NAME

Win32 - Interfaces to some Win32 API Functions

DESCRIPTION

The Win32 module contains functions to access Win32 APIs.

Alphabetical Listing of Win32 Functions

It is recommended to `use Win32;` before any of these functions; however, for backwards compatibility, those marked as [CORE] will automatically do this for you.

In the function descriptions below the term *Unicode string* is used to indicate that the string may contain characters outside the system codepage. The caveat *If supported by the core Perl version* generally means Perl 5.8.9 and later, though some Unicode pathname functionality may work on earlier versions.

Win32::AbortSystemShutdown(MACHINE)

Aborts a system shutdown (started by the InitiateSystemShutdown function) on the specified MACHINE.

Win32::BuildNumber()

[CORE] Returns the ActivePerl build number. This function is only available in the ActivePerl binary distribution.

Win32::CopyFile(FROM, TO, OVERWRITE)

[CORE] The Win32::CopyFile() function copies an existing file to a new file. All file information like creation time and file attributes will be copied to the new file. However it will **not** copy the security information. If the destination file already exists it will only be overwritten when the OVERWRITE parameter is true. But even this will not overwrite a read-only file; you have to unlink() it first yourself.

Win32::CreateDirectory(DIRECTORY)

Creates the DIRECTORY and returns a true value on success. Check $^E on failure for extended error information.

DIRECTORY may contain Unicode characters outside the system codepage. Once the directory has been created you can use Win32::GetANSIPathName() to get a name that can be passed to system calls and external programs.

Win32::CreateFile(FILE)

Creates the FILE and returns a true value on success. Check $^E on failure for extended error information.

FILE may contain Unicode characters outside the system codepage. Once the file has been created you can use Win32::GetANSIPathName() to get a name that can be passed to system calls and external programs.

Win32::DomainName()

[CORE] Returns the name of the Microsoft Network domain or workgroup that the owner of the current perl process is logged into. The "Workstation" service must be running to determine this information. This function does **not** work on Windows 9x.

Win32::ExpandEnvironmentStrings(STRING)

Takes STRING and replaces all referenced environment variable names with their defined values. References to environment variables take the form `%VariableName%`. Case is ignored when looking up the VariableName in the environment. If the variable is not found then the original `%VariableName%` text is retained. Has the same effect as the following:

```
$string =~ s/%([^%]*)%/$ENV{$1} || "%$1%"/eg
```

However, this function may return a Unicode string if the environment variable being expanded hasn't been assigned to via %ENV. Access to %ENV is currently always using byte semantics.

Win32::FormatMessage(ERRORCODE)

[CORE] Converts the supplied Win32 error number (e.g. returned by Win32::GetLastError()) to a descriptive string. Analogous to the perror() standard-C library function. Note that $^E used in a string context has much the same effect.

```
C:\> perl -e "$^E = 26; print $^E;"
The specified disk or diskette cannot be accessed
```

Win32::FsType()

[CORE] Returns the name of the filesystem of the currently active drive (like 'FAT' or 'NTFS'). In list context it returns three values: (FSTYPE, FLAGS, MAXCOMPLEN). FSTYPE is the filesystem type as before. FLAGS is a combination of values of the following table:

```
0x00000001  supports case-sensitive filenames
0x00000002  preserves the case of filenames
0x00000004  supports Unicode in filenames
0x00000008  preserves and enforces ACLs
0x00000010  supports file-based compression
0x00000020  supports disk quotas
0x00000040  supports sparse files
0x00000080  supports reparse points
0x00000100  supports remote storage
0x00008000  is a compressed volume (e.g. DoubleSpace)
0x00010000  supports object identifiers
0x00020000  supports the Encrypted File System (EFS)
```

MAXCOMPLEN is the maximum length of a filename component (the part between two backslashes) on this file system.

Win32::FreeLibrary(HANDLE)

Unloads a previously loaded dynamic-link library. The HANDLE is no longer valid after this call. See *LoadLibrary* for information on dynamically loading a library.

Win32::GetANSIPathName(FILENAME)

Returns an ANSI version of FILENAME. This may be the short name if the long name cannot be represented in the system codepage.

While not currently implemented, it is possible that in the future this function will convert only parts of the path to FILENAME to a short form.

If FILENAME doesn't exist on the filesystem, or if the filesystem doesn't support short ANSI filenames, then this function will translate the Unicode name into the system codepage using replacement characters.

Win32::GetArchName()

Use of this function is deprecated. It is equivalent with $ENV{PROCESSOR_ARCHITECTURE}. This might not work on Win9X.

