Why are there four functions for parsing strings into GUIDs, and why are they in three different DLLs?

devblogs.microsoft.com/oldnewthing/20160331-00

March 31, 2016

Raymond Chen

Some time ago, we discussed the differences among various functions that take a string and produce a GUID-like thing. Let's look at that table.

Function	Exported by
`UuidFromString`	rpcrt4.dll
`IIDFromString`	ole32.dll
`CLSIDFromString`	ole32.dll
`GUIDFromString`	shell32.dll

Why are there four such functions, and more importantly, why are they in three different DLLs?

As you might expect, the answer comes from history.

The first two functions on the scene are the ones in the middle of the table. `IIDFromString` and `CLSIDFromString` come from the original 32-bit OLE library. They differ in their intended use. The second one is for parsing strings that represent OLE objects. It so happens that you are allowed to do this either by specifying the raw GUID as a string, or by specifying the programmatic ID for the class. That's why `CLSIDFromString` does the extra work of looking in `HKEY_CLASSES_ROOT` to convert the string to a CLSID.

On the other hand, interface IDs have no such alternate notation, so the `IIDFromString` function accepts only stringized GUIDs.

At this point in time, OLE was a monolithic DLL. It then became apparent that the monolithic OLE DLL was really doing several things: It managed document linking and embedding (OLE). As part of that work, it also had to manage the component object model

(COM). And in the case where the components are in different processes, it needs to perform remote procedure calls (RPC).

The remote procedure call functionality was useful in its own right, so the OLE team spun it off into its own library, and OLE would be one of many clients of the new library. That new library was called RPCRT4, which I'm guessing stands for "remote procedure call runtime, fourth attempt" (?).

The remote procedure call library therefore had to have its own parser for stringized GUIDs; it couldn't call up into OLE because that would be a layering violation. (RPC is the low-level component and OLE is the high-level component.) And besides, the components which were using the raw RPC layer were doing so because they explicitly didn't want OLE. Having the string parsing function in OLE would force components to load OLE, which ruined the point of splitting RPC into its own library. For want a string-parsing function the kingdom was lost.

The last function on the scene is `GUIDFromString`. This was written by the shell team back in the days of OLE Chicken. (not to be confused with Chicken Ole). The shell needed only a limited subset of OLE in order to function. To avoid the performance impact of loading all of OLE (and allocating a whopping 32KB of memory), it contained a miniature copy of OLE; just enough to let the shell do what it needed. And one of the things in that miniature copy of OLE was a function to parse strings into GUIDs.

Raymond Chen

Follow