If you want to receive a message that is broadcast to top-level windows, you need a top-level window

devblogs.microsoft.com/oldnewthing/20160101-00

January 1, 2016

Raymond Chen

A customer wanted to suppress autorun from a wizard page. They started with this page on MSDN but found that their wizard page did not receive the `QueryCancelAutoPlay` message.

```
default:
  if (g_uQueryCancelAutoPlay == 0) {
    g_uQueryCancelAutoPlay =
      RegisterWindowMessage(TEXT("QueryCancelAutoPlay"));
  }
  if (uMsg && uMsg == g_uQueryCancelAutoPlay) {
    SetWindowLongPtr(hwndDlg, DWLP_MSGRESULT, TRUE);
    return TRUE;
  }
  break;
```

The customer reported that their dialog procedure never received the `QueryCancelAuto-Play` message. They even called `ChangeWindowMessageFilterEx` to explicitly allow the `g_uQueryCancelAutoPlay` message to be received, but that didn't help.

The original code had other issues which distracted me from the actual problem. The source of the problem is hidden in their opening statement: They are trying to receive this message *in a wizard*, which means that this dialog procedure does not correspond to a top-level window. It is a child dialog inside the wizard.

Since broadcast messages go only to top-level windows, you need to have a top-level window that can receive the message. Creating your own top-level window will not work in this case, because `QueryCancelAutoPlay` is sent only to the foreground window. Therefore, you will have to subclass your existing top-level window so you can snoop in on the messages. You can find the top-level window by calling `GetAncestor(hwnd, GA_ROOT)`, and you can use `SetWindowSubclass` to intercept the messages.

You need to be careful about this, because you don’t want to reject autoplay unconditionally. You should reject autoplay only when your page is the active page. It would be bad if the “Please insert the DVD” page suppressed autorun even before the user reached that page, or after the used moved beyond that page all the way to “Congratulations. Everything is ready to go.”

Therefore, you need to listen for the `PSN_SETACTIVE` and `PSN_KILLACTIVE` notifications. When your page becomes active, you subclass the root window; when it is no longer active, you remove the subclass. Alternatively, you can set and clear a flag, and have the subclass function check the flag to see whether it should respond to the message or let it go through.

Okay, let’s do it. First, let’s have a three-page wizard that doesn’t try to do anything fancy with autoplay.

```
#include <windows.h>
#include <commctrl.h>

HINSTANCE g_hinst;

// For demonstration purposes only. In real life, of course,
// you would check if the correct DVD is inserted.

bool IsCorrectDVD()
{
  SYSTEMTIME st;
  GetSystemTime(&st);
  return st.wMinute % 2 == 0;
}

INT_PTR CALLBACK WelcomeDlgProc(
    HWND hdlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  switch (message) {
  case WM_NOTIFY:
    {
      LPNMHDR pnmh = (LPNMHDR)lParam;
      switch (pnmh->code) {
      case PSN_SETACTIVE:
        PropSheet_SetWizButtons(pnmh->hwndFrom, PSWIZB_NEXT);
        return TRUE;
      }
    }
    break;
  }
  return FALSE;
}

INT_PTR CALLBACK InsertDlgProc(
    HWND hdlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  switch (message) {
  case WM_NOTIFY:
    {
      LPNMHDR pnmh = (LPNMHDR)lParam;
      switch (pnmh->code) {
      case PSN_SETACTIVE:
        PropSheet_SetWizButtons(pnmh->hwndFrom, PSWIZB_BACK | PSWIZB_NEXT);
        return TRUE;
      case PSN_WIZNEXT:
        if (!IsCorrectDVD()) {
           MessageBox(hdlg, TEXT("Please insert the correct DVD."),
                      TEXT("Error"), MB_OK);
           SetWindowLongPtr(hdlg, DWLP_MSGRESULT, -1);
           return TRUE;
        }
        break;
      }
    }
    break;
  }
  return FALSE;
}

INT_PTR CALLBACK FinishedDlgProc(
    HWND hdlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  switch (message) {
  case WM_NOTIFY:
    {
      LPNMHDR pnmh = (LPNMHDR)lParam;
      switch (pnmh->code) {
      case PSN_SETACTIVE:
        PropSheet_SetWizButtons(pnmh->hwndFrom, PSWIZB_FINISH);
        return TRUE;
      }
    }
    break;
  }
  return FALSE;
}

HPROPSHEETPAGE CreateWizardPage(PCTSTR dialogTemplate, DLGPROC dlgProc)
{
  PROPSHEETPAGE psp = { sizeof(psp) };
  psp.hInstance = g_hinst;
  psp.lParam = 0;
  psp.dwFlags = PSP_DEFAULT;
  psp.pszTemplate = dialogTemplate;
  psp.pfnDlgProc = dlgProc;
  return CreatePropertySheetPage(&psp);
}

int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
  HPROPSHEETPAGE pages[3] = {
    CreateWizardPage(MAKEINTRESOURCE(1), WelcomeDlgProc),
    CreateWizardPage(MAKEINTRESOURCE(2), InsertDlgProc),
    CreateWizardPage(MAKEINTRESOURCE(3), FinishedDlgProc),
  };

  PROPSHEETHEADER psh = { sizeof(psh) };
  psh.hInstance = hinst;
  psh.dwFlags = PSH_WIZARD;
  psh.pszCaption = TEXT("Awesome Wizard");
  psh.phpage = pages;
  psh.nPages = 3;
  PropertySheet(&psh);
  return 0;
}

// scratch.rc
#include <windows.h>
#include <commctrl.h>

1 DIALOGEX 0, 0, WIZ_CXDLG, WIZ_CYDLG
STYLE DS_SHELLFONT | WS_CHILD
FONT 8, "MS Shell Dlg 2"
BEGIN
    LTEXT "Welcome.", -1, 0, 8, 216, 38
END

2 DIALOGEX 0, 0, WIZ_CXDLG, WIZ_CYDLG
STYLE DS_SHELLFONT | WS_CHILD
FONT 8, "MS Shell Dlg 2"
BEGIN
    LTEXT "Insert DVD.", -1, 0, 8, 216, 38
END

3 DIALOGEX 0, 0, WIZ_CXDLG, WIZ_CYDLG
STYLE DS_SHELLFONT | WS_CHILD
FONT 8, "MS Shell Dlg 2"
BEGIN
    LTEXT "Finished.", -1, 0, 8, 216, 38
END
```

