The mystery of the icon that never appears

devblogs.microsoft.com/oldnewthing/20140514-00

May 14, 2014

Raymond Chen

A customer reported a problem showing an icon on their dialog box.

> We verified that this code does execute during the handling of the `WM_INITDIALOG` message. No assertion fires, yet no icon appears either.
>
> ```
> SHFILEINFO sfi = { 0 };
> DWORD_PTR dwResult = SHGetFileInfo(m_pszFile, &sfi,
>                                    sizeof(sfi), SHGFI_ICON);
> assert(dwResult != 0);
> m_hico = sfi.hIcon;
> assert(m_hico != nullptr);
> assert(GetDlgItem(hdlg, IDI_CUSTOMICON) != nullptr);
> SendDlgItemMessage(hdlg, IDI_CUSTOMICON,
>                    WM_SETICON, ICON_BIG, (LPARAM)m_hico);
> assert(SendDlgItemMessage(hdlg, IDI_CUSTOMICON,
>                    WM_GETICON, ICON_BIG, 0) == (LPARAM)m_hico);
> ```
>
> Our dialog template says
>
> ```
>   ICON "", IDI_CUSTOMICON, 10, 10, 0, 0
> ```

The customer did some helpful preliminary troubleshooting:

- Verify that the code does indeed execute. It sounds obvious, but some people forget to check this. They get distracted trying to figure out why a function isn't working, when in fact the root cause is that *you forgot to call the function in the first place*.
- Verify that the `SHGetFileInfo` call succeeded. That rules out the case that the static control is displaying nothing because you didn't give it anything to display.
- Verify via `GetDlgItem` that the control you're trying to talk to really does exist. That rules out the case that you are talking to an empty room. (For example, maybe you added the control to the wrong template.)
- Verify via `WM_GETICON` that the attempt to change the icon really worked.

The problem is that the customer is using the wrong icon-setting message.

The `WM_SETICON` message lets you customize the icon that is displayed in the window's caption bar. For this to have any effect, your window naturally needs to have the `WS_CAPTION` style. If you don't have a caption, then telling the window manager, "Please display this icon in my caption" is mostly a waste of time. It's like signing up for a lawn-mowing service when you don't have a lawn.

The message to change the icon displayed *inside* a static control is `STM_SETICON`.

```
SendDlgItemMessage(hdlg, IDI_CUSTOMICON,
                   STM_SETICON, (LPARAM)m_hico, 0);
```

Red herring: Some of you may have noticed that the customer set their control size to 0×0. "You aren't seeing an icon because you set the control to zero size!" But since this control was not created with `SS_REALSIZECONTROL` or `SS_CENTERIMAGE`, the control will resize itself to match the size of the icon.

Here's a sample program to show both types of icons set on the same window, so you can see the difference.

```
#include <windows.h>
#include <commctrl.h>
LRESULT CALLBACK SubclassProc(HWND hwnd, UINT uMsg, WPARAM wParam,
    LPARAM lParam, UINT_PTR uIdSubclass, DWORD_PTR dwRefData)
{
 switch (uMsg) {
 case WM_NCDESTROY:
  RemoveWindowSubclass(hwnd, SubclassProc, 0);
  PostQuitMessage(0);
  break;
 }
 return DefSubclassProc(hwnd, uMsg, wParam, lParam);
}
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   PSTR lpCmdLine, int nShowCmd)
{
 HWND hwnd = CreateWindow("static", nullptr,
               WS_OVERLAPPEDWINDOW | WS_VISIBLE |
               SS_ICON | SS_CENTERIMAGE,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               nullptr, nullptr, hinst, nullptr);
 SetWindowSubclass(hwnd, SubclassProc, 0, 0);
 HICON hicoCaption = LoadIcon(nullptr, IDI_EXCLAMATION)
 SendMessage(hwnd, WM_SETICON, ICON_BIG,
             reinterpret_cast<LPARAM>(hicoCaption));
 HICON hicoClient = LoadIcon(nullptr, IDI_QUESTION);
 SendMessage(hwnd, STM_SETICON,
             reinterpret_cast<LPARAM>(hicoClient), 0);
 MSG msg;
 while (GetMessage(&msg, NULL, 0, 0)) {
  TranslateMessage(&msg);
  DispatchMessage(&msg);
 }
 DestroyIcon(hicoClient);
 DestroyIcon(hicoCaption);
 return 0;
}
```

We create a top-level static window, which is highly unusual, since static controls are nearly always children of some other window. I'm doing this specifically to show the two different icons. You don't want to do this in a real program.

The static control has the `SS_ICON` style, because we want it to display an icon, and the `SS_CENTERIMAGE` style, because we just want it to center the icon in its client area without resizing. (We will control the size.)

We subclass the window so that we can post a quit message to exit the program when the window is destroyed, which the user can do by pressing `Alt` + `F4`. (Hey, this is just a demo program. Catching clicks on the × button is just extra code that will distract from the purpose

of the demonstration. Heck, this entire subclass thing is already distracting from the purpose of the demonstration!)

We load up two icons, an exclamation point, which we set as our caption icon, and a question mark, which we put in our client area. (We could have used the `Static_SetIcon` macro in `windowsx.h` to send the `STM_SETICON` message, but I did it manually just to make the message explicit.)

Run the program, and there you can see the two different types of icons: The exclamation point goes in the caption, and the question mark goes in the client area.

Raymond Chen

Follow