nono&About&PrintyesTRUEnoyesyesyesNEW FEATURES Version 5.0newyes03/10/94

New in WinBatch Version 5

Table of Contents

Help file produced by **HELLLP!** v2.3a , a product of Guy Software, on 10/03/1994 for WILSON WINDOWWARE, INC..

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

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

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

Programming:

Structured programming is now a feature in a batch language for Windows with these functions: GoSub, Select Case, Switch Case, For Next, While End While, If Then Else, and If Else EndIf.

Speed and Operating Improvements:

WinBatch is now up to 500% faster in basic operations and sending keystrokes to other applications. Speed is also enhanced with the capability of sending menu item selections to parent and child windows. Operations in WinBatch have been extended to controlling specifically named child windows. SendKey has been extended to include sending keystrokes to named windows. SendKey also sends keystrokes directly to DOS windows; pasting keystrokes through the clipboard is now a trick of the past.

Date and Time Functionality:

There are several new functions for making scheduling processes easier. GetExactTime and GetTickCount handle fine timing. Eight other functions handle time differences, Julian dates and more.

Networking:

Network operations are handled differently. Previous versions of WIL tried to handle all networks with universal functions. This effort at standardization proved to be cumbersome, inflexible, and inefficient. The WIL Language now supports modular addition of network functions through the function, AddExtender.

This new method of extending the WIL Language will have two clear advantages. First, network support will be needed for only the networks in use. Second, modular network additions will provide network support tailored to specific networks and versions of them. Network changes and updates will be accomplished with the copying of a simple extender.

Floating Point Support:

New in version 5 is floating point support. Many of the new functions take advantage of this. Now WinBatch can serve financial data entry functions.

Also, statistical analyses are easily done. Floating point support is a capability necessary for using WIL in the areas of process control and instrumentation. Now, WIL can attach to a data server with DDE, get data, and send it to a DDE client, such as a spreadsheet, for analysis.

Custom Functions

The WIL Language has always offered convenient access to the capabilities of the Windows Application Programming Interface. Now custom functions can be added in two ways. Custom language extensions can be created to accomplish a wide variety of tasks. These can be added to the functions in WIL in the same way that networking functions are added using the new AddExtender() function.

The second method of adding functionality is through the DllCall() function. Using the C or Pascal languages, a programmer can create a library of custom functions. These can be used throughout the DllCall function of the WIL Language.

List of New Functions

Besides the new Networking functions there are over 100 more new functions. Most are the direct results of requests our users have been kind enough to bring to our attention. While some are clearly breaking new ground for WIL, others continue our interest in making WIL, and Windows programming, easier to use on a daily basis. Many of the functions here qualify as programming "nicities": those functions that do in one function what takes line after line of coding in other language products.

