WinBatch User's Guide

.

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

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

CONTENTS

INTRODUCTION

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

With almost two hundred functions and commands, WinBatch can:

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

And much, much more.

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

System Requirements
WinBatch requires an IBM PC or compatible with a minimum of 640K memory running Microsoft Windows version 3.0 or higher.

About This Manual

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

This User's Guide includes only topics and functions which are exclusive to WinBatch or which behave differently in WinBatch, as well as additions and changes that have been made since the WIL Reference Manual went to press. It also covers the use of **WinMacro**, a separate program which is included in the WinBatch package.

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

Notational Conventions
Throughout this manual, we use the following conventions to distinguish elements of text:
ALL-CAPS
Used for filenames.

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

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

`Small fixed-width`
Used for WIL sample code.

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

Acknowledgements
WinBatch software developed by Morrie Wilson.

Documentation written by Richard Merit.

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

GETTING STARTED

WinBatch is easy to install. A diskette with your software on it is part of your WinBatch package. You will need to insert it into one of your floppy diskette drives. You will also need to have Windows running and the Program Manager active.

To install WinBatch, you need to run the file WSETUP.EXE from your installation diskette. You can do this from whichever floppy drive (a or b) you will use to install WinBatch. You can use any Windows shell (Program Manager, File Manager, or any other program with a **File, Run** menu item).

You do this by first selecting **File, Run** from the main menu of the shell. Then you enter either A:\WSETUP or B:\WSETUP into the dialog depending on whether you are running wsetup from your a or your b drive. Hit **enter** and WSETUP will take over from there. WSETUP will copy or create the necessary files in a directory of your choice.

Using WinBatch

Creating WinBatch files

WinBatch is a batch file interpreter. Therefore, before you can do anything useful with WinBatch, you must create at least one WinBatch file to run.

WinBatch files are plain text files, which you can create with **Notepad** or a comparable text editor.

Each WIL statement appears on a separate line, and can be a maximum of 255 characters long (refer to the **WIL Reference Manual** for information on the commands you can use in WinBatch). Indentation does not matter.

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

Command-Line Parameters

WinBatch is run with the following command line:

WINBATCH filename.wbt p1 p2 ... p9

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

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

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

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

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

WinBatch Functions--Introduction

This help file 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.

All of the WIL Reference Manual is included in the WIL Reference help file. Cross references to functions described there will automatically appear if both this help file and the WIL Reference help file, WIL.HLP, are in the same directory. You can also switch to the WIL Reference help file from the WIL Reference button on the button bar. You can switch back by using the "Back" button on the button bar, or by using the "History" button and selecting a previous help page. The history list displays the name of the help file along with the page in it.

Two utilities, WinInfo and the WIL Dialog Editor, are included with WinBatch. There are buttons for launching these. The buttons will work only if the files WININFO.EXE and WWWDLGED.EXE are also in the same directory with this help file.

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.

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

IntControl(request#, p1, p2, p3, p4)
Internal control functions.

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

BoxOpen

Opens a WinBatch message box.

Syntax:
BoxOpen (title, text)

Parameters:
(s) title title of the message box.

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

Returns:
(i) always 1.

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

Example:

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

See Also:
BoxShut, BoxText, BoxTitle, Display , Message

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")
BoxTitle("Still processing")
BoxText("Almost done")
BoxShut()
```

See Also:
BoxOpen, BoxText, BoxTitle

BoxText

Changes the text in the WinBatch message box.

Syntax:

BoxText (text)

Parameters:

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

Returns:

(i) always 1.

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

Example:

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

See Also:

BoxOpen, BoxShut, BoxTitle

BoxTitle

Changes the title of the WinBatch message box.

Syntax:
BoxTitle (title)

Parameters:
(s) title title of the message box.

Returns:
(i) always 1.

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

Example:

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

See Also:
BoxOpen, BoxShut, BoxText, WinTitle

CallExt

Calls another WBT file as a separate subprogram.

Syntax:

CallExt (filename, parameters)

Parameters:

(s) filename.wbt — the WBT file you are calling (the extension is required).

(s) parameters — the parameters to pass to the file, if any, in the form "p1 p2 p3 ... pn".

Returns:

(i) always 0.

This function is used to pass control temporarily to a secondary WBT file. The main WBT file can optionally pass parameters to the secondary WBT file. All variables are exclusive (**local**) to their respective files, so that neither WBT file "knows about" variables being used by the other. The secondary WBT file should end with a Return statement, to pass control back to the main WBT file. If a string of parameters is passed to the secondary WBT file, it will automatically be parsed into individual variables with the names **param1**, **param2**, etc. (maximum of nine parameters). The variable **param0** will be a count of the total number of parameters in the string.

Example:

