L2/11-176

Subject:	CLDR request for change in DUCET
From:	CLDR-TC
Date:	2011-05-10
Live document:	http://goo.gl/VPz3X

The CLDR-TC changed the default sorting in CLDR to be a tailoring of DUCET in the 1.9 release. This has been widely deployed, with no problems. The committee recommends that a PRI be issued for incorporating these changes into the next version of the DUCET (corresponding to Unicode 6.1).

There are two changes:

1. Symbols and punctuation characters are grouped instead of interspersed.
2. Special functions are given to two noncharacters.

There is one further change which should not be incorporated directly into DUCET, but could be reflected as an additional option in http://unicode.org/reports/tr10/#Variable_Weighting

3. Symbols being non-ignorable.

Much of the following is from: http://unicode.org/Public/UCA/6.0.0/CollationAuxiliary.html

Grouping Symbols and Punctuation

Here is the text from CollationAuxiliary:

Reordering of Common characters. The DUCET ordering puts characters into *roughly* the following ordering:
- First "common characters": whitespace, punctuation and general symbols, currency symbols, and numbers.
- Then "script characters": Latin, Greek, and the rest of the scripts.

The CLDR root locale tailored orders the common characters *strictly* by category:
- whitespace, punctuation, general symbols, currency symbols, and numbers.

The relative order within each of these groups still matches the DUCET. Symbols, punctuation, and numbers that are grouped with a particular script stay with that script. The only two exceptions are two currency symbols that are moved up to be with the other currency symbols:
- U+20A8 (Rs) RUPEE SIGN

- U+FDFC (﷼) RIAL SIGN

This regrouping only matters in comparison where a common character in one group is compared to a common character in another, such as if “I♥NY” were compared to “I-NY”, where a symbol is compared to a punctuation mark. What the regrouping allows is for users to parametrically reorder the groups. For example, users can reorder numbers after all scripts, or reorder Greek before Latin.

One of the primary reasons that applications and users wish to customize collation order—and one way in which locales also tailor collations—is to reorder entire categories of characters (such as punctuation, digits, and letters) with respect to other categories (for example, moving digits after letters). The above approach allows this to be done parametrically.

For example, the rules for Greek would be able to specify that the sorting order is:

  - punctuation < Greek letters < numbers < currency symbols < Latin letters < other scripts and characters.

While the same effect can be achieved with tailorings, that is very cumbersome, requiring excessive memory consumption (from multiple tables), or runtime checks that hurt performance.

The rearrangement involves a relatively small number of characters. They are all characters that users do not have solid expectations for, regarding sorting position. CLDR currently has an option for getting the ducet behavior, which shows the characters involved. See http://unicode.org/repos/cldr/trunk/common/collation/root.xml, under <collation type="ducet">.

Tailored noncharacter weights

The code point U+FFFF is tailored to have a weight higher than all other characters. This allows reliable specification of a range, such as “Sch” ≤ X ≤ “Sch\uFFFF” to include all strings starting with "sch" or equivalent.

The code point U+FFFE is tailored to have a weight lower than all other characters. This allows for Interleaved_Levels within code point space.

In CLDR, these values are not further tailorable, and nothing can tailor to them. That is, neither can occur in a collation rule: for example, the following rules are illegal:

```
& \uFFFF < x
& x <\uFFFF
```

Symbols non-variable

There are two options in the UCA for symbols and punctuation: *non-ignorable,* or *shifted*. With the *shifted* option, basically whitespace, symbols, and punctuation are ignored -- except at a fourth level. However, in a great many situations, it is important to be able to ignore spaces and punctuation, but not whitespace. CLDR does this in its root ordering, for example.

For this purpose it is useful to have another option in the algorithm:

ignoreSP for *ignore spaces & punctuation* (NOTE better names are welcome)