0)
```

See the WIL manual for more information on **IntControl**.

ItemSelect Listbox

Use the ItemSelect Listbox to allow the user to choose an item from a list box. This option is similar to the WIL commands **AskItemList**, and **ItemSelect**. Set your variable to display a list of items delimited by a tab.

Use **@tab,** a predefined constant, as the delimiter.

```
tunes="My Shirona%@tab%In the Mood%@tab%Staying Alive%@tab%
                              RockLobster%@tab%Tequila"
```

Note: When an ItemSelect Listbox is used, the dialog editor assumes that a file must be chosen before it proceeds. Add the following WIL command to the top of your script if you wish to allow the dialog to proceed without a file selection.

```
IntControl(4, 0,0,0,0)
```

See the WIL manual for more information on **IntControl**.

yesTRUEyesyesyesHELLLP! Generatedextjumpsyesyes13/10/94

Table of Contents

Help file produced by **HELLLP!** v2.3a , a product of Guy Software, on 10/13/1994 for WILSON WINDOWWARE, INC..
The above table of contents will be automatically completed and will also provide an excellent cross-reference for context strings and topic titles. You may leave it as your main table of contents for your help file, or you may create your own and cause it to be displayed instead by using the I button on the toolbar. This page will not be displayed as a topic. It is given a context string of _:_ and a HelpContextID property of 32517, but these are not presented for jump selection.
HINT: If you do not wish some of your topics to appear in the table of contents as displayed to your users (you may want them ONLY as PopUps), move the lines with their titles and contexts to below this point. If you do this remember to move the whole line, not part. As an alternative, you may wish to set up your own table of contents, see Help under The Structure of a Help File.
Do not delete any codes in the area above the Table of Contents title, they are used internally by HELLLP!

Note:

The songs that appear in the ItemSelect Listbox are listed earlier in the script on one continuous line as the variable, tunes.
ie.

```
tunes="My Shirona%@tab%In the Mood%@tab%Staying Alive%@tab%
                                    RockLobster%@tab%Tequila"
```

Variables can be defined above the dialog script or in another WBT file above the statement which calls the dialog file.

View the Script

Take a peek at the resulting script with the **File "ShowScript"** command to begin to get used to what WIL Dialog Scripts look like.

See: Decipher the Script

ShowScript

Here is an example of what a WIL Dialog Editor script looks like. For information on what it all means, see Decipher the Script.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
                                              your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

Decipher the Script

The Dialog Editor follows a specific format when creating your script. For example, here is a dialog box script we created.

The first line sets the format and specifies the version of the Dialog Editor being used.

```
ExampleFormat=`WWWDLGED,5.0`
```

The next section establishes the caption which will appear in the title bar of the dialog box along with the coordinates, size and number of controls in the dialog box.

```
ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12
```

The third section contains the code for the actual controls. Each line has specific information.