```
; MAIN.WBT
old = AskLine ("RENAME", "File to rename", "")
If !FileExist (old) Then Exit
new = AskLine("RENAME", "New name for %old%", "")
If FileExist(new) Then Exit
CallExt("rename.wbt", "%old% %new%")

; RENAME.WBT
old = param1
new = param2
FileRename (old, new)
Return
```

See Also:

Call, ParseData, Return

DirExist

Determines if a directory exists.

Syntax:

DirExist ([d:]path)

Parameters:

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

Returns:

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

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

Examples:

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

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

See Also:

DirChange, DirMake, DirRemove, DirRename, FileExist

IntControl

Internal control functions.

Syntax:
IntControl (request#, p1, p2, p3, p4)

Parameters:
(i) request# specifies which sub-function is to be performed (see below).

(s) p1 - p4 parameters which may be required by the function (see below).

Returns:
(s) varies (see below).

Short for Internal Control, a special function that permits numerous internal operations. The first parameter of IntControl defines exactly what the function does, the other parameters are possible arguments to the function.
Refer to the **WIL Reference Manual** for information on other sub-functions.

IntControl(12, p1, 0, 0, 0)
Used to direct WinBatch to allow itself to be terminated without warning when Windows shuts down and a batch file is still running

P1	Meaning
0	WinBatch complains on shutdown (default)
1	WinBatch will terminate quietly

IntControl (15, 0, 0, 0, 0)
Returns currently executing WBT file name; the same as the "paramfile" variable.

MsgTextGet

Returns the contents of a Windows message box.

Syntax:

MsgTextGet (window-name)

Parameters:

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

Returns:

(s) contents of the message box.

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

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

Example:

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

WinMacro

WinMacro is a standalone companion program included in the WinBatch package, which lets you create macro files and "attach" them to the control menu of any Windows application. These macros can then be executed, either by selecting them from the control menu, or through the use of a "hotkey." WinMacro also has the ability to "record" keystrokes, which can later be "played back" virtually anywhere in the Windows environment.

A number of help pages describe WinMacro. Use the Browse buttons to page through them.

Starting WinMacro

You can run WINMACRO.EXE just like any other Windows program, using your favorite Windows-program-starting method (keyboard, mouse, Program Manager, File Manager, MS-DOS Executive, Command Post, File Commander, WinBatch, etc.). However, if you will be using WinMacro on a regular basis, you may wish to have it load automatically when you start up Windows. You can do this by adding WINMACRO.EXE to the **LOAD=** line in your WIN.INI file (consult your Microsoft Windows manual for more information).
WinBatch starts up as an icon, and remains active until you either close it or end your Windows session (whichever comes first).

Macro Definition Files

WinMacro definition (**WDF**) files are plain ASCII files which you create and edit. They *must* have a WDF extension, and they *must* be located in the same directory as WINMACRO.EXE. A WDF file contains any number of definition lines, each of which represents an individual command. Each line has the following format:

Title [\ optional hotkey] : program to be executed

Title is the name which will appear on the application's control menu to identify the command.

The **hotkey** is optional; if it is included, it must be preceded by a backslash (\). This is followed by a colon (**:**), and then the **program** which should be executed when the command is selected, with any required parameters. This can be any Windows or DOS EXE, COM, PIF, or BAT file, and you must include the appropriate file extension.

If the program isn't located either in the current directory or on your DOS path, you must include a path specification for it.

To run a WinBatch file, run WINBATCH.EXE, with the name of the WBT file as a parameter.

Let's create a WinMacro definition file, named GLOBAL.WDF:

