Why am I getting mojibake when I try to create a shell link?

devblogs.microsoft.com/oldnewthing/20190103-00

January 3, 2019

Raymond Chen

A customer couldn't get the `IShellLink` interface to work. They tried to set the shortcut target to a path, but it came out as Chinese mojibake.

Here's a reduction of their code to its simplest form.

```
HRESULT CreateLink()
{
  HRESULT hr;
  IShellLinkA* psl;

  hr = CoCreateInstance(CLSID_ShellLink, NULL, CLSCTX_INPROC_SERVER,
                        IID_IShellLink, (LPVOID*)&psl);
  if (SUCCEEDED(hr)) {
    IPersistFile* ppf;

    psl->SetPath("C:\\Windows"); // this comes out as mojibake

    hr = psl->QueryInterface(IID_IPersistFile, (LPVOID*)&ppf);
    if (SUCCEEDED(hr)) {
      hr = ppf->Save(L"C:\\Test\\Test.lnk", TRUE);
      ppf->Release();
    }
    psl->Release();
  }
  return hr;
}
```

(You can see that this customer used to be a C programmer, because all variable declarations are at the start of blocks. Also, because they aren't using RAII.)

The problem is hidden in the call to `CoCreateInstance`:

```
  hr = CoCreateInstance(CLSID_ShellLink, NULL, CLSCTX_INPROC_SERVER,
                        IID_IShellLink, (LPVOID*)&psl);
  //                    --------------  -------------
```

Observe that the requested interface is `IID_IShellLink`, but the result is placed into a pointer to `IShellLinkA`. This mismatch should raise a warning flag.

It appears that the program is being compiled with Unicode as the default character set, which means that `IID_IShellLink` is really `IID_IShellLinkW`. Consequently, the requested interface is `IShellLinkW`, and the result is placed into a pointer to `IShellLinkA`. As a result of this mismatch, the call to `psl->SetPath` thinks that it's calling `IShellLinkA::SetPath`, but in reality it is calling `IShellLinkW::SetPath`. (The `IShellLinkA` and `IShellLinkW` interfaces have the same methods in the same order, except that one uses ANSI strings and the other uses Unicode strings.)

That is where the mojibake is coming from. An ANSI string is passed where a Unicode string is expected.

Mismatches like this can be avoided by using the `IID_PPV_ARGS` macro. This macro looks at the type of the pointer you pass it and autogenerates the matching `REFIID`, as well as casting the pointer to `void**`.

```
hr = CoCreateInstance(CLSID_ShellLink, NULL, CLSCTX_INPROC_SERVER,
                      IID_PPV_ARGS(&psl));
```

While they're at it, the customer should consider abandoning the ANSI version altogether and just using the Unicode one.

Raymond Chen

Follow