Pluralization on compact numbers

Robin Leroy (egg.robin.leroy@gmail.com)

Mark Davis (mark.edward.davis@gmail.com)

Overview

ICU and CLDR provide a way to display numbers in a (short or long) compact decimal format, e.g., 3 141 592 ↦ “3.1M”, “3.1 million”. However, there is no explicit support for pluralization on a number formatted in that way. That is, there may be grammatical differences in a message based on whether a number is written as 3.1M vs 3,100,000.

English tends to be simple, *e.g.*,

```
{number_of_views, plural,
 =0 {No views}
 one {# view}
 other {# views}}
```
↦ “3.1M views”, “3.1 million views”.

It turns out that naïvely using the displayed value to select the plural case (which is already nontrivial with ICU, see Annex B) fails in many Romance languages, including French, Italian, and Spanish.

As the use of compact number formatting in numeric placeholders has become more commonplace (see, for instance, view counts on the YouTube homepage), this issue has become more visible.

This document proposes a solution involving:

- A change in UTS #35, CLDR, and ICU to add a new plural operand c, which records suppressed exponents, eg for “2.3 M”, the value of c would be 6.
- An additional plural category in CLDR for the affected languages, using the new variable. Example for fr.xml:
  - `<pluralRule count="many">c = 0 and i % 1000000 = 0 and v = 0 or c != 0 .. 5</pluralRule>`
- An enhanced API in ICU to use in cases like compact number formatting. The code for ICU should be done before adding the operand/rules to CLDR, so that it can be used in the CLDR code for testing and examples.

ƵNote that there is pluralisation *within* the compact decimal format, so that the format is *1 million* but *2 millions* in French. This pluralization (with numeric values 1 and 2 in these examples) is irrelevant to, and should not be confused with, the issue at hand, which is pluralization *on* the compact number (e.g. “1 million views”).

ƵNote also that while the notation 1.2c6 resembles scientific notation, these pluralization rules do not apply to numbers in scientific notation, which is outside the scope of this proposal. For instance we have *1,2 × 10⁶ (*de) chats*, the new `many` case

would be incorrect. An earlier version of this document called the new operand e, and used the 1.2e6 to represent, e.g., “1.2 million”. This was changed to c in order to reduce the risk of confusion.

Notation

This document uses a prefix asterisk “*” to denote intentional *grammaticals errors used for demonstrative purposes, which can be combined with parentheses to indicate required or proscribed words:

— the Romans go *(to) the house — the “to” is required
— the Romans go (*to) home — the “to” is forbidden

Problem Cases

Consider the following French translation of the aforementioned view count message, `{number_of_views, plural, =0 {Aucune vue} one {# vue} other {# vues}}`.

A French speaker will immediately recognize that substituting the number 2 000 000 in compact long format (*2 millions*) in the `other` case `# vues`, yielding

**2 millions vues*,

is glaringly incorrect.

The correct form requires a preposition, *2 millions *(de) vues*. The same applies to Italian *2 milioni *(di) visualizzazioni*, and Spanish *2 millones *(de) vistas*, and likely to Portuguese *2 milhões *(de) visualizações* — there may be other languages with similar issues.

In fact, the compact short form also has this property, as it often is an abbreviation of the long form, *e.g.*, Italian *2 Mln *(di) visualizzazioni*, Catalan *2M *(de) visualitzacions*, etc.

The same applies in French to the word for 1 000 000 000, and higher powers of 1 000, so that, at first glance, this is easily solved within the current framework of UTS #35, Part 3, Section 5, by adding a new plural case with the following rule to affected Romance languages:

```
<pluralRule count="many">n mod 1000000 = 0</pluralRule>
```

Indeed, French requires the preposition even when the number is written in digits, *1 000 000 000 *(de) vues*. This is somewhat similar to the situation of the Romanian plural rules.

The above all works within the framework of the existing standards. Unfortunately, things get trickier with fractional millions.