```
Run  Notepad                  : notepad.exe
Play Solitaire        \ ^F9 : winbatch.exe solitare.wbt
```

(This second line assumes that you have created SOLITARE.WBT as part of the WIL tutorial. If not, just substitute any WBT file name).

GLOBAL.WDF is a special file name. When WinMacro starts up, it looks for this file.

If present, WinMacro loads it, and attaches its contents to the control menu of every window currently running, as well as any windows that may subsequently be opened (the **control menu,** also known as the system menu, is the menu that you access by pressing **Alt-Space**, or by clicking the little box which appears at the left side of the title bar of almost all application windows).

Go ahead and start up WinMacro, then access the control menu of any open window.

You should see that the two commands in your GLOBAL.WDF file have been attached to the control menu, and both are now available for your use.

You can run these user-defined commands by selecting them from the menu. In addition, because you have defined a hotkey for the "Play Solitaire" command, you can run it from any window by pressing **Ctrl-F9**.

Hot keys

You can assign a hotkey to any WinMacro definition line. A hotkey consists of the **Ctrl** key plus any letter (**A** - **Z**) or function (**F1** - **F16**) key. In addition, you can optionally use the **Alt** and **Shift** keys:

Key	Char
Ctrl	**^**
Alt	**!**
Shift	**+**

Here are some examples of valid key combinations:

Hotkey	Equivalent keystrokes
^F5	Ctrl-F5
^!F5	Ctrl-Alt-F5
^+F5	Ctrl-Shift-F5
^!+F5	Ctrl-Alt-Shift-F5
^D	Ctrl-D
^!D	Ctrl-Alt-D
^+D	Ctrl-Shift-D
^!+D	Ctrl-Alt-Shift-D

Application Specific Macros

In addition to GLOBAL.WDF, you can create **application-specific** WinMacro definition files.

They have the form **progname.WDF**, where "progname" is the name of the application's COM or EXE file.

So, if you wanted to have a WDF file which would apply only to Notepad, you would name it NOTEPAD.WDF. Its contents would be attached *only* to *Notepad's* control menu, and its hotkeys would be active *only* when *Notepad* was the active window.

WinMacro loads application-specific WDF files *after* GLOBAL.WDF, so if you have, for example, a NOTEPAD.WDF file, it's contents will be attached to Notepad's control menu *in addition to* (not instead of) GLOBAL.WDF. If you define the same hotkey in GLOBAL.WDF *and* NOTEPAD.WDF, the one in NOTEPAD.WDF will apply.

If you edit a WDF file while WinMacro is running, and want to see the changes reflected in the current menus, select **About/Reload** from the WinMacro icon's menu. All windows will be updated.

Recording Keystrokes

Another feature of WinMacro is the ability to record keystrokes to a file, which can be played back at a later time.

To do this, make sure that WinMacro is running, and then type **Ctrl**-**Shift**-**Home** from any window, or select **Begin Macro Record** from the WinMacro icon's menu.

WinMacro will present you with a menu of existing WBM files. If you want to overwrite an existing file, select its name from the menu; otherwise, enter a name for the file you wish to create in the edit box (a WBM extension will automatically be added), and press the **Enter** key or click on the **OK** button**.** At this point, the icon will begin flashing, indicating that you are in record mode.

Once you are in record mode, every keystroke you type will be recorded to your WBM file. Mouse movement and mouse clicks are *not* recorded.

To end record mode, type **Ctrl**-**Shift**-**End** from any window**,** or click on the flashing WinMacro icon and select **End Macro Record** from the menu. The icon will stop flashing.

Once you have created a WBM keystroke macro file, you can assign it to a hotkey in a WDF file, using the steps outlined above. You use WinBatch to run WBM files, the same way you do with WBT files.

WinMacro Example

Let's create a macro for Solitaire which will cycle to the next deck back design (sound familiar?).

First, WinMacro should be running.

Next, start up Solitaire, and make sure that it is the current window.

Now, activate keystroke record mode, as outlined above, and name the file SOLITARE.WBM.

Once the WinMacro icon begins flashing, we're ready to record. Enter the following series of keystrokes:

Alt-G
C
Cursor right
Space
Enter

And end record mode with **Ctrl Shft End**. Now, create a WinMacro definition file named SOL.WDF, containing the following entry:

```
Change deck design          \ ^C  : winbatch.exe solitare.wbm
```

Finally, select **About/Reload** from the WinMacro icon's menu. Your new command is now available from the Solitaire control menu, or simply by typing **Ctrl**-**C** when the Solitaire window is active.

WBM Files

If you look at a WBM file, you will see that it is nothing more than a series of one or more **SendKey** statements.

For example, the SOLITARE.WBM file that we just created looks something like this

