

**yesnono&About&PrintyesTRUEyesyesyesErrata  
and Tech Help Techyes19/04/95**

# Errata and Tech Help

## Table of Contents

### Your Most Common [Questions](#)

### Technical Stuff (How To...)

[Add and Delete](#)

[Program Manager Groups and Icons](#)

[System.Ini and Device= lines](#)

[Carriage Return Line Feed](#)

[Dialog Editor Button Selection](#)

[Error Trapping](#)

[Exit Windows Quietly](#)

[Hide Icon](#)

[Hourglass / Mouse Pointer](#)

[Recovering from Cancel](#)

[Restarting Windows - Internal Control Functions](#)

[Terminating WIL processing](#)

[Timing Script - TimeWait Function](#)

[Uninstall / Remove Old Version](#)

[Using Multiple lines in Messages](#)

[Window on Top](#)

### [Parameters](#)

#### [Command-Line Parameters](#)

[Passing Parameters From the Command Line](#)

[Passing Parameters Between WBT's](#)

[Passing Information Between WinBatch Exe's and Calls](#)

[Using Association to Pass Parameters](#)

[Passing Parameters Between a WBT and a DOS Batch](#)

### ERRATA Section and Things We Shoulda Told Ya

#### Manual Goofs or Changes

[EnvironmentSet Example Change](#)

[Problem.wbt](#)

[WinActivChild](#)

#### More Info

Buttonnames

Converting Integer Numbers to Floating Point Numbers

More DllCall Info

FileExtension

FileRead Example

GetTickCount

Partial Window Names

Screensaver Info

Sending Keystrokes to DOS Boxes

TimeDiffSecs and TimeDiffDays

UNC - Unified Naming Convention

## Unusual Errors / Problems

Encrypted Encoded verification failed

Dialog Box Interface and CTL3DV2.DLL

RunWait Failing

EXE Keeps Launching

IgnoreInput

Windows NT - Registration Database Entries

## Questions

How do I get rid of the icon while WinBatch is running?

How do I add an Icon into the Program Manager Group?

How do I add a Device= line to my System.ini without overwriting an existing line?

What does it mean when I get the "Encrypted/Encoded Verification Failed" error message in a compiled EXE file?

My WinActiveChild command is not working?

How do I get my long lines to wrap to the next line?

How do I tell which button was pushed by the user when exiting a dialog box generated by the Dialog Editor?

How do I trap errors?

How do I get WinBatch to not terminate when a Cancel button is pushed?

How do I restart Windows?

How do I terminate a WinBatch Script?

How do I get a window to stay on top?

How do I Pass Parameters?

How do I cram multiple lines into a message or more information into a function without exceeding the maximum line length.

Is there a Timing Function?

What does UNC mean?

How do I determine if the Screensaver is on?

How do I send special keys to a DOS app?

When I run a compiled EXE it seems to launch itself in a continuous loop.

Does IgnoreInput ignore everything?

When using RunWait, why does WinBatch continue before the program has finished loading?

My Dialog Boxes look UGLY and bizarre!! What can I do?

## Program Manager Groups and Icons



How do I add an Icon into the Program Manager Group?

There is a good example of how to add and delete Program Manager Groups and Icons in the WinBatch sub directory, WBTCODE. Look for DDETEST.WBT.

## System.Ini and Device= lines



How do I add a Device= line to my System.ini without overwriting an existing line?

The Binary Functions must be used to add Device= lines to the [386 enhanced] section of the System.ini.

The following example is from the BinaryPokeStr function:

```
; This example writes a new device= line to SYSTEM.INI
; It is *very* fast
NewDevice = "DEVICE=COOLAPP.386"
;
; Change to the Windows Directory
DirChange(DirWindows(0))
;
; Obtain filesize and allocate binary buffers
fsl=FileSize("SYSTEM.INI")
srcbuf = BinaryAlloc(fsl)
editbuf = BinaryAlloc(fsl+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)
```

## Carriage Return Line Feed



How do I get my long lines to wrap to the next line?

