

# Why am I getting a weird error about `promise_type` when I try to write a coroutine?

 [devblogs.microsoft.com/oldnewthing/20210809-00](https://devblogs.microsoft.com/oldnewthing/20210809-00)

August 9, 2021



Raymond Chen

A customer was following [a tutorial by Kenny Kerr](#) and tried to create the world's simplest coroutine:

```
using namespace winrt::Windows::Foundation;

IAsyncAction Dummy()
{
    co_return;
}
```

This resulted in an error:

```
// MSVC
error C2039: 'promise_type': is not a member of 'std::experimental::coroutine_traits<winrt::Windows::Foundation::IAsyncAction>'

// gcc
error: unable to find the promise type for this coroutine

// clang
this function cannot be a coroutine: 'std::experimental::coroutine_traits<winrt::Windows::Foundation::IAsyncAction>' has no member named 'promise_type'
```

What is this error trying to say?

Recall that one of the steps in the coroutine transformation is taking the formal return type of the coroutine function, combining it with the formal parameters, and looking up a corresponding specialization of the `coroutine_traits` type, specifically to find a nested type name `promise_type`.

If no such specialization exists, then the lookup for `promise_type` fails.

And that's what the error message is complaining about. The error message is wearing compiler-colored glasses.

To be fair, it's not like the error message is intentionally being obtuse. What's happening is that the compiler front-end performs the coroutine transformation, and then the result goes through the type resolver, and it's the type resolver that can't find a `promise_type`. The type resolver doesn't know that the request for `promise_type` was the result of a coroutine transformation. It just reports it as a failed type lookup in the coroutine traits type.

It looks like gcc goes the extra mile and passes some information to the type resolver that "This type is coming from a coroutine transformation", so that it can generate a coroutine-specific error message.

Okay, so now that we understand what the error message means, how do we fix it?

To fix it, you need to provide the appropriate specialization of the `coroutine_traits` type. In the case of C++/WinRT, you get it by doing a

```
#include <winrt/Windows.Foundation.h>
```

This is consistent with the overall C++/WinRT rules that you must include the header file for any namespace you use.

Raymond Chen

**Follow**