So far, so boring. The Welcome page enables the Next button; the Insert page has both Back and Next buttons; the Finished page has a Finish button. If the user clicks Next on the Insert page, but the correct DVD is not inserted, then display an error message and stay on the page.

Okay, now the problem: If the user inserts the DVD when instructed, it will autoplay, so let's suppress autoplay while the Insert page is displayed.

```
LRESULT CALLBACK MainWindowSubclassProc(
    HWND hwnd,
    UINT message,
    WPARAM wParam,
    LPARAM lParam,
    UINT_PTR uIdSubclass,
    DWORD_PTR dwRefData)
{
  static UINT wmQueryCancelAutoPlay;

  if (wmQueryCancelAutoPlay == 0) {
    wmQueryCancelAutoPlay =
      RegisterWindowMessage(TEXT("QueryCancelAutoPlay"));
  }

  if (message && message == wmQueryCancelAutoPlay) {
    HWND hwndPropSheet = (HWND)dwRefData;
    PropSheet_PressButton(hwndPropSheet, PSBTN_NEXT);
    return TRUE;
  }
  return DefSubclassProc(hwnd, message, wParam, lParam);
}

INT_PTR CALLBACK InsertDlgProc(
    HWND hdlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  HWND hwndRoot;
  switch (message) {
  case WM_NOTIFY:
    {
      LPNMHDR pnmh = (LPNMHDR)lParam;
      switch (pnmh->code) {
      case PSN_SETACTIVE:
        PropSheet_SetWizButtons(pnmh->hwndFrom, PSWIZB_BACK | PSWIZB_NEXT);
        hwndRoot = GetAncestor(hdlg, GA_ROOT);
        SetWindowSubclass(hwndRoot, MainWindowSubclassProc, 0, (DWORD_PTR)pnmh-
>hwndFrom);
        SetWindowLongPtr(hdlg, DWLP_USER, (LPARAM)hwndRoot);
        return TRUE;
      case PSN_WIZNEXT:
        if (!IsCorrectDVD()) {
           MessageBox(hdlg, TEXT("Please insert the correct DVD."),
                      TEXT("Error"), MB_OK);
           SetWindowLongPtr(hdlg, DWLP_MSGRESULT, -1);
           return TRUE;
        }
        break;
      case PSN_KILLACTIVE:
        hwndRoot = (HWND)SetWindowLongPtr(hdlg, DWLP_USER, 0);
        RemoveWindowSubclass(hwndRoot, MainWindowSubclassProc, 0);
        break;
      }
    }
    break;
  }
  return FALSE;
}
```

Our subclass procedure receives the property sheet window as its reference data. It checks whether the message is `QueryCancelAutoPlay`. If so, then it presses the Next button in the property sheet and returns `TRUE` to cancel the autoplay.

Exercise: I thought you returned a value from a dialog procedure by using `SetWindow-LongPtr` with `DWLP_MSGRESULT`, but here, we're just returning `TRUE` directly. What's going on?

The dialog procedure for the page installs the subclass when the page is activated, and it removes the subclass when the page is deactivated. The window to be subclassed is the root window for the wizard. We save that in our `DWLP_USER` so we know which window to unsubclass when we lose activation. Note also that we pass the property sheet as the reference data for `SetWindowSubclass`, so that the subclass procedure has the correct window handle for the `PropSheet_PressButton` macro.

By the way, as a courtesy, the property sheet manager forwards the following messages to all pages:

- `WM_DISPLAYCHANGE`
- `WM_SETTINGSCHANGE`
- `WM_SYSCOLORCHANGE`

And it forwards the following messages to the active page:

- `WM_ACTIVATEAPP`
- `WM_ACTIVATE`
- `WM_DEVICECHANGE`
- `WM_ENABLE`
- `WM_ENDSESSION`
- `WM_PALETTECHANGED` (but only if the `wParam` is not the property sheet itself, to avoid an infinite palette realization loop)
- `WM_QUERYENDSESSION`
- `WM_QUERYNEWPALETTE`

But for other messages (like `QueryCancelAutoPlay`), you're on your own.

Armed with this knowledge, you can answer this question, which as it happens, arrived eight months later from the same customer!

> We have a scenario where we need some pages of a wizard to know that a `WM_POWERBROADCAST` message has arrived. What is the best way to pass this message to the wizard pages? Also, how would we ensure that this message only gets forwarded to the page that is currently active?

Raymond Chen

Follow