What is this [uuid(…)] in front of my C++ class declaration?

devblogs.microsoft.com/oldnewthing/20230331-00

March 31, 2023

Raymond Chen

A customer was dealing with some legacy code, and when they included one of their header files in a new project:

```
// foo.h

[uuid(a6107c25-4c22-4a12-8440-7eb8f5972e50)]
class Widget : public IWidget
{
    /* ... */
};
```

They ran into a weird compiler error:

```
error C2337: 'uuid': attribute not found
error C3688: invalid literal suffix 'a6107c25'; literal operator or literal operator
template 'operator ""a6107c25' not found
```

First question: What is this `[uuid(...)]` notation anyway?

Answer: It is a Visual C++ nonstandard extension that associates a UUID with a type, extractable by the `__uuid` extension. This is part of a larger set of attributes designed for COM and .NET.

Okay, so now that we know what it is, why isn't it working?

If you set the `/permissive-` compiler flag, then the Visual C++ compiler no longer supports this nonstandard attribute syntax. This is mentioned in the documentation for `/permissive-`:

> Microsoft-specific ATL attributes can cause issues under `/permissive-`:
>
> ```
> // Example 1
> [uuid("594382D9-44B0-461A-8DE3-E06A3E73C5EB")]
> class A {};
> ```
>
> You can fix the issue by using the `__declspec` form instead:
>
> ```
> // Fix for example 1
> class __declspec(uuid("594382D9-44B0-461A-8DE3-E06A3E73C5EB")) B {};
> ```

The `__declspec(uuid(...))` syntax is still nonstandard, but at least it's nonstandard in a standard-compliant way: Identifiers which begins with two underscores are reserved for the implementation.

Final mystery: Why did this problem show up all of a sudden? Their other projects worked fine.

The reason is that the customer created a new project, and `/permissive-` is the default for new projects. The corresponding property in the `.vcxproj` file is `<ConformanceMode>true</ConformanceMode>`.

And the solution is to update the header file to use `__declspec(uuid(...))` instead of `[uuid(...)]`:

```
// foo.h

class
__declspec(uuid("a6107c25-4c22-4a12-8440-7eb8f5972e50"))
Widget : public IWidget
{
    /* ... */
};
```

You can use the Windows-defined macro `DECLSPEC_UUID` as an alternative:

```
// foo.h

class
DECLSPEC_UUID("a6107c25-4c22-4a12-8440-7eb8f5972e50")
Widget : public IWidget
{
    /* ... */
};
```

The macro allows for more flexibility since it allows other toolchains to redefine the macro to expand to whatever makes sense for them (either some other custom extension, or simply nothing at all).

In both cases, note the placement of the declaration: It goes immediately before the name being declared, after the type specifier keyword `struct` or `class`.

If you are not at liberty to update the header file, then you'll have to disable `/permissive-` in order to get the header file to compile.