Indeed, we have:

— 2,*3 millions *(de) vues* (read *deux virgule trois millions de vues*, the preposition is required), but
— *2 300 000 (*de) vues* (read *deux millions trois-cent-mille vues*, the preposition is proscribed).

For non-integer numbers in non-compact decimal form, the value does not solely depend on the displayed numeric value (plural operands n, i, f, w), and quantities derived from insignificant 0s (plural operands v, t) are needed for plural case selection. Similarly, it follows from the above example that numbers in compact decimal form require information beyond the numeric value being displayed.

Unicode Technical Standard #35, revision 51

Plural rules currently support only pluralization on numbers of the form ⟨digits⟩[,⟨digits⟩], with quantities n, i, v, w, f, and t (*plural operands*) derived from that form being used in the plural rules; see UTS #35 Part 3, Section 5.

Exponent

The compact decimal formats make this more complicated; indeed, a number in compact decimal format is not in the form ⟨digits⟩[,⟨digits⟩], even disregarding grouping marks.

To solve the problem, a new variable is proposed for the plural category rules.

Let c be the exponent of the power of 10 expressed by the compact decimal format, *i.e.*, 3 for “thousands”, 6 for “millions”, *etc.*, or 0 if a non-compact format is used (the introduction of this plural operand would be an amendment to UTS #35 Part 3, Section 5.1.1 “Operands”).

The French rule for the new `many` case would then be

```
<pluralRule count="many">c = 0 and i % 1000000 = 0 and v = 0 or
                         c != 0 .. 5</pluralRule>
```

and the French translation of the message would be

```
{number_of_views, plural,
 =0 {Aucune vue}
 one {# vue}
 many {# de vues}
 other {# vues}}
```

with similar rules for the other affected Romance languages.

The following table summarizes the values of the relevant plural operands for some interesting formatted numbers. Since we are considering only integers, *i*=*n*, and *v*, *w*, *f*, and *t* are 0.

Formatted number	*n*	*c*
1 284 043	1 284 043	0
1 200 000	1 200 000	0
1,2 million	1 200 000	6
1,2 M	1 200 000	6
1 000 000	1 000 000	0
1 million	1 000 000	6
1 M	1 000 000	6

Annex A: Outline of changes and usage

We will now summarize the proposed changes and their effect on the standard, on the libraries, and to the data per language. Additions are underlined and deletions ~~struck through~~.

Unicode Technical Standard #35, Part 3, Section 5.1

In the first paragraph,

> The xml value for each pluralRule is a condition with a boolean result that specifies whether that rule (i.e. that plural form) applies to a given numeric value n, where n can be expressed as a decimal fraction or with compact decimal formatting, denoted by scientific notation in the syntax, e.g., “1.2c6” for “1.2M”. Clients of CLDR may express all the rules for a locale using the following syntax:

In the syntax,

```
samples = ('@integer' sampleList)?
          ('@decimal' sampleList)?
          ('@compact' sampleList)?

operand       = 'n' | 'i' | 'f' | 't' | 'v' | 'w' | 'c'

decimalValue  = value ('.' value)?('c' value)?
```

In 5.1.1 “Operands”, in the table “Plural Operand Meanings”, amend the “n” row and append a “c” row, as follows:

> **Symbol** **Value**

n	absolute value of the source number (integer, ~~and~~ decimals, and exponent in compact decimal format).
⋮	⋮
c	exponent of the power of 10 used in compact decimal formatting.

In 5.1.1 “Operands”, in the table “Plural Operand Examples”, add a “c” column with 0s in the existing rows, and add three rows as follows.

n	i	v	w	f	t	c
⋮	⋮	⋮	⋮	⋮	⋮	0
1200000	1200000	0	0	0	0	0
1.2c6	1200000	0	0	0	0	6
123c6	123000000	0	0	0	0	6

In 5.1.2 “Relations”,

> The values of relations are defined according to the operand as follows. Importantly, the results may depend on the visible decimals in the source, including trailing zeros, or on the compact decimal exponent.

