EN 此内容没有您的语言版本，但有英语版本。

Registry Value Types

A registry value can store data in various formats. When you store data under a registry value, for instance by calling the **RegSetValueEx** function, you can specify one of the following values to indicate the type of data being stored. When you retrieve a registry value, functions such as **RegQueryValueEx** use these values to indicate the type of data retrieved.

The following registry value types are defined in Winnt.h.

Value	Type
REG_BINARY	Binary data in any form.
REG_DWORD	A 32-bit number.
REG_DWORD_LITTLE_ENDIAN	A 32-bit number in little-endian format. Windows is designed to run on little-endian computer architectures. Therefore, this value is defined as REG_DWORD in the Windows header files.
REG_DWORD_BIG_ENDIAN	A 32-bit number in big-endian format. Some UNIX systems support big-endian architectures.
REG_EXPAND_SZ	A null-terminated string that contains unexpanded references to environment variables (for example, "%PATH%"). It will be a Unicode or ANSI string depending on whether you use the Unicode or ANSI functions. To expand the environment variable references, use the **ExpandEnvironmentStrings** function.
REG_LINK	A null-terminated Unicode string that contains the target path of a symbolic link that was created by calling the **RegCreateKeyEx** function with REG_OPTION_CREATE_LINK.
REG_MULTI_SZ	A sequence of null-terminated strings, terminated by an empty string (\0). The following is an example: String1\0String2\0String3\0LastString\0\0 The first \0 terminates the first string, the second to the last \0 terminates the last string, and the final \0 terminates the sequence. Note that the final terminator must be factored into the length of the string.

REG_NONE	No defined value type.
REG_QWORD	A 64-bit number.
REG_QWORD_LITTLE_ENDIAN	A 64-bit number in little-endian format. Windows is designed to run on little-endian computer architectures. Therefore, this value is defined as REG_QWORD in the Windows header files.
REG_SZ	A null-terminated string. This will be either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions.

String Values

If data has the REG_SZ, REG_MULTI_SZ, or REG_EXPAND_SZ type, the string may not have been stored with the proper terminating null characters. Therefore, when reading a string from the registry, you must ensure that the string is properly terminated before using it; otherwise, it may overwrite a buffer. (Note that REG_MULTI_SZ strings should have two terminating null characters.)

When writing a string to the registry, you must specify the length of the string, including the terminating null character (\0). A common error is to use the **strlen** function to determine the length of the string, but to forget that **strlen** returns only the number of characters in the string, not including the terminating null. Therefore, the length of the string should be calculated as follows: strlen(string) + 1

A REG_MULTI_SZ string ends with a string of length 0. Therefore, it is not possible to include a zero-length string in the sequence. An empty sequence would be defined as follows: \0.

The following example walks a REG_MULTI_SZ string.

C++

```
#include <windows.h>
#include <tchar.h>
#include <stdio.h>

void SampleSzz(PTSTR pszz)
{
   _tprintf(_TEXT("\tBegin multi-sz string\n"));
   while (*pszz)
   {
      _tprintf(_TEXT("\t\t%s\n"), pszz);
      pszz = pszz + _tcslen(pszz) + 1;
   }
   _tprintf(_TEXT("\tEnd multi-sz\n"));
}

int __cdecl main(int argc, char **argv)
{
```

```
    // Because the compiler adds a \0 at the end of quoted strings,
    // there are two \0 terminators at the end.

    _tprintf(_TEXT("Conventional multi-sz string:\n"));
    SampleSzz(_TEXT("String1\0String2\0String3\0LastString\0"));

    _tprintf(_TEXT("\nTest case with no strings:\n"));
    SampleSzz(_TEXT(""));

    return 0;
}
```

Byte Formats

In little-endian format, a multi-byte value is stored in memory from the lowest byte (the "little end") to the highest byte. For example, the value 0x12345678 is stored as (0x78 0x56 0x34 0x12) in little-endian format.

In big-endian format, a multi-byte value is stored in memory from the highest byte (the "big end") to the lowest byte. For example, the value 0x12345678 is stored as (0x12 0x34 0x56 0x78) in big-endian format.