FALSEyesnono&About&Printyesyesyesyes WinBatch Help FileWinBatchyes19/04/95

WinBatch
User's Guide

Table of Contents

Overview

From R. Petersen on CompuServe; Our entire company is becoming 100% dependent on dozens of WinBatch programs that make everything hang together.

WinBatch can automate Windows and Windows NT. WinBatch manipulates the Windows interface, Windows applications, and network connections.

So, any operations in or from Windows, or Windows NT, can be done at the click of a mouse button with WinBatch.

WinBatch includes keystroke record and playback , and much more. WinBatch works from script files, so recorded events can be combined with advanced capabilities to automate operations impossible to record.

Testing values, getting system information, working with directories, logging events, and manipulating files are just a few of these capabilities.

WinBatch is often used to assemble reports, install software, automate testing, control processes, acquire data, and add efficiency to the Windows workstation.

WinBatch excels in tailoring the Windows interface to fit any user. Standard operations are easy to program in WinBatch.

WinBatch utilities manipulate:

- The operating system
- The Windows interface
- Any and all Windows applications
- Most MS DOS applications
- Most networks.

WinBatch has two components:

- A system control language called WIL (Windows Interface Language)
- An interpreter that reads a text file written in the WIL language and performs the required manipulations.

Separate versions of WinBatch are available for Microsoft Windows and for Windows NT. WinBatch is continually updated to function with future versions of Microsoft Windows.

It is easy to get started in Windows programming with WinBatch. Useful system utilities are produced quickly with WinBatch. All the things you couldnt do before in Windows are suddenly just a few minutes away.

When projects demand an advanced solution, the depth in WinBatch is ready to speed development. A visual dialog editor, a window information grabber, a debugger and the power of structured programming are part of the WinBatch software.

WinBatch has these capabilities: engineering functions, text manipulation, binary file editing completely in memory, network connectivity, and Windows system manipulation.

Many WinBatch functions accomplish with one line operations that take of pages of forms design, property setting and coding in other programming languages.

WinBatch is optimized for making quick work of custom system

management utilities.

Registering Your Copy of WinBatch

Registered users of WinBatch get manuals, technical support, use of Wilson WindowWare on-line information services, and special offers on new versions of WinBatch and other Wilson WindowWare products.

You must register your copy to obtain these benefits.

You can register WinBatch by mailing your registration card, faxing your registration card, or calling Wilson WindowWare.

WinBatch OnLine

Registered users can share their WinBatch experience with other users on the Wilson WindowWare BBS. They can also share information on the Wilson WindowWare forums on America Online and CompuServe.

The latest versions of WinBatch are available on-line. The addresses here may change at any time check your installation sheet.

Wilson WindowWare BBS: (206.935.5198) requires N,8,1 logon parameters.

CompuServe forum: GO WILSON at Section 15.

America Online: keyword WINDOWWARE.

Internet FTP: www.windowware.com in /wwwftp/wilson.

Internet Web page: http://www.windowware.com/wilson/pages/index.html

How WinBatch is Used

Activate one WinBatch icon or file and you can run from one to thousands of operations. One WinBatch script can squeeze any number of operations into a single batch file that runs just like a Windows program. It can run from a Windows shell or any application that can run another application.

WinBatch excels in controlling other software-both Windows and MS DOS. From getting system information, through controlling software, to accessing the network, WinBatch can do it all from Windows.

What WinBatch Can Do

With 269 general functions and commands, 64 networking functions, 74 physical constants, 24 operators, and 397 exception handling routines, WinBatch can:

- Solve numerous system management problems.
- Run Windows and DOS programs.
- Send keystrokes directly to applications.
- Send menu items directly to Windows 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.
- Call Dynamic Link Libraries.
- Act as an OLE 2.0 automation client.

And much, much more.

About This Manual

WinBatch is an application which uses Wilson WindowWares 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 in WinBatch).

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

Network manipulation functions are dealt with in extensions to WIL. The extenders are included in dynamic link libraries accessed with the WIL AddExtender ( ) function. Each extender has its own help file.

> **Note**: WinBatch is a **batch file** based implementation of WIL. A WinBatch batch file is a text file containing one or more lines of WIL functions and commands.

System Requirements

WinBatch requires an IBM PC or compatible running Microsoft Windows version 3.1 or higher. WinBatch 32 requires a 32 bit version of Microsoft Windows or Windows NT.

WinBatch scripts use about 150 kilobytes of system memory and 2% of system resources. This memory is returned to the system when the WinBatch utility ends.

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, Tina Browning and Jim Stiles.

Installing and Configuring WinBatch

WinBatch is easy to install. You will find the necessary diskettes in your WinBatch package.

If you have purchased WinBatch+Compiler, go to Appendix B for instructions on installing WinBatch+Compiler.

The WinBatch installation program is itself a Windows application, so make sure Windows is running.

Insert your WinBatch disk into your A: or B: disk drive. From the **File Run** menu in **Program Manager, File Manager**, type A:\WSETUP or B:\WSETUP, depending on which floppy drive contains the WinBatch diskette. Follow the prompts from WSETUP. WSETUP will install the necessary files in a directory of your choice. You will be asked for additional diskettes and license numbers.

Note: If you want to install WinBatch over a network, copy the diskettes in the WinBatch package to a temporary directory on a server. Share that directory with read-only permission. Then attach to that directory from a workstation and run the installation from there. The WinBatch installation program, a Windows or Windows NT program, is called WSETUP.EXE. Do not install WinBatch from a floppy drive shared over a network.

License Numbers: Temporary and Permanent

The WinBatch installation program will request entry of both a control number and an ID number. Either upper or lower case will do. These numbers are located inside the back cover of this WinBatch Users guide.

Purchase of the software includes technical support, a full package of the software materials, and notification of updates and enhancements.

WinBatch will run without license numbers, but a screen will be appear to remind users to register the software.

Keep license numbers in a safe place. They will be needed whenever WinBatch is reinstalled.

If you have been issued a temporary license number, it will expire and the evaluation screens will appear. To prevent this, obtain a permanent license number in place of a temporary one.

Once you register your copy of WinBatch, you can enter your registration information. To make the registration screen appear, hold the shift key down while starting a WinBatch utility. Enter the license numbers into the screen that will pop up.

USING WINBATCH

Creating WinBatch Script Files

WinBatch is a script file interpreter. Before you can do anything useful with the WinBatch interpreter, you must have at least one WinBatch script file to interpret.

Your WinBatch installation puts several sample scripts into your WinBatch directory. Suitable icons for these scripts were added to the WinBatch group in the Windows Program Manager, or to the usual place programs are accessed in your version of Windows.