In 5.1.3 “Samples”,

> Samples are provided if sample indicator (@integer, @compact, or @decimal) is present on any rule. (CLDR always provides samples.)
>
> Where samples are provided, the absence of one of the sample indicators indicates that no numeric values can satisfy that rule. For example, the rule "i = 1 and v = 0" can only have integer samples, so @decimal must not occur; if c=0 then there can be no compact samples, so @compact must not occur.
>
> The sampleRanges have a special notation: **start~end**. The **start** and **end** values must have the same number of decimal digits; if they are compact values, they must have the same exponent. The range encompasses all and only values those value v where **start** ≤ **v** ≤ **end**, and where **v** has the same number of decimal places as **start** and **end**.
>
> Samples must indicate whether they are infinite or not. The '…' marker must be present if and only infinitely many values (integer or decimal) can satisfy the rule. If a set is not infinite, it must list all the possible values.

Rules	**Comments**
@integer 1, 3~5	1, 3, 4, 5.

@integer 3~5, 103~105, …	Infinite set: 3, 4, 5, 103, 104, 105, …
@decimal 1.3~1.5, 1.03~1.05, …	Infinite set: 1.3, 1.4, 1.5, 1.03, 1.04, 1.05, …
@compact 1c6~3c6, 1.0c6~1.2c6, …	Infinite set: 1c6, 2c6, 3c6, 1.0c6, 1.1c6, 1.2c6, …

In determining whether a set of samples is infinite, leading zero integer digits, compact decimal exponent, and trailing zero decimals are not significant. Thus "i = 1000 and f = 0" is satisfied by 01000, 1000, 1000.0, 1000.00, 1000.000, 1c3, etc. but is still considered finite.

ICU changes

We need to have a *Datatype* that we can use to pass a number for formatting and selection, one that represents something logically like 123.450c9, with (a) specified digits, (b) where the decimal goes, and (c) what the exponent is.

The best choice for how to express this will be decided via consultation with Shane and other interested parties on the ICU team. This could either take the form of an enhanced FormattedNumber (but with one or more constructors to pass in the exponent), or by introducing a `DecimalFloat`, a type equivalent to a `BigDecimal` together with an integer exponent, i.e., a representation of the `decimalValue` as amended above, with an exponent in additional to the integer and fractional parts.

Introduce `FormattedNumber::displayedValue`, the *Datatype* corresponding to the value of `FormattedNumber::toString`, such that we have the following correspondence:

`toString()`	`displayedValue()`
1,284,043	1284043
1,200,000	1200000
1.2 million	1.2c6
11.2M	11.2c6
1,000,000	1000000
1 million	1c6
1 thousand	1c3
1M	1c6
234.5 million	234.5c6

If we choose to use a new Datatype, add PluralRules::select(DecimalFloat), a pluralization selection function that implements plural selection as amended (with c rules). Otherwise, make the PluralRules::select(FormattedNumber) handle the c case.

We can now consider examples of the result of pluralized message formatting using `plural_rules.select(formatted_number.displayedValue())` for case selection on the new plural rules, in some relevant languages.

CLDR changes and example usage

English

Plural rules are unchanged.

Message:

```
{number_of_views, plural, =0 {No views} one {# view} other {# views}}
```

Plural case selection and message formatting:

Formatted number	Plural case	Reason	Message
1,284,043	other	$i \neq 1$	1,284,043 views
1,200,000	other	$i \neq 1$	1,200,000 views
1.2 million	other	$i \neq 1$	1.2 million views
1.2M	other	$i \neq 1$	1.2M views
1,000,000	other	$i \neq 1$	1,000,000 views
1 million	other	$i \neq 1$	1 million views
1 thousand	other	$i \neq 1$	1 thousand views
1M	other	$i \neq 1$	1M views
234.5 million	other	$i \neq 1$	234.5M views

