UTC #187 properties feedback & recommendations

Markus Scherer & Josh Hadley / Unicode properties & algorithms group, 2026-apr-16

Participants

The following people have contributed to this document:

Markus Scherer (chair), Josh Hadley (vice chair), Amrit Kaur, Asmus Freytag, Charlotte Buff, John Wilcock, Ken Whistler, Manish Goregaokar, Mark Davis, Ned Holbrook, Peter Constable, Robin Leroy, Roozbeh Pournader

1. Core Spec

1.1 L2/26-070 over-restricts shorthand format controls (PRI-540) [#540]

Recommended UTC actions

1. **No Action**: PAG has posted a revised document L2/26-070R, removing the definition and most of the discussion of misplaced default ignorable code points. PAG intends to revisit this topic.

Feedback (verbatim)

PRI-540 (general feedback)

Date/Time: Tue Mar 17 21:44:21 PT 2026
ReportID: ID20260317214421
Name: David Corbett
Report Type: Report Error in Publication/Data
Opt Subject: L2/26-070 over-restricts shorthand format controls

Two of L2/26-070's proposals for "Misplaced Default Ignorable Code Point" are too restrictive for Duployan.

Criterion 3 flags consecutive identical non-tag default ignorable code points. However, Duployan uses sequences of U+1BCA0 SHORTHAND FORMAT LETTER OVERLAP and/or U+1BCA1 SHORTHAND FORMAT CONTINUING OVERLAP to represent multiple letters overlapping one letter. Two consecutive overlap controls can be identical.

Criterion 6 requires shorthand format controls to be adjacent to Duployan. However, there can be three or more overlap controls in a row, in which case the middle controls are not adjacent to Duployan. Also, a Duployan character with a common-script combining mark may precede a shorthand format control, in which case the control is not adjacent to the Duployan character. Also, non-Duployan base characters like U+003D EQUALS SIGN can participate in Duployan overlap sequences.

The solution is for shorthand format controls to not be default ignorable. This would address the security issue without breaking Duployan. Compare the Egyptian hieroglyph format controls, which are not default ignorable.

Background information / discussion

PAG has discussed this area some more.
We plan to address other default ignorable code points in the Unicode 18 core spec, especially including tag characters, but have agreed in the meantime to refocus L2/26-070 on Variation Selectors.

2. UCD

2.1 Inconsistent delimiter for UAX #57 kEH_FVal [#437]

Recommended UTC actions

1. **Action Item** for Michel Suignard, SAH: In Unicode Standard Annex #57, change the delimiter of kEH_FVal from “/ or | (see description)” to “|”. For Unicode Version 18.0. See L2/26-096 item 2.1.

PAG input

From John Wilcock, PAG

kEH_FVal is using | (with leading and trailing spaces) and / (without leading and trailing spaces) as delimiters for multi-value attributes.

1. Is there a difference between the two delimiters?
2. If not, can we pick one or the other?
3. Can we also be consistent on the leading/trailing spaces around the chosen character?

Background information / discussion

Clearly, `/` is not actually being used as a machine-readable delimiter: Unikemet.txt has entries such as

```
None
U+132E8	kEH_FVal	ḫtm.w/y
```

for S19 □, and a look at the relevant Thot Sign List entry https://thotsignlist.org/mysign?id=5622 confirms that the corresponding phonetic values are meant to be ḫtm.w and ḫtm.ty.

`|` does appear to be used as an actual delimiter between fully independent alternatives, even if those alternatives may themselves describe multiple values in a human-readable way, as ḫtm.w/y above, or as in

```
None
U+133D4	kEH_FVal	t | sn, fḳ□
```

cf. https://thotsignlist.org/mysign?id=6695, this is either a logogram for *t*, or a phono-repeater for *sn* or *fḳ*□, as indeed described by

```
None
U+133D4	kEH_Func	Classifier bread, festival; Phonemogram t; Logogram t
(bread); Phono-repeater: sn, fḳ□
```

Here the lower-level human-readable delimiter is `,`, but like `/` it should not be listed as a delimiter for this property.

This is similar to the case of `;` in kDefinition, on which see UTC-178-A35, UTC-179-A12, and documents and feedback referenced therein.

