What does error E_ILLEGAL_DELEGATE_ASSIGNMENT mean?

devblogs.microsoft.com/oldnewthing/20210226-00

February 26, 2021

Raymond Chen

A customer reported that they program was crashing because it got the error `E_ILLEGAL_DELEGATE_ASSIGNMENT`. The description of this error is “A delegate was assigned when not allowed,” but what does that mean?

The most common source of this error is attempting to set a completion callback for a Windows Runtime `IAsyncXxx` asynchronous operation when a callback has already been set. There can be only one completion callback for an asynchronous operation, and you get this error if you try to set a second one.

You typically do not set completion callbacks explicitly. They are usually set for you as part of your programming framework. In the case of C++/WinRT, the most common place it it happens is when you `co_await` an asynchronous operation. (Less common sources are functions like `when_all` or `when_any`.)

The good news is that the stack trace from the crash dump usually points to the code that attempted to register the second callback.

```
SomeClass::SomeClass()
{
    m_start = StartAsync();
}

IAsyncAction SomeClass::DoThing1Async()
{
    co_await m_start;
    Thing1();
}

IAsyncAction SomeClass::DoThing2Async()
{
    co_await m_start; // ← crash here
    Thing2();
}
```

The idea here is that the object initializes itself at construction, but does so asynchronously. When the `DoThing` methods are called, they first wait for the initialization to complete, and then proceed with the actual operation.

The problem is that we are `co_await`ing the `m_start` object more than once. Each coroutine's call to `co_await` will try to set its own continuation as the completion handler so it can resume execution when the operation completes. The first one to perform the `co_await` succeeds. The rest fail with `E_ILLEGAL_DELEGATE_ASSIGNMENT`.

The rule that we walk away with is "Each `IAsyncXxx` can be awaited only once."

Okay, so what do we do if we need to await something more than once, like here, where we want all of the operations to wait for the initialization to complete before proceeding with work?

You'll have to find something that can be awaited more than once.

We'll look into possible solutions next time.

Raymond Chen

Follow