Why does Windows have trouble finding my Win32 resource if it contains an accented character?

devblogs.microsoft.com/oldnewthing/20250430-00/?p=111129

April 30, 2025

Maurice Kayser reported an issue with Win32 API loading of PE resources containing lowercase letters. Maurice did some experiments adding resources named `MyIcon`, `MyIcÖn`, and `MyIcön`, then trying to load them using various names, and built up a table of results. I've broken it up into three tables depending on the nature of the accented character.

Arg	Can load `MyIcon`	Can load `MyIcÖn`	Can load `MyIcön`
`myicon`	Yes	No	No
`MyIcon`	Yes	No	No
`mYiCoN`	Yes	No	No
`MYICON`	Yes	No	No

This table shouldn't be surprising. The argument passed to `LoadResource` is compared case-insensitively with the name of the resource, treating accented characters as different from their unaccented versions.

Here's the next batch.

Arg	Can load `MyIcon`	Can load `MyIcÖn`	Can load `MyIcön`
`myicÖn`	No	Yes	No (!)
`MyIcÖn`	No	Yes	No (!)
`mYiCÖN`	No	Yes	No (!)
`MYICÖN`	No	Yes	No (!)

The first column is consistent with our previous result, namely that unaccented characters are treated as not the same as accented characters.

The second column is not surprising either, since the strings do match according to a case-insensitive comparison.

The third column is surprising. It seems that accented characters are case-sensitive, even though the documentation says that the comparison is case-insensitive.

Okay, here's the third block.

Arg	Can load `MyIcon`	Can load `MyIcÖn`	Can load `MyIcön`
`myicön`	No	Yes (?)	No (!)
`MyIcön`	No	Yes (?)	No (!)
`mYiCöN`	No	Yes (?)	No (!)
`MYICöN`	No	Yes (?)	No (!)

The PE specification says that the resources are sorted "in ascending order", and the names are sorted "by case-sensitive string."[1]

That's all it says. The rest is left to interpretation.

First of all, even though the file format specification says that the resource names can be in any case, the `FindResource` function converts all names to uppercase before searching, so any names with lowercase characters are effectively unfindable. Fortunately, the Resource Compiler also converts names to uppercase before storing them in the resources, so it all cancels out, right?

Well, it cancels out only if the Resource Compiler and the `FindResource` function agree on how the names are converted to uppercase.

The Resource Compiler uses `_wcsupr` to convert the names to uppercase, and `_wcsupr` uses the default C locale,[2] which as we noted before, is not a very interesting locale. It converts Latin unaccented lowercase letters a-z to Latin unaccented uppercase letters A-Z, and that's all.

Let's update the top row of the table by converting the names to uppercase according to the C locale.

Arg	Can load `MYICON`	Can load `MYICÖN`	Can load `MYICöN`

How does the `FindResource` function convert strings to uppercase? It uses the uppercase table corresponding to the system default language. It is almost certain that `Ö` and `ö` are uppercase and lowercase partners in the system default language. That means that the left columns are all effectively `MYICON` in the first table, and that they are all effectively `MYICÖN` in the second and third tables.

With these adjustments, the tables make more sense.

Arg	Loaded as	Can load `MyIcon` Stored as `MYICON`	Can load `MyIcÖn` Stored as `MYICÖN`	Can load `MyIcön` Stored as load `MYICöN`
`myicon`	`MYICON`	Yes	No	No
`MyIcon`				
`mYiCoN`				
`MYICON`				
`myicÖn`	`MYICÖN`	No	Yes	No
`MyIcÖn`				
`mYiCÖN`				
`MYICÖN`				
`myicön`				
`MyIcön`				
`mYiCöN`				
`MYICöN`				

Okay, so after we have accounted for how the Resource Compiler stores names and how `FindResource` searches for names, the table looks less bonkers.

The moral of the story, I think, is that you should just stick to ASCII characters for resource names. Everybody agrees on that subset.

[1] Note that the specification is incomplete: It doesn't say what collation to use for sorting. Does it use a locale-sensitive sort, so that `Ö` comes before `P` in German, but after `P` in Swedish?[3] Does it use a case-sensitive sort where all punctuation come before all alphabetics? The `FindResource` function assumes that the resources are sorted lexicographically by code unit (not code point) numerical value. Which is a good thing, because you don't want a file compiled on a German system to be considered corrupted by a Swedish system.

[2] But what about the `#pragme code_page()` directive? That directive tells the Resource Compiler how to convert quoted strings to Unicode, but it does not affect character mapping or collation.

[3] In German dictionary sorting, the letter `Ö` is sorted as if it had no accent mark. But in German phone book sorting, the letter `Ö` is sorted as if it were two characters `O` + `e`. And in Austrian phone book sorting, the letter `Ö` is sorted as if it were two characters `O` + ¨, where the ¨ is treated as a character that comes after `Z`. And in Swedish, the letter `Ö` is treated as one of the three accented characters that come after `Z`.