Win32::GetChipName()

Returns the processor type: 386, 486 or 586 for Intel processors, 21064 for the Alpha chip.

Win32::GetCwd()

[CORE] Returns the current active drive and directory. This function does not return a UNC path, since the functionality required for such a feature is not available under Windows 95.

If supported by the core Perl version, this function will return an ANSI path name for the current directory if the long pathname cannot be represented in the system codepage.

Win32::GetCurrentThreadId()

Returns the thread identifier of the calling thread. Until the thread terminates, the thread identifier uniquely identifies the thread throughout the system.

Note: the current process identifier is available via the predefined $$ variable.

Win32::GetFileVersion(FILENAME)

Returns the file version number from the VERSIONINFO resource of the executable file or DLL. This is a tuple of four 16 bit numbers. In list context these four numbers will be returned. In scalar context they are concatenated into a string, separated by dots.

Win32::GetFolderPath(FOLDER [, CREATE])

Returns the full pathname of one of the Windows special folders. The folder will be created if it doesn't exist and the optional CREATE argument is true. The following FOLDER constants are defined by the Win32 module, but only exported on demand:

```
CSIDL_ADMINTOOLS
CSIDL_APPDATA
CSIDL_CDBURN_AREA
CSIDL_COMMON_ADMINTOOLS
CSIDL_COMMON_APPDATA
CSIDL_COMMON_DESKTOPDIRECTORY
CSIDL_COMMON_DOCUMENTS
CSIDL_COMMON_FAVORITES
CSIDL_COMMON_MUSIC
CSIDL_COMMON_PICTURES
CSIDL_COMMON_PROGRAMS
CSIDL_COMMON_STARTMENU
CSIDL_COMMON_STARTUP
CSIDL_COMMON_TEMPLATES
CSIDL_COMMON_VIDEO
CSIDL_COOKIES
CSIDL_DESKTOP
CSIDL_DESKTOPDIRECTORY
CSIDL_FAVORITES
CSIDL_FONTS
CSIDL_HISTORY
CSIDL_INTERNET_CACHE
CSIDL_LOCAL_APPDATA
CSIDL_MYMUSIC
CSIDL_MYPICTURES
CSIDL_MYVIDEO
CSIDL_NETHOOD
CSIDL_PERSONAL
CSIDL_PRINTHOOD
CSIDL_PROFILE
CSIDL_PROGRAMS
CSIDL_PROGRAM_FILES
CSIDL_PROGRAM_FILES_COMMON
CSIDL_RECENT
CSIDL_RESOURCES
CSIDL_RESOURCES_LOCALIZED
CSIDL_SENDTO
CSIDL_STARTMENU
```

```
CSIDL_STARTUP
CSIDL_SYSTEM
CSIDL_TEMPLATES
CSIDL_WINDOWS
```

Note that not all folders are defined on all versions of Windows.

Please refer to the MSDN documentation of the CSIDL constants, currently available at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/enums/csidl.asp

This function will return an ANSI folder path if the long name cannot be represented in the system codepage. Use Win32::GetLongPathName() on the result of Win32::GetFolderPath() if you want the Unicode version of the folder name.

Win32::GetFullPathName(FILENAME)

[CORE] GetFullPathName combines the FILENAME with the current drive and directory name and returns a fully qualified (aka, absolute) path name. In list context it returns two elements: (PATH, FILE) where PATH is the complete pathname component (including trailing backslash) and FILE is just the filename part. Note that no attempt is made to convert 8.3 components in the supplied FILENAME to longnames or vice-versa. Compare with Win32::GetShortPathName() and Win32::GetLongPathName().

If supported by the core Perl version, this function will return an ANSI path name if the full pathname cannot be represented in the system codepage.

Win32::GetLastError()

[CORE] Returns the last error value generated by a call to a Win32 API function. Note that $^E used in a numeric context amounts to the same value.

Win32::GetLongPathName(PATHNAME)

[CORE] Returns a representation of PATHNAME composed of longname components (if any). The result may not necessarily be longer than PATHNAME. No attempt is made to convert PATHNAME to the absolute path. Compare with Win32::GetShortPathName() and Win32::GetFullPathName().

This function may return the pathname in Unicode if it cannot be represented in the system codepage. Use Win32::GetANSIPathName() before passing the path to a system call or another program.

Win32::GetNextAvailDrive()

[CORE] Returns a string in the form of "<d>:" where <d> is the first available drive letter.