One way, besides using the built in `@crlf` and `@tab` string constants is to use the functions **Num2Char** or **StrCat** to accomplish this.

### Example:

```
cr=Num2Char(13) ; 13 is a carriage-return
lf=Num2Char(10) ; 10 is a line feed
Message("", "This is line one %cr% %lf% This is line two")
```

Or...

```
cr=Num2Char(13)
lf=Num2Char(10)
crlf=StrCat(cr, lf)
Message("", "This is line one %crlf% This is line two")
```

**Note:** `@crlf` and `@tab` are explained in more detail in the WIL Tutorial section under the heading **Nicer Messages**.



## Dialog Editor Button Selection



How do I tell which button was pushed by the user when exiting a dialog box generated by the Dialog Editor?

The Dialog box returns the value of the selected push-button. The Dialog Editor sets the variable by generating the following line for you.

```
ButtonPushed=Dialog("MyDialog")
```

Use ButtonPushed as your variable.

```
;i.e.  
If ButtonPushed == 1 then Run("program.exe", " ")
```

You can of course, change the variable to something which has more meaning in your script.

More information is available in the WinBatch User's Guide.

## Error Trapping



How do I trap errors?

In front of the Run Command, do a call to the last error function.

### Example

```
LastError( )           ;this sets the error memory to zero
ErrorMode(@OFF)        ;this tells the WB we are handling the errors
                        ;ourselves and not to bother the script.
Run( statement         ;just like it originally was)
ErrorMode(@CANCEL)     ;this hands control back to WB which will look for
                        ;error messages.
If LastError !=1932 then Message("Script Died","Error not 1932" ) then Exit
                        ;this stomps on other errors but lets 1932 go through
                        ;so it is never seen.
```

Note: != means Not Equal.

## To Exit Windows Quietly

Sometimes when using WinBatch to exit windows, the system complains. "Are you sure you want to terminate batch processing now?" To bypass this message and exit quietly use IntControl 12.

### **IntControl (12, p1, p2, 0, 0)**

**IntControl** 12 is used to direct WIL and it's parent application (if the parent application supports this function) as to how to handle users either terminating WinBatch via the "Ctrl-Break" keystroke sequence or perhaps a menu item, or by simply exiting windows.

#### **Example:**

```
; We want to refuse termination requests and refuse any attempt to
; exit Windows until the WIL script is complete
;Add codes 2 and 8 making 10

IntControl(12,10,"Close Net apps before exiting Windows", 0, 0 )
```

## Hide Icon



How do I get rid of the icon while WinBatch is running?

There are two ways to get rid of the icon.

1) After the script is debugged and working correctly you can add the following to the first statement of the script.

```
WinHide(" ")
```

If you add it before your script is finished, you won't be able to test that your program is running correctly.

or.....

2) Go into the WWW-PROD.INI file. Under the section [WB16I] add the following statement.

```
HideIcon=1
```

By adding the statement in the .ini file, all .wbt programs you run will be hidden.

## Hourglass / Mouse Pointer



How do I change the mouse pointer to a hourglass?

You can use the new DllCall function to change the cursor to an arrowglass, and to do pretty much anything else that's possible in Windows. Here's an example:

```
; these numbers are obtained from the Windows API documentation
arrowcur = 32512
hglascur = 32514
varowcur = 32516

Display(1, "Next", "Cursor will change to an hourglass")
hcur = DllCall("user.exe", word:"LoadCursor", word:0, long:hglascur)
DllCall("user.exe", word:"SetCursor", word:hcur)
Delay(2)

Display(1, "Next", "Cursor will change to a vertical arrow")
hcur = DllCall("user.exe", word:"LoadCursor", word:0, long:varowcur)
DllCall("user.exe", word:"SetCursor", word:hcur)
Delay(2)

Display(1, "Next", "Cursor will change to a normal arrow")
hcur = DllCall("user.exe", word:"LoadCursor", word:0, long:arrowcur)
DllCall("user.exe", word:"SetCursor", word:hcur)
Delay(2)
```

Note that WinBatch programs automatically restore the cursor to normal on exit, and many WinBatch functions set and reset the cursor as well.