WinBatch script files must be formatted as plain text files. You can create them with WinEdit (Wilson WindowWares optional text editor for programmers), the Windows Notepad or another text editor.

Word processors like WordPerfect, AmiPro, and Word can also save scripts in plain text formatted files.

The .WBT extension is used in this manual for batch file extensions, but, you can use others just as well. If you want to click on a batch file and have Windows run it, be sure that you associate it in Windows with your WinBatch executable program file. When you installed WinBatch, an association is automatically established between WinBatch and .WBT files.

Each line in a WinBatch script file contains a statement written in WIL, Wilson WindowWares Windows Interface Language.

A statement 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. A statement can contain functions, commands, and comments.

You can give each WinBatch script file a name which has an extension of WBT (e.g. TEST.WBT). We'll use the terms WinBatch script files and WBT files interchangeably.

Running WinBatch Utilities

WinBatch system utilities are very versatile. They can be run from icons in the Windows Program Manager.

- as automatic execution macros for Windows via the Run= line in the Windows Win.ini file.
- from macros in word processors and spreadsheets.
- from a command line entry such as the File Run... in the Windows Program and File Managers.
- by double clicking or dragging and dropping file names in the Windows File Manager.
- from menu items on the Windows control menu using WinMacro, an accessory program included with WinBatch.
- from other WinBatch scripts to serve as single or multiple agents, event handlers, or schedulers.
- from any Windows application or application macro language that can execute another Windows program. Software suite macro languages and application builders like Visual Basic and PowerBuilder are examples of these.

Running WinBatch System Utilities

WinBatch utilities run like any other Windows programs. They can run from a command line, an icon in a shell program like the Program Manager in Windows 3.1 and Windows NT, or from a file listing such as the Windows and Windows NT File Managers.

WinBatch utilities are usually run as files with the extension .WBT. When some WinBatch utilities are used, they need information passed to them when they run. This is easily done by passing command line parameters to them.

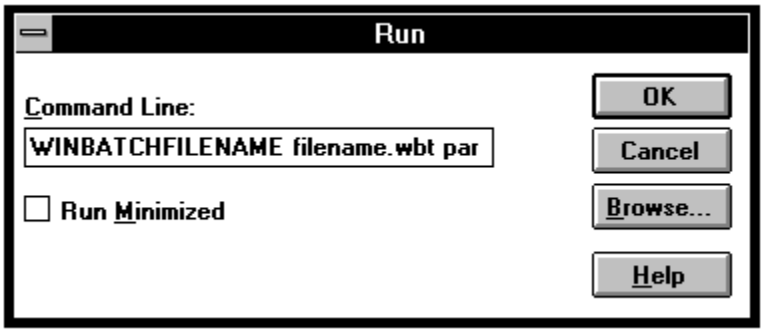
This capability can be used from the command line in the File Run menu items of both the Windows File Manager and the Program Manager. An example dialog is shown below.

Parameters can be also be passed through the command line entry included in the item properties of any icon in Program Manager. Finally, an application can send parameters to a WinBatch utility it launches from a command line or from a function in a macro language.

A command like this runs a WinBatch system utility from a command line or an icon:

```
WinBatchfilename filename.wbt param1 param2 ... param9
```

This command line can be entered into a Command Line text entry box like this one from Program Manager:



The command line is longer than the dialog can show, but it can be easily edited with the arrow keys.

WINBATCHFILENAME is the generic name of your WinBatch executable. The specific, or actual, name for the WinBatch application will change to reflect the operating system in use: Windows 3.1, Windows 95, and the different Windows NT versions.

(See Appendix A, page for more information on file names).

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

"p1 p2 ... p9" are optional parameters (there are a maximum of nine of these) to be passed to the WBT file on startup. Each is delimited from the next by one space character.

Using Icons to Pass Parameters to WinBatch Utilities

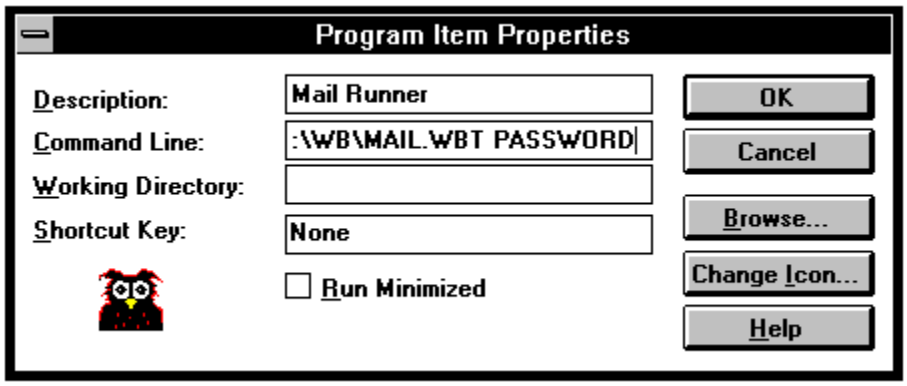
*In order to pass parameters to a WinBatch script file, you **must** run the WinBatch executable, itself, and it must be followed by the name of the WinBatch script file and any other desired parameters.*

WBT files run from the Program Manager as icons must have their complete path in the Properties dialog box in order for command line parameters to be received.

For example, the command line for "**MAIL.WBT**", an imaginary WinBatch utility that runs mail with a password passed as a parameter might be:

"C:\WB\WBAT16I.EXE C:\WB\MAIL.WBT PASSWORD". (The actual command line entered does not include the quotation marks.)

To edit icon properties, highlight the icon, hold down ALT, and press ENTER. The program item properties box should look like the following:



Parameters passed to a WBT file will be automatically inserted into variables named **param1**, **param2**, etc. The WinBatch utility will be able to use these. An additional variable, **param0**, gives you the total number of command-line parameters.

Displaying Passed Parameters in a Message Box.

To display the total number of command line parameters, use **param0** as a variable in a message box. WinBatch works like the DOS Batch language to put parameters into text. Enclosing them in percent (%) signs works in WinBatch, too. This example is a simple one line WinBatch function that:

1. Designs a dialog box with an OK button.
2. Specifies a title.
3. Specifies a message.
4. Puts varying information into the title or the message.
5. Formats the message in more than one line.
6. Returns a value that can indicate whether the operation has succeeded or not.

