Contents

The purpose of the library 1
Languages supported 1
Implemented methods 1
Samples 2
How to use 2
How to redistribute 2
Limitations of trial version 2

The purpose of the library

Long Path library allows accessing and performing typical operations with a file that has:

1. very long path (more than 256 characters)
2. unsupported characters in its name (e.g. trailing spaces)

Both NTFS and FAT volumes are supported.

Languages supported

The Long Path library is implemented as a DLL. It can be used with different environments and languages. The ones that were tried are:

- .NET Languages (C#, VB, Managed C++, etc.);
- C++ (MS Visual studio family, Borland C++ Builder, etc.);
- VB 6.0;
- and others.

Implemented methods

There are a number of low level functions you can use to substitute the common Win32 API:

- CreateDirectoryL
- RemoveDirectoryL
- DirectoryExistsL
- CreateFileL
- CopyFileL
- MoveFileL
- DeleteFileL
- FindFirstFileL
- FindNextFileL
- FindCloseL
- GetFileAttributesL

There are also classes like **LongFileAndDirectoryManager** (.NET), implementing some high level functionality using underlying low level functions.

Samples

There are several samples that demonstrate the library usage:

- C# sample, which demonstrates the file browser
- C# sample, which removes trailing spaces from a specified folder recursively
- C++ sample of library usage
- VB 6.0 sample of library usage

How to use

It is very easy to include it in your project. To use it with your project, you should perform the following steps:

- .NET Copy FileDirectoryManager.cs file to your project folder and include it into your solution.
- C++ include Import.h file to the solution.
- VB 6 Copy and include module LongPathModule.bas

Copy LongPathLibraryDLL.dll to your project folder and to Debug and Release folders.
For more information, please check the samples folder.

How to redistribute

Copy LongPathLibraryDLL.dll to your application folder and that's it.

Limitations of trial version

Some functions in trial version are not available; all these functions are setting error code 0x5 (ERROR_ACCESS_DENIED) with SetLastError function. You can check the results using the following code.

In .NET Marshal.GetLastWin32Error() == 0x5.
In C++ GetLastError() == ERROR_ACCESS_DENIED.

Following functions are limited:

- RemoveDirectoryL
- CreateFileL (limits opening existing files that are longer than 100 kb)

- `MoveFileL`
- `DeleteFileL`