2.2 DoNotEmit for ⧵ [#521]

Recommended UTC actions

1. **Action Item** for Roozbeh Pournader, PAG: Add DoNotEmit sequences <U+221E, U+20D2> for U+29DE ⧵ INFINITY NEGATED WITH VERTICAL BAR, <U+1DB11, U+20D2> for U+1DB12 LEIBNIZIAN CONGRUENCE WITH VERTICAL BAR, and <U+223E, U+20D2> for U+1DB1B LEIBNIZIAN DISSIMILARITY, with type = Precomposed_Form. For Unicode Version 18.0. See L2/26-096 item 2.2.
2. **Action Item** for Ken Whistler, EDC: In the Core Specification, document the DoNotEmit sequences <U+221E, U+20D2> for U+29DE ⧵ INFINITY NEGATED WITH VERTICAL BAR, <U+1DB11, U+20D2> for U+1DB12 LEIBNIZIAN CONGRUENCE WITH VERTICAL BAR, and <U+223E, U+20D2> for U+1DB1B LEIBNIZIAN DISSIMILARITY. For Unicode Version 18.0. See L2/26-096 item 2.2.

PAG input

From Robin Leroy, PAG: While looking at the Leibnizian relations, the proposal for which mentions ⧵ *diversitas seu confossa obelo identitas* (WG2-N5334R2 p. 40), I noticed that ⧵ was a ∞ negated with a vertical bar, but contrary to most negations with vertical bar (see, e.g., https://www.unicode.org/reports/tr25/tr25-16d2.html#table_negated_relations), is not represented by a sequence with U+20D2.

It seems that a DoNotEmit.txt entry is in order.

The same would apply to two of the characters proposed in that document and accepted by decision UTC-186-C33, to wit, LEIBNIZIAN CONGRUENCE WITH VERTICAL BAR (cf. LEIBNIZIAN CONGRUENCE ibidem) and LEIBNIZIAN DISSIMILARITY (cf. U+223E ∾ INVERTED LAZY S).

2.3 Proto-cuneiform: suspect wrong data [#524]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The properties are assigned as intended, as proposed, and as documented.

Feedback (verbatim)

Date/Time: Mon Feb 9 06:36:22 PT 2026
ReportID: ID20260209063622
Name: Mikhail Merkuryev
Report Type: Report Error in Publication/Data
Opt Subject: Proto-cuneiform: suspect wrong data

Chars 12550…125A7 are Xsux (cuneiform)

12A58…1264B are Pcun (proto-cuneiform)

1264C…12686 are Xsux again?

Are you sure what you are doing? Shouldn't they be all Pcun?

Background information / discussion

These property assignments are correct; we previously reported on them to the UTC in L2/24-224, p. 13; the properties are as proposed in L2/24-210R, for the reasons given in that document.

Specifically:

- 12550..125A7 comprise, as described in the Δ charts:
  - *Common numerals*: the N01 series, the N08 series, the N14 series, the N34 series, the N48 series, the N45 series, and the N50 series;
  - *Numerals used for land areas*: land area signs specific to the Early Dynastic (ED) period, as well as the N22 series
  - *Early Dynastic capacity measures*;
  - *Early Dynastic weight fractions*.
- These are all used in the third millennium, including in the Early Dynastic III period, with the rest of Cuneiform (Xsux) script (including characters from both the main Cuneiform block, and the Early Dynastic Cuneiform block which has characters specific to the ED III period). Note that in scholarship pertaining to that period, the N-series of numerals would instead be transliterated similarly to the existing cuneiform numerals, e.g., in ATF, `n(aš@c)` for the N01 series, cf. `n(aš)` for the already-encoded AŠ series (□–□). Crossreferences and informative aliases in the names list document these conventions.
- 1264C…12686 are used in the Early Dynastic I–II period (earlier in the third millennium), as noted in the names list; as noted in L2/25-211 §4.1, we also consider the script of that period to be Xsux, not Pcun, for the purposes of encoding.
- In contrast, 12A58..1264B are *only* used in the fourth millennium, together with the Proto-Cuneiform (Pcun) script.

Those characters that are used in both the fourth and third millennium have Script=Xsux and Script_Extensions=Pcun Xsux; the rationale for that is given on p. 3 of L2/24-210R.