```
; Recorded Macro D:\WINDOWS\BATCH\SOLITARE.WBM
SendKey (`!gc{RIGHT} {ENTER}`)
; End Recorded Macro
```

If you glance back at the SOLITARE.WBT file which appears at the end of the **Tutorial** section of the **WIL Reference Manual**, you will find a line which looks amazingly like the middle one above (**~** has the same meaning as **{ENTER}**). This demonstrates that WBM files are simply WBT files in disguise.

So, why do we use different extensions for the two types of files? Consider, if you will, that a WBT file is a standalone program, which can be run from the Program Manager or File Manager.
It starts up whatever other programs it needs, does its work, and cleans up after itself.

A WBM file, on the other hand, is only a program fragment. When called, it sends a sequence of keystrokes to the active window, but it neither knows nor cares what window that may happen to be.

In Solitaire, **Alt-G** selects the **Game** menu; in another program, it may trigger the **Goodbye** function.

Needless to say, WBM files should be played back *only* in the window where they were recorded, and the easiest way to ensure this is to attach them to *application-specific* WDF files, as we have done here with Solitaire. That's why we distinguish them from regular WBT files.

However, because SendKey is a perfectly respectable WinBatch function – and because WinMacro *does* generate **SendKey** statements – it is quite useful to be able to record a WBM file, and later incorporate it into a full-fledged WinBatch file.

Suppose that we had a one-line WinBatch WBT file like this:

```
RunZoom ("sol.exe", "")
```

and we wanted to follow that with a **SendKey** statement to change the deck design every time the file was run. Instead of laboring over the WinBatch manual to find the cryptic symbols necessary to accomplish such a feat, we could simply use the WinMacro record feature to create a WBM file, as we did above, and then paste the resulting **SendKey** statement into the WinBatch WBT file:

```
RunZoom("sol.exe", "")
SendKey(`!gc{RIGHT} {ENTER}`)
```

You can also use your favorite editor to remove any accidental keystrokes you make when you are recording a WBM file.

Unrecordable Areas in WinMacro

WinMacro is unable to record keystrokes entered in Windows' **System Modal Dialog Boxes**. These include the dialog boxes in the MS-DOS Executive window, as well as dialog boxes generated by severe system errors. By the same token, WinBatch cannot play back keystrokes in these types of dialog boxes.

UTILITIES

Dialog Editor

The WIL Dialog Editor (WWWDLGED.EXE) provides a convenient method of creating dialog box templates for use with the **Dialog** function.

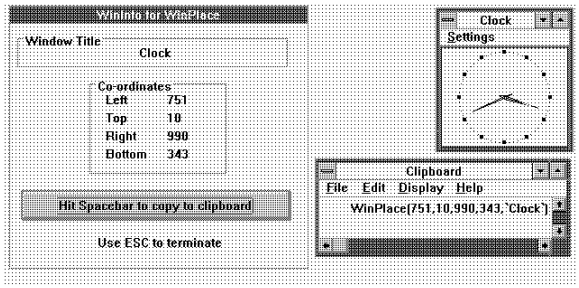
It displays a graphical representation of a dialog box, and allows you to create, modify, and move individual controls which appear in the dialog box.

After you have defined your dialog box, the Dialog Editor will generate the appropriate WIL code, which you can save to a file or copy to the Clipboard for pasting into your WIL program.

The WIL Dialog Editor comes with an online help file (WWWDGEDT.HLP). Simply select the **Help** function in the Dialog Editor for detailed instructions on using the program.
If the file WWWDLGED.EXE is located in the FIL-CMDR.HLP directory , you can try it by clicking the button here. You can also load it from the menu button bar at the top of this screen.

WinInfo

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



You'll need a mouse to use **WinInfo**. While **WinInfo** is the active window, place the mouse over the window you wish to create the **WinPlace** statement for, and press the spacebar. The new statement will be placed into the Clipboard. Then press the **Esc** key to close **WinInfo**.

If the file WININFO.EXE is located in the FIL-CMDR.HLP directory, you can try it by clicking the button here. You can also load it from the menu button bar at the top of this screen.