This file contains the most commonly asked questions and requested sample scripts, along with information on changes in the documentation.

The easiest way to find information on your topic is to search using a keyword. We've included an abundance of keywords to help you find answers quickly.

If you were just wondering "How do I..." look below for your answer.

Check out these topics if you're frustrated because you can't figure out why something isn't working and you're sure you're doing it right. See Fixes.txt for more information on WIL changes.

EnvironmentSet Example Change
Problem.wbt
WinActivChild
BinaryStrCnt
FileExtension
FileRead Example
IntControl(29, p1, 0, 0, 0)
File Delimiters

These topics contain extra information on using WIL functions. See Fixes.txt for more information on WIL changes.

These topics cover common errors which may be encountered.

How do I Pass Parameters?

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?

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

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

How do I create an event driven pause or DELAY?

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?

Question 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
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)
```

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**.

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.

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.

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 )
```

Question 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.

Question How do I change the mouse pointer to a hourglass?

You can use the new DllCall function to change the cursor to an hourglass, 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.

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

If the user presses the **Cancel** 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 **Cancel**.

How to have several different cancels in a script.

```
;; TOP OF FILE
ONCANCEL="EXIT"

Dialog1(xxxxx etc

ONCANCEL="GOTO DIALOG"


Dialog2(xxxxx etc.

ONCANCEL="GOTO DIALOG2"


Dialog(xxxx etc.
```

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, thus 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.**

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.

Question How do I create an event driven pause or delay?

There is no single command which will delay the .wbt script and wait for an event to occur.

The options for creating a delay are:

- Use the **TimeDelay** command. However, the time interval must be specified.

- Use the **RunWait** command. Not always applicable.

- Learn something about the application which is traceable. Things you can check for: files that exist, windows, dialog boxes, system resource percents etc.

For example, a dialog box may be displayed when printing is in progress. Write a loop in the script which checks for the existence of the dialog box. When the dialog box closes the script will continue.

Example of a while loop checking for a window title.

```
while WinExist("Notepad")
          delay(2)
endwhile
```

Question 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()   ;;Get the current date and time

year=itemextract(1,day,":")
month=itemextract(2,day,":")
day=itemextract(3,day,":")

;;concatenate the parts back together adding the time field to the end.
time=StrCat(year, ":", month, ":", day, ":", "13:45:00")

TimeWait(time)
Run("thereal.wbt", "")
```

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

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))
```

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

```
;Check to see whether a 32 bit version of Windows is running.
If WinMetrics(-4)>=4
    ; Set our windows to be on top
      OurWnd=DllhWnd("")
      xx=-1
      DaDll=strcat(DirWindows(1),"USER32.DLL")

      DllCall(DaDll,long:"SetWindowPos",long:OurWnd,long:xx,long:0,long:0, long:0,
long:0,long:3)
      Message("This Window","Should stay on top")
      ;
      ;Try it

      ;Return to normal mode
      xx= -2
      DllCall(DaDll,long:"SetWindowPos",long:OurWnd,long:xx,long:0,long:0,
long:0,long:0,long:3)
      ;Try it

      Message("This Window","Should be normal and fall behind")
      exit
else
      ; 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)
      Message("This Window","Should stay on top")
      ;
      ;Try it

      ;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
endif
```

Test Example

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
- Between WBT's
- Using Association
- Between WinBatch Exe's and Calls
- Between a WBT and a DOS Batch

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.

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%")
```

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)
```

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 param1.

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=""
```

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
```

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

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= " "
a11= "iQuestions = 42"
a12= "iCorrect = 37"
a13= "Score = iCorrect / iQuestions"

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)
```

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.

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

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

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)
```

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
```

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 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.

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 (~).

A tilde (~) used as the first character of the window name will match any window containing the specified string anywhere in its title. For example, **WinShow("~Notepad")** will match a window title of "(Untitled) - Notepad" and a window title of "My Notepad Application", as well as a window title of "Notepad - (Untitled)".

A tilde (~) used as the last character of the window name indicates that the name must match the window title through to the end of the title. For example, **WinShow("Note~")** would only match a window whose title was "Note"; it would **not** match "Notepad". Furthermore, **WinShow("~Notepad~")** will match a window title of "Notepad" and a window title of "(Untitled) -Notepad", but will **not** match a window title of "Notepad - (Untitled)".

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.

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)
```

For an example of working with a screensaver, see the sample file Saver.wbt.

