Can the MTA thread exit while keeping its COM class registrations alive?

devblogs.microsoft.com/oldnewthing/20191114-00

November 14, 2019

Raymond Chen

A customer has an app whose main thread is a single-threaded apartment (STA). The app also has a dedicated multi-threaded apartment (MTA) thread whose job is to register some class objects. Right now, they register the objects on the dedicated thread and hold the thread hostage for the lifetime of the process.

Here's a sketch of what they do:

```
// thread start
CoInitializeEx(nullptr, COINIT_MULTITHREADED);

CoRegisterClassObject(CLSID_Something1, something1Factory,
    CLSCTX_LOCAL_SRVER, REGCLS_MULTI_SEPARATE, &token1);
CoRegisterClassObject(CLSID_Something2, something2Factory,
    CLSCTX_LOCAL_SRVER, REGCLS_MULTI_SEPARATE, &token2);
// etc.

PumpMessageAndWaitForAnExitSignal();

// Clean up
CoRevokeClassObject(token1);
CoRevokeClassObject(token2);
// etc.

CoUninitialize();
// end of thread
```

The sole purpose of the thread is to keep the MTA alive for the class factories that it registered.

The customer wanted to know if there was a way to register the class factories and then exit the thread immediately. That way, they don't need to burn a thread for the sole purpose of keeping the MTA alive.

Normally, when the last MTA thread exits, the class factories are automatically unregistered, so exiting the thread doesn't seem to be a good idea. But we can put `CoIncrementMTAUsage` to work for us: We can use it to keep the MTA alive despite not having a thread dedicated to it. This is the same trick we used some time ago to avoid creating an accidental choke point when handing work to a background thread.

```
// thread start
CoInitializeEx(nullptr, COINIT_MULTITHREADED);

CoRegisterClassObject(CLSID_Something1, something1Factory,
    CLSCTX_LOCAL_SRVER, REGCLS_MULTI_SEPARATE, &token1);
CoRegisterClassObject(CLSID_Something2, something2Factory,
    CLSCTX_LOCAL_SRVER, REGCLS_MULTI_SEPARATE, &token2);
// etc.

CoIncrementMTAUsage(&cookie);

CoUninitialize();
// end of thread
```

Later, when you decide that it's time to clean up:

```
// Clean up
CoRevokeClassObject(token1);
CoRevokeClassObject(token2);
// etc.

CoDecrementMTAUsage(cookie);
```

The MTA usage cookie keeps the MTA alive without requiring a dedicated thread.

Raymond Chen

Follow