About( ) Displays the About message box.
Acos(fp_num) Calculates the arccosine.
AddExtender(dllfilename) Installs a WIL extender Dll.
Asin(fp_num) Calculates the arcsine.
AskFileText(title, filename, sort mode, select mode) Allows the user to choose an item from a list box initialized with data from a file.
AskFileName(title, directory, filetypes, default filename, flag) Returns the filename as selected by a FileOpen dialog box.
AskItemlist(title, list, delimiter, sort mode, select mode) Allows the user to choose an item from a list box initialized with a list variable.
Atan(fp_num) Calculates the arc tangent.
BinaryAlloc(buffsize) Allocates a memory buffer of the desired size.
BinaryCopy(handle targ, offset targ, handle src, offset src, bytecount) Copies bytes of data from one binary buffer to another.
BinaryEodGet(handle) Returns the offset of the free byte just after the last byte of stored data.
BinaryEodSet(handle, offset) Sets the EOD value of a buffer.
BinaryFree(handle) Frees a buffer previously allocated with Binary Alloc.
BinaryIndex(handle, offset, search string, direction) Searches a buffer for a string.
BinaryPeek(handle, offset) Returns the value of a byte from a binary buffer.
BinaryPeekStr(handle, offset, maxsize) Extracts a string from a binary buffer.
BinaryPoke(handle, offset, value) Pokes a new value into a binary buffer at offset.
BinaryPokeStr(handle, offset, string) Writes a string into a binary buffer.
BinaryRead(handle, filename) Reads a file into a binary buffer.
BinaryStrCnt(handle, start-offset, end-offset, string) Counts the occurrences of a string in some or all of a binary buffer.
BinaryWrite(handle, filename) Writes a binary buffer to a file.
Break Used to exit a conditional flow control statement.
Ceiling(fp_num) Calculates the ceiling of a value.
Continue Transfers control to the beginning of a For or While loop or to a different case statement.
Cos(fp_num) Calculates the cosine.
Cosh(fp_num) Calculates the hyperbolic cosine.
Decimals(#digits) Sets the number of decimal points used with floating point numbers.
DirExist(pathname) Determines if a directory exists.
DllCall(dllfilename/dllhandle, returntype:entrypoint [,paramtype:param...]) Calls an external Dll.
DllFree(dllhandle) Frees a Dll that was loaded via the DllLoad function.
DllHinst(partial-winname) Obtains an application instance handle for use in DllCall's when required.
DllHwnd(partial-winname) Obtains a window handle for use in DllCall's when required.
DllLoad(dllname) Loads a Dll for later use via the DllCall function.
EnvironSet(env-varname, newvalue) Changes LOCAL Environment variables.
EnvItemize( ) Returns a delimited list of the current environment.
ExeTypeInfo(exefilename) Returns an integer describing the type of EXE file specified.
Exp(fp_num) Calculates the exponential.
Fabs(fp_num) Calculates the absolute value of a floating-point argument.
FileCompare(filename1, filename2) Compares two files.
FileFullName(partial filename) Returns a file name with drive and path information.
FileMapName(filename, mapping-data) Transforms a filename with a file wild-card mask and returns a new filename.
FileTimeCode(filename) Returns a machine readable/computable code for a file time.

FileTimeSet(list, ymdhms) Sets the date and time of one or more files.
FileYmdHms(filename) Returns a file time in the YmdHms date/time format.
Floor(fp_num) Calculates the floor of a value. Forvarname = initial value to final value [by increment] Controls the looping of a block of code base in an incrementing index.
GetExactTime( ) Returns the current time in hundredths of a second.
GetTickCount( ) Returns the number of clock ticks used by Windows since Windows started.
GoSub Transfers control of WIL processing while saving location of the next statement.
IconReplace(filename, iconfilename) Replaces an existing icon with a new icon.
If/ Else / Endif expression Conditionally performs a function.
Int(string/fp_num) Converts a floating point number or a string to an integer.
IsFloat(value) Tests whether a number can be converted to a floating point number.
IsInt(string) Tests whether a number can be converted into a valid number.
KeyToggleGet(@key) Returns the status of a toggle key.
KeyToggleSet(@key, value) Sets the state of a toggle key and returns the previous value.
Log10(fp_num) Calculates the base-10 logarithm.
LogE(fp_num) Calculates the natural logarithm.
MsgTextGet(window-name) Returns the contents of a Windows message box.
NetInfo(requestcode) Determines network(s) installed.
Num2Char(integer) Converts a number to its character equivalent.
ObjectClose(objecthandle) Closes OLE 2.0 automation object.
ObjectOpen(objectname) Opens or creates an OLE 2.0 automation object.
Print(data file, directory, display mode, waitflag) Instructs an application associated to a file to print the file on the default printer.
RegCloseKey(keyhandle) Closes a key to the registration database.
RegCreateKey(keyhandle, sub-key string) Returns a handle to a new registration database key.
RegDeleteKey(keyhandle, sub-key string) Deletes a key and data items associated with the key.
RegOpenKey(keyhandle, sub-key string) Returns a handle to an existing registration database key.
RegQueryKey(keyhandle, index) Returns sub keys of the specified key.
RegQueryValue(keyhandle, keyname) Returns data item string at sub-key position.
RegSetValue(keyhandle, sub-key string, value) Sets the value of a data item in the registration database.
RunEnviron(program-name, parameters, displaymode, waitflag) Launches a program in the current environment as set with the
EnvironSet command.
RunExit(program-name, parameters) Exits Windows, runs a DOS program or batch file then restarts windows when DOS is finished.
RunShell(program-name, params, directory, displaymode, waitflag) Runs a program via the Windows ShellExecute Command.
Selectvarname Allows selection among multiple blocks of statements.
SendKeysChild(partial-parent-winname, partial-child-winname, sendkey-string) Sends keystrokes to the active child window.
SendKeysTo(partial-parent-winname, sendkey-string) Sends keystrokes to a "windowname".
SendMenusTo(partial-parent-winname, menuname) Activates a window and sends a specified menu option.
Sin(fp_num) Calculates the sine.
Sinh(fp_num) Calculates the hyperbolic sine.
Sqrt(fp_num) Calculates the square root.
StrCharCount(string) Counts the number of characters in a string.
StrFixChars(base-string, pad-string, length) Pads or truncates a string To a fixed length using characters.
Switchvarname Allows selection among multiple blocks of statements.
Tan(fp_num) Calculates the tangent.
Tanh(fp_num) Calculates the hyperbolic tangent.
TimeAdd(YmdHms, YmdHms) Adds two YmdHms variables.
TimeDate( ) Provides the current date and time in a readable format.
TimeDelay(seconds) Pauses execution for a specified amount of time.