## Recovering from Cancel



How do I get WinBatch to not terminate when a Cancel button is pushed?

If the user presses the **C**ancel button (in most any dialog which has one), the label **:CANCEL** will be searched for in the WIL program, and, if found, control will be transferred there. If no label **:CANCEL** is found, processing simply stops. This allows the program developer to perform various bits of cleanup processing after a user presses **C**ancel.

## Restarting Windows - Internal Control Functions



How do I restart Windows?

WinBatch has several Internal Control functions, **IntControl**, which permit numerous internal operations. If you are having trouble finding a specific command, you may find a solution here. For example, **IntControl** can perform a warm boot and restart windows.

### **IntControl (66, 0, 0, 0, 0)**

In Windows, this function restarts Windows, just like exiting to DOS and typing WIN again. Could be used to restart Windows after editing the SYSTEM.INI file to change video modes.

In 32 bit versions, this function logs the user out of the current session

### **IntControl (67, 0, 0, 0, 0)**

Performs a warm boot of the system, just like <Ctrl-Alt-Del>. Could be used to reboot the system after editing the AUTOEXEC.BAT or CONFIG.SYS files.

In 32 bit versions will cause of reboot of Windows NT machines.

### **IntControl (68, 0, 0, 0, 0)**

Performs a warm boot of the system, just like <Ctrl-Alt-Del>. Could be used to reboot the system after editing the AUTOEXEC.BAT or CONFIG.SYS files.

In 32 bit versions will cause a shutdown of the machine, awaiting power off.

Check out the versatility of **IntControl** in the **WIL Function Reference**.

## Terminating WIL processing



How do I terminate a WinBatch Script?

A currently-executing WIL program can be terminated immediately by pressing the **<CtrlBreak>** key combination. You may need to hold it a second or two. `IntControl(12,...)` can be used to suppress the ability of the user to terminate the batch file. One would suggest the batch file is completely debugged before doing this.



## Timing Script - TimeWait Function



Is there a Timing Function?

TimeWait pauses execution and waits for the date/time to pass. This function can be used from a WinBatch script as a timer. It waits until the time has been reached and then continues execution. Launch a second WBT when the time is reached.

### Example:

```
;Build the current day and attach your desired time using StrCat.
day=timeymdhms()
year=itemextract(1,day,":")
month=itemextract(2,day,":")
day=itemextract(3,day,":")
time=StrCat(year, ":", month, ":", day, ":", "13:45:00")
TimeWait(time)
Run("thereal.wbt", "")
```

## Uninstall / Remove Old Version

When manually removing WB 4.0 from your system, remove the following from your Windows directory

### **Remove these files**

- winmachk.dll
- wwwbat11.dll
- wwwdealr.dll
- wwwdlang.dll
- w10net.dll
- wbanyan.dll
- wlanman.dll
- wnovell.dll
- wtastic.dll
- wsetup2.ovl
- wwwdnetx.dll

These >may< be used by other network apps - unlikely though...

- nwconn.dll
- nwcore.dll
- nwmisc.dll

These two are from Microsoft.

- commdlg.dll
- ctl3dv2.dll

Commdlg.dll is almost certainly used by other apps, it is recommended that you leave it there.

Ctl3dv2.dll is the 3-D effects DLL from Microsoft. It is likely to be used by more and more applications.

**IT MUST BE IN THE WINDOWS/SYSTEM** directory, or else it will whine

## Using Multiple lines in Messages



How do I cram multiple lines into a message or more information into a function without exceeding the maximum line length.

Sometimes it is necessary to have long lines of text appear in a message type box. However, the maximum line length can limit the size of your message. In order to increase the length of a message or function line, concatenate the information together. Set a variable for each of your lines and then use StrCat to glue the lines together.

