A library is a collection of prewritten code such as functions, types and data that can be reused in other projects.

Note hereafter functions, types and global objects are referred to as symbols.

A library can be a static link library or a dynamic link library (DLL).

Static link library symbols are linked to another project at link time whereas DLL symbols are linked at runtime.

Both static link library and dynamic link library symbols can be linked to another static library or dynamic link library or application executable.

Only symbols which are exported from the DLL can be linked to another project. The other project is said to import such symbols from the DLL.

To export a symbol from the DLL, the symbol needs to be decorated with \_declspec(dllexport) declaration, in the DLL project. Symbols which are not decorated with \_declspec(dllexport) are known as internal or private symbols. To import a symbol from the DLL, the symbol needs to be decorated with \_declspec(dllimport) in another project. Refer to our DLL project for further clarity.

When a DLL project is built a dll, a lib and an exp files are produced. The lib file produced is called an import library. It doesn’t contain compiled code but contains a list of symbols which are exported from the DLL. The compiled definition of symbols is present in DLL.

A dll can be linked to a project implicitly or explicitly. When linked implicitly, the DLL is loaded at the start of the application and remains loaded till the end of the application. When linked explicitly, the DLL is loaded when needed and unloaded when not needed.

To link DLL implicitly, you need to specify import library name in Project > Properties > Linker > Input > Additional Dependencies property. Also when DLL is linked implicitly, it is searched in various paths in the following sequence:

application directory

System32 directory

Windows directory

Directories mentioned in PATH environment variable

If DLL is not found in any of the above paths, then loading error occurs.

To link DLL explicitly, first do not specify import library name in Addition Dependencies property. Instead use LoadLibrary API to load DLL and FreeLibrary API to unload the DLL.

A static link library developer will have to share .h and .lib with another library or application developer.

A DLL developer will have to share .h, .lib, .exp and .dll with another library or application developer.

An application developer will have to share .exe and .dll with the end user.