2.4 Diacritic inconsistencies for new 18.0 Arabic Quranic marks [#525]

Recommended UTC actions

1. **Consensus**: Assign the Diacritic property to U+08D3 ARABIC SMALL LOW WAW, U+0656 ARABIC SUBSCRIPT ALEF, and U+06E2 ARABIC SMALL HIGH MEEM ISOLATED FORM. For Unicode Version 18.0. See L2/26-096 item 2.4.
2. **Action Item** for Robin Leroy, PAG: In UCD file PropList.txt, assign the Diacritic property to U+08D3 ARABIC SMALL LOW WAW, U+0656 ARABIC SUBSCRIPT ALEF, and U+06E2 ARABIC SMALL HIGH MEEM ISOLATED FORM. For Unicode Version 18.0. See L2/26-096 item 2.4.
3. **Action Item** for Robin Leroy, PAG: In UCD file PropList.txt, assign the Diacritic property to the following characters added in Unicode 18.0: U+10ECC and U+10ECD (arrowheads below), U+10EF3..U+10EF6 (small high and low noon with fatha and damma), U+10EF7 ARABIC SMALL HIGH HEH INITIAL FORM, and U+10EF9 ARABIC MARK CROWN. For Unicode Version 18.0. See L2/26-096 item 2.4.

PAG input

While Ken Whistler was preparing collation data for the 18.0α répertoire, he noticed some inconsistencies in the Diacritic assignments for the new Arabic marks.

These assignments were made based on comparanda suggested by Roozbeh Pournader, but it is well-known that the Diacritic assignments of existing Arabic marks are defective; indeed L2/26-006, p. 9, states that

> The Diacritic and Alphabetic properties have been assigned according to the above correspondences. However, the assignments of the existing characters are currently under review; when they get updated, the new characters will likely follow according to these correspondences.

Since these Diacritic assignments may matter for collation of characters new in 18.0, now is a good time to revisit them.

Recall the definition of the Diacritic property:

> Characters that linguistically modify the meaning of another character to which they apply. Some diacritics are not combining characters, and some combining characters are not diacritics. Typical examples include accent marks, tone marks or letters, and phonetic modifier letters. The Diacritic property is used in tooling which assigns default primary weights for characters, for generation of the DUCET table used by the Unicode Collation Algorithm (UCA).

U+10EF9 functions as an uppercasing modifier on some letters; in collation, those crown letters that are encoded atomically are equivalent to their non-crown counterparts followed by U+10EF9. This is a diacritic.

The other new characters, their pre-existing comparanda, and their documented functions are as follows. The proposed diacritic assignments follow from the documented functions.

New character	Comparandum	Diacritic assignment of both in 18.0α	Proposed	Names list comment of new character
ARABIC NORTHEAST POINTING ARROWHEAD ABOVE	ARABIC INVERTED DAMMA	Yes	Yes (no change)	used to mark changing of vowel to fatha or damma when starting recitation from the word
ARABIC NORTHEAST POINTING ARROWHEAD BELOW	ARABIC SUBSCRIPT ALEF	No	Yes	used to mark changing of vowel to kasra when starting recitation from the word
ARABIC SOUTHWEST POINTING ARROWHEAD BELOW	ARABIC SUBSCRIPT ALEF	No	Yes	used to mark changing of vowel to kasra when pausing recitation from the word
ARABIC SMALL HIGH NOON WITH FATHA	ARABIC SMALL HIGH MEEM ISOLATED FORM	No	Yes	These characters are used to mark naql, when there is a hamza after tanween, according to Warsh narration, the vowel of the hamza is moved to tanween so that it is pronounced as the letter noon
ARABIC SMALL LOW NOON WITH FATHA	ARABIC SMALL LOW WAW	No	Yes	〃
ARABIC SMALL HIGH	ARABIC SMALL HIGH	No	Yes	〃

NOON WITH DAMMA	MEEM ISOLATED FORM			
ARABIC SMALL LOW NOON WITH DAMMA	ARABIC SMALL LOW WAW	No	Yes	〃
ARABIC SMALL HIGH HEH INITIAL FORM	ARABIC SMALL HIGH MEEM ISOLATED FORM	No	Yes	used to indicate the end of the word should be pronounced with /h/ sound when pausing recitation
ARABIC SMALL HIGH WORD KABBIR	ARABIC SMALL HIGH WORD QIF	No	No (no change)	used to mark place of takbeer at the end of the chapter

In order to maintain the correspondences, this entails assigning the Diacritic property to the following four pre-existing characters: U+08D3 ARABIC SMALL LOW WAW, U+0656 ARABIC SUBSCRIPT ALEF, and U+06E2 ARABIC SMALL HIGH MEEM ISOLATED FORM.

Further, we can confirm that no change to the assignment Diacritic=No is warranted for the existing ARABIC SMALL HIGH WORD QIF, nor to the assignment Diacritic=Yes for the existing ARABIC INVERTED DAMMA.

This is justified by their functions, which can be reconstructed from documentation in the UTC register as follows.