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.

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)
```

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

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
```

Note: If you're using the 32-bit version under Windows 95, you always get 3D dialogs -- the "PrettyDialogs" setting is ignored.

Making the change on more than one pc

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.

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.

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.

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.

Note: IgnoreInput does not work in 32 bit versions of Windows.

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")
```

Question 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)
```

How do I set a DialogEditor dialog box to default to the center of the screen?

Set the initial DialogX and DialogY parameters to a large number (3000)

If WinBatch sees that the dialog box will be offscreen, it overrides it and centers the dialog box.

The example in the manual and help file for BinaryStrCnt is wrong. It should be:

```
; Find number of Device, DEVICE= and device= lines in config.sys
fs1 = FileSize( "C:\CONFIG.SYS" )
binbuf1 = binaryalloc( fs1 )
BinaryRead( binbuf1, "C:\CONFIG.SYS" )
a = BinaryStrCnt( binbuf1, 0, fs1 - 1, "Device=")
b= BinaryStrCnt( binbuf1, 0, fs1 - 1, "DEVICE=")
c= BinaryStrCnt( binbuf1, 0, fs1 - 1, "device=")
BinaryFree( binbuf1 )
d = a + b + c
Message( "Hmmm", "Total Device= lines found in Config.Sys is %d% " )
```

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

Sample files PROGMAN.WIL and DDETEST.WBT are in the subdirectory WBTCode.

PROGMAN.WIL Commands:

AddGroup 'Group Title'
DelGroup 'Group Title'
AddIcon 'Group Title' 'File and Path' 'Icon description' 'Wrk Dir'
DelIcon 'Group Title' 'Icon Description'
ShowGroup 'Group Title' ShowCommand'
ReplaceItem 'Group Title' 'Icon Description'
Reload 'Group Title'
IconDump 'Group Title' 'Icon Description' 'Variable Name'
ProgBuild 'Variable Name'
AddNewData '%grptitle%', '%Variable Name%'

PROGMAN.WIL is a script we've written to make it easier for you to access the Program Manager Groups and Icons. The code in PROGMAN.WIL does not need to be added into your original script. Simply add a one line **Call** statement to your script passing the desired parameters. See DDETEST.WBT for a working example.

The following line would be used to delete an icon from the Accessories group.

```
Call ("PROGMAN.WIL", "DelIcon 'Accessories' 'Notepad'")
```

The **Call** statement is made up of two strings. The first string is the name of the executable being called, in this case, PROGMAN.WIL. The second string contains the parameters to be passed to PROGMAN.WIL.

If you are "calling" PROGMAN.WIL from a compiled WinBatch script, you will need to compile PROGMAN.WIL with the third option on the Compiler, "Encode for Call's from Exe's". After you've compiled PROGMAN.WIL, the extension will change from .WIL to .WBC. You'll need to change the **Call** statement in your WinBatch script to reflect the new file extension.

From a WinBatch executable script your line will be;

```
Call ("PROGMAN.WBC", "DelIcon 'Accessories' 'Notepad'")
```

Instructs Program Manager to create a new group or activate the window of an existing group.

Syntax:

```
Call("PROGMAN.WIL", "AddGroup 'Group Title'")
```

Parameters:

(s)	AddGroup	Entrypoint label into PROGMAN.WIL
(s)	GroupTitle	The group name. If a group already exists with the name specified, the group window will be activated.

See Also:

AddIcon, DelIcon, DelGroup, ProgBuild

Instructs Program Manager to delete an existing group.

Syntax:

```
Call("PROGMAN.WIL", "DelGroup 'Group Title'")
```

Parameters:

(s) DelGroup Entrypoint label into PROGMAN.WIL
(s) GroupTitle The GROUP name.

