How do I cancel autoplay from a wizard page?

devblogs.microsoft.com/oldnewthing/20161006-00

October 6, 2016

Raymond Chen

A customer wanted to suppress autoplay from their wizard. They looked at the documentation and followed the dialog procedure example in their own wizard page dialog procedure:

```
// ... in the dialog procedure ...
  default:
    if (g_uQueryCancelAutoPlay == 0) {
      g_uQueryCancelAutoPlay =
        RegisterWindowMessage(TEXT("QueryCancelAutoPlay"));
      if (ChangeWindowMessageFilterEx(hwndDlg,
                                      g_uQueryCancelAutoPlay,
                                      MSGFLT_ALLOW,
                                      NULL) == FALSE) {
        MessageBox(NULL, L"ChangeWindowMessageFilterEx failed",
                   L"error", MB_OK);
      }
    }
    if (uMsg && uMsg == g_uQueryCancelAutoPlay) {
      SetWindowLongPtr(hwndDlg, DWLP_MSGRESULT, TRUE);
      return TRUE;
    }
```

However, the code didn't seem to work. The `QueryCancelAutoPlay` message was never received. What's wrong with the code? It was copied almost directly from MSDN!

Well, one of the things that's wrong is that the code doesn't change the window message filter until after it receives the message, which creates a bit of a chicken-and-egg problem: You don't change the filter until after you get the message, but you won't get the message until you change the filter! That's easy to fix: Initialize the `g_uQueryCancelAutoPlay` variable and change the message filter immediately after creating the dialog box. That allows the `QueryCancelAutoPlay` message to come through when the system generates it.

But even with that fix, the message won't get through.

What's wrong with the code is that wizard dialog pages are not top-level windows. The `QueryCancelAutoPlay` message is sent to the foreground window, but wizard pages are child dialogs inside the outer wizard frame and therefore can never be the foreground window. Activation and foreground are a properties of top-level windows.

Since the `QueryCancelAutoPlay` message is sent to the foreground window, you need to listen for the message on the foreground window. The standard way of doing this is subclassing, and the cleanest way to do this is with a function like `SetWindowSubclass`:

```
LRESULT CALLBACK CancelAutoPlaySubclassProc(
    HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam,
    UINT_PTR uIdSubclass, DWORD_PTR dwRefData)
{
  if (uMsg == g_uQueryCancelAutoPlay) {
    return TRUE;
  }
  return DefSubclassProc(hwnd, uMsg, wParam, lParam);
}

// when you want to start blocking autoplay
SetWindowSubclass(hwndDlg, CancelAutoPlaySubclassProc, 0, 0);

// when you want to stop blocking autoplay
RemoveWindowSubclass(hwndDlg, CancelAutoPlaySubclassProc, 0);
```

I discussed the `SetWindowSubclass` family of functions some time ago.

There is a fine point here: You want to set the subclass only when you want to start blocking autoplay, and remove it when you want to allow it again. And you should block autoplay only when your page is the active page in the wizard. In other words, the earliest you should block autoplay is in your `PSN_SETACTIVE` notification handler, and the latest you should remove the block is in your `PSN_KILLACTIVE` notification handler.

If you mess up and block autoplay when your page is not the active page, then you will be blocking autoplay even when there is nothing in the wizard that needs to block autoplay. Consider:

- Page 1: "Welcome to the XYZ Wizard."
- Page 2: "Insert the CD."
- Page 3: "Please wait while we copy stuff from the CD."
- Page 4: "All done. This completes the XYZ Wizard."

Autoplay should be disabled only on page 2, because that's the only place the user expects autoplay to be blocked. When the user is on the welcome page, they haven't started anything yet, and inserting a CD should start playing music (if that's how the user configured the system). Similarly, when the user is on the "All done" page, the wizard is basically finished, and inserting a CD at that point should also start playing music.

Armed with this knowledge, you can answer this question from a customer:

> We have a scenario where we need a child page of an AeroWizard to know that a `WM_POWER-BROADCAST` message has arrived. What is the best way to get this message from the main window procedure to the child pages of the wizard? Also, how would I ensure that this message only gets forwarded to the page that is currently active?

Bonus chatter: I didn't realize it at the time the question is asked, but writing up this article just now, I noticed that the customer who asked the above question about the `WM_POWERBROADCAST` message is the exact same customer who asked the original question about the `QueryCancelAutoPlay` message! The questions were eight months apart.

Raymond Chen

Follow