One possible reason why ShellExecute returns SE_ERR_ACCESSDENIED and ShellExecuteEx returns ERROR_ACCESS_DENIED

devblogs.microsoft.com/oldnewthing/20101118-00

November 18, 2010

Raymond Chen

(The strangely-phrased subject line is for search engine optimization.) A customer reported that when they called `ShellExecute`, the function sometimes fails with `SE_ERR_ACCESSDENIED`, depending on what they are trying to execute. (If they had tried `ShellExecuteEx` they would have gotten the error `ERROR_ACCESS_DENIED`.) After a good amount of back-and-forth examing file type registrations, a member of the development team had psychic insight to ask, "Are you calling it from an MTA?" "Yes," the customer replied. "ShellExecute is being called from a dedicated MTA thread. Would that cause the failure?" Why yes, as a matter of fact, and it's called out in the documentation for `ShellExecute`.

> Because **ShellExecute** can delegate execution to Shell extensions (data sources, context menu handlers, verb implementations) that are activated using Component Object Model (COM), COM should be initialized before **ShellExecute** is called. Some Shell extensions require the COM single-threaded apartment (STA) type.

As a general rule, shell functions require STA. Recall that MTA implies no user interface. If you try to use an apartment-threaded object from your MTA thread, a marshaller is required, and if no such marshaller exists, the call fails.

This also explains why the failure occurs only for certain file types: If handling the file type happens not to involve creating a COM object, then the MTA/STA mismatch situation never occurs.

Raymond Chen

Follow