

Re: A solution to warning C4251 – class needs to have dll-interface...?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2007-10/msg00387.html>

- *From:* "Doug Harrison [MVP]" <dsh@xxxxxxx>
 - *Date:* Wed, 17 Oct 2007 15:24:00 -0500
-

On Wed, 17 Oct 2007 19:22:37 +0200, "Niels Dekker – no return address" <unknown@xxxxxxxxxxxxxxxx> wrote:

It sounds like this scenario won't happen when using an STL container within my dllexport class, right? Because the functions of the STL containers are completely defined by their header files. We're using both VC++ 7.1 and 8.0, with the included STL.

I think you should be OK, but there are some important restrictions which I talk about below.

Anyway, you're just talking about link errors, right? I was afraid that the warning indicated that my program might have runtime crashes! Especially because of Q172396, "You may experience an access violation when you access an STL object through a pointer or reference in a different DLL or EXE".
<http://support.microsoft.com/kb/q172396/>

Q172396 says: "Most classes in the Standard C++ Libraries use static data members..." Apparently this is the cause of those access violations.

Yes, and it's a tangential issue. The problem is that each module will get its own instance of the static data, instead of there being one instance of the static data for the entire program. The standard workaround for that is to explicitly instantiate and dllexport all the specializations you're going to use.

So you should never return a reference to an STL container from an dllexport class, right? Is it even unsafe to return a reference to an std::string, or a reference to a vector?

That should be fine, and the static data usage by the classes Q172396 is talking about was eliminated in VC7. (Dinkumware also published patches for

Re: A solution to warning C4251 – class needs to have dll-interface...?

the associative containers for VC6.)

Note that sharing C++ objects in this way between modules must be considered equivalent to static linking for compilation dependency purposes. In particular, all the modules should be compiled with the same options, and they must all use the same CRT DLL. It is very important to understand this, because otherwise, each module will end up with its own CRT state, including heap, file descriptors, and so forth, and you won't be able to fully share C++ objects between modules. For example, if modules X and Y use different CRTs, you won't be able to delete an object in one that was created by the other.

--

Doug Harrison
Visual C++ MVP

.