```
Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
```

When the first line in the example above is broken down, the parts are as follows.

Code	Definition
Example	Dialog Variable Name
01	Control Number
27,113,76,DEFAULT	Coordinates of the control
PUSHBUTTON	Control Type
"DEFAULT",	Variable name
OK	Text
1	Value

Each Dialog script will end with the following line, making it easy to test the PushButton return values.

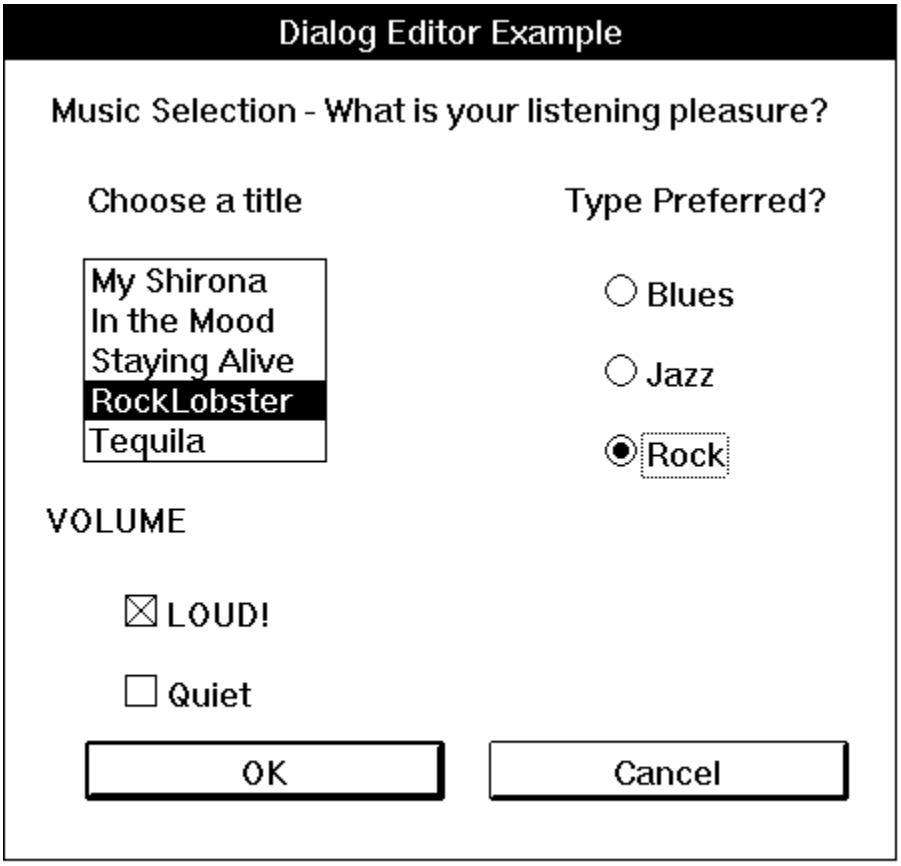
```
ButtonPushed=Dialog("Example")
```

Put all the parts together and the completed script looks like the following.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
                                              your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

Note:

Here is the completed dialog box.



WILSON WINDOWWARE ORDER FORM

Name: _____

Company:_____

Address:_____

_____

City: _____ **St:**_____ **Zip:**_____

Phone: (_____)_____ **Country:**_____

____ WinBatch @ $69.95 : _____.____

____ WinBatch Compiler @$395.00 : _____.____

____ WinBatch 32 @ $99.95 : _____.____

____ WinBatch 32 Compiler @$495.00 : _____.____

____ LetterBox Pro for Win 3.1 @$195.00 : _____.____

Upgrades

____ WinBatch to WinBatch 32 @ $30.00 : _____.____

____ WinBatch Compiler to WinBatch 32 Compiler @$100.00 : _____.____

Shipping

____ US and Canada shipping @ $5.00 : _____.____

____ Foreign air shipping (except Canada) @ $12.50 : _____.____

Total: _____.____

Please enclose a check payable to Wilson WindowWare or you may use Access, Amex, Visa, MasterCharge, or EuroCard. For credit cards, please enter the information below:

Card #:__ __ __ __ - __ __ __ __ - __ __ __ __ - __ __ __ __ **Expiration date: ____/____**

Signature: _____

Where did you hear about or get a copy of our products?

_____

International customers please see note on previous page.

Version 4.0 Example - Devadd.wbt

This is an example of adding "Device=" line(s) to the [386ENH] section of ;the SYSTEM.INI. It is still possible to use this example, however ;there is an alternative, preferred solution for Version 5.0.

EXAMPLE

```
windir = DirWindows(0)
sysini = "%windir%system.ini"
systmp = "%windir%system.tmp"

hIn = FileOpen(sysini, "READ")
hOut = FileOpen(systmp, "WRITE")
found = 0

:nextline
line = FileRead(hIn)
If line == "*EOF*" Then Goto done
FileWrite(hOut, line)
If found == 1 Then Goto nextline
If StriCmp(line, "[386ENH]") != 0 Then Goto nextline
found = 1

; here's where we add the new line(s)
FileWrite(hOut, "Device=DUMMY1.386")
FileWrite(hOut, "Device=DUMMY2.386")

Goto nextline

:done
FileClose(hIn)
FileClose(hOut)
If found == 1 Then FileCopy(systmp, sysini, @FALSE)
Then Message("DEVADD", "Operation complete")
Else Message("DEVADD", "[386ENH] section not found")
FileDelete(systmp)
```

Version 5.0 EXAMPLE - BinaryPokeStr

This example writes a new device= line to SYSTEM.INI. It is *very* fast

Example:

```
NewDevice = "DEVICE=COOLAPP.386"
;
; Change to the Windows Directory
DirChange(DirWindows(0))
;
; Obtain filesize and allocate binary buffers
fs1=FileSize("SYSTEM.INI")
srcbuf  = BinaryAlloc(fs1)
editbuf = BinaryAlloc(fs1+100)
;
; Read existing system.ini into memory
BinaryRead( srcbuf, "SYSTEM.INI")
;
; See if this change was already installed.  If so, quit
a = BinaryIndex( srcbuf, 0, "COOLAPP.386", @FWDSCAN)
if a != 0 then goto AlreadyDone
;
; Find 386Enh section.
a = BinaryIndex( srcbuf, 0, "[386Enh]", @FWDSCAN)
;
;
; Find beginning of next line ( add 2 to skip over our crlf )
cuthere = BinaryIndex( srcbuf, a, @CRLF, @FWDSCAN) + 2
;
; Copy data from beginning of file to just after [386Enh}
; to the edit buffer
BinaryCopy( editbuf, 0, srcbuf, 0, cuthere)
;
; Add the device= line to the end of the edit buffer, and add a CRLF
BinaryPokeStr( editbuf, BinaryEodGet(editbuf), strcat(NewDevice,@CRLF))
;
; Copy remaining part of source buffer to the edit buffer
a = BinaryEodGet(editbuf)
b = BinaryEodGet(srcbuf)
BinaryCopy( editbuf, a, srcbuf, cuthere, b-cuthere)
;
; Save file out to disk.  Use system.tst until it is
; completely debugged
BinaryWrite( editbuf, "SYSTEM.TST")
;
; Close binary buffers
:AlreadyDone
BinaryFree(editbuf)
BinaryFree(srcbuf)
```