See Also:

AddGroup, AddIcon,DelIcon, ProgBuild

Instructs Program Manager to add an item to an existing group.

Syntax:

```
Call("PROGMAN.WIL", "AddIcon 'Group Title' 'File and Path' 'Icon Description' 'Wrk Dir'")
```

Parameters:

(s)	AddIcon	Entrypoint label into PROGMAN.WIL
(s)	GroupTitle	The Group name.
(s)	File and Path	The full command line to run the application.
(s)	Icon Description	The title that is displayed below the icon in the group window.
(s)	Wrk Dir	The name of the default or working directory.

See Also:

AddNewData, IconDump,
AddGroup, DelIcon, ProgBuild

Instructs Progam Manager to delete an item from the currently active group.

Syntax:

```
Call("PROGMAN.WIL",   "DelIcon 'Group Title' 'Icon Description'")
```

Parameters:

(s)	DelIcon	Entrypoint label into PROGMAN.WIL
(s)	GroupTitle	The GROUP name.
(s)	Icon Description	The title that is displayed below the icon in the group window.

See Also:

AddIcon, DelGroup,
AddGroup, ProgBuild

Instructs Program Manager to minimize, maximize, or restore the window of an existing group.

Syntax:

```
Call("PROGMAN.WIL", "ShowGroup 'Group Title' ShowCommand")
```

Parameters:

(s) ShowGroup — Entrypoint label into PROGMAN.WIL

(s) GroupTitle — The group name.

(i) Show Command — An integer which specifies the action Program Manager is to perform on the group window.

Show command:

VALUE	ACTION
1	Activates and displays group restored to its original size and position.
2	Activates and displays group as an icon.
3	Activates and displays group as a maximized window.
4	Displays group in its most recent size and position. The currently active window remains active.
5	Activates the group and displays it in its current size and position.
6	Minimizes the group window.
7	Displays group as an icon. The currently active window remains active.
8	Displays group in its current state. The currently active window remains active.

See Also:

AddGroup, DelGroup,
ProgBuild

Instructs Progam Manager to delete an item and record the position of the deleted item. A new item is added at this recorded position by the next AddIcon command.

Syntax:

```
Call("PROGMAN.WIL", "ReplaceItem 'Group Title' 'Icon Description'")
```

Parameters:

(s) ReplaceItem Entrypoint label into PROGMAN.WIL
(s) GroupTitle The group name.
(s) Icon Description The title that is displayed below the icon in the group window.

See Also:

AddIcon, AddNewData, IconDump,

Instructs Progam Manager to remove and reload an existing group. This command can be used to update the groups after modifications have been made by other applications.

Syntax:

```
Call("PROGMAN.WIL", "Reload 'Group Title'")
```

Parameters:

(s)	Reload	Entrypoint label into PROGMAN.WIL
(s)	GroupTitle	The group name.

See Also:

AddGroup, DelGroup, ProgBuild

Queries the Program Manager for information about an item.

Syntax:

Call("PROGMAN.WIL", "IconDump 'Group Title' 'Icon Description' 'Variable Name'")

Parameters:

(s) IconDump Entrypoint label into PROGMAN.WIL
(s) GroupTitle The group name.
(s) Icon Description The title that is displayed below the icon in the group window.
(s) Variable Name Variable name the "calling" program uses to access contents of the return.

Return:

(s) DDE Item Syntax Syntax is returned to the "calling" progam via the variable name.

IconDump returns the DDE Item Syntax for the specified group and icon. The syntax will be returned to the variable name, the fourth parameter of **IconDump**.
PROGMAN.WIL sets the specified group title as the variable - *grptitle*, which can be retreived from the "calling" program.

Each item in a Program Manager group has 9 parameters.