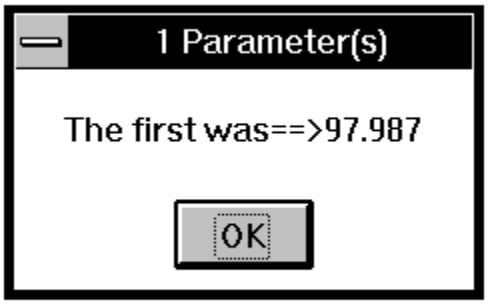
The Message function has this form:

```
Message(title in quotes,message in quotes)
```

The actual statement used to produce this dialog box was:

```
Message("%param0% Parameter(s)", "The first was==> %param1%")
```

It produced:



The command line that executed the utility producing the statement above was:

```
c:\www\wb50\wbat16i.exe c:\www\message.wbt 97.987
```

Note: Full path names were used for both the WinBatch executable file and for the WinBatch utility. Spaces separate the three parts of the command line.

Passing Parameters Between WinBatch Script Files:

You can pass command line parameters from one WinBatch script file to another WinBatch script file. To do this, place percent characters (%) around the variables as in: %variable%.

Example:

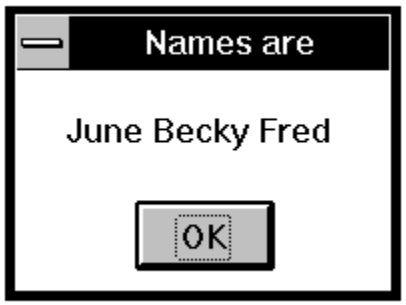
The first WBT calls a second WBT then passes three parameters.

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

TEST.WBT contains the following line:

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

which produces:



WINBATCH FUNCTIONS

Function Reference 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.

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.

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.

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.

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.

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.

Note: The CallExt function is not supported in compiled Exe's.

Example:

```
;This script is a short utility that politely asks for old and
;new names for a file.  Then it checks to be sure that a file of
;the new name does not already exist.

;To illustrate parameter passing, the old and new names are
;passed as parameters to a second script that actually does the
;renaming operation.  A final, and very polite, indeed, dialog
;informs that the deed has been done.

;In a practical version, most of this script would be combined
;and the script would take as passed parameters the old and
;new file names. Then with one CallExt function and two
;parameters, all the messages, and so forth, would work whenever
;needed.

;A CallExt() routine is another way to create new, and unique,
;functions.
```

```
;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

Windows 32 / Windows NT Network Extender

Introduction

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

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.

The following is an abbreviated summary of the network extenders. Refer to the extenders in the on-line help file for function names and more details.

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 Filenames: Appendix A, for more information on DLL filenames.

For more information and a list of functions see the Netware3 Extender Help file. 

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 Filenames: Appendix A for more information on DLL filenames.

For more information and a list of functions see the Netware4 Extender Help file. 

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 Filenames: Appendix A for more information on DLL filenames.

For more information and a list of functions see the Basic Net Extender Help file. 

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 Filenames: Appendix A for more information on DLL filenames.

For more information and a list of functions see the MultiNet Extender Help file. 

Windows 32 / Windows NT Network Extender

This extender provides standard support for computers running 32 bit versions of Windows, such as Windows NT. It may be used in conjunction with other 32 bit Intel extenders.

This extender is only for 32 bit versions of Windows

32 Bit Intel Version

AddExtender("wwnet32i.dll")

32 Bit Dec Alpha Version

AddExtender("wwnet32d.dll")

32 Bit Mips Version

AddExtender("wwnet32m.dll")

32 Bit PowerPC Version

AddExtender("wwnet32p.dll")

Other required DLL's: none

For more information and a list of functions see the Win32 Extender Help file. 

See Filenames: Appendix A for more information on DLL filenames.

DIALOG EDITOR

Dialog Editor

Visual programming of dialog boxes is quick and accurate. Use generic variable names so you can reuse your favorite dialogs.

The WIL Dialog Editor (see Filenames: Appendix A 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.

Note: The WIL Dialog Editor comes with an on-line help file (For the name of the help file see Filenames: Appendix A, 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.

You can have as many as 100 controls in a WinBatch dialog. However, too many controls can be confusing. Aim for simple dialogs with a consistent appearance between different ones.

The WIL Dialog Editor offers quick production of custom dialog boxes for your WinBatch programs.

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", "")
```

For more information see the Dialog Editor Help file.

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.

WinInfo

WinInfo can grab window position settings from windows on display on your monitor.

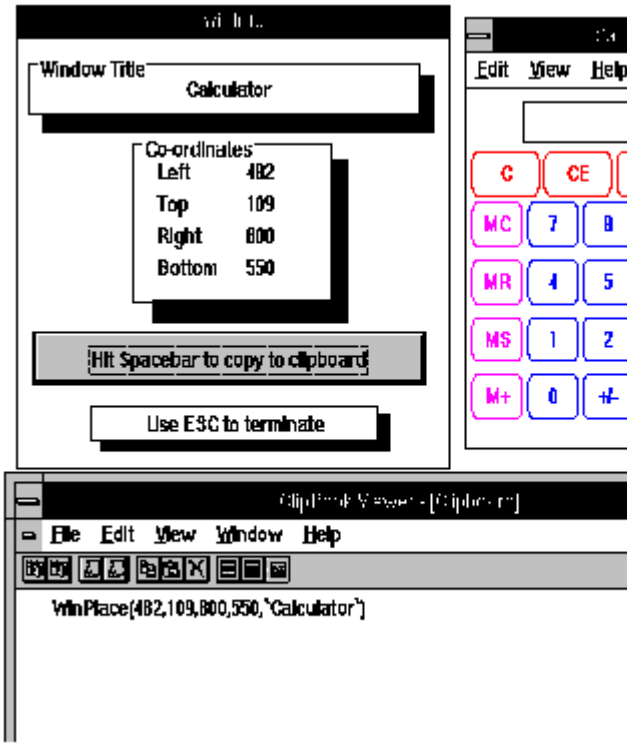
Using WinInfo

WinInfo is a handy window name and position grabber

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.

WinInfo captures coordinates in a 1000 by 1000 format that is relative to the current screen size. Since WinBatch considers every screen to have a 1000 by 1000 size, your sizing will always take up the same percentage of the users screen. One eighth of a screen at 1024 by 768 screen resolution is actually much larger than the same eighth is at 640 by 480 pixels resolution.

Design your dialog boxes to be about 250 by 250 in size or larger. Then they will be prominent at all resolutions.



WinInfo captures relative screen coordinates. 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**.

WinMacro

Recording Keystrokes

Running Macros from the Control Menus

For more information, see the WinMacro Help file  **.**

Recording Keystrokes

WinMacro is the keystroke recorder and program launcher included with WinBatch.

