Windows Data Types

The data types supported by Windows are used to define function return values, function and message parameters, and structure members. They define the size and meaning of these elements. For more information about the underlying C/C++ data types, see Data Type Ranges.

The following table contains the following types: character, integer, Boolean, pointer, and handle. The character, integer, and Boolean types are common to most C compilers. Most of the pointer-type names begin with a prefix of P or LP. Handles refer to a resource that has been loaded into memory.

For more information about handling 64-bit integers, see Large Integers.

Data type	Description
APIENTRY	The calling convention for system functions. This type is declared in WinDef.h as follows: `#define APIENTRY WINAPI`
ATOM	An atom. For more information, see About Atom Tables. This type is declared in WinDef.h as follows: `typedef WORD ATOM;`
BOOL	A Boolean variable (should be **TRUE** or **FALSE**). This type is declared in WinDef.h as follows: `typedef int BOOL;`
BOOLEAN	A Boolean variable (should be **TRUE** or **FALSE**). This type is declared in WinNT.h as follows: `typedef BYTE BOOLEAN;`
BYTE	A byte (8 bits). This type is declared in WinDef.h as follows: `typedef unsigned char BYTE;`

CALLBACK	The calling convention for callback functions. This type is declared in WinDef.h as follows: `#define CALLBACK __stdcall` **CALLBACK**, **WINAPI**, and **APIENTRY** are all used to define functions with the __stdcall calling convention. Most functions in the Windows API are declared using **WINAPI**. You may wish to use **CALLBACK** for the callback functions that you implement to help identify the function as a callback function.
CCHAR	An 8-bit Windows (ANSI) character. This type is declared in WinNT.h as follows: `typedef char CCHAR;`
CHAR	An 8-bit Windows (ANSI) character. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef char CHAR;`
COLORREF	The red, green, blue (RGB) color value (32 bits). See COLORREF for information on this type. This type is declared in WinDef.h as follows: `typedef DWORD COLORREF;`
CONST	A variable whose value is to remain constant during execution. This type is declared in WinDef.h as follows: `#define CONST const`
DWORD	A 32-bit unsigned integer. The range is 0 through 4294967295 decimal. This type is declared in IntSafe.h as follows: `typedef unsigned long DWORD;`
DWORDLONG	A 64-bit unsigned integer. The range is 0 through 18446744073709551615 decimal. This type is declared in IntSafe.h as follows: `typedef unsigned __int64 DWORDLONG;`

DWORD_PTR	An unsigned long type for pointer precision. Use when casting a pointer to a long type to perform pointer arithmetic. (Also commonly used for general 32-bit parameters that have been extended to 64 bits in 64-bit Windows.) This type is declared in BaseTsd.h as follows: `typedef ULONG_PTR DWORD_PTR;`
DWORD32	A 32-bit unsigned integer. This type is declared in BaseTsd.h as follows: `typedef unsigned int DWORD32;`
DWORD64	A 64-bit unsigned integer. This type is declared in BaseTsd.h as follows: `typedef unsigned __int64 DWORD64;`
FLOAT	A floating-point variable. This type is declared in WinDef.h as follows: `typedef float FLOAT;`
HACCEL	A handle to an accelerator table. This type is declared in WinDef.h as follows: `typedef HANDLE HACCEL;`

HALF_PTR	Half the size of a pointer. Use within a structure that contains a pointer and two small fields.
	
	This type is declared in BaseTsd.h as follows:
	
	C++
	
	`#ifdef _WIN64`
	` typedef int HALF_PTR;`
	`#else`
	` typedef short HALF_PTR;`
	`#endif`

HANDLE	A handle to an object. This type is declared in WinNT.h as follows: `typedef PVOID HANDLE;`
HBITMAP	A handle to a bitmap. This type is declared in WinDef.h as follows: `typedef HANDLE HBITMAP;`
HBRUSH	A handle to a brush. This type is declared in WinDef.h as follows: `typedef HANDLE HBRUSH;`
HCOLORSPACE	A handle to a color space. This type is declared in WinDef.h as follows: `typedef HANDLE HCOLORSPACE;`
HCONV	A handle to a dynamic data exchange (DDE) conversation. This type is declared in Ddeml.h as follows: `typedef HANDLE HCONV;`
HCONVLIST	A handle to a DDE conversation list. This type is declared in Ddeml.h as follows: `typedef HANDLE HCONVLIST;`
HCURSOR	A handle to a cursor. This type is declared in WinDef.h as follows: `typedef HICON HCURSOR;`
HDC	A handle to a device context (DC). This type is declared in WinDef.h as follows: `typedef HANDLE HDC;`