French

Plural rules:

```
<pluralRule count="one">
  i = 0,1
  @integer 0, 1
  @decimal 0.0~1.5
</pluralRule>
<pluralRule count="many">
  c = 0 and i % 1000000 = 0 and v = 0 or c != 0 .. 5
  @integer 1000000, 2000000, 10000000, 100000000, 1000000000, …
  @compact 1c6~9c6, 1.0c6~1.9c6, 1c9~9c9, …
</pluralRule>
```

```
<pluralRule count="other">
  @integer 2~17, 100, 1000, 10000, 100000, 1000000, …
  @decimal 2.0~3.5, 10.0, 100.0, 1000.0, 10000.0, 100000.0, 1000000.0, …
  @compact 1c3~9c3, …
</pluralRule>
```

Message:

```
{number_of_views, plural,
 =0 {Aucune vue}
 one {# vue}
 many {# de vues}
 other {# vues}}
```

Plural case selection and message formatting:

Formatted number	Plural case	Reason	Message
1,284,043	other	$i \notin \{0,1\}$; $i \bmod 1\,000\,000 \neq 0$ and $c \in [0,5]$	*1 284 043 vues*
1 200 000	other	$i \notin \{0,1\}$; $i \bmod 1\,000\,000 \neq 0$ and $c \in [0,5]$	*1 200 000 vues*
1,2 million	many	$i \notin \{0,1\}$; $c \notin [0,5]$	*1,2 million de vues*
1,2 M	many	$i \notin \{0,1\}$; $c \notin [0,5]$	*1,2M de vues*
1 000 000	many	$i \notin \{0,1\}$; $c = i \bmod 1\,000\,000 = v = 0$	*1 000 000 de vues*
1 million	many	$i \notin \{0,1\}$; $c \notin [0,5]$	*1 million de vues*
mille[1]	other	$i \notin \{0,1\}$; $i \bmod 1\,000\,000 \neq 0$ and $c \in [0,5]$	*mille vues*
1 M	many	$i \notin \{0,1\}$; $c \notin [0,5]$	*1 M de vues*
234,5 millions	many	$i \notin \{0,1\}$; $c \notin [0,5]$	*234.5M de vues*

Italian, Spanish (Europe/Latin America), Portuguese (Europe/Latin America), Catalan, …

Once we have the above working for French, we will generate a questionaire for related languages to see what behavior they have. For Italian, for example, we will figure out whether the new case applies to 1 000 000 in digits in all the other languages; this question aside, this is similar to French, with the *mille* issue being

[1] This requires that https://unicode.org/cldr/trac/ticket/11045 (Google issue b/116777167) be fixed, the reason being that the current compact-long format for the one case, *1 millier*, differs from the way 1 000 would normally be read, *mille*, and they require different plural forms: *1 000 (*de) vues*, but *1 millier *(de) vues*.

even worse there, as the current **1 mille* is outright wrong in Italian (there is no equivalent of *millier*).

Slavic and other plural rules

For practical purposes, the existing rules should work fine in slavic languages: 1 200 has the same plural case as 1 000, and thus pluralizing *1,2 тыс* as 1 200 is correct. In theory however, one could use the compact decimal format with four significant digits, thus *1,243 тыс* for 1 243; the pluralization would then be incorrect in the absence of an c rule, the correct case being `few` for 1 243, but `many` for 1.243c3.

In order to avoid issues of that kind (and possibly likelier ones that we have not thought of), it may be best to add a “default” c rule to all languages

```
<pluralRule count="appropriate_case">
  c != 0
</pluralRule>
```

Where `appropriate_case` is the current plural case for the powers of 10 used by compact decimal formatting (if the plural case differs between the relevant powers of 10 for some language, that language likely needs more specialized c rules).

This effectively means ‘treat “12.34 million” as “1 million” for the purposes of pluralization’.

Annex B: displayed value in current ICU

In the foregoing treatment of pluralization on compact number, our baseline was pluralization based on the displayed value, *e.g.*, pluralization based on 1 200 000 when the number 1 284 001 is shown as “1.2M”, regardless of the hidden digits.

