How can I get the original target of a shortcut without applying any 32-bit adjustments?

devblogs.microsoft.com/oldnewthing/20230208-00

February 8, 2023

Raymond Chen

If 64-bit code creates a shortcut to, say, `C:\Program Files\Contoso\Contoso.exe`, and 32-bit code tries to read the target of that shortcut, the 32-bit code is told that the shortcut target is `C:\Program Files (x86)\Contoso\Contoso.exe`. What's going on, and how can I get the original path?

What's going on is that when you create a shortcut, the shortcut code says, "Oh, this path is relative to the current `%ProgramFiles%`. Let me remember that." Later, when you open the shortcut, the shortcut code says, "Oh, this is an environment variable-relative path, so I'll expand it relative to the current value of `%ProgramFiles%`."

Recording paths as relative to environment variables and special folders is handy when the shortcut moves to another machine with a different directory structure, or if the directory itself moves. For example, you might move your Documents folder to a new location. Tracking a shortcut target by its location relative to the `FOLDERID_Documents` special folder keeps those shortcuts working even after you've moved it.

This feature does get in the way sometimes, such as if you want to get the original path without all this "assistance".

We start with a program that prints the path stored in a shortcut.

```
#include <windows.h>
#include <winrt/base.h>
#include <shlobj.h>

int wmain(int argc, wchar_t** argv) try
{
    winrt::init_apartment(winrt::apartment_type::single_threaded);

    auto link = winrt::create_instance<IShellLinkW>(CLSID_ShellLink);
    winrt::check_hresult(link.as<IPersistFile>()->Load(argv[1], STGM_READ));

    wchar_t buffer[MAX_PATH];
    WIN32_FIND_DATA wfd;
    winrt::check_hresult(link->GetPath(buffer, MAX_PATH, &wfd, 0));
    printf("Path is %ls\n", buffer);
} catch (...) {
    printf("Error: %ls\n" winrt::to_message().c_str());
    return winrt::to_hresult();
}
```

This simple program takes the path to a shortcut file as its command line parameter and prints the target as a path. If you compile this as a 32-bit program and give it the path to a shortcut created from a 64-bit program, then paths that point into `C:\Program Files` are written as `C:\Program Files (x86)`.

Let's disable that "helpful" conversion.

```
int wmain(int argc, wchar_t** argv) try
{
    winrt::init_apartment(winrt::apartment_type::single_threaded);

    auto link = winrt::create_instance<IShellLinkW>(CLSID_ShellLink);
    winrt::check_hresult(link.as<IPersistFile>()->Load(argv[1], STGM_READ));

    DWORD flags;
    winrt::check_hresult(link.as<IShellLinkDataList>()->GetFlags(&flags));
    flags |= SLDF_DISABLE_KNOWNFOLDER_RELATIVE_TRACKING;
    winrt::check_hresult(link.as<IShellLinkDataList>()->SetFlags(flags));

    winrt::com_ptr<IStream> stm;
    winrt::check_hresult(CreateStreamOnHGlobal(nullptr, TRUE, stm.put()));
    winrt::check_hresult(link.as<IPersistStream>()->Save(stm.get(), true));
    winrt::check_hresult(stm->Seek({ 0 }, 0, nullptr));
    winrt::check_hresult(link.as<IPersistStream>()->Load(stm.get()));

    wchar_t buffer[MAX_PATH];
    WIN32_FIND_DATA wfd;
    winrt::check_hresult(link->GetPath(buffer, MAX_PATH, &wfd, 0));
    printf("Path is %ls\n", buffer);
} catch (...) {
    printf("Error: %ls\n" winrt::to_message().c_str());
    return winrt::to_hresult();
}
```

After loading the shortcut, we query for `IShellLinkDataList`, which gives us access to the flags that alter the shortcut behavior. In our case, we want to turn on “disable known-folder-relative tracking”.

Once we’ve updated the flags, we have to save and reload the shortcut for it to take effect. We don’t have to save it to disk, though. We save it to a memory stream, and then load it back from the same stream.

Once reloaded, we can ask for the path, and since known-folder-relative tracking is disabled, you get the original raw path.

Bonus chatter: If you want to create a shortcut with this flag set in the shortcut itself (rather than merely set in memory like we did here), then use `IShellLinkDataList` to set the “disable known-folder-relative tracking” flag before saving it.