HDDEDATA	A handle to DDE data. This type is declared in Ddeml.h as follows: `typedef HANDLE HDDEDATA;`
HDESK	A handle to a desktop. This type is declared in WinDef.h as follows: `typedef HANDLE HDESK;`
HDROP	A handle to an internal drop structure. This type is declared in ShellApi.h as follows: `typedef HANDLE HDROP;`
HDWP	A handle to a deferred window position structure. This type is declared in WinUser.h as follows: `typedef HANDLE HDWP;`
HENHMETAFILE	A handle to an enhanced metafile. This type is declared in WinDef.h as follows: `typedef HANDLE HENHMETAFILE;`
HFILE	A handle to a file opened by **OpenFile**, not **CreateFile**. This type is declared in WinDef.h as follows: `typedef int HFILE;`
HFONT	A handle to a font. This type is declared in WinDef.h as follows: `typedef HANDLE HFONT;`
HGDIOBJ	A handle to a GDI object. This type is declared in WinDef.h as follows: `typedef HANDLE HGDIOBJ;`

HGLOBAL	A handle to a global memory block. This type is declared in WinDef.h as follows: `typedef HANDLE HGLOBAL;`
HHOOK	A handle to a hook. This type is declared in WinDef.h as follows: `typedef HANDLE HHOOK;`
HICON	A handle to an icon. This type is declared in WinDef.h as follows: `typedef HANDLE HICON;`
HINSTANCE	A handle to an instance. This is the base address of the module in memory. **HMODULE** and **HINSTANCE** are the same today, but represented different things in 16-bit Windows. This type is declared in WinDef.h as follows: `typedef HANDLE HINSTANCE;`
HKEY	A handle to a registry key. This type is declared in WinDef.h as follows: `typedef HANDLE HKEY;`
HKL	An input locale identifier. This type is declared in WinDef.h as follows: `typedef HANDLE HKL;`
HLOCAL	A handle to a local memory block. This type is declared in WinDef.h as follows: `typedef HANDLE HLOCAL;`
HMENU	A handle to a menu. This type is declared in WinDef.h as follows:

	`typedef HANDLE HMENU;`
HMETAFILE	A handle to a metafile. This type is declared in WinDef.h as follows: `typedef HANDLE HMETAFILE;`
HMODULE	A handle to a module. The is the base address of the module in memory. **HMODULE** and **HINSTANCE** are the same in current versions of Windows, but represented different things in 16-bit Windows. This type is declared in WinDef.h as follows: `typedef HINSTANCE HMODULE;`
HMONITOR	A handle to a display monitor. This type is declared in WinDef.h as follows: `if(WINVER >= 0x0500) typedef HANDLE HMONITOR;`
HPALETTE	A handle to a palette. This type is declared in WinDef.h as follows: `typedef HANDLE HPALETTE;`
HPEN	A handle to a pen. This type is declared in WinDef.h as follows: `typedef HANDLE HPEN;`
HRESULT	The return codes used by COM interfaces. For more information, see Structure of the COM Error Codes. To test an **HRESULT** value, use the FAILED and SUCCEEDED macros. This type is declared in WinNT.h as follows: `typedef LONG HRESULT;`
HRGN	A handle to a region. This type is declared in WinDef.h as follows: `typedef HANDLE HRGN;`

HRSRC	A handle to a resource. This type is declared in WinDef.h as follows: `typedef HANDLE HRSRC;`
HSZ	A handle to a DDE string. This type is declared in Ddeml.h as follows: `typedef HANDLE HSZ;`
HWINSTA	A handle to a window station. This type is declared in WinDef.h as follows: `typedef HANDLE WINSTA;`
HWND	A handle to a window. This type is declared in WinDef.h as follows: `typedef HANDLE HWND;`
INT	A 32-bit signed integer. The range is -2147483648 through 2147483647 decimal. This type is declared in WinDef.h as follows: `typedef int INT;`

INT_PTR	A signed integer type for pointer precision. Use when casting a pointer to an integer to perform pointer arithmetic.
	
	This type is declared in BaseTsd.h as follows:
	
	C++
	
	`#if defined(_WIN64)`
	` typedef __int64 INT_PTR;`
	`#else`
	` typedef int INT_PTR;`
	`#endif`

INT8	An 8-bit signed integer.