Here is an example of concatenating several lines of text together to appear in a message.

### Example:

```
info1="                                     Hi!"
info1="This mini app diddles with the Flying Windows Screensaver."
info2="Use this to modify the SSFLYWIN.SCR file to insert a new character."
info3="Don't worry, you can return to the original at any time."
info4="Do you want to continue?"

Message("FunSaver", StrCat(info1,@crlf,info2,@crlf,info3,@crlf,info4))
```

## Window on Top



How do I get a window to stay on top?

The following script is an example of how to get a window to stay on top.

### Example

```
; Hello

; Set our windows to be on top
OurWnd=DllhWnd("")
DllCall("USER.EXE",void:"SetWindowPos",word:OurWnd,word:65535,word:0
        ,word:0,word:0,word:0,word:3)
;
;Try it

Message("This Window","Should stay on top")

;Return to normal mode
DllCall("USER.EXE",void:"SetWindowPos",word:OurWnd,word:65534,word:0
        ,word:0,word:0,word:0,word:3)
;Try it

Message("This Window","Should be normal and fall behind")

exit
```



Test Example

# Parameters

## Command-Line Parameters

### Passing Parameters

From the Command Line

Between WBT's

Using Association

Between WinBatch Exe's and Calls

Between a WBT and a DOS Batch

## 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.

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

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

## Passing Parameters From the Command Line

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 the WinBatch Executable, followed by the name of the WBT file and any other desired parameters.

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

## Passing Parameters Between WBT's

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

For Example:

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

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

TEST.WBT contains the following line:

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



## Passing Information Between WinBatch Exe's and Calls

There are (at least) five ways to pass information from a called WinBatch EXE back to the calling WinBatch EXE:

1. Via an INI file (IniWrite[Pvt] / IniRead[Pvt])
2. Via a text file (FileWrite / FileRead)
3. Via a file name (FileCopy or FileRename / FileExist)
4. Via the clipboard (ClipPut / ClipGet)
5. Via a window (Display or Message / MsgTextGet)

Each method involves a write/read function pair (shown in parentheses): first the called EXE places the desired information in an accessible location using the "write" function, then the calling EXE retrieves the information using the "read" function.

Here is an example using the first (INI) method, since it's easy, clean, efficient, and safe:

### Example:

```
; MAIN.WBT (which can be compiled to an EXE)
number = AskLine("SquareIt", "Please enter a number:", "")
If number == "" Then number = 0
RunWait("squareit.exe", number)
result = IniRead("SquareIt", "Result", "")
Message("SquareIt", "%number% squared is %result%")

; SQUAREIT.WBT (which should be compiled to SQUAREIT.EXE)
square = param1 * param1
IniWrite("SquareIt", "Result", square)
```

## Using Association to Pass Parameters

Associate a file type as you normally would by using File Associate except choose the WBT or WBT.EXE which launches the application instead.

To pass parameters your code should look like:

```
RunWait("Winword.exe", %param1%)
```

If a DOC file is clicked on in the File Manager the filename will be passed as param 1.

To keep the application from failing when launched from an icon add the following line above the run statement to define param 1.

```
If Param0==0 then Param1=""
```

## Passing Parameters Between a WBT and a DOS Batch

Here is an example of how to pass parameters between WinBatch and a DOS Batch file.

The WinBatch script:

```
A="Fred"
b="Sally"
c="123.456"

xxx=strcat(A," ",b," ",c)

Runwait("zing.bat",xxx)
```

or...

```
RunWait("zing.bat", %A%, %B%, %C%)
```

The DOS batch script.

```
REM ZING.BAT

echo First parameter %1
echo Second Parameter %2
echo Third Parameter %3
echo Forth Parameter %4
pause
```

## EnvironmentSet Example Change

The example with the EnvironSet example in the book is wrong. It should be:

**Example:**

```
EnvironSet("DUMMY", "")  
EnvironSet("PATH", "X:\EXCEL")  
RunEnviron("excel.exe", "/E", @NORMAL, @WAIT)
```

Note: The @WAIT is mandatory for 16 bit versions of Windows

## Problem.wbt

This example might not work exactly how you think it will. If you take two integers for example 32 and 37 and divide 32 by 37, you will not necessarily get a floating point answer. This integer divide will result in an answer of 0. Add 0.0 to one of the numbers to get a true floating point answer.

