Why can't I create my dialog with DialogBox, DialogBoxParam, CreateDialog, CreateDialogParam, or the indirect versions of same?

devblogs.microsoft.com/oldnewthing/20131129-00

November 29, 2013

Raymond Chen

One of the purposes of my dialog manager series was to help people diagnose problems with their dialog boxes. But since I embedded the tips inside the series body, it's hard for people to find them, and I still end up answering the same questions over and over.

So here it is in a separate article that hopefully people can find.

Why your call to `DialogBox` or `CreateDialog` is failing. This also goes for `CDialog::DoModal`, but perhaps extra so because the MFC source code says

```
* 3) CreateDialogIndirect() did NOT create the window (ie. due to error in template)
*    and returns NULL.
```

even though the problem is rarely due to an error in the template. A wrong comment is worse than no comment at all.[1]

I've decided to put the reasons in most-likely-error-first order rather than chronological.

> Your dialog template uses a custom control which has not been registered.

This is by far the number one reason why dialog boxes fail to be created. As we saw in Part 3, if a child control cannot be created from the template, then the dialog creation fails. (There is a special dialog box style `DS_NOFAILCREATE` that tells the dialog manager to ignore the error and just continue without the child control.)

For example, maybe your dialog box template uses a List View control, but you forgot to call `InitCommonControls` or `InitCommonControlsEx`. Or it uses a Rich Edit control and you forgot to load the appropriate library. Or it contains an ActiveX control but you forgot to call `AfxEnableControlContainer` to enable ActiveX control hosting.

A special case of this error is where your dialog template uses a custom control which was registered in a different activation context. If you are trying to use a control from version 6 of the common controls, then you must use the appropriate manifest and make sure that the manifest's activation context is active at the time you call `DialogBox` or whatever. If you forget to do this, then you will be using whatever activation context happens to be lying around at the time you call `DialogBox`, and it may not be the one you want. (It's like walking up to a computer and looking on the Desktop and not seeing the file you created yesterday. "Hey, who deleted my file!" But you forgot to do a Switch User to your own account. You're looking at the Desktop of whoever used the computer last. You're in the wrong context.)

Your resource name is incorrectly declared in the resource template.

One of the annoying features of the Resource Compiler is that it happily accepts typos! For example, suppose you have the following resource header file:

```
// ids.h
#define IDD_BRILLIANT 100
```

and the following resource file:

```
IDD_BRILLANT DIALOG ...
```

Since you misspelled `IDD_BRILLIANT`, you are not in fact creating a dialog box whose ID is 100 (which is what would have happened if you had spelled it correctly). Instead, you're creating a dialog whose name is the string `"IDD_BRILLANT"`.

Later, your code passes `MAKEINTRESOURCE(IDD_BRILLIANT)` to ask for dialog 100, and the dialog manager says, "Sorry, I don't see one of those." (To get the typo'd dialog template, you would have to pass `TEXT("IDD_BRILLANT")` as the template name.)

Even if you spell everything correctly, you will also have this problem if you forget to `#include "ids.h"` in your resource template file in the first place!

To diagnose this error, you can add a diagnostic call to `FindResource` (or simply trace through the dialog manager's call to the same function) to see if it can find the resource. Many IDEs will let you load a DLL and inspect its resources interactively. You can check the dialog template to see whether it is listed as resource 100 or as resource `"IDD_BRILLANT"`.

You passed the wrong instance handle or dialog ID.

This is a generalization of the previous error. The dialog template needs to exist in the module you passed, with the ID or name you passed. If you pass the wrong module or the wrong name, then you're not going to find it.

An even more generalized version of this error is the case where you forgot to add the dialog to the module's resources in the first place. (Maybe you forgot to add the resource file to your project.)

The diagnosis for this case is the same as the case of the misspelled dialog identifier.

> The control refused its creation.

In rare cases, a control may fail its creation by returning `FALSE` in response to the `WM_NC-CREATE` message or `-1` in response to the `WM_CREATE` message.

> You passed a bad window handle as the `hwndParent`.

An invalid parameter will naturally result in the function failing. This rarely occurs in practice because you should be using the handle of a window under your control, so it shouldn't be destroyed out from under you.

> There is an error in the dialog template.

I have never seen this occur. The Resource Compiler is pretty good about not generating erroneous templates.

[1] Perhaps the author meant to use *e.g.* (*exempli gratia*, which means "for example") instead of *i.e.* (*id est*, which means "that is"). As written, the comment is saying that an error in the template is the only reason that `CreateDialogIndirect` could have failed, when in fact it is only one example of a failure. Getting the two Latin abbreviations confused is not just a pedantic error; here, it created genuine confusion and probably wasted a lot of developers' time.

Raymond Chen

Follow