An even more naïve approach would be to choose the plural case based on the (unrounded) underlying value, *i.e.*, to pluralize on 1 284 001 when showing “1.2M”.

Rosa, rosa, rosam…

Cursory knowledge of pluralization in Slavic languages is sufficient to realize that this approach is flawed. For instance, in Russian, the plural case is[2] `few` for 1 284 043, it is `one` for 1 284 001, and it is `many` for 1 284 000, yet all three get formatted as[3] *1,2 млн*, for which the correct plural case is `many`.

With plural selection on the unrounded number, this leads to effectively arbitrary selection depending on invisible digits.

This issue occurred[4] at YouTube, whose view count display would sample the declension of the word “view” seemingly at random, haphazardly showing *просмотров*, *просмотр*, and *просмотра*, as can be seen on a screenshot from March 2018 of the homepage in Russian.

ICU support

Unfortunately, ICU provides poor support for obtaining the value 1 200 000 when formatting a number as *1,2 млн*.

In ICU4J, `FormattedNumber#toBigDecimal()` yields the `BigDecimal m` = 1.2 when an underlying value `k` gets formatted as *1,2 млн*, from which 1 200 000 can be obtained as `m.movePointRight(floor(log10(k/m)))`. there is no equivalent of `toBigDecimal` in ICU4C. In C++, the easiest way to compute the displayed value 1 200 000 in this situation ends up being, *horresco referens*, counting the number of code points with `General_Category Nd`[5] in the string formatted by `CompactDecimalFormat`.

The function `FormattedNumber::displayedValue` proposed in Annex A provides a superset of this functionality, obviating the need for digit-counting.

---

[2] www.unicode.org/cldr/charts/latest/supplemental/language_plural_rules.html#ru.

[3] www.unicode.org/cldr/charts/34/verify/numbers/ru.html.

[4] YouTube has since worked around the issue; the homepage now consistently shows *просмотров* for view counts with compact numbers.

[5] `Decimal_Number`, see unicode.org/reports/tr44/#General_Category_Values.

Limitations

This solution does not purport to handle all potential issues regarding compact number formatting.

Applicability to the compact short format

This approach continues to assume that the plural case of the compact short “1.2M” is the same as that of the compact long “1.2 million” in all languages.

In many cases, the compact short format is an abbreviated version of the compact long, *e.g.*, Russian *тыс.* for *тысяча*, French *M* and *Md* for *million* and *milliard*, Italian *Mln* and *Mrd* (formerly *Mld*) for *millione* and *milliardi*, *etc*. In those cases, abbreviation does not alter pluralization, *i.e.*, since we have *1 millione *(di) visualizzazioni*, we also have *1 Mln *(di) visualizzazioni*, so that no special treatment is needed for the compact short format.

Within the French, Italian, and Spanish data, only the French compact short thousand abbreviation (the SI prefix k, oddly not used as a prefix) fails to fit the “abbreviated long” pattern; since thousands do not use the new plural case anyway, this is not a concern.

Context-dependent formatting

This approach also does not (purport to) solve the inflection issue. Since the problem that we are trying to solve is placeholder substitution (with plural case selection) in a string that supports both a compact and a non-compact number, the formatted number does not depend on the surrounding message.

In other words, this approach would not work in a language and message where the form of the word “million” changes when inserted into the message. This is not the case in the aforementioned Romance languages.

In a language with declension, and in a message where the number does not occur in the nominative, *e.g.*,
`{number_of_cats, plural, one {I see a cat} other {I see # cats}}`,
this approach would fail in Russian with the compact-long form, *Я вижу тысячу кошек*, with the accusative *тысячу* rather than the nominative *тысяча*.

However, the compact short form (being abbreviated) does not generally change with case, and even where the compact long form is used, most usages are in the nominative (e.g., “50 million views”, rather than “share with 50 million people”).

We consider declension of compact numbers to be out of scope.