### Example:

```
a1= "An unexpected problem can occur when dividing numbers."
a2= "The problem is in deciding between an integer divide "
a3= "(where the remainder, if any, is discarded) and a floating "
a4= "point divide (where a floating point number is returned)."
```

```
a5= ""
a6= ""
a7= "Let's assume a test. There are 42 questions."
a8= "A student gets 37 of them correct,"
a9= "what is the student's score."
a10= " "
```

```
iQuestions = 42
iCorrect = 37
Score = iCorrect / iQuestions
```

```
a14= " "
a15= "The unexpected result is that the score is %Score%"
```

```
a16= "Reasonable problem? The trap is that WIL will perform an"
a17= "integer divide and return the unexpected answer of Zero."
a18= " "
```

```
a19= "To dig your code out of this trap, simply use floating point"
a20= "numbers when you want a floating point answer."
a21= " "
```

```
a22= "fQuestions = 42.0"
a23= "fCorrect = 37.0"
```

```
fQuestions = 42.0
fCorrect = 37.0
Score = fCorrect / fQuestions
```

```
a24= "Score = fCorrect / fQuestions"
a25= "The correct score is %Score%"
a26= " "
```

```
a27= "Or make the answer look nicer by using the Decimals function"
a28= "and a little formatting."
a29= ""
```

```
a30= "Decimals(0)"
a31= "Score=Score*100"
Decimals(0)
Score=Score*100
a32= ""
a33= "The correct score is %Score%%%"
```

```
text=""
for i=1 to 15
    text=strcat(text,a%i%,@crlf)
next

text2=""
for i=16 to 33
    text2=strcat(text2,a%i%,@crlf)
next

Message("Integer Divide Problem",text)
Message("Floating point solution",text2)
```

## WinActiveChild



My WinActiveChild command is not working?

The command is WinActivChild, without an "E" at the end of Active. There is a 13 character limit to function names.

The "E" was accidentally replaced. Whoops!

Use WINACTIVCHILD (without an "E") instead.

## **Buttonnames**

This function changes the name of buttons which appear in WIL dialogs.  
However, it does not change the name in all functions which display buttons.  
Supported functions are:

- AskLine
- AskFileText
- AskItemList
- AskPassword



## **Converting Integer Numbers to Floating Point Numbers**

Either add 0.0 or multiply by 1.0 to convert an integer to a floating point number

## More DllCall Info

A number of third party DLL's like to write to an address passed in a DllCall. Oftentimes the third-party documentation uses LPSTR's to do this. WinBatch does not detect if a LPSTR was modified, and consequentially does not capture the changed data. In order to do this type of operation, a Binary buffer must be used. In addition, it must be remembered to use the BinaryEODSet function to alter the BinaryBuffer EOD point. It goes something like this:

### Example:

```
DaBinBuf=BinaryAllocate(1024)
BinaryEODSet(DaBinBuf,1024)
flag = DllCall("party3.dll",word:"DaEntryPoint",lpBinary:DaBinBuf)
if flag == 1
    NewData=BinaryPeekStr(DaBinBuf,0,1024)
else
    NewData=""
endif
Message("The New Data is", NewData)
```

## FileExtension

The example in FileExtension is correct but has a catch. The extension you are looking for must be in uppercase.

i.e.

```
If (ext == "COM") || (ext == "EXE")
```

### Complete Example:

```
; prevent the user from editing a COM or EXE file
allfiles = FileItemize("*.*.")
editfile = AskItemList("Select file to edit", allfiles, " ", @unsorted,
    @single)
ext = FileExtension(editfile)
If (ext == "COM") || (ext == "EXE")
    Message ("Sorry", "You may not edit a program file")
else
    Run("notepad.exe", editfile)
endif
```

## FileRead Example

The FileRead example in the book needs an additional asterisk.