	This type is declared in BaseTsd.h as follows: `typedef signed char INT8;`
INT16	A 16-bit signed integer. This type is declared in BaseTsd.h as follows: `typedef signed short INT16;`
INT32	A 32-bit signed integer. The range is -2147483648 through 2147483647 decimal. This type is declared in BaseTsd.h as follows: `typedef signed int INT32;`
INT64	A 64-bit signed integer. The range is –9223372036854775808 through 9223372036854775807 decimal. This type is declared in BaseTsd.h as follows: `typedef signed __int64 INT64;`
LANGID	A language identifier. For more information, see Language Identifiers. This type is declared in WinNT.h as follows: `typedef WORD LANGID;`
LCID	A locale identifier. For more information, see Locale Identifiers. This type is declared in WinNT.h as follows: `typedef DWORD LCID;`
LCTYPE	A locale information type. For a list, see Locale Information Constants. This type is declared in WinNls.h as follows: `typedef DWORD LCTYPE;`
LGRPID	A language group identifier. For a list, see EnumLanguageGroupLocales. This type is declared in WinNls.h as follows: `typedef DWORD LGRPID;`
LONG	A 32-bit signed integer. The range is –2147483648 through 2147483647 decimal.

	This type is declared in WinNT.h as follows: `typedef long LONG;`
LONGLONG	A 64-bit signed integer. The range is –9223372036854775808 through 9223372036854775807 decimal. This type is declared in WinNT.h as follows: C++ `#if !defined(_M_IX86)` ` typedef __int64 LONGLONG;` `#else` ` typedef double LONGLONG;` `#endif`
LONG_PTR	A signed long type for pointer precision. Use when casting a pointer to a long to perform pointer arithmetic. This type is declared in BaseTsd.h as follows: C++ `#if defined(_WIN64)` ` typedef __int64 LONG_PTR;` `#else` ` typedef long LONG_PTR;` `#endif`
LONG32	A 32-bit signed integer. The range is –2147483648 through 2147483647 decimal. This type is declared in BaseTsd.h as follows: `typedef signed int LONG32;`
LONG64	A 64-bit signed integer. The range is –9223372036854775808 through 9223372036854775807 decimal. This type is declared in BaseTsd.h as follows:

	`typedef __int64 LONG64;`
LPARAM	A message parameter. This type is declared in WinDef.h as follows: `typedef LONG_PTR LPARAM;`
LPBOOL	A pointer to a BOOL. This type is declared in WinDef.h as follows: `typedef BOOL far *LPBOOL;`
LPBYTE	A pointer to a BYTE. This type is declared in WinDef.h as follows: `typedef BYTE far *LPBYTE;`
LPCOLORREF	A pointer to a COLORREF value. This type is declared in WinDef.h as follows: `typedef DWORD *LPCOLORREF;`
LPCSTR	A pointer to a constant null-terminated string of 8-bit Windows (ANSI) characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef __nullterminated CONST CHAR *LPCSTR;`

LPCTSTR	An LPCWSTR if **UNICODE** is defined, an LPCSTR otherwise. For more information, see Windows Data Types for Strings.
	
	This type is declared in WinNT.h as follows:
	
	C++
	`#ifdef UNICODE`
	` typedef LPCWSTR LPCTSTR;`
	`#else`
	` typedef LPCSTR LPCTSTR;`
	`#endif`

LPCVOID	A pointer to a constant of any type. This type is declared in WinDef.h as follows: `typedef CONST void *LPCVOID;`
LPCWSTR	A pointer to a constant null-terminated string of 16-bit Unicode characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef CONST WCHAR *LPCWSTR;`
LPDWORD	A pointer to a DWORD. This type is declared in WinDef.h as follows: `typedef DWORD *LPDWORD;`
LPHANDLE	A pointer to a HANDLE. This type is declared in WinDef.h as follows: `typedef HANDLE *LPHANDLE;`
LPINT	A pointer to an INT. This type is declared in WinDef.h as follows: `typedef int *LPINT;`
LPLONG	A pointer to a LONG. This type is declared in WinDef.h as follows: `typedef long *LPLONG;`
LPSTR	A pointer to a null-terminated string of 8-bit Windows (ANSI) characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef CHAR *LPSTR;`
LPTSTR	An LPWSTR if **UNICODE** is defined, an LPSTR otherwise. For more information, see Windows Data Types for Strings.

	This type is declared in WinNT.h as follows: C++
	```
	#ifdef UNICODE
	 typedef LPWSTR LPTSTR;
	#else
	 typedef LPSTR LPTSTR;
	#endif
	```

LPVOID	A pointer to any type. This type is declared in WinDef.h as follows: `typedef void *LPVOID;`
LPWORD	A pointer to a WORD. This type is declared in WinDef.h as follows: `typedef WORD *LPWORD;`
LPWSTR	A pointer to a null-terminated string of 16-bit Unicode characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef WCHAR *LPWSTR;`
LRESULT	Signed result of message processing. This type is declared in WinDef.h as follows: `typedef LONG_PTR LRESULT;`
PBOOL	A pointer to a BOOL. This type is declared in WinDef.h as follows: `typedef BOOL *PBOOL;`
PBOOLEAN	A pointer to a BOOLEAN.