Keystrokes can be sent only to the active application. The SendKeys() function requires that the destination application have the current focus. This can be done with an appropriate WinActivate(). The function SendKeysTo() does all this in one statement. Keystrokes cannot be sent to hidden or to full screen DOS applications.

To use the keystroke recorder, first start WinMacro if you havent already. Find the WinMacro icon and display its icon menu with a mouse click or an Alt Space with the keyboard.

Start recording by selecting the **Begin Macro Record** menu item to start your recording. You can try it immediately to get a feel for it. After recording a few sample keystrokes, click on the icon and select End Macro Record to end the recording process. (Control Shift Home and Control Shift End are the keyboard hot key combinations you can use to begin and end recording.)

The recording will be saved in your WinBatch directory with a name you choose. A .wbm extension will be added.

You can inspect it with a text editor. Note that much of the hard work has been done for you.

Running Macros from the Control Menu

The Windows Control Menu is found at the upper left corner of every main (parent) window.

WinMacro wears another hat. It doubles as a handy script runner always available from any, or all, of your Windows applications. WinBatch utilities and applications can be run from the control menu of any or all applications. You can even add macros to applications that do not have their own internal language.

You can leave WinMacro running to add menu items to the control menu in every Windows program. This menu drops down when the space bar icon in the upper left corner of a Windows program window is activated either by a mouse click or by an Alt Space keystroke combination.

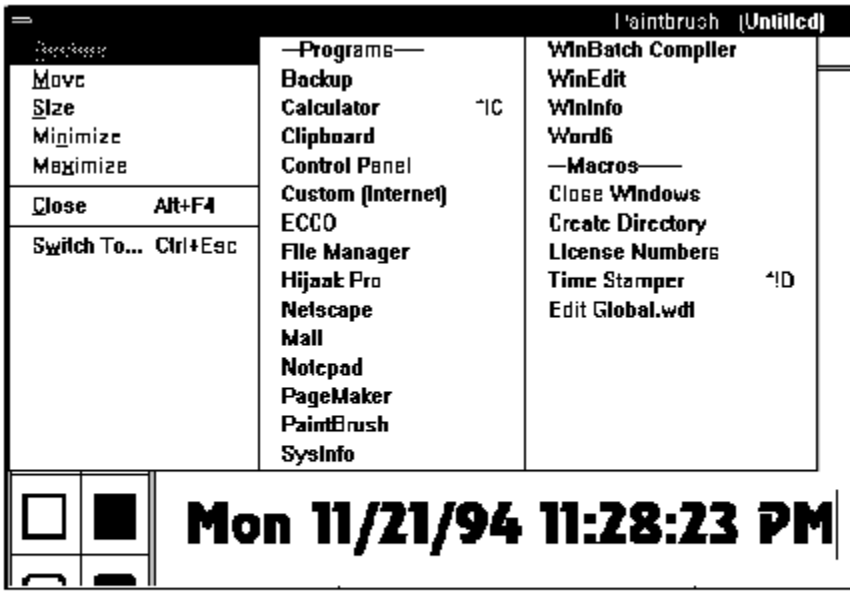
The example here shows how WinMacro can be used to run applications and macros. It is handy to run macros from within any application. Some general macros can work across applications and work within any one of them. For instance, this time and date stamp can be made from a WinBatch script containing nothing more than these three lines:

Example:

```
;TimeDate.WBT
;Sends time and date as keystrokes. Does not use clipboard. Does not
clear clipboard.
focus=WinGetActive()
now=TimeDate()
SendKeysTo(focus,now)
```

In this screen shot, the Time Stamper appears on the right. It is always available by keying the Ctrl-Alt-D key combination. Whether the application is a word processor, a personal information manager, a database, or even a paint program, the time and date is easily entered.

This easy access to system management utilities may be simple, but easy time-savers like this yield major benefits. Minutes saved for one person add up quickly when multiplied over the number of times used and the number of people doing the using.



To add items to the control menu, go to your WinBatch directory and use WinEdit (from Wilson WindowWare) or the Windows Notepad to open the file global.wdf. It will look like this:

Example:

```
Clipboard           : Clipbrd.exe
Control Panel  \ ^P : control.exe
WinEdit             c:\programs\we31\winedit.exe
Notepad             : Notepad.exe
Word6               :\programs\word6\winword.exe
```

Handy access to the sysinfo macro can be a great assistance to technical support efforts. Are there other time savers you could develop?

To add a WinBatch macro that will run on a Windows 3.1 computer, you will need to add a line like this one to the GLOBAL.WDF file. It will let the computer user easily access the sysinfo script you added to your WinBatch directory during the installation of WinBatch.

Example:

```
SysInfo    \^!+S       : c:\wbdir\wbat16.exe c:\wbdir*\sysinfo.wbt
```

Of course, the WinBatch directory, wbdir*, in the example will need to be changed to the directory you use for WinBatch. The \^!+S is optional. It specifies that the Control Alt Shift S keystroke combination can be used to launch this system information display macro.

You can also create .WDF files specifically for your applications. In this way you can add macros to applications that do not have a macro language.

For instance, adding several macros to PageMaker 4.0 can be done through creating a .wdf file called PM4.WDF. WinMacro will know when the executable file PM4.EXE is loaded and will attach the menu file to that application.

Note: You may need to do some detective work to uncover the executable file name for some applications. If you find that your menus do not attach to your application, check the properties for the icon used to launch that application from its Program Manager icon. The file name listed there should work for your WDF file name. The WinBatch directory includes an accessory program called wdf-exe.wbt. Run it and follow directions to uncover the name of an obscure executable file.

FILENAME APPENDIX A

WinBatch and Accessories

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.

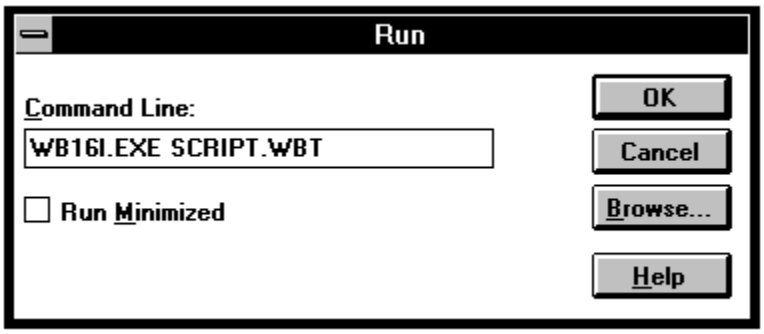
File Name Summary

File names are important in these areas:

1. Running WinBatch scripts.
   WinBatch scripts are text files the WinBatch interpreter translates into action. To do this from a program launcher such as the Windows Program Managers icons or File Run menu item, the file names of WinBatch has to be entered first and it must be followed by a space and the name of a script.