On ARABIC SMALL LOW WAW,

much ink has flowed (L2/15-329, L2/16-044, L2/16-056, L2/16-100, L2/16-102, L2/16-153, L2/16-268). I found this in L2/16-102:

> **Kamal Mansour:** Functionally, this proposed SMALL WAW BELOW (08C7) appears identical to the existing 08F3 SMALL WAW ABOVE (recte: ARABIC SMALL HIGH WAW). One would need to compare some verses of a Libyan edition to those of a Cairo edition to determine whether the proposed character is simply an alternative graphic shape.
> **Lorna Evans:** I suspect #10 should be encoded. When we proposed the tone marks for Rohingya I argued that the above and below mark was one tone and it just depended on whether the tone followed an above or below vowel marking. I was told that kind of behavior would be too hard to

> implement and we needed to encode an above and below mark for each tone. We've also got the meem above and below encoded as well as some others.

Well then, let us look at this comparandum comparandi U+08F3. L2/09-419R has this:

> The other missing character, ARABIC SMALL HIGH WAW, is exemplified in the Arabeyes document, although not as a new character. The character appears mid-word in Koranic text, as shown in Figure 3 (lefthand).
>
> U+06E5 ARABIC SMALL WAW cannot be used for this purpose, as it is a non-joining spacing character, and would result in the first part of the word getting disconnected from the second part.
>
> Although similar, U+064C ARABIC DAMMA cannot be used for this purpose either, as there are clear semantic and visual differences. Apart from having slightly different shapes, damma and small high waw are used for writing two different sounds ([u] vs [u:]). A Koran reciter is aware of the visual difference and will pronounce them differently because of the visual hint.

U+08F3 is Diacritic=Yes; so if we accept Kamal Mansour's assessment, **the Diacritic property assignments of U+08F3 and U+08D3 are inconsistent**.

And the caption of Figure 3:

> **Figure 3.** On the right side, there are two examples of spacing U+06E5 ARABIC SMALL WAW at the end of each word. On the left side, there is a combining version (applied to either a seen, or a tatweel, and followed by a small madda), not encoded yet. Note the visual difference between ARABIC SMALL HIGH WAW (typically indicating a long [u:] sound) and a normal U+064C ARABIC DAMMA (typically indicating a short [u]). Both samples from [1], page 282.

On ARABIC SMALL HIGH WORD QIF:

From L2/14-095 (character names etc. revised later by L2/14-105R)

> **ARABIC SMALL HIGH SIGN QIF –** This is the mandatory pause sign. Recited (sic) must take a pause here.

On ARABIC SUBSCRIPT ALEF and ARABIC INVERTED DAMMA:

L2/00-135 proposes ARABIC LETTER SUBSCRIPT ALEF in a list prefixed by the comment

> These characters are used to assist in pronunciation.

L2/01-425 then proposes ARABIC SUBSCRIPT ALEF and ARABIC TURNED DAMMA with the respective comments

- Used to indicate a long /i:/ vowel, or /i/ as contrasted with /e/

and

- Used to indicate a long /u:/ vowel, or /u/ contrasted with /o/

Surely U+0656 ARABIC SUBSCRIPT ALEF and U+0657 ARABIC INVERTED DAMMA should then have the same Diacritic assignment; the Diacritic property does not discriminate between vowel qualities.

On ARABIC SMALL HIGH MEEM ISOLATED FORM:

This character was added in Unicode 1.1; the record is sparse and sparsely digitized in that time period. However, the character is well-known, and its function clearly fits the criteria for Diacritic=Yes; Roozbeh Pournader provided the following page from a Quran, which documents it as "للدلالة على وجود الإقلاب", meaning "to indicate the presence of iqlāb", a form of formal nasal assimilation where a /n/ followed by a /b/ becomes a /m/.

3. Proposed new scripts & characters

PAG participants have reviewed the following character encoding proposals and prepared draft UCD data as described below. Where the descriptions below compare proposed characters with already-encoded ones, tests are in place to check that their property assignments are consistent.

Work has not yet been carried out on non-UCD data files, such as those for UTS #10 (collation), UTS #39 (security), or UTS #46 (IDNA). When these files are generated, the draft UCD properties may be revised in light of implications in these Unicode Technical Standards.