TimeDiffDays(Ymd[Hms], Ymd[Hms]) Returns the difference in days between the two dates.
TimeDiffSecs(YmdHms, YmdHms) Returns the time difference in seconds between the two date times.
TimeJulianDay(Ymd[Hms]) Returns the julian day given a date/time.
TimeWait(YmdHms) Pauses execution and waits for the date/time to pass.
TimeYmdHms( ) Returns current date/time in the date/time format.
While expression Conditionally and/or repeatedly executes a series of statements.
WinActivChild(partial-parent-winname, partial-child-winname) Activates a previously running child window. (Note misspelling)
WinExistChild(partial-parent-winname, partial-child-winname) Tells if a specified child window exists.
WinIsDos(partial-winname) Tells whether or not a particular window is a DOS or console-type window.
WinItemChild(partial-parent-winname) Returns a list of all the child windows under this parent.

Constants

New functions and, particularly, the floating point capability, mean more constants are in order. Here is a listing: See Constants for more details.

OS Dependent Constants

@CAPSLOCK
@NUMLOCK
@REGCLASSES
@REGCURRENT
@REGMACHINE
@REGROOT
@REGUSERS
@SCROLLLOCK
@WHOLESECTION
Miscellaneous
@MULTIPLE
@NOWAIT
@OPEN
@ROWS
@SAVE
@SINGLE
@SORTED
@UNSORTED
@WAIT

String Constants

@CRLF 0x13,0x10 cr,lf
@TAB 0x09 tab

Floating Point Constants

@AMC	Atomic Mass Constant	1.66043E-27
@AVOGADRO	Avogadro's Constant	6.02252E23
@BOLTZMANN	Boltzmann Entropy Constant	1.38054e-23
@DEG2RAD	Degrees to Radians Conversion Constant	0.017453292519943
@E		2.718281828459045
@ELECTRIC	Electric Field Constant	8.8541853e-12
@EULERS	Eulers's Constant	0.5772156649015338
@FARADAY	Faraday Constant	9.64870e4
@GFTSEC	Gravitational Acceleration feet/sec2	32.174
@GMTSEC	Gravitational Acceleration meters/sec2	9.80665
@GOLDENRATIO	Goldenratio	1.6180339887498948
@GRAVITATION	Gravity Constant	6.670e-11
@LIGHTMPS	Lightmps Light miles/sec	186272
@LIGHTMTPS	Lightmtps meters/sec	2.997925e8
@MAGFIELD	Magnetic Field Constant	1.256637
@PARSEC	Parsec in AU	206.265
@PI	Pi	3.141592653589793
@PLANCKERGS	Planck's Constant in Ergs	6.6252E-27
@PLANCKJOULES	Planck's Constant in joules	6.6256E-34
@RAD2DEG	Radians to Degrees Conversion Constant	57.29577951308232