Win32::GetOSVersion()

[CORE] Returns the list (STRING, MAJOR, MINOR, BUILD, ID), where the elements are, respectively: An arbitrary descriptive string, the major version number of the operating system, the minor version number, the build number, and a digit indicating the actual operating system. For the ID, the values are 0 for Win32s, 1 for Windows 9X/Me and 2 for Windows NT/2000/XP/2003/Vista. In scalar context it returns just the ID.

Currently known values for ID MAJOR and MINOR are as follows:

OS	ID	MAJOR	MINOR
Win32s	0	-	-
Windows 95	1	4	0
Windows 98	1	4	10
Windows Me	1	4	90
Windows NT 3.51	2	3	51
Windows NT 4	2	4	0
Windows 2000	2	5	0

```
Windows XP              2       5       1
Windows Server 2003     2       5       2
Windows Vista           2       6       0
```

On Windows NT 4 SP6 and later this function returns the following additional values: SPMAJOR, SPMINOR, SUITEMASK, PRODUCTTYPE.

SPMAJOR and SPMINOR are are the version numbers of the latest installed service pack.

SUITEMASK is a bitfield identifying the product suites available on the system. Known bits are:

```
VER_SUITE_SMALLBUSINESS             0x00000001
VER_SUITE_ENTERPRISE                0x00000002
VER_SUITE_BACKOFFICE                0x00000004
VER_SUITE_COMMUNICATIONS            0x00000008
VER_SUITE_TERMINAL                  0x00000010
VER_SUITE_SMALLBUSINESS_RESTRICTED  0x00000020
VER_SUITE_EMBEDDEDNT                0x00000040
VER_SUITE_DATACENTER                0x00000080
VER_SUITE_SINGLEUSERTS              0x00000100
VER_SUITE_PERSONAL                  0x00000200
VER_SUITE_BLADE                     0x00000400
VER_SUITE_EMBEDDED_RESTRICTED       0x00000800
VER_SUITE_SECURITY_APPLIANCE        0x00001000
```

The VER_SUITE_xxx names are listed here to crossreference the Microsoft documentation. The Win32 module does not provide symbolic names for these constants.

PRODUCTTYPE provides additional information about the system. It should be one of the following integer values:

```
1 - Workstation (NT 4, 2000 Pro, XP Home, XP Pro, Vista)
2 - Domaincontroller
3 - Server
```

Win32::GetOSName()

In scalar context returns the name of the Win32 operating system being used. In list context returns a two element list of the OS name and whatever edition information is known about the particular build (for Win9X boxes) and whatever service packs have been installed. The latter is roughly equivalent to the first item returned by GetOSVersion() in list context.

Currently the possible values for the OS name are

```
Win32s Win95 Win98 WinMe WinNT3.51 WinNT4 Win2000 WinXP/.Net Win2003
WinVista
```

This routine is just a simple interface into GetOSVersion(). More specific or demanding situations should use that instead. Another option would be to use POSIX::uname(), however the latter appears to report only the OS family name and not the specific OS. In scalar context it returns just the ID.

The name "WinXP/.Net" is used for historical reasons only, to maintain backwards compatibility of the Win32 module. Windows .NET Server has been renamed as Windows 2003 Server before final release and uses a different major/minor version number than Windows XP.

Win32::GetShortPathName(PATHNAME)

[CORE] Returns a representation of PATHNAME that is composed of short (8.3) path components where available. For path components where the file system has not generated the short form the returned path will use the long form, so this function might still for instance

return a path containing spaces. Returns `undef` when the PATHNAME does not exist. Compare with Win32::GetFullPathName() and Win32::GetLongPathName().

Win32::GetProcAddress(INSTANCE, PROCNAME)

Returns the address of a function inside a loaded library. The information about what you can do with this address has been lost in the mist of time. Use the Win32::API module instead of this deprecated function.

Win32::GetTickCount()

[CORE] Returns the number of milliseconds elapsed since the last system boot. Resolution is limited to system timer ticks (about 10ms on WinNT and 55ms on Win9X).

Win32::GuidGen()

Creates a globally unique 128 bit integer that can be used as a persistent identifier in a distributed setting. To a very high degree of certainty this function returns a unique value. No other invocation, on the same or any other system (networked or not), should return the same value.