	This type is declared in WinNT.h as follows: `typedef BOOLEAN *PBOOLEAN;`
PBYTE	A pointer to a BYTE. This type is declared in WinDef.h as follows: `typedef BYTE *PBYTE;`
PCHAR	A pointer to a CHAR. This type is declared in WinNT.h as follows: `typedef CHAR *PCHAR;`
PCSTR	A pointer to a constant null-terminated string of 8-bit Windows (ANSI) characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef CONST CHAR *PCSTR;`
PCTSTR	A PCWSTR if **UNICODE** is defined, a PCSTR otherwise. For more information, see Windows Data Types for Strings. This type is declared in WinNT.h as follows: C++ `#ifdef UNICODE` `typedef LPCWSTR PCTSTR;` `#else` `typedef LPCSTR PCTSTR;` `#endif`
PCWSTR	A pointer to a constant null-terminated string of 16-bit Unicode characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef CONST WCHAR *PCWSTR;`
PDWORD	A pointer to a DWORD.

	This type is declared in WinDef.h as follows: `typedef DWORD *PDWORD;`
PDWORDLONG	A pointer to a DWORDLONG. This type is declared in WinNT.h as follows: `typedef DWORDLONG *PDWORDLONG;`
PDWORD_PTR	A pointer to a DWORD_PTR. This type is declared in BaseTsd.h as follows: `typedef DWORD_PTR *PDWORD_PTR;`
PDWORD32	A pointer to a DWORD32. This type is declared in BaseTsd.h as follows: `typedef DWORD32 *PDWORD32;`
PDWORD64	A pointer to a DWORD64. This type is declared in BaseTsd.h as follows: `typedef DWORD64 *PDWORD64;`
PFLOAT	A pointer to a FLOAT. This type is declared in WinDef.h as follows: `typedef FLOAT *PFLOAT;`
PHALF_PTR	A pointer to a HALF_PTR. This type is declared in BaseTsd.h as follows: C++ `#ifdef _WIN64` `typedef HALF_PTR *PHALF_PTR;` `#else` `typedef HALF_PTR *PHALF_PTR;` `#endif`

PHANDLE	A pointer to a HANDLE. This type is declared in WinNT.h as follows: `typedef HANDLE *PHANDLE;`
PHKEY	A pointer to an HKEY. This type is declared in WinDef.h as follows: `typedef HKEY *PHKEY;`
PINT	A pointer to an INT. This type is declared in WinDef.h as follows: `typedef int *PINT;`
PINT_PTR	A pointer to an INT_PTR. This type is declared in BaseTsd.h as follows: `typedef INT_PTR *PINT_PTR;`
PINT8	A pointer to an INT8. This type is declared in BaseTsd.h as follows: `typedef INT8 *PINT8;`
PINT16	A pointer to an INT16. This type is declared in BaseTsd.h as follows: `typedef INT16 *PINT16;`
PINT32	A pointer to an INT32. This type is declared in BaseTsd.h as follows: `typedef INT32 *PINT32;`
PINT64	A pointer to an INT64. This type is declared in BaseTsd.h as follows: `typedef INT64 *PINT64;`

PLCID	A pointer to an LCID. This type is declared in WinNT.h as follows: `typedef PDWORD PLCID;`
PLONG	A pointer to a LONG. This type is declared in WinNT.h as follows: `typedef LONG *PLONG;`
PLONGLONG	A pointer to a LONGLONG. This type is declared in WinNT.h as follows: `typedef LONGLONG *PLONGLONG;`
PLONG_PTR	A pointer to a LONG_PTR. This type is declared in BaseTsd.h as follows: `typedef LONG_PTR *PLONG_PTR;`
PLONG32	A pointer to a LONG32. This type is declared in BaseTsd.h as follows: `typedef LONG32 *PLONG32;`
PLONG64	A pointer to a LONG64. This type is declared in BaseTsd.h as follows: `typedef LONG64 *PLONG64;`

POINTER_32	A 32-bit pointer. On a 32-bit system, this is a native pointer. On a 64-bit system, this is a truncated 64-bit pointer.
	