DDE Item Syntax

icondesc	The title that is displayed below the icon in the group window.
cmdline	The name of the executable file for the application. Can include full path and any parameters required by the application.
wrkdir	Specifies the default or working directory.
icofile	The filename for the .ICO file or .EXE file.
xpos	Horizontal position of the icon in the group window.
ypos	Vertical position of the icon in the group window.
index	The integer position of the icon in the file identified by the ICOFILE parameter.
hotkey	An integer specifying the shortcut key.
minimize	Specifies whether an application window is to be minimized when displayed.

(Parameters are in the order returned.)

Syntax Example:

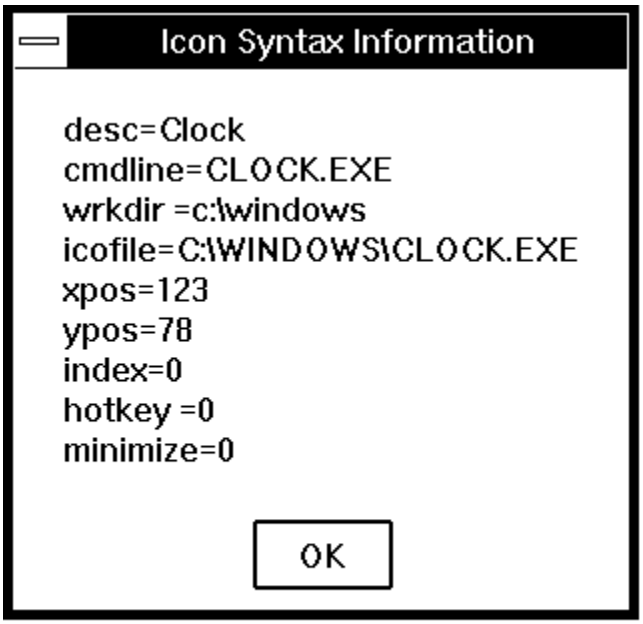
```
"Clock","CLOCK.EXE",c:\windows,C:\WINDOWS\CLOCK.EXE,123,78,0,0,1
```

Example:

```
Call("PROGMAN.WIL", "Icondump 'Accessories' 'Clock' 'joan'")
Display(4,"Dump of Progbuild from group:%grptitle%", joan)

Clipput(joan)    ;;Put the string into the clipboard using ClipPut.

;;;To break the syntax into pieces use the following code.

joan=StrReplace(joan,'"','')           ;;replace quotes with null-string.
icondesc= ItemExtract( 1, joan, ",")    ;;extract each of the items.
```

```
Cmdline = ItemExtract( 2, joan, ",")
wrkdir  = ItemExtract( 3, joan, ",")
icofile = ItemExtract( 4, joan, ",")
xpos    = ItemExtract( 5, joan, ",")
ypos    = ItemExtract( 6, joan, ",")
index   = ItemExtract( 7, joan, ",")
hotkey  = ItemExtract( 8, joan, ",")
minimize= ItemExtract( 9, joan, ",")

dadata=StrCat('desc=',icondesc, @crlf, 'cmdline=',cmdline, @crlf, 'wrkdir
=',wrkdir, @crlf, 'icofile=',icofile,@crlf, 'xpos=',xpos, @crlf,  'ypos=',ypos,
@crlf,  'index=',index, @crlf, 'hotkey =',hotkey,@crlf,  'minimize=',minimize)
message("Icon Syntax Information", dadata)
```

Displays:



To change a parameter in the DDE Item Syntax, see AddNewData.

See Also:

AddIcon, AddNewData, ProgBuild

Queries the Program Manager to allow user to select from a group list, then an icon list.

Syntax:

Call("PROGMAN.WIL", "ProgBuild 'Variable Name'")

Parameters:

(s) ProgBuild Entrypoint label into PROGMAN.WIL
(s) Variable Name Variable Name for the Calling program to access contents of return.

Return:

(s) DDE Item Syntax Syntax is returned to the "calling" progam via the variable name.

ProgBuild creates a list of Program Manager groups. The user selects a group from the displayed list box. A list of icons is created from the selected group. The DDE Item Syntax is returned from the item selected by the user.

The syntax is returned to the Variable Name, the second parameter of **ProgBuild**. PROGMAN.WIL sets the specified Group Title as the variable, *grptitle*, which can be retreived from the "calling" program.

