How do I perform a case-insensitive comparison of two strings in the Deseret script?

devblogs.microsoft.com/oldnewthing/20230913-00

September 13, 2023

Raymond Chen

A customer reported that they were having difficulty sorting strings in the Deseret script in a case-insensitive manner. The `CompareStringEx` function, when called with the `LOCALE_NAME_INVARIANT` locale and the `NORM_IGNORECASE` flag, reported the strings as unequal even if they differed only in case.

```
// U+10412 (DESERET CAPITAL LETTER BEE)
wchar_t Bee[] = L"\xD801\xDC12";

// U+1043A (DESERET SMALL LETTER BEE)
wchar_t bee[] = L"\xD801\xDC3A";

auto result = CompareStringEx(LOCALE_NAME_INVARIANT,
NORM_IGNORECASE, Bee, -1, bee, -1, NULL, NULL, 0));
```

The customer suspected that they were using the wrong locale, but they couldn't find a Deseret locale. Is there one?

No, Windows does not have a Deseret locale, and that means that there is no custom sorting information for the Deseret script, which means that the Windows locale system doesn't know that U+10412 and U+1043A are case variants.

The Unicode Common Locale Data Repository (CLDR) does have an entry for Deseret, known as en_Dsrt, but it is a seed locale, meaning that it contains only minimal data and consequently is of limited/questionable quality.

The locale team was curious about how the customer is using Deseret script, because the next step will vary depending on the use case.

If the customer has a real-world corpus of text in Deseret script that they are processing, then they should share some details with the Windows locale team so that they can better understand how Deseret script is being used by customers, which may affect prioritization of future work. In the meantime, the customer can use the International Components for

Unicode (ICU) library (now included with Windows) to collate strings encoded in Deseret script. For example, the `ucol_strcoll` function understands the case mapping rules for Deseret script.

If the customer merely noticed this discrepancy while running automated testing over the entire Unicode character set, then adding support in their program for text in the Deseret script may very well be unnecessary, since the usage case was artificial. If they have no organic use of Deseret script among their install base, it could be a significant architectural change to their program for no real-world benefit. In that case, they should just mark those tests as exceptions.

The team speculated that maybe the customer was doing genealogy or processing historical documents. (They ruled out the possibility that the customer was just tinkering around as a hobby, since those types of customers typically wouldn't bother their customer liaison about it, though sometimes they do it anyway.) My quick reading of the Wikipedia page for the Deseret alphabet suggests that the script underwent reform during its brief lifetime. If that reform included changes to collation (I don't know whether it did), then the customer will also have to take into account *when* the text was generated in order to collate it correctly.

We never heard back from the customer.