**Example:**

```
handle = FileOpen("autoexec.bat", "READ")
line=""
while line != "*EOF"
    line = FileRead(handle)
```

Add an asterisk into the While line on the other side of "\*EOF". The following line is correct.

```
while line != "*EOF*"
```

## **GetTickCount**

GetTickCount does not return the actual number of hardware ticks that occurred, it returns a the number of "Virtual" clock ticks that have occurred since Windows was started.

For normal Windows 3.1, each virtual clock tick is about a millisecond ( 1/1000 of a second). There are times when the tick count is suspended, such as when running a DOS Box full screen.

## Partial Window Names

Those WIL functions which take a partial windowname as a parameter can be directed to accept only an exact match by ending the window name with a tilde (~). For example, **WinShow("Note~")** would only match a window whose title was "Note"; it would **not** match "Notepad".

When using partial windownames as parameters, you can specify the full name if you wish, but in most circumstances, it isn't necessary. Remember that the case (upper or lower) of the title is significant. If the case is not correct, a match will not be made.

## Screensaver Info



How do I determine if the Screensaver is on?

To check if Screensaver is on, find the name of the .scr file in the System.ini.  
then:

### Example:

```
If AppExist("screensaver.scr") then a=1  
else a=0  
message("It is", a)
```

## **Sending Keystrokes to DOS Boxes**

To send keystrokes to a DOS box, the DOS box must be in a window (Not Full Screen). Most keystrokes can be sent to a full screen DOS box, but not all (Due to some unfathomable Windows quirk).

One especially irritating one is the enter key. SendKey can only send the [ENTER](#) key to a Windowed DOS Box.

In order to SendKey the [UP/DOWN](#) arrows to a DOS app NumLock must be off. Use KeyToggleSet to turn the NumLock to off.



## TimeDiffSecs and TimeDiffDays

Note that in order to get a positive time value, Tim1 must be the later time, and Time2 must be the earlier time.

The example in TimeDiffSecs is incorrect. In the third line, Seconds=, the function is misspelled. The function is TimeDiffSecs, NOT TimeDiffSeconds.

### Example:

```
Now=TimeYmdHms()  
Midnight=strcat(strsub(Now,1,9), "00:00:00")  
Seconds=TimeDiffSecs(Now, Midnight)  
Message("Seconds since midnight", Seconds)
```

## UNC - Unified Naming Convention



What does UNC mean?

UNC stands for Unified Naming Convention. UNC names are used to bring order into chaos by using a single naming convention for all names.

An example of a UNC path and filename:

`\\DEPT07\SYS\Excel\Excel.exe`

Server name -	\\DEPT07
Volume or share name -	SYS
Directory -	Excel
Filename -	Excel.exe

## Dialog Box Interface and CTL3DV2.DLL



My Dialog Boxes look not only UGLY but bizarre!! Can I change how they look?

or

Receiving an Error: "CTL3DV2.Dll is incorrectly installed"

The interface of the Dialog Boxes is controlled by CTL3DV2.DLL. This is a DLL from Microsoft used to add the 3D effects to controls and boxes. Per Microsoft specs, it MUST be installed in the **Windows System** directory.

If you have an old Dll or if the Dll is not in the Windows System directory your dialog boxes can look strange or you might possibly receive an error. You can correct this by either turning the 3D effects off or by making sure the Dll is in the proper place.

### To get rid of the 3D effects by telling WinBatch to ignore the Dll.

In the WWWBATCH.INI file add PrettyDialogs=0 to the Main section.

**Example:**

```
[MAIN]
prettydialogs=0
```

If you have several users, you may want to use WinBatch to make this change. At the top of a script use the function IniWritePvt to change the WWWBATCH.INI file.