Example:

File Run from Program Manager produces this dialog:



2. Compiling WinBatch files with WinBatch Compiler.
   If you have the WinBatch Compiler, you have the option of including in the executable batch file all, or just the minimum, number of files WinBatch needs to run a particular script. The Compiler includes selection dialogs for choosing options. The file name tables are here for general information.

3. Using Accessories.
   WinBatch comes with a launcher called WinMacro. It also has a window position and name grabber called WinInfo. Finally, WinBatch comes with a Dialog Editor. File names are used to run these.

WinBatch and Compiler Programs: File Names

Environment	WinBatch	Compiler
Windows 3.1, 3.11	WBAT16I.EXE	WBC-16I.EXE
Windows NT-Intel	WBAT32I.EXE	WBC-32I.EXE
Windows NT-DEC Alpha	WBAT32D.EXE	WBC-32D.EXE
Windows NT-PowerPC	WBAT32P.EXE	WBC-32P.EXE
Windows NT-MIPS	WBAT32M.EXE	WBC-32M.EXE

WinBatch Required DLLs: File Names

(The ?? stands for a two letter code unique to a set of WinBatch installation disks.)

Environment	First	Second
Windows 3.1, 3.11	WBO??16I.EXE	WBD??16I.EXE
Windows NT-Intel	WBO??32I.EXE	WBD??32I.EXE
Windows NT-DEC Alpha	WBO??32D.EXE	WBD??32D.EXE
Windows NT-MIPS	WBO??32M.EXE	WBD??32M.EXE
Windows NT PowerPC	WBO??32P.EXE	WBD??32P.EXE

WinBatch Accessories: File Names

Environment	Dialog Editor	WinInfo	WinMacro
Windows 3.1, 3.11	WWDLG16I.EXE	WINFO16I.EXE	WWMAC16I.EXE
Windows NT-Intel	WWDLG32I.EXE	WINFO32I.EXE	WWMAC32I.EXE
Windows NT -DEC Alpha	WWDLG32D.EXE	WINFO32D.EXE	WWMAC32D.EXE
Windows NT-MIPS	WWDLG32M.EXE	WINFO32M.EXE	WWMAC32M.EXE
Windows NT-PowerPC	WWDLG32P.EXE	WINFO32P.EXE	WWMAC32P.EXE

Network Extenders: File Names

Environment	Novell 3.x	Novell 4.x
Windows 3.1, 3.11	WWN3X16I.DLL	WWN4X16I.DLL
Windows NT-Intel	WWN3X32I.DLL	WWN4X32I.DLL
Windows NT-DEC Alpha	WWN3X32D.DLL	WWN4X32D.DLL
Windows NT-MIPS	WWN3X32M.DLL	WWN4X32M.DLL
Windows NT-PowerPC	WWN3X32P.DLL	WWN4X32P.DLL

Environment	Microsoft	Generic
Windows 3.1, 3.11	WWWN16I.DLL	WWW3A16I.DLL
Windows NT-Intel	WWWN32I.DLL	WWW3A32I.DLL
Windows NT-DEC Alpha	WWWN32D.DLL	WWW3A32D.DLL
Windows NT-MIPS	WWWN32M.DLL	WWW3A32M.DLL
Windows NT-PowerPC	WWWN32P.DLL	WWW3A32P.DLL

File Naming Conventions

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

WinBatch for Windows 3.1 running on a PC with an Intel, or compatible, microprocessor (the majority of installed PCs)will have the file name, WBAT16I.EXE. WinInfo is WINFO16I.EXE. The Dialog Editor is WWDLG16I.EXE. The WinBatch Compiler is WBC16I.EXE.

The First 4--5 Digits in the File Name

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

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

The Second 3 Digits in the File Name

Platform	Filename
Intel 16-bit version (Windows 3.1)	16I
Intel 32-bit version	32I
DEC Alpha 32-bit version	32D
MIPS 32-bit version	32M
PowerPC 32-bit version	32P

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 combinations above will exist.

WinBatch DLLs

A WinBatch utility needs two DLLs to function: a WBO DLL and a WBD DLL.

For WinBatch to find and use them, they must be either in the directory holding the WinBatch utility, or on a DOS or network search path. They can be copied there manually, or automatically with the Large EXEstandalone option of the Compiler.

When a script is compiled with the Large EXE option, all the necessary DLLs will be added to the executable utility. When it runs, these DLLs are extracted and saved in the directory where the WinBatch utility is run.

To decrease file sizes, the Compiler also has a Small EXE option.

Small WinBatch executables will need to find the WinBatch DLLs. They can be in the current directory, or on the DOS path or search path. The easiest way to get them there is to create a simple WinBatch utility that uses all the DLLs, extenders, and so forth. Run this once in any directory on the DOS or network search path.

Once the DLLs are extracted, they can be copied anywhere they will be needed. A convenient place for them is often in the Windows directory since it is always on the search path.

Names for the WinBatch DLLs

The WinBatch DLL names are made up of 3 parts. The first three digits identify the DLL type. The second two digits are used for identification purposes and have no importance in use. The final three digits reference the operating environment of the DLL.

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

Platform	Filename
Intel 16-bit version (Windows 3.1)	16I
Intel 32-bit version	32I
DEC Alpha 32-bit version	32D
MIPS 32-bit version	32M
PowerPC 32-bit version	32P

Here is are examples of a pair of DLLs for use on 16-bit versions of Windows on Intel 386, 486, and 586 class processors.

WBDAK16I.DLL

WBOAK16I.DLL

WinBatch+Compiler

Installing and Using WinBatch+Compiler

NOTE: This section is applicable only if you purchased WinBatch+Compiler. This is NOT a shareware software product. 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.
Because WinBatch+Compiler includes both WinBatch and the WinBatch Compiler, registered users of WinBatch can always upgrade to WinBatch+Compiler at a special price.

The WinBatch Compiler can change a WinBatch .WBT file into any one of the following:

- A small Windows EXE file.
- A standalone Windows EXE file.
- An encoded and encrypted WinBatch script file.
- A password protected WinBatch script file.

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

COMPILER INSTALLATION

WinBatch and the Compiler install from one set of diskettes in your WinBatch+Compiler package. 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 Compiler diskette. Follow whatever instructions WSETUP gives you. WSETUP will create the necessary files in a directory of your choice.

The first time you run 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 and test a WinBatch script file. The Compiler will not test WinBatch macro scripts. Each WinBatch macro script file should have a file extension of .WBT, .WBM, or .WIL.