- L2/26-041: Request to Encode the Omani Rial Currency Symbol in the Unicode Standard — Central Bank of Oman [#778]
    - Propertywise like the Saudi Riyal; in particular for the Line_Break property, lb=PR (Prefix_Numeric). Currency symbols are always lb=PR or lb=PO.
    - Attachment 3 ("Photos and screenshots showing usage of Omani Rial symbol") in L2/26-041 motivates the choice of lb=PR.
- L2/25-131: Proposal to encode 7 historic alchemical symbols — Mayer, et al [#539]
    - From UTC-186-C30 and UTC-186-C36, three more alchemical symbols, propertywise like U+1F746 🝆 ALCHEMICAL SYMBOL FOR OIL (the one that is not inverted).
- 186-C31 / WG 2 N5333R2: 11 Cossic characters (from Leibniz's work) [#545]
    - The names are from UTC-186-C31. As noted in UTC-186 discussion, the general categories suggested in WG 2 N5333R2 are wrong. The characters with SMALL LETTER or SMALL LIGATURE in their name should have General_Category=Lowercase_Letter.
    - Of those letters, all but 1DF95 LATIN SMALL LIGATURE LONG S WITH DESCENDER S and 1D6A6 MATHEMATICAL ITALIC SMALL LIGATURE LONG S WITH DESCENDER S are straightforwardly lowercase Latin letters without uppercase counterparts, propertywise like U+1DF01 □ LATIN SMALL LETTER REVERSED SCRIPT G.
    - The iterated square roots, being derived from and co-occurrent with the ordinary square root √, should have similar properties; in particular, they should be Bidi_Mirrored, *contra* WG 2 N5333R2. Since √ has some legacy in East Asian typesetting, the properties of the new characters are more like those of ∜ than those of √ (East_Asian_Width=Neutral rather than Ambiguous, Line_Break=Alphabetic rather than Ambiguous).
    - LONG S WITH DESCENDER S is more interesting. The encoding model here is inspired by that for the Greek variant letterforms: ϑ, ϕ, ϖ are separately-encoded variants of the regular θ, φ, π; their glyphs lie within the general-purpose glyphic range of θ, φ, π for Greek text, but they must be distinguished in mathematical usage; see §7.2.1, *sub* Variant Letterforms.
    - Likewise, 1DF95 LATIN SMALL LIGATURE LONG S WITH DESCENDER S resembles some modern glyphs for ß, but must be distinguished from it to properly represent Rudolff's *Behend und hübſch Rechnung durch die kunſtreichen regeln Algebre, so gemeinicklich die Coſs genennt werden* (1525). See, for instance, the contrast between weiß at the beginning of the second paragraph of fol. 30 (p. 8 of WG 2 N5333R2) and 1DF95 below on the same page. This character is however seen as somewhat equivalent to ß, see, e.g., fol. 70r of the 1553 reprint which uses a Fraktur ß (ſz ligature).
    - In terms of properties, MATHEMATICAL ITALIC SMALL LIGATURE LONG S WITH DESCENDER S and LATIN SMALL LIGATURE LONG S WITH DESCENDER S should thus be to ß SHARP S as those of 𝜛 and ϖ are to π. This means that, as proposed, LATIN SMALL LIGATURE LONG S WITH DESCENDER S `<compat>` decomposes to SHARP S. It also means that LATIN SMALL LIGATURE LONG S WITH DESCENDER S has an Uppercase_Mapping to

