A customer question about shortcuts that don't have a target path

devblogs.microsoft.com/oldnewthing/20160902-00

September 2, 2016

Raymond Chen

A customer had a question about the Minesweeper shortcut on Windows 7:

> We found something strange when we tried to retrieve target path information from the Minesweeper shortcut, specifically, the system built-in shortcut for Minesweeper application which is normally found in `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Games\Minesweeper.lnk`. When we tried to view the property of this shortcut file, we got the information below:
>
> | | |
> |---|---|
> | Target type: | Minesweeper |
> | Target location: | Games |
> | Target: | Minesweeper |
>
> On the other hand, if we manually create a shortcut to Minesweeper, the shortcut properties say
>
> | | |
> |---|---|
> | Target type: | Application |
> | Target location: | Minesweeper |
> | Target: | "C:\Program Files\Microsoft Games\Minesweeper\Minesweeper.exe" |
>
> We are unable to obtain the target path from the system built-in shortcut, but we can get it from the one we created manually.
>
> Can you explain the difference between the two shortcuts? How can we get the target path from both types of shortcuts?

As we saw some time ago, a shortcut can point to anything in the shell namespace, and the target path is meaningful only for shortcuts whose targets are, y'know, paths. if the target of the shortcut is not a file system object, then you can't get a path to it because those virtual objects *don't have paths*.

But it wasn't clear why the customer is trying to take these shortcuts apart. I asked, "Why is the customer obtaining the target path? Is this for some sort of software inventory tool?"

The customer liaison explained:

> This is an issue raised by my customer. The customer is retrieving the target path of a shortcut by using `IShellLink::GetPath()`, but they are unable to get the target path information from some shortcuts, of which `Minesweeper.lnk` is just an example. The customer wants to retrieve the path information for arbitrary shortcuts including those which point to third-party applications. I don't understand how the underlying mechanism works, so I don't know whether there is a workaround that would allow the customer to retrieve the path information from shortcuts like `Minesweeper.lnk`. Is there some other way to retrieve the target path information?

The customer liaison gave a long response that didn't actually answer the question.

With some prodding, we got an answer from the custom liaison: The customer is a laptop manufacturer, and their laptop keyboard has some dedicated function keys. They are writing software that lets the user assign shortcuts to those function keys, so that when the user presses the function key, the corresponding app launches. They want to get the target path of the shortcut so they know what program to launch.

Y'know, if all you want to do is launch the thing that a shortcut refers to, you can just launch the shortcut.

```
ShellExecute(hwnd, nullptr, "Minesweeper.lnk",
             nullptr, nullptr, SW_SHOWNORMAL);
```

This has the advantage of also respecting the command line arguments, the current directory, the hotkey, any application compatibility settings, console settings (if the target is a console application), and all the other information stored in the shortcut. It also means that if the target of the shortcut gets moved or renamed, the standard shortcut resolution code will kick in and try to find the target, wherever it ended up.

This is like designing an overhead console in an automobile for a garage door opener. One way to do it is to have the user bring the garage door opener to the car, and press a button that causes the car to take apart the garage door opener and analyze it to figure out the transmission frequency, extract the rolling code, and whatever other information you need in order to perfectly replicate the operation of the garage door opener.

Or you could design the overhead console so it had a compartment where the user can place the garage door opener, and when they want to open the garage door, they *push the button*.

The customer liaison replied that the customer accepted the workaround of calling `ShellExecute` and it is working fine.

I didn't bother pointing out that calling `ShellExecute` isn't the workaround. It's actually the standard technique. The thing about extracting the target path and manually executing the program: *That* is the thing that smells like a workaround.

Raymond Chen

Follow