The return value is formatted according to OLE conventions, as groups of hex digits with surrounding braces. For example:

```
{09531CF1-D0C7-4860-840C-1C8C8735E2AD}
```

Win32::InitiateSystemShutdown

(MACHINE, MESSAGE, TIMEOUT, FORCECLOSE, REBOOT)

Shutsdown the specified MACHINE, notifying users with the supplied MESSAGE, within the specified TIMEOUT interval. Forces closing of all documents without prompting the user if FORCECLOSE is true, and reboots the machine if REBOOT is true. This function works only on WinNT.

Win32::IsAdminUser()

Returns non zero if the account in whose security context the current process/thread is running belongs to the local group of Administrators in the built-in system domain; returns 0 if not. On Windows Vista it will only return non-zero if the process is actually running with elevated privileges. Returns `undef` and prints a warning if an error occurred. This function always returns 1 on Win9X.

Win32::IsWinNT()

[CORE] Returns non zero if the Win32 subsystem is Windows NT.

Win32::IsWin95()

[CORE] Returns non zero if the Win32 subsystem is Windows 95.

Win32::LoadLibrary(LIBNAME)

Loads a dynamic link library into memory and returns its module handle. This handle can be used with Win32::GetProcAddress() and Win32::FreeLibrary(). This function is deprecated. Use the Win32::API module instead.

Win32::LoginName()

[CORE] Returns the username of the owner of the current perl process. The return value may be a Unicode string.

Win32::LookupAccountName(SYSTEM, ACCOUNT, DOMAIN, SID, SIDTYPE)

Looks up ACCOUNT on SYSTEM and returns the domain name the SID and the SID type.

Win32::LookupAccountSID(SYSTEM, SID, ACCOUNT, DOMAIN, SIDTYPE)

Looks up SID on SYSTEM and returns the account name, domain name, and the SID type.

Win32::MsgBox(MESSAGE [, FLAGS [, TITLE]])

Create a dialogbox containing MESSAGE. FLAGS specifies the required icon and buttons according to the following table:

```
0 = OK
1 = OK and Cancel
2 = Abort, Retry, and Ignore
3 = Yes, No and Cancel
4 = Yes and No
5 = Retry and Cancel

MB_ICONSTOP          "X" in a red circle
MB_ICONQUESTION      question mark in a bubble
MB_ICONEXCLAMATION   exclamation mark in a yellow triangle
MB_ICONINFORMATION   "i" in a bubble
```

TITLE specifies an optional window title. The default is "Perl".

The function returns the menu id of the selected push button:

```
0  Error

1  OK
2  Cancel
3  Abort
4  Retry
5  Ignore
6  Yes
7  No
```

Win32::NodeName()

[CORE] Returns the Microsoft Network node-name of the current machine.

Win32::OutputDebugString(STRING)

Sends a string to the application or system debugger for display. The function does nothing if there is no active debugger.

Alternatively one can use the *Debug Viewer* application to watch the OutputDebugString() output:

http://www.microsoft.com/technet/sysinternals/utilities/debugview.mspx

Win32::RegisterServer(LIBRARYNAME)

Loads the DLL LIBRARYNAME and calls the function DllRegisterServer.

Win32::SetChildShowWindow(SHOWWINDOW)

[CORE] Sets the *ShowMode* of child processes started by system(). By default system() will create a new console window for child processes if Perl itself is not running from a console. Calling SetChildShowWindow(0) will make these new console windows invisible. Calling SetChildShowWindow() without arguments reverts system() to the default behavior. The return value of SetChildShowWindow() is the previous setting or `undef`.

The following symbolic constants for SHOWWINDOW are available (but not exported) from the Win32 module: SW_HIDE, SW_SHOWNORMAL, SW_SHOWMINIMIZED, SW_SHOWMAXIMIZED and SW_SHOWNOACTIVATE.

Win32::SetCwd(NEWDIRECTORY)

[CORE] Sets the current active drive and directory. This function does not work with UNC paths, since the functionality required to required for such a feature is not available under Windows 95.

Win32::SetLastError(ERROR)

[CORE] Sets the value of the last error encountered to ERROR. This is that value that will be returned by the Win32::GetLastError() function.

Win32::Sleep(TIME)

[CORE] Pauses for TIME milliseconds. The timeslices are made available to other processes and threads.

Win32::Spawn(COMMAND, ARGS, PID)

[CORE] Spawns a new process using the supplied COMMAND, passing in arguments in the string ARGS. The pid of the new process is stored in PID. This function is deprecated. Please use the Win32::Process module instead.

Win32::UnregisterServer(LIBRARYNAME)

Loads the DLL LIBRARYNAME and calls the function DllUnregisterServer.