Running the Compiler in Interactive Mode

Running the Compiler in Interactive Mode

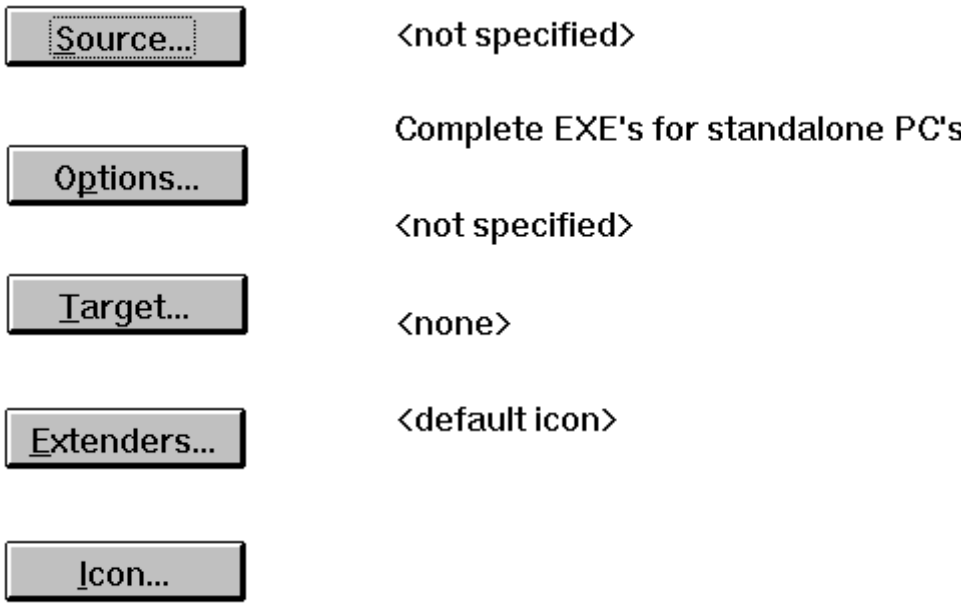
Start the compiler by double-clicking the compiler icon or the Compiler.EXE file name. (or by choosing the appropriate item in any menu system you may be using).

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 displaying the following buttons will be displayed:



SOURCE

Source

The SOURCE button displays a File Selection Box. Select your file or key in 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 1: Keeping source and target names:

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.

Note 2: A WinBatch executable cannot have the same name as an executable application it runs:

Your compiled file with have an extension of EXE. If your WinBatch utility has the same name as the program you want to run from the WinBatch utility, you have a problem you must resolve. The result of this situation is that your utility will run itself. This cannot be resolved by using full path names for the program you want to run.

The solution is to make certain that the WinBatch utility and the other application have different names. Either choose a different name for your utility, or rename the other application and run it with that name.

Note 3: Running an application ONLY from a WinBatch utility:

You can prevent users from running an application from outside of a WinBatch utility. A WIL Run() function can run an executable file name like this:

Run(excel*lib,)

The application can be renamed to excel.lib, an action that will prevent it from being run under Windows. Setting excel.lib to be read only, especially if it is located on a network server with full security capabilities, will make this operation more secure.

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
(includes accessory DLLs, Extenders, OLE 2.0, etc.)

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 automatically written into the current directory. If for some reason, they cannot be written to that directory (perhaps the directory is set to be Read Only), the large compiled file will not run.

The DLLs can also be copied into a directory on a computers PATH and the compiled EXE will find them there and run. The Compiler has a small EXE option that takes advantage of this.

The DLLs need to be placed on the PATH only once. 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.

Small EXE for Networked PC's (without accessory files)

This option is suitable for network file server installation, or for distribution with separate DLL files. DLLs external to the WinBatch utility that uses them must be available in order to run small utilities.

When a small WinBatch utility 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 utility on a PC in which a large standalone utility has been run previously, the small utility can use the same DLLs the large one installed.

> **Hint:** You can automatically install the DLLs on the PATH in a computer. Create a large executable containing only a single statement: Display(1,WinBatch,WinBatch installed. Thank You.). You can change this statement as you like. Then, compile this as a large EXE with all the DLLs your scripts are ever likely to need. Copy it into the Windows System directory, and run it from there. The DLLs will be installed once and for all. Any subsequent batch files run on that computer can be small compiled Exes that dont need the DLLs already installed on that computer.

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 WinBatch interpreter (WBAT16I.EXE, or version specific WinBatch file) 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 a WBT file is to prevent unauthorized personnel from executing it.

Since encryption is easily added to WinBatch utilities, this option is rarely used. In fact, no one has ever been known to use it. Like the human appendix, it reminds one of evolutionary events while avoiding the performance of any useful function.

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.

Note: Your Target exe should not be the same name as the EXE file launched from within the compiled WBT. If you use the same name, Windows will ignore the path in the run command and run what it recognizes as the current exe, the compiled WinBatch executable, again.

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.

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.

WinBatch+Compiler comes with icons you can use. These are in an ICONS subdirectory of your WinBatch directory.

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 operation.

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 a File.Run dialog box.

More Command line info:

For Standalone (Large) EXE compiles
For Compiles of Small EXES
For Encoded EXE'S
For Encrypted WinBatch Utilities

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 Appendix A: Filenames for more information.

For Standalone (Large) EXE compiles:

Five parameters are required beyond the executable name of the compiler. Separate them with spaces.

Parameter 1: The first parameter is the number 1.

Parameter 2: The second parameter must be the source WBT file name. Example: source.wbt

Parameter 3: The third parameter must be the target executable utility file name. Example: utility.exe

Parameter 4: The fourth parameter must be either the name of an icon file or the word NONE. Example: balloon.ico

Parameter 5: The fifth parameter must be either the word NONE or the name of an extender DLL. Example: extender.dll

NOTE: NONE must be used in upper case.