	This type is declared in BaseTsd.h as follows:
	
	C++
	
	`#if defined(_WIN64)`
	` #define POINTER_32 __ptr32`
	`#else`

	` #define POINTER_32`
	`#endif`

POINTER_64	A 64-bit pointer. On a 64-bit system, this is a native pointer. On a 32-bit system, this is a sign-extended 32-bit pointer.
	
	Note that it is not safe to assume the state of the high pointer bit.
	
	This type is declared in BaseTsd.h as follows:
	
	C++
	
	`#if (_MSC_VER >= 1300)`
	` #define POINTER_64 __ptr64`
	`#else`
	` #define POINTER_64`
	`#endif`

POINTER_SIGNED	A signed pointer. This type is declared in BaseTsd.h as follows: `#define POINTER_SIGNED __sptr`
POINTER_UNSIGNED	An unsigned pointer. This type is declared in BaseTsd.h as follows: `#define POINTER_UNSIGNED __uptr`
PSHORT	A pointer to a SHORT. This type is declared in WinNT.h as follows: `typedef SHORT *PSHORT;`
PSIZE_T	A pointer to a SIZE_T. This type is declared in BaseTsd.h as follows: `typedef SIZE_T *PSIZE_T;`

PSSIZE_T	A pointer to a SSIZE_T. This type is declared in BaseTsd.h as follows: `typedef SSIZE_T *PSSIZE_T;`
PSTR	A pointer to a null-terminated string of 8-bit Windows (ANSI) characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef CHAR *PSTR;`
PTBYTE	A pointer to a TBYTE. This type is declared in WinNT.h as follows: `typedef TBYTE *PTBYTE;`
PTCHAR	A pointer to a TCHAR. This type is declared in WinNT.h as follows: `typedef TCHAR *PTCHAR;`

PTSTR	A PWSTR if **UNICODE** is defined, a PSTR otherwise. For more information, see Windows Data Types for Strings.
	
	This type is declared in WinNT.h as follows:
	
	C++
	
	`#ifdef UNICODE`
	` typedef LPWSTR PTSTR;`
	`#else typedef LPSTR PTSTR;`
	`#endif`

PUCHAR	A pointer to a UCHAR. This type is declared in WinDef.h as follows: `typedef UCHAR *PUCHAR;`
PUHALF_PT	A pointer to a UHALF_PTR.

R	This type is declared in BaseTsd.h as follows:
	
	C++
	`#ifdef _WIN64`
	` typedef UHALF_PTR *PUHALF_PTR;`
	`#else`
	` typedef UHALF_PTR *PUHALF_PTR;`
	`#endif`

PUINT	A pointer to a UINT. This type is declared in WinDef.h as follows: `typedef UINT *PUINT;`
PUINT_PTR	A pointer to a UINT_PTR. This type is declared in BaseTsd.h as follows: `typedef UINT_PTR *PUINT_PTR;`
PUINT8	A pointer to a UINT8. This type is declared in BaseTsd.h as follows: `typedef UINT8 *PUINT8;`
PUINT16	A pointer to a UINT16. This type is declared in BaseTsd.h as follows: `typedef UINT16 *PUINT16;`
PUINT32	A pointer to a UINT32. This type is declared in BaseTsd.h as follows: `typedef UINT32 *PUINT32;`
PUINT64	A pointer to a UINT64. This type is declared in BaseTsd.h as follows:

	`typedef UINT64 *PUINT64;`
PULONG	A pointer to a ULONG. This type is declared in WinDef.h as follows: `typedef ULONG *PULONG;`
PULONGLONG	A pointer to a ULONGLONG. This type is declared in WinDef.h as follows: `typedef ULONGLONG *PULONGLONG;`
PULONG_PTR	A pointer to a ULONG_PTR. This type is declared in BaseTsd.h as follows: `typedef ULONG_PTR *PULONG_PTR;`
PULONG32	A pointer to a ULONG32. This type is declared in BaseTsd.h as follows: `typedef ULONG32 *PULONG32;`
PULONG64	A pointer to a ULONG64. This type is declared in BaseTsd.h as follows: `typedef ULONG64 *PULONG64;`
PUSHORT	A pointer to a USHORT. This type is declared in WinDef.h as follows: `typedef USHORT *PUSHORT;`
PVOID	A pointer to any type. This type is declared in WinNT.h as follows: `typedef void *PVOID;`
PWCHAR	A pointer to a WCHAR. This type is declared in WinNT.h as follows:

	`typedef WCHAR *PWCHAR;`
PWORD	A pointer to a WORD. This type is declared in WinDef.h as follows: `typedef WORD *PWORD;`
PWSTR	A pointer to a null-terminated string of 16-bit Unicode characters. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef WCHAR *PWSTR;`
QWORD	A 64-bit unsigned integer. This type is declared as follows: `typedef unsigned __int64 QWORD;`
SC_HANDLE	A handle to a service control manager database. For more information, see SCM Handles. This type is declared in WinSvc.h as follows: `typedef HANDLE SC_HANDLE;`
SC_LOCK	A lock to a service control manager database. For more information, see SCM Handles. This type is declared in WinSvc.h as follows: `typedef LPVOID SC_LOCK;`
SERVICE_STATUS_HANDLE	A handle to a service status value. For more information, see SCM Handles. This type is declared in WinSvc.h as follows: `typedef HANDLE SERVICE_STATUS_HANDLE;`
SHORT	A 16-bit integer. The range is –32768 through 32767 decimal. This type is declared in WinNT.h as follows: `typedef short SHORT;`
SIZE_T	The maximum number of bytes to which a pointer can point. Use for a count that must span the full range of a pointer.

	This type is declared in BaseTsd.h as follows: `typedef ULONG_PTR SIZE_T;`
SSIZE_T	A signed version of **SIZE_T**. This type is declared in BaseTsd.h as follows: `typedef LONG_PTR SSIZE_T;`

TBYTE	A **WCHAR** if **UNICODE** is defined, a **CHAR** otherwise.
	
	This type is declared in WinNT.h as follows:
	
	C++
	`#ifdef UNICODE`
	` typedef WCHAR TBYTE;`
	`#else`
	` typedef unsigned char TBYTE;`
	`#endif`

TCHAR	A **WCHAR** if **UNICODE** is defined, a **CHAR** otherwise.
	
	This type is declared in WinNT.h as follows:
	
	C++
	`#ifdef UNICODE`
	` typedef WCHAR TCHAR;`
	`#else`
	` typedef char TCHAR;`
	`#endif`

UCHAR	An unsigned **CHAR**. This type is declared in WinDef.h as follows: `typedef unsigned char UCHAR;`

UHALF_PTR

An unsigned HALF_PTR. Use within a structure that contains a pointer and two small fields.

This type is declared in BaseTsd.h as follows:

C++

```cpp
#ifdef _WIN64
 typedef unsigned int UHALF_PTR;
#else
 typedef unsigned short UHALF_PTR;
#endif
```

UINT

An unsigned INT. The range is 0 through 4294967295 decimal.

This type is declared in WinDef.h as follows:

`typedef unsigned int UINT;`

UINT_PTR

An unsigned INT_PTR.

This type is declared in BaseTsd.h as follows:

C++

```cpp
#if defined(_WIN64)
 typedef unsigned __int64 UINT_PTR;
#else
 typedef unsigned int UINT_PTR;
#endif
```

UINT8

An unsigned INT8.

This type is declared in BaseTsd.h as follows:

`typedef unsigned char UINT8;`

UINT16

An unsigned INT16.

This type is declared in BaseTsd.h as follows:

	`typedef unsigned short UINT16;`
UINT32	An unsigned INT32. The range is 0 through 4294967295 decimal. This type is declared in BaseTsd.h as follows: `typedef unsigned int UINT32;`
UINT64	An unsigned INT64. The range is 0 through 18446744073709551615 decimal. This type is declared in BaseTsd.h as follows: `typedef usigned __int 64 UINT64;`
ULONG	An unsigned LONG. The range is 0 through 4294967295 decimal. This type is declared in WinDef.h as follows: `typedef unsigned long ULONG;`

ULONGLONG	A 64-bit unsigned integer. The range is 0 through 18446744073709551615 decimal.
	
	This type is declared in WinNT.h as follows:
	
	C++
	
	`#if !defined(_M_IX86)`
	` typedef unsigned __int64 ULONGLONG;`
	`#else`
	` typedef double ULONGLONG;`
	`#endif`

ULONG_PTR	An unsigned LONG_PTR.
	
	This type is declared in BaseTsd.h as follows:
	
	C++
	
	`#if defined(_WIN64)`
	` typedef unsigned __int64 ULONG_PTR;`
	`#else`
	` typedef unsigned long ULONG_PTR;`
	`#endif`

ULONG32	An unsigned LONG32. The range is 0 through 4294967295 decimal. This type is declared in BaseTsd.h as follows: `typedef unsigned int ULONG32;`
ULONG64	An unsigned LONG64. The range is 0 through 18446744073709551615 decimal. This type is declared in BaseTsd.h as follows: `typedef unsigned __int64 ULONG64;`

UNICODE_STRING	A Unicode string.
	
