# Why do I get E_ACCESSDENIED when trying to access my brokered Windows Runtime object?COM is double-checking the trust level.

devblogs.microsoft.com/oldnewthing/20240808-00

August 8, 2024

Raymond Chen

A customer implemented a Windows Runtime object but found that it didn't work from the UWP app container, so they changed it from an in-process server to a brokered server by manifesting it as PartialTrust. They double-checked the registration by calling `RoGetActivatableClassRegistration` and then asking for the `RegisteredTrustLevel`, and it did indeed report as `PartialTrust`, yet when they tried to create the object, they code `E_ACCESSDENIED`. What is going on?

COM is being cautious and watching out for people trying sneaky tricks in order to activate an object with the wrong trust level. When it obtains the activation factory or an object from that factory, it asks the factory or object for its `TrustLevel` directly, and if the answer disagrees with the manifest, then COM assumes that somebody is trying to pull a fast one, and it says, "The manifest declares one trust level, but when I ask the object, it claims to be a different trust level. I get a bad feeling from this, so I'm not going to let this one go through."

In this case, what this means is that when you converted your object from BaseTrust to PartialTrust, you forgot to update the `GetTrustLevel()` method to return `PartialTrust`.

If you implemented your object and object factory in C++/WinRT, you need to override the default reported trust level of `BaseTrust` with a custom trust level:

```
namespace winrt::Contoso::implementation
{
    struct Widget : WidgetT<Widget>
    {
        auto GetTrustLevel() { return TrustLevel::PartialTrust; }

        ⟦ remainder of class ⟧
    };
}

namespace winrt::Contoso::factory_implementation
{
    struct Widget : WidgetT<Widget>
    {
        auto GetTrustLevel() { return TrustLevel::PartialTrust; }

        ⟦ remainder of class ⟧
    };
}
```

If you implemented your objects in C++/WRL, then you report your trust level in the `InspectableClass` macro of your object type.

```
class Widget : public Microsoft::WRL::RuntimeClass<⟦ whatever ⟧>
{
    InspectableClass(RuntimeClass_Contoso_Widget, PartialTrust);

    ⟦ remainder of class ⟧
};
```

The factory trust level takes its cue from the object trust level, so you don’t have to make any explicit changes to the factory.

If you have a static class (so there are no objects from which the factory class can infer its trust level), you put the trust level in the second parameter to your existing `Inspectable-ClassStatic` macro:

```
class WidgetFactory : public Microsoft::WRL::AgileActivationFactory<⟦ whatever ⟧>
{
    InspectableClassStatic(RuntimeClass_Contoso_Widget, PartialTrust);

    ⟦ remainder of class ⟧
};
```