Example:

A complete example with no icon or extender specified:

```
WBC-xxx.exe 1 source.wbt target.exe NONE   NONE
```

A complete example with an icon and an extender specified:

```
WBC-xxx.exe 1 source.wbt target.exe baloon.ico Ext.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 Compiles of Small EXES with no DLLs (for use where DLLs are already on the path, i.e.-on a network):

Four parameters are required.

Parameter 1: The first parameter is the number 2.

Parameter 2: The second parameter must be the source WBT file name. Example: source.wbt

Parameter 3: The third parameter must be the target executable utility file name. Example: utility.exe

Parameter 4: The fourth parameter must be either the name of an icon file or the word NONE. Example: icon.ico. NONE must be used in upper case.

Example:

```
WBC-xxx.exe 2 source.wbt utility.exe icon.ico
or
WBC-xxx.exe 2 source.wbt target.exe NONE
```

For Encoded EXE'S (used where the batch file is used as a subprogram called by a parent WinBatch executable program):

Encoded

Three parameters are required.

Parameter 1: The first parameter is the number 3.

Parameter 2: The second parameter must be the source WBT file name. Example: source.wbt

Parameter 3: The third parameter must be the target file name. Example: utility.wbc

Example:

```
WBC-xxx.exe 3 source.wbt utility.wbc
```

For Encrypted WinBatch Utilities:

Encrypted WinBatch utilities must be executed by the WinBatch interpreter, they are not stand alone executables. The encrypted option is rarely used because the capability of password protection is easily inserted into a compiled WinBatch utility.

Four parameters are required.

Parameter 1: The first parameter is the number 4.

Parameter 2: The second parameter must be the source WBT file name. Example: source.wbt

Parameter 3: The third parameter must be the target utility file name. In this case it takes the extension of WBE. Example: utility.wbe

Parameter 4: The fourth parameter must be the password for access to the compiled WinBatch utility. It is case sensitive.

Example:

```
WBC-xxx.exe 4 source.wbt utility.wbe password
```

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 user's WINDOWS 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 Exes.

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

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

BASIC31yesTRUEBasic Windows 3.1 Network Extender Help Filenono&About&Printyesyesyesyesyes1 9/04/95

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 04/19/1995 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)
```

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 dialogyes19/04/95

DIALOG EDITOR

Table of Contents

Introduction

DIALOG EDITOR

Visual programming of dialog boxes is quick and accurate. Use generic variable names so you can reuse your favorite dialogs.

The WIL Dialog Editor (see Filenames: Appendix A, page **Error! Bookmark not defined.** 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.

Note: The WIL Dialog Editor comes with an on-line help file (For the name of the help file see Filenames: Appendix A, page **Error! Bookmark not defined.**), 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.

You can have as many as 100 controls in a WinBatch dialog. However, too many controls can be confusing. Aim for simple dialogs with a consistent appearance between different ones.

The WIL Dialog Editor offers quick production of custom dialog boxes for your WinBatch programs.

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

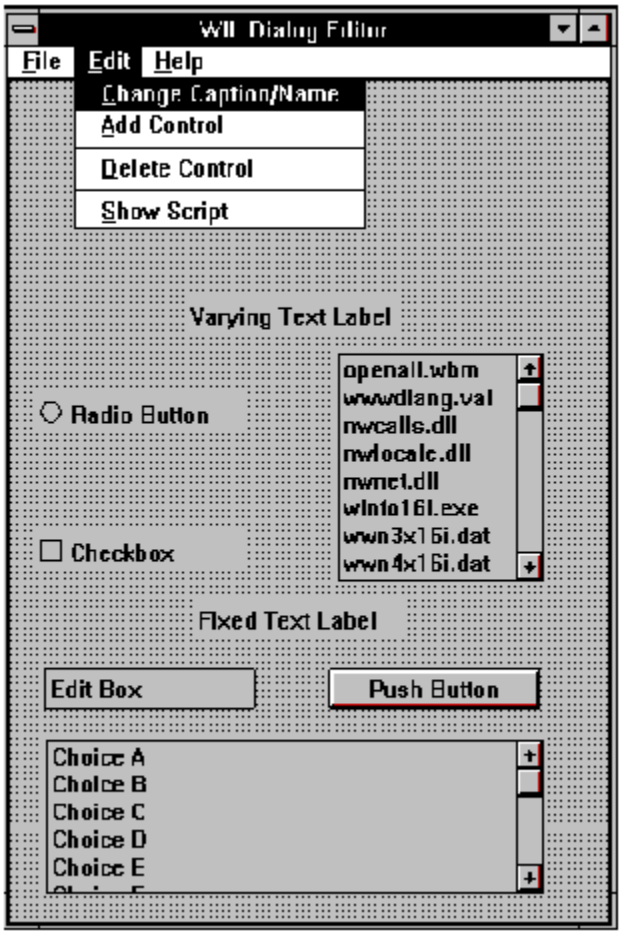
Using the Dialog Editor is easy. Once it is loaded, these hints offer a quick way to become comfortable with dialog box construction.

The dialog editor filename for 16 bit Windows use is: wwd1g16i.exe.

Launch the dialog editor executable, (see Filenames: Appendix A on page **Error! Bookmark not defined.** for filename).

The editor will look like the following:

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.



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.

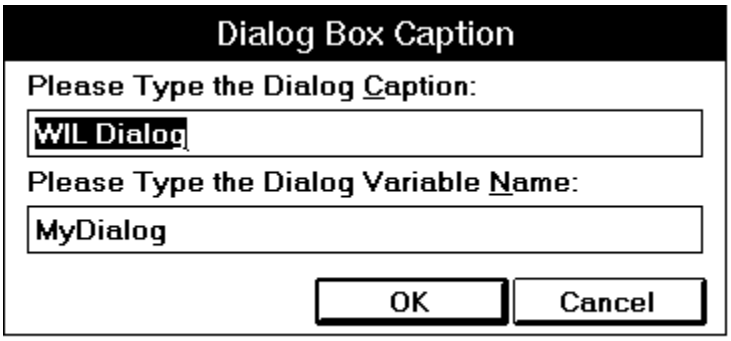
About
Displays the WIL Dialog Editor About dialog which includes the version number of the program.

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.

