# How can a COM local server keep itself alive even though there are no active clients?

devblogs.microsoft.com/oldnewthing/20170127-00

January 27, 2017

Raymond Chen

Suppose you have a COM local server, which means that the COM object is provided by a process different from the clients. And suppose you have a method which triggers some sort of background operation, and the rule for your object is that even if all references to the object are released, the background operation must still run to completion. (This is the model followed by Windows Runtime objects, for example. An asynchronous operation will continue to run independently of the object which initiated the operation.)

The question, then, is how to communicate to the COM infrastructure that you are still doing work and the server should not be shut down just because there are no outstanding references to it.

Assuming the object is MTA-threaded or free-threaded, the recommendation for out-of-process servers is to use `CoAddRefServerProcess` and `CoReleaseServerProcess` to manage their process lifetime. In particular, `CoReleaseServerProcess` will automatically call `CoSuspendClassObjects` when the process reference count reaches zero, thereby avoiding the race condition where one thread drops the reference count to zero, and before it can call `CoSuspendClassObjects`, another thread sneaks in and creates an object.

I'm not the subject matter expert here, so I'll leave you with some fascinating reading on the subject.

- This article discusses the issue at hand and covers many of the race conditions that can occur.
- This article is a more general discussion of free-threaded COM objects.

**Warning**: Both articles were originally magazine articles, and the tall narrow formatting is carried over to the Web presentation.

Raymond Chen

**Follow**