Exporting from a DLL Using __declspec(dllexport)

05/06/2019 • 2 minutes to read • 

In this article

You can export data, functions, classes, or class member functions from a DLL using the `__declspec(dllexport)` keyword. `__declspec(dllexport)` adds the export directive to the object file so you do not need to use a .def file.

This convenience is most apparent when trying to export decorated C++ function names. Because there is no standard specification for name decoration, the name of an exported function might change between compiler versions. If you use `__declspec(dllexport)`, recompiling the DLL and dependent .exe files is necessary only to account for any naming convention changes.

Many export directives, such as ordinals, NONAME, and PRIVATE, can be made only in a .def file, and there is no way to specify these attributes without a .def file. However, using `__declspec(dllexport)` in addition to using a .def file does not cause build errors.

To export functions, the `__declspec(dllexport)` keyword must appear to the left of the calling-convention keyword, if a keyword is specified. For example:

Copy

```
__declspec(dllexport) void __cdecl Function1(void);
```

To export all of the public data members and member functions in a class, the keyword must appear to the left of the class name as follows:

Copy

```
class __declspec(dllexport) CExampleExport : public CObject
{ ... class definition ... };
```

> **Note**
>
> `__declspec(dllexport)` cannot be applied to a function with the `__clrcall` calling convention.

When building your DLL, you typically create a header file that contains the function prototypes and/or classes you are exporting and add **`__declspec(dllexport)`** to the declarations in the header file. To make your code more readable, define a macro for **`__declspec(dllexport)`** and use the macro with each symbol you are exporting:

```
#define DllExport   __declspec( dllexport )
```

`__declspec(dllexport)` stores function names in the DLL's export table. If you want to optimize the table's size, see Exporting Functions from a DLL by Ordinal Rather Than by Name.

What do you want to do?

- Export from a DLL using .def files
- Export and import using AFX_EXT_CLASS
- Export C++ functions for use in C-language executables
- Export C functions for use in C or C++-language executables
- Determine which exporting method to use
- Import into an application using __declspec(dllimport)
- Initialize a DLL

What do you want to know more about?

- The __declspec keyword
- Importing and exporting inline functions
- Mutual imports

See also

Exporting from a DLL

Is this page helpful?

Yes No