	This type is declared in Winternl.h as follows:
	
	C++
	`typedef struct _UNICODE_STRING {`
	`  USHORT  Length;`
	`  USHORT  MaximumLength;`
	`  PWSTR  Buffer;`
	`} UNICODE_STRING;`
	`typedef UNICODE_STRING *PUNICODE_STRING;`
	`typedef const UNICODE_STRING *PCUNICODE_STRING;`

USHORT	An unsigned SHORT. The range is 0 through 65535 decimal. This type is declared in WinDef.h as follows: `typedef unsigned short USHORT;`
USN	An update sequence number (USN). This type is declared in WinNT.h as follows: `typedef LONGLONG USN;`
VOID	Any type. This type is declared in WinNT.h as follows:

	`#define VOID void`
WCHAR	A 16-bit Unicode character. For more information, see Character Sets Used By Fonts. This type is declared in WinNT.h as follows: `typedef wchar_t WCHAR;`
WINAPI	The calling convention for system functions. This type is declared in WinDef.h as follows: `#define WINAPI __stdcall` **CALLBACK**, **WINAPI**, and **APIENTRY** are all used to define functions with the __stdcall calling convention. Most functions in the Windows API are declared using **WINAPI**. You may wish to use **CALLBACK** for the callback functions that you implement to help identify the function as a callback function.
WORD	A 16-bit unsigned integer. The range is 0 through 65535 decimal. This type is declared in WinDef.h as follows: `typedef unsigned short WORD;`
WPARAM	A message parameter. This type is declared in WinDef.h as follows: `typedef UINT_PTR WPARAM;`

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]

Header	BaseTsd.h;
	WinDef.h;
	WinNT.h

Community Additions

mapi email

mapi email program

gary gunter 32
5/30/2014

DWORD length

If DWORD is typedef to unsigned long, then its length varies based on CPU type, not necessarily 32bit.

Jacky Hsiang
6/6/2013

Error in HWINSTA

It says the name of the handle is HWINSTA, but the code says "typedef HANDLE WINSTA". I suppose the true code is "typedef HANDLE HWINSTA".

AnUser123
12/26/2012

Visual Basic 9 Equivalents for PInvoke

MSDN Type	**Visual Basic 9 Type**
ATOM	UShort
BOOL	Integer
BOOLEAN	Byte
BYTE	Byte
CALLBACK	Delegate
CHAR	SByte
COLORREF	UInteger
CONST	Const
DWORD	UInteger
DWORDLONG	ULong
DWORD_PTR	UInteger (ULong)
DWORD32	UInteger
DWORD64	Long
FLOAT	Single
HACCEL	IntPtr

HALF_PTR Short (Integer)
HANDLE IntPtr
HBITMAP IntPtr
HBRUSH IntPtr
HCONV IntPtr
HCONVLIST IntPtr
HCURSOR IntPtr
HDC IntPtr
HDDEDATA IntPtr
HDESK IntPtr
HDROP IntPtr
HDWP IntPtr
HENHMETAFILE IntPtr
HFILE Integer
HFONT IntPtr
HGIDOBJ IntPtr
HGLOBAL IntPtr
HHOOK IntPtr
HICON IntPtr
HINSTANCE IntPtr
HKEY IntPtr
HKL IntPtr
HLOCAL IntPtr
HMENU IntPtr
HMETAFILE IntPtr
HMODULE IntPtr
HMONITOR IntPtr
HPALETTE IntPtr
HPEN IntPtr
HRESULT Integer
HRGN IntPtr
HRSRC IntPtr
HSZ IntPtr
HWINSTA IntPtr
HWND IntPtr
INT_PTR Integer (Long)
INT32 Integer
INT64 Long
LANGID UShort
LCID UInteger
LGRPID UInteger
LONG Integer
LONGLONG Long
LONG_PTR Integer (Long)
LONG32 Integer
LONG64 Long
LPARAM Integer (Long)
LPBOOL ByRef Integer
LPBYTE ByRef Byte
LPCOLORREF UInteger
LPCSTR ByRef SByte
LPCTSTR ByRef Char
LPCWSTR ByRef Char

