Psychic debugging: Why does FormatMessage say the resource couldn't be found?

devblogs.microsoft.com/oldnewthing/20070529-00

May 29, 2007

Raymond Chen

Solving this next problem should be a snap with your nascent psychic powers:

> I'm trying use `FormatMessage` to load a resource string with one insertion in it, and this doesn't work for some reason. The string is "Blah blah blah %1. Blah blah blah." The call to `FormatMessage` fails, and `GetLastError()` returns `ERROR_RESOURCE_TYPE_NOT_FOUND`. What am I doing wrong?
>
> ```
> LPTSTR pszInsertion = TEXT("Sample");
> LPTSTR pszResult;
> FormatMessage(
>         FORMAT_MESSAGE_ALLOCATE_BUFFER |
>         FORMAT_MESSAGE_FROM_HMODULE |
>         FORMAT_MESSAGE_ARGUMENT_ARRAY,
>         //I also tried an instance handle and NULL.
>         GetModuleHandle(NULL),
>         IDS_MY_CUSTOM_MESSAGE,
>         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // default language
>         (LPTSTR) &pszResult,
>         0,
>         (va_list*) &pszInsertion);
> ```

Hint: Take a closer look at the parameter `IDS_MY_CUSTOM_MESSAGE`.

Hint 2: What does "`IDS_`" tell you?

Resource identifiers that begin with "`IDS_`" are typically string resource identifiers, not message resource identifiers. There is no strong consensus on the naming convention for message resource identifiers, although I've seen "`MSG_`". Part of the reason why there is no strong consensus on the naming convention for message resource identifiers is that almost nobody uses message resources! I don't understand why they were added to Win32, since there was already a way of embedding strings in resources, namely, string resources.

That's why you're getting `ERROR_RESOURCE_TYPE_NOT_FOUND`. There is no message resource in your module. If you're not going to use a message resource, you'll have to use the `FORMAT_MESSAGE_FROM_STRING` flag and pass the format string explicitly.

```
DWORD_PTR rgdwInsertions[1] = { (DWORD_PTR)TEXT("Sample") };
TCHAR szFormat[256];
LoadString(hInstance, IDS_MY_CUSTOM_MESSAGE, szFormat, 256);
LPTSTR pszResult;
FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER |
        FORMAT_MESSAGE_FROM_STRING |
        FORMAT_MESSAGE_ARGUMENT_ARRAY,
        szFormat,
        0,
        0,
        (LPTSTR) &pszResult,
        0,
        (va_list*) &rgdwInsertions);
```

I also made a slight change to the final parameter. When you use `FORMAT_MESSAGE_ARGUMENT_ARRAY`, the last parameter must be an array of `DWORD_PTR`s. (The parameter must be cast to `va_list*` to keep the compiler happy.) It so happens that the original code got away with this mistake since `sizeof(DWORD_PTR) == sizeof(LPTSTR)` and they both have the same alignment requirements. On the other hand, if the insertion were a `DWORD`, passing `(va_list*)&dwValue` is definitely wrong and can crash if you're sufficiently unlucky. (Determining the conditions under which your luck runs out is left as an exercise.)

Raymond Chen

Follow