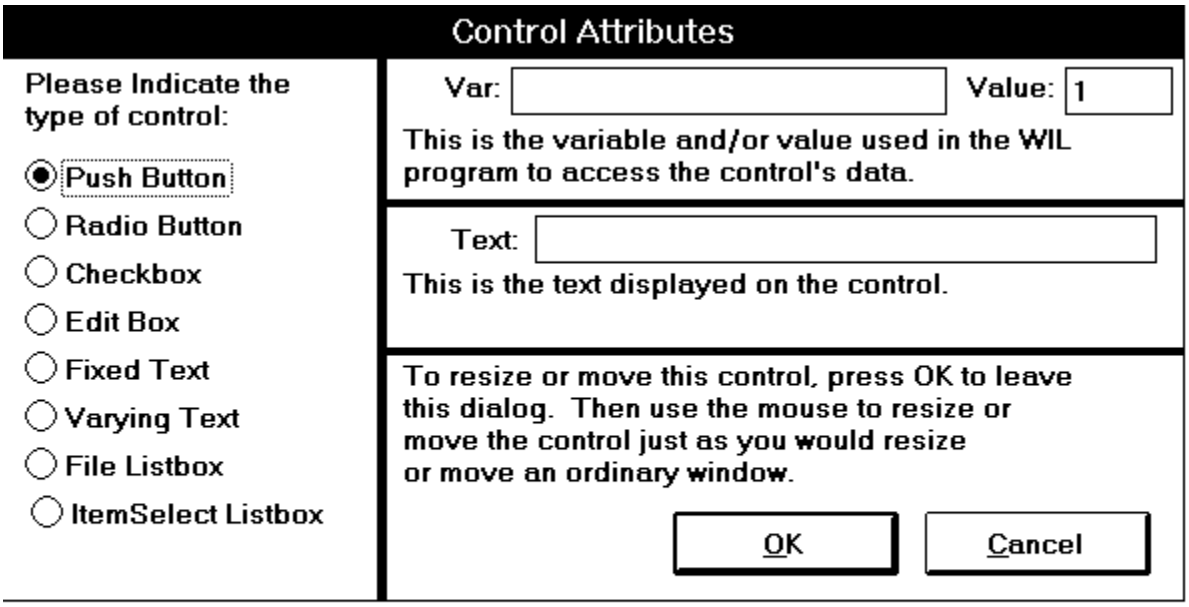
This information can be entered or changed at any time. However, we suggest filling it whenever you start a new dialog box. To change the caption double click on the workspace, (not on a control) with the left mouse button.



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")
```

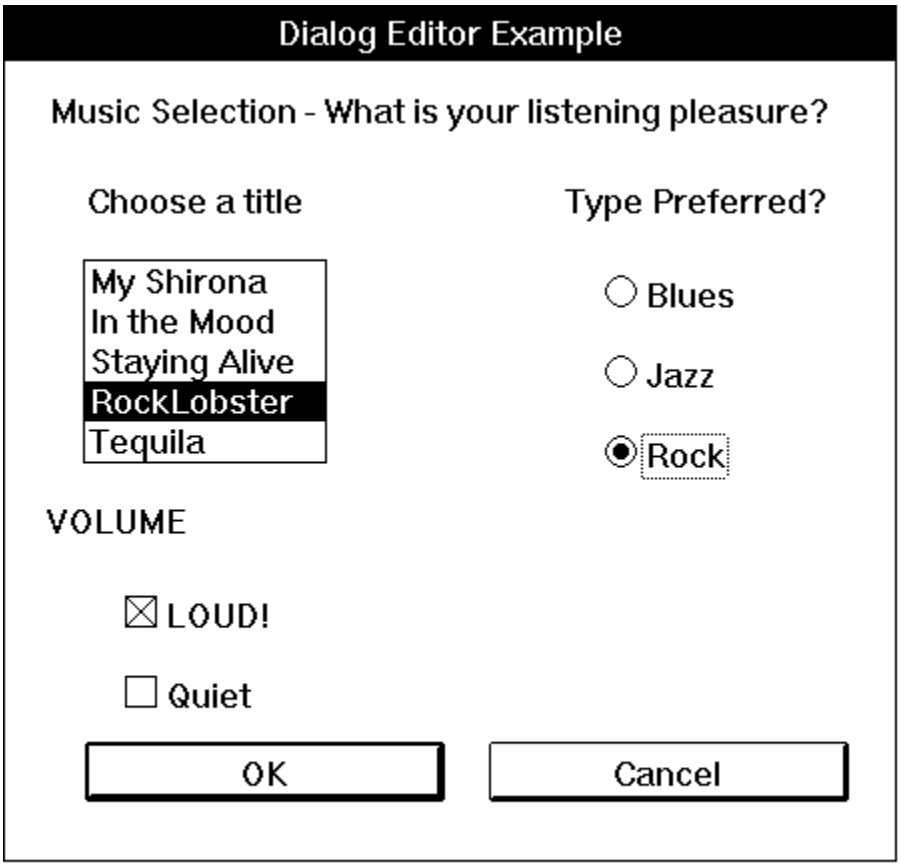
Put all the parts together and the completed script looks like the following.

```
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 "music" 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.

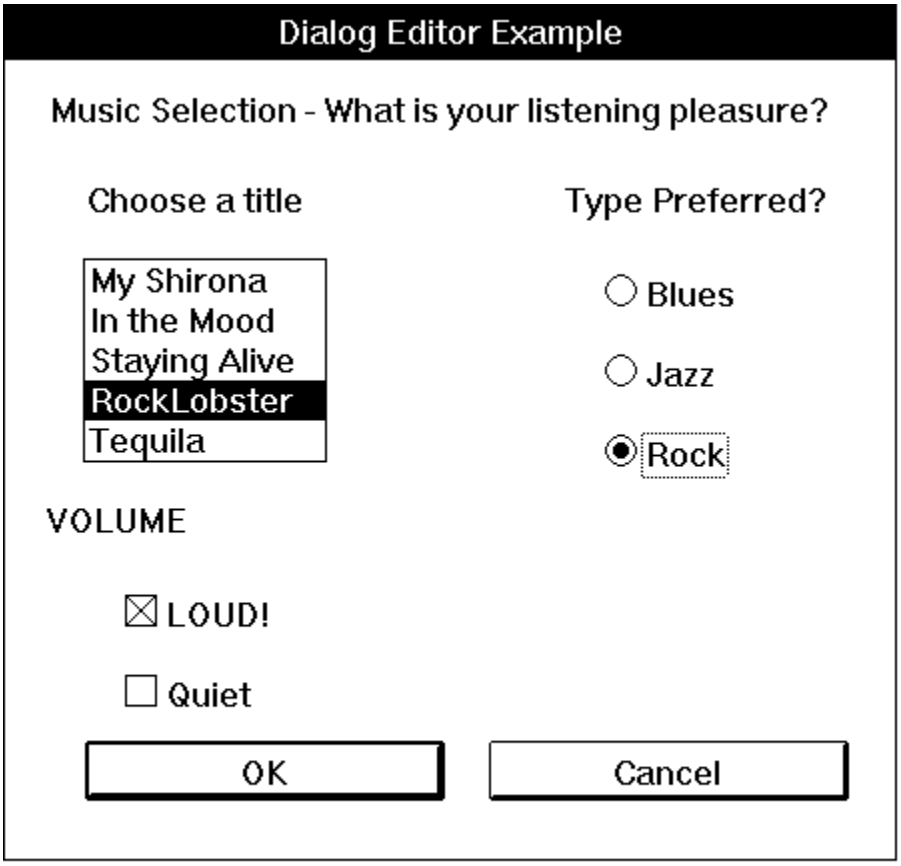
```
ButtonPushed=Dialog("Example")
```

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)
```