**Example:**

```
IniWritePvt("main", "prettydialogs", "0", "c:\windows\wwwbatch.ini")
```

You may still receive the error message the first time this script is run. However, the next time the script is run, the change will have already occurred and no error will result.

### To Locate the Dll and the Windows System Directory.

Use the following code to figure out where the DLL is and where the Windows System Directory is located. If they match up, then its installed properly.

**Example:**

```
a=FileLocate("CTL3DV2.DLL")
b=DirWindows(1)
Message(a,b)
```

WinBatch tries to install a current copy of CTL3DV2.DLL. If it finds an old version in your Windows System directory it installs the dll with an extension of NEW. To update to the

current version, rename CTL3DV2.NEW to CTL3DV2.DLL and delete the old dll.

## Encrypted Encoded verification failed



What does it mean when I get the "Encrypted/Encoded Verification Failed" error message in a compiled EXE file?

Generally, this message is telling you that you have not compiled a WBT script that the main EXE is Calling.

Scripts that are called by the main executable must be compiled using the **"Encode for Call's from EXE files"** option. This option will compile the SOURCE.WBT into a SOURCE.WBC file. Your main program will then need to be changed to reflect the change of the name and then recompiled.

In short the steps are:

1. Main EXE - Change the Call statement from source.wbt to source.wbc and recompile using the original option.
2. Compile the Source.wbt using the **"Encode for Call's from EXE files"** option. The name will change to Source.wbc.

## EXE Keeps Launching



When I run a compiled EXE it seems to launch itself in a continuous loop.

Windows is doing it, not us. If a particular EXE file is running, and another one of the same name, regardless of path is launched, it will run a new copy of the original.

Try this

copy NOTEPAD.EXE to a temp directory.

Rename it to calc.exe

run the file. Notice notepad pop up.

Return to the windows directory and run calc.exe

Notice another copy of notepad pops up.

## IgnoreInput



Does IgnoreInput ignore everything?

Unfortunately, IgnoreInput now ignores everything, even SendKey. You must turn it off and then turn it back on as a work around.

## RunWait Failing / Segment Load Failure



When using RunWait, why does WinBatch continue before the program has finished loading?

Many programs now have "Loader" programs which call the REAL EXE, launch it and then exit. WinBatch is following instructions. When the "Loader" exits, WinBatch continues with the script. Therefore the network connections are being deleted before the REAL EXE has had the chance to finish loading. This can cause a **"Segment Load Failure"** or at the very least doesn't work as expected.

You can fix this in one of two ways.

### 1. Find out the real name of the EXE.

Use an example we've provided. WDF-EXE.WBT will tell the REAL name of a program. Launch the program and then run this script. Put the mouse pointer on the program and the dialog box will display the REAL EXE name.

### 2. Instead of using RunWait use a Run in conjunction with a WinWaitClose.

#### Example:

```
Run("xxx.exe", "")
delay(3)
WinWaitClose("MainWindowName")
```



## Windows NT - Registration Database Entries



How do I change the registration database in Windows NT

Windows NT added named data types to the registration database entries. As a result there is a special way to access the named data entries in Windows NT registration databases. The steps are as follows:

- 1) Open a key pointing to the group of data items that contains the desired data item.
- 2) Use the RegSetValue or the RegQueryValue functions to access the data value. The "subkey-string" must contain only the data item name enclosed in square brackets.
- 3) Be sure to close the key when operations are complete.

For example, here is a WIL script which modifies the default printer in Windows NT.

```
newprt = "LJ3,winspool,LPT1:" ;the printer you want to assign as the
                                ;default
regkey = RegOpenKey(@REGCURRENT, "Software\Microsoft\Windows
                        NT\CurrentVersion\Windows")
defprt = RegQueryValue(regkey, "[Device]")
Message("Previous Default printer", defprt)
RegSetValue(regkey, "[Device]", newprt)
defprt = RegQueryValue(regkey, "[Device]")
Message("New Default printer", defprt)
RegCloseKey(regkey)
```