Each item in a Program Manager group has 9 parameters.

DDE Item Syntax

icondesc	The title that is displayed below the icon in the group window.
cmdline	The name of the executable file for the application. Can include full path and any parameters required by the application.
wrkdir	Specifies the default or working directory.
icofile	The filename for the .ICO file or .EXE file.
xpos	Horizontal position of the icon in the group window.
ypos	Vertical position of the icon in the group window.
index	The integer position of the icon in the file identified by the ICOFILE parameter.
hotkey	An integer specifying the shortcut key.
minimize	Specifies whether an application window is to be minimized when displayed.

(Parameters are in the order returned.)

Syntax Example:

```
"Clock","CLOCK.EXE",c:\windows,C:\WINDOWS\CLOCK.EXE,123,78,0,0,1
```

Example:

```
;;;Accessories group / Clock icon chosen for example.
Call("PROGMAN.WIL", "Progbuild 'joan'")
Display(4,"Dump of Progbuild from group:%grptitle%", joan)

Clipput(joan)    ;;Put the string into the clipboard using ClipPut.
```

```
;;;To break the syntax into pieces use the following code.

joan=StrReplace(joan,'"','')           ;;replace quotes with null-string.
icondesc= ItemExtract( 1, joan, ",")    ;;extract each of the items.
Cmdline = ItemExtract( 2, joan, ",")
wrkdir  = ItemExtract( 3, joan, ",")
icofile = ItemExtract( 4, joan, ",")
xpos    = ItemExtract( 5, joan, ",")
ypos    = ItemExtract( 6, joan, ",")
index   = ItemExtract( 7, joan, ",")
hotkey  = ItemExtract( 8, joan, ",")
minimize= ItemExtract( 9, joan, ",")

dadata=StrCat('desc=',icondesc, @crlf, 'cmdline=',cmdline, @crlf, 'wrkdir
=',wrkdir, @crlf, 'icofile=',icofile,@crlf, 'xpos=',xpos, @crlf,  'ypos=',ypos,
@crlf,  'index=',index, @crlf, 'hotkey =',hotkey,@crlf,  'minimize=',minimize)
message("Icon Syntax Information", dadata)
```

Displays:

To change a parameter in the DDE Item syntax, see AddNewData.

See Also:

AddIcon, AddNewData, IconDump,

Adds an item to a group by passing the entire DDE Item Syntax string.

Syntax:

Call("PROGMAN.WIL", "AddNewData '%*grptitle*%', '%Variable Name%'")

Parameters:

(s) AddNewData Entrypoint label into PROGMAN.WIL
(s) *grptitle* The Variable Name returned from **ProgBuild** or **IconDump** containing the original group title.
(s) Variable Name Variable containing the concantenated DDE Item Syntax.

AddNewData allows an easy way to add an item when the contents of the DDE Item Syntax need to be changed. Use **ProgBuild** or **IconDump** to request the DDE Item Syntax of an icon. The syntax can then be parsed, changed and reassembled.

Note: Variable name *grptitle* is returned from **IconDump** or **ProgBuild**. A group name can be supplied instead.

Each item in a Program Manager group has 9 parameters.

DDE Item Syntax

icondesc	The title that is displayed below the icon in the group window.
cmdline	The name of the executable file for the application. Can include full path and any parameters required by the application.
wrkdir	Specifies the default or working directory.
icofile	The filename for the .ICO file or .EXE file.
xpos	Horizontal position of the icon in the group window.
ypos	Vertical position of the icon in the group window.
index	The integer position of the icon in the file identified by the ICOFILE parameter.
hotkey	An integer specifying the shortcut key.
minimize	Specifies whether an application window is to be minimized when displayed.

(Parameters are in order returned.)