SS and a Titlecase_Mapping to Ss, just like ϖ has those mappings to Π (this is achieved by means of an entry in SpecialCasing.txt). Note that just like italic ϖ has no case mappings, neither does MATHEMATICAL ITALIC SMALL LIGATURE LONG S WITH DESCENDER S; further, neither of the new characters have simple case mappings, as SS is not a single code point.
    - These case mappings in turn imply that LATIN SMALL LIGATURE LONG S WITH DESCENDER S case folds to ss, and its mathematical italic counterpart does so under NFKC_Casefold. Since UTC-175-C19, simple case folding is the restriction to single code points of full case folding. This means that since LATIN SMALL LIGATURE LONG S WITH DESCENDER S and SHARP S are equivalent under full case folding, they must also be equivalent under simple case folding, and thus LATIN SMALL LIGATURE LONG S WITH DESCENDER S must map to SHARP S under simple case folding, and its mathematical italic counterpart does so under NFKC_Simple_Casefold. This is similar to the situation of the ligature ft mapping to 𝔰t.
    - Note that an earlier revision of the proposal, WG 2 N5333R, had LATIN SMALL LIGATURE LONG S WITH DESCENDER S decomposing to ſs rather than ß. Applying the same logic to this decomposition would lead to the same case foldings and mappings and the same primary equivalences in collation, so the choice is of little importance for practical purposes of loose matching.
- 186-C33 / WG 2 N5334R2: Proposal to encode historical mathematical relations [#690]
    - 29 characters, with names as in UTC-186-C33. The properties are generally the same as those of other mathematical symbols, except that none of the new characters are Pattern_Syntax, whereas all of the comparanda below are. The main difficulty lies in the bidi mirroring. While the proposal suggests they all be Bidi_Mirrored=N, this should not be the case for all characters.
    - LEIBNIZIAN EQUALS SIGN, LEIBNIZIAN EQUALS SIGN WITH DOUBLE VERTICALS, LEIBNIZIAN CONGRUENCE-2 and its relatives, LEIBNIZIAN SIMILARITY are all symmetrical about the vertical axis, and should indeed be Bidi_Mirrored=N.
    - LEIBNIZIAN EQUALS SIGN WITH SMALL S is such a symbol adorned with a Latin letter, like 𝔪, and should likewise be Bidi_Mirrored=N.
    - One of the proposed characters, CARTESIAN EQUALS SIGN, is described in WG 2 N5534R2 p. 34 in relation to 221D ∝ PROPORTIONAL TO. ∝ has long been Bidi_Mirrored without a Bidi_Mirroring_Glyph. It now acquires a Bidi_Mirroring_Glyph in the form of CARTESIAN EQUALS SIGN, and the new character is Bidi_Mirrored with a Bidi_Mirroring_Glyph of ∝.
    - Three of the characters, LEIBNIZIAN COINCIDENCE, INVERTED LAZY S OVER LAZY S, and LEIBNIZIAN DISSIMILARITY, incorporate a lazy S or inverted lazy S (ꭥ) in their glyph; they are expected to be used similarly and together with ꭥ, and should likewise have Bidi_Mirrored=Yes with no Bidi_Mirroring_Glyph.
    - Two of the characters, INVERTED SQUARE LEFT OPEN BOX OPERATOR and INVERTED SQUARE RIGHT OPEN BOX OPERATOR, are inverted versions of the existing ⊏ SQUARE LEFT OPEN BOX OPERATOR and ⊐ SQUARE RIGHT OPEN BOX OPERATOR. Like the existing operators, they should be Bidi_Mirrored and mapped to each other by Bidi_Mirroring_Glyph.
    - LEIBNIZIAN GREATER-THAN and LEIBNIZIAN LESS-THAN are asymmetric symbols whose mirror image represents the converse relation. Such mathematical symbols are always Bidi_Mirrored, with a Bidi_Mirroring_Glyph being the converse; thus these two should be Bidi_Mirrored, and mapped to each other by Bidi_Mirroring_Glyph. Likewise for LEIBNIZIAN GREATER-THAN WITH SMALL P and LEIBNIZIAN LESS-THAN WITH SMALL P; the presence of text on these relations does not affect these properties, cf. ⪾ and ⪿.

- For the remaining characters, the question of the Bidi_Mirrored assignment is unclear (it is not like these 17th century characters have ever been used in right-to-left mathematical notation, so absent obvious analogues in modern notation there is no usage to inform us). Many of these are derived from letters (FACIT SYMBOL < 𝑓, LEIBNIZIAN CONGRUENCE < c), but that is not enough to determine the properties: the quantifier ∃ (< E) is Bidi_Mirrored, whereas the quantifier ⅁ (< G, used in game theory) is not. Since those letter-like symbols that are mirrored (see also ∂) are more common than those that are not (cf. ℵ), and since the proposed characters are extremely obscure, we go with the simpler Bidi_Mirrored=N.

4. Line Break

4.1 Again breaking by hyphen [#520]

Recommended UTC actions

1. **No Action**: PAG recommends no action; this feedback should be considered as part of prior action 185-A102.

Feedback (verbatim)

Note: The following feedback appears to be an addendum to feedback addressed in 2025-sep; see

> [185-A102] Action Item for Robin Leroy, PAG: Investigate the feasibility of changing line breaking around hyphens to avoid isolating single letters. For Unicode Version 18.0. See L2/25-228 item 4.2.

Date/Time: Sun Jan 08 23:01:12 PT 2026
ReportID: ID20260108230112
Name: Mikhail Merkuryev
Report Type: Report Error in Publication/Data
Opt Subject: Again breaking by hyphen

I'm pleased that you've taken my issue to discussion. I suggest writing these things to TR14 section 5.3 Use of Hyphen. Brush up as you wish.