LPDWORD UInteger
LPHANDLE ByRef IntPtr
LPINT Integer (Long)
LPLONG Integer
LPSTR ByRef SByte
LPTSTR ByRef Char
LPVOID IntPtr
LPWORD UShort
LPWSTR ByRef Char
LRESULT Integer (Long)
PBOOL Integer (Long)
PBOOLEAN ByRef Byte
PBYTE ByRef Byte
PCHAR ByRef SByte
PCSTR ByRef SByte
PCTSTR ByRef Char
PCWSTR ByRef Char
PDWORD UInteger
PDWORDLONG ByRef ULong
PDWORD_PTR ByRef UInteger (ULong)
PDWORD32 ByRef UInteger
PDWORD64 ByRef Long
PFLOAT ByRef Single
PHALF_PTR ByRef Short (Integer)
PHANDLE ByRef IntPtr
PHKEY ByRef IntPtr
PINT Integer (Long)
PINT_PTR ByRef Integer (Long)
PINT32 ByRef Integer
PINT64 ByRef Long
PLCID UInteger
PLONG Integer
PLONGLONG ByRef Long
PLONG_PTR ByRef Integer (Long)
PLONG32 ByRef Integer
PLONG64 ByRef Long
POINTER_32 (IntPtr)
POINTER_64 IntPtr
POINTER_SIGNED IntPtr
POINTER_UNSIGNED UIntPtr
PSHORT Short
PSIZE_T ByRef UInteger (ULong)
PSSIZE_T ByRef Integer (Long)
PSTR ByRef SByte
PTBYTE ByRef Char
PTCHAR ByRef Char
PTSTR ByRef Char
PUCHAR ByRef Byte
PUHALF_PTR ByRef UShort (UInteger)
PUINT ByRef UInteger
PUINT_PTR ByRef UInteger (ULong)
PUINT32 ByRef UInteger
PUINT64 ByRef ULong

PULONG UInteger
PULONGLONG ByRef ULong
PULONG_PTR ByRef UInteger (ULong)
PULONG32 ByRef UInteger
PULONG64 ByRef ULong
PUSHORT UShort
PVOID IntPtr
PWCHAR ByRef Char
PWORD UShort
PWSTR ByRef Char
SC_HANDLE IntPtr
SC_LOCK IntPtr
SERVICE_STATUS_HANDLE IntPtr
SHORT Short
SIZE_T UInteger (ULong)
SSIZE_T Integer (Long)
TBYTE Char
TCHAR Char
UCHAR Byte
UHALF_PTR UShort (UInteger)
UINT UInteger
UINT_PTR UInteger (ULong)
UINT32 UInteger
UINT64 ULong
ULONG UInteger
ULONGLONG ULong
ULONG_PTR UInteger (ULong)
ULONG32 UInteger
ULONG64 ULong
UNICODE_STRING Structure UNICODE_STRING : Dim Lenght As UShort, MaximumLenght As UShort, ByRef Buffer As Char : End Structure
USHORT UShort
USN Long
VOID Object
WCHAR Char
WIANPI Delegate
WORD UShort
WPARAM UInteger (ULong)

2 types means 32bit plaform (64bit platform)
Assumes #Unicode directive
Assumest highest Windows version possible

See full table http://spreadsheets.google.com/ccc?key=pK5CEcdG9GYGeO7K2dmEcBg

yic81
10/7/2012

LONGLONG - defined via double?

LONGLONG

64-bit signed integer.
The range is –9223372036854775808 through 9223372036854775807 decimal.

This type is declared in WinNT.h as follows:

```
#if !defined(_M_IX86)

typedef __int64 LONGLONG;

#else

typedef double LONGLONG;

#endif
```

Is it in above is typing error?

The datatype "double" is defined:

> Type double is a floating type that is larger than or equal to type float, but shorter than or equal to the size of type longdouble.1
>
> http://msdn.microsoft.com/en-us/library/cc953fe1.aspx

yic81
10/7/2012

This article needs reviewing

When was it last reviewed? 15 years ago?
The statement `typedef HANDLE HINSTANCE;` is totally incorrect, as many other typedef HANDLEs. Vast majority of them are now DECLARE_HANDLE() structs. Please review and fix this article. See this KB83456 http://support.microsoft.com/kb/83456 (last updated November 1999) for more details

yic81
10/7/2012

DOUBLE and CY are undocumented

The types DOUBLE and CY are undocumented here, although their existence is attested by the documentation page for VARIANT.

The type CY is defined aside with CURRENCY instead.

yic81
10/7/2012

qword isn't defined

Apparently, QWORD isn't defined in any of the windows header files for MSVC 2010.

yic81
10/7/2012

HWND can't be read

Note that
though HWND is a "pointer to void *"
or (in the VBasic example) an IntPtr (pointer to an int). So size of a pointer.

you can't actually read the value it "is a pointer to," or write to that location. It's just a pointer into some deep dark windows data structure, and the fact that it points into that exact location is all it gives you. You'll get a memory read exception if you try to read from that location.

yic81
10/7/2012