Syntax Example:

```
"Clock","CLOCK.EXE",c:\windows,C:\WINDOWS\CLOCK.EXE,123,78,0,0,1
```

Note: When the syntax is used to add an item to a group, the order changes slightly.

```
cmdline    icondesc    icofile    index    xpos    ypos    wrkdir    hotkey
minimize
```

Example:

```
Call("PROGMAN.WIL", "Icondump 'Accessories' 'Clock' 'joan'")
Display(4,"Dump of Progbuild from group:%grptitle%", joan)

Clipput(joan)    ;;Put the string into the clipboard using ClipPut.
```

```
;;;To break the syntax into pieces use the following code.

joan=StrReplace(joan,'"','')           ;;replace quotes with null-string.
icondesc= ItemExtract( 1, joan, ",")    ;;extract each of the items.
Cmdline = ItemExtract( 2, joan, ",")
wrkdir  = ItemExtract( 3, joan, ",")
icofile = ItemExtract( 4, joan, ",")
xpos    = ItemExtract( 5, joan, ",")
ypos    = ItemExtract( 6, joan, ",")
index   = ItemExtract( 7, joan, ",")
hotkey  = ItemExtract( 8, joan, ",")
minimize= ItemExtract( 9, joan, ",")

;;make changes by resetting the variables.
minimize="0"
icofile="moricons.dll"
index="22"

;;Delete the original Icon.  Otherwise a second icon will be added.
Call("PROGMAN.WIL",  "DelIcon 'Accessories' 'Clock'")

;;Put the parts back together in proper order.
data=StrCat(cmdline,',',icondesc,',',icofile,',',index,',',xpos,',',ypos,',',wrkdir
,',',hotkey,',',minimize)
Call("PROGMAN.WIL", "AddNewData '%grptitle%', '%data%'")
```

See Also:

AddIcon, DelIcon, IconDump, ProgBuild

Changes the "default" file delimiter.

p1 new file delimiter

The first parameter for IntControl is the new file delimiter you want to use, and must be a single character. The return value of the function is the previous file delimiter character. If you specify an empty string ("") as the first parameter, the function will return the current file delimiter character but the file delimiter will not be changed.

If you are using the 32-bit version of WIL, and want to make the file delimiter a space for compatibility with existing scripts, you can place the following line at the beginning of each of your scripts:

```
IntControl(29, " ", 0, 0, 0)
```

Conversely, if you want to standardize on a TAB delimiter, you can use:

```
IntControl(29, @TAB, 0, 0, 0)
```

See Also:

File Delimiters

In order to support long file names in Windows NT and Windows 95, which can contain embedded spaces, we have changed the default file delimiter, used to delimit lists of files and directories, to a TAB in the 32-bit version of WIL. In the 16-bit version of WIL, the default delimiter has not changed, and remains a space.

Note that this is the "default" file delimiter. We have added the ability to change the file delimiter to a character of your own choosing, using the new IntControl 29.

The most important functions affected by this change are:

DirItemize
DiskScan
FileItemize

which now return lists delimited by the current file delimiter character.

The following functions, which take file or directory lists as input parameters, now expect the lists to be delimited by the current file delimiter character. However, they now also accept lists delimited with a TAB or a vertical bar ("|", which may be easier to code in a WIL script):

DirItemize
DirRemove
DiskFree
FileAppend
FileAttrSet
FileCopy
FileDelete
FileItemize
FileMove
FileRename
FileSize
FileTimeSet
FileTimeTouch

Note that DiskFree will continue to accept space-delimited lists as input.

See Also:

IntControl 29

When using the INI Private commands, WinBatch always assumes that the INI file is in the Windows directory unless a full path is specified.

WinItemChild

WinItemChild generally returns a list of all the child windows under a parent window. However, if a child windowname is given as the parameter instead of the parent name, WinItemChild will try to return the contents of the child window. Right down to the buttons it finds.

Syntax:

WinItemChild("partial-parent-windowname")

Parameters:

(s) partial-parent-windowname the initial part of, or an entire, window name.

Returns:

(s) a list of all the child windows under the parent.

```
Run("notepad.exe", "")
SendKeysTo("~ Notepad", "!fo")
a=WinItemChild("Open")
message("Items in child window are..." , a)
```