Unless you've done morphological analysis, we strongly discourage you from:

Breaking out one character: 7- / bit, да- / с (Russian: yes milord)
And discourage you from:

Breaking out two characters: кто- / то (Russian: someone)
Breaking out short numbers: 128- / bit
No change in formal algorithms.

4.2 No HH in field 1 description in LineBreak.txt [#522]

Recommended UTC actions

1. **No Action**: PAG recommends no action: the comment will be corrected for β review.

Feedback (verbatim)

Date/Time: Thu Feb 5 15:56:17 PT 2026
ReportID: ID20260205155617
Name: Sergiusz Wolicki
Report Type: Report Error in Publication/Data
Opt Subject: No HH in field 1 description in LineBreak.txt

https://www.unicode.org/Public/UCD/latest/ucd/LineBreak.txt:

```
None
# Field 1: Line_Break property, consisting of one of the following values:

#   Non-tailorable:

#         "BK", "CM", "CR", "GL", "LF", "NL", "SP", "WJ", "ZW", "ZWJ"

#   Tailorable:

#         "AI", "AK", "AL", "AP", "AS", "B2", "BA", "BB", "CB", "CJ",

#         "CL", "CP", "EB", "EM", "EX", "H2", "H3", "HL", "HY", "ID",

#         "IN", "IS", "JL", "JT", "JV", "NS", "NU", "OP", "PO", "PR",

#         "QU", "RI", "SA", "SG", "SY", "VF", "VI", "XX"
```

The new HH property values is missing from the list.

4.3 Incorrect Line Break Property of U+FE51 (PRI-536) [#526]

Recommended UTC actions

1. **Consensus**: Change the Line_Break assignment of U+FE51 SMALL IDEOGRAPHIC COMMA from Ideographic (ID) to Close_Punctuation (CL). For Unicode Version 18.0. See L2/26-096 item 4.3.
2. **Action Item** for Robin Leroy, PAG: In UCD file LineBreak.txt and derived files, change the Line_Break assignment of U+FE51 SMALL IDEOGRAPHIC COMMA from Ideographic (ID) to Close_Punctuation (CL). For Unicode Version 18.0. See L2/26-096 item 4.3.

Feedback (verbatim)

PRI-536

Date/Time: Tue Feb 17 05:50:09 PT 2026
ReportID: ID20260217055009
Name: Ruixi Zhangy
Report Type: Public Review Issue
Opt Subject: 536 - Incorrect Line Break Property of U+FE51

In LineBreak.txt, U+FE51 SMALL IDEOGRAPHIC COMMA has the property "ID", which doesn't match the fact that it is a "Po"st punctuation mark. It should
have the same "CL" property as U+FE50 SMALL COMMA and U+FE52 SMALL FULL STOP.

Background information / discussion

The assignment appears to be an omission in 1999 from the list of exceptions assigned CL, which included the neighbouring FE50 and FE52, as well as the compatibility equivalent 3001; see the the draft then-UTR #14 L2/99-189, the initial derivation L2/99-179, and the resulting assignments L2/99-180; absent the exception, the rule Po → ID(w) applied to FF51.

The purpose of the small forms from CNS 11643 is unclear, and they are rarely used. Absent further information, assigning the Line_Break property consistently with their compatibility equivalent seems reasonable, and their rarity means we are unlikely to break users relying on the current behaviour.

4.4 Ambiguous line breaking for 02DF CROSS ACCENT [#536]

Recommended UTC actions

1. **No Action**: PAG recommends no action as it negatively impacts intended use of the character

Feedback (verbatim)

PRI-536 Unicode 18.0 alpha

Date/Time: Mon Mar 02 15:34:57 PT 2026
ReportID: ID20260302153457
Name: Mikhail Merkuryev
Report Type: Public Review Issue
Opt Subject: Ambiguous line breaking for 02DF [PAG]

At least in Soviet tradition, people wrote “10ˣ” for optical magnification.

IDK how to coin the rule. Options…

1. Just add a note: Sometimes “10ˣ” means optical magnification. To prevent breaking, insert a 2060 WJ between these characters.
2. Make a rule: If other characters provide an opportunity to break after BA, and do not provide before → do not break before.
3. Make a special class BX=Conditional Break Before for some or all six European BB’s. The rules are the same: do not break before if you see an opportunity after.

Background information / discussion

In discussion it was decided that this particular usage is too specialized to override generic considerations for the intended use as "cross accent".

However, intra-word break opportunities (other than for ideographs) should never result in orphans or widows. This is being addressed by a slight broadening of an existing action item UTC185-A102. It would result in keeping together expressions like "10ˣ" when they are surrounded, or in this case, followed, by non-word characters.

"10ˣ magnification" would then work as expected, while "10ˣmagnification" would not. That seems an acceptable outcome for a general, default algorithm.

4.5 Should LB12a apply to – EN DASH followed by NO-BREAK SPACE? (Yes.) [#543]

Recommended UTC actions

1. **Consensus**: Change the Line_Break assignments of U+2012 – FIGURE DASH and U+2013 – EN DASH from Line_Break=Unambiguous_Hyphen to Line_Break=Break_After, and U+00AD SOFT HYPHEN from Line_Break=Break_After to Line_Break=Unambiguous_Hyphen, for Unicode Version 18.0. See L2/26-096 item 4.5.
2. **Action Item** for Robin Leroy, PAG: In UCD file LineBreak.txt and derived files, Change the Line_Break assignments of U+2012 – FIGURE DASH and U+2013 – EN DASH from Line_Break=Unambiguous_Hyphen to Line_Break=Break_After, and U+00AD SOFT HYPHEN from Line_Break=Break_After to Line_Break=Unambiguous_Hyphen. For Unicode Version 18.0. See L2/26-096 item 4.5.
3. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, update the descriptions of classes Break_After and Unambiguous_Hyphen to reflect the changes to the assignments of U+2012, U+2013, and U+00AD. For Unicode Version 18.0. See L2/26-096 item 4.5.
4. **Consensus**: Change rule LB12a from [^SP BA HY HH] × GL to [^SP HY HH] × GL, disallowing line breaks between characters of class Break_After and characters of class Glue, for Unicode Version 18.0. See L2/26-096 item 4.5.
5. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, Change rule LB12a to [^SP HY HH] × GL. For Unicode Version 18.0. See L2/26-096 item 4.5.
6. **Action Item** for Robin Leroy, PAG: In UCD files LineBreakTest.txt and LineBreakTest.html, change rule LB12a to [^SP HY HH] × GL. For Unicode Version 18.0. See L2/26-096 item 4.5.
7. **Action Item** for Robin Leroy, PAG: In UCD files LineBreakTest.txt and LineBreakTest.html, add realistic tests exercising the changes to the behaviour of en dashes set off from their contents with non-breaking spaces. For Unicode Version 18.0. See L2/26-096 item 4.5.

Feedback (verbatim)

PRI-536 Unicode 18.0 Alpha Review

Date/Time: Tue Mar 31 05:37:59 PT 2026
ReportID: ID20260331053759
Name: Léane Grasser
Report Type: Public Review Issue
Opt Subject: PRI 536: Allowing breaks before en dashes [PAG]

In French, at least in France, en dashes are now commonly used for parentheticals as an alternative to em dashes. However, UAX #14 does not define the same line-breaking behavior for EN DASH (U+2013) and EM DASH (U+2014). LB12a + LB21 result in undesirable breaks after en dashes.

Therefore, I would like to suggest allowing line breaks before EN DASH for the same reason explained in the B2 class description [1].

[1]https://www.unicode.org/reports/tr14/tr14-55.html#B2

From Léane Grasser via the unicode mailing list:

> Hi,
>
> Recently, while setting up typographic conventions for the French newspaper I contribute to, I noticed an issue that affected en dashes (U+2013), but not em dashes (U+2014).
>
> We collectively decided to use en dashes for parentheticals rather than em dashes, because of their limited size, improved readability over hyphen-minus, and Unicode "compliance". We also put a non-breaking space (U+00A0) inside the parenthetical and a regular space (U+0020) outside.
>
> Example, with "--" as an en dash: Nous mangions des pommes[SP]--[NBSP]les plus rouges au monde[NBSP]--[SP]sous les arbres.
>
> However, since en dashes are considered unambiguous hyphens (HH) in UAX #14, the tailorable line breaking rule LB12a means that there *can* be a line break after en dashes, even with a non-breaking space. LB21 also specifies that there shouldn't be a line break before HHs.
>
> This is a problem in our case, as we, like many other major French publications, use en dashes for parentheticals and therefore can't join the opening dash and the word after with a simple NBSP. This results in the opening dash being placed at the end of the line, which is undesirable. Em dashes are unaffected due to being categorized as B2 rather than HH.
>
> We eventually decided to resort to the WORD JOINER + NBSP combo, rather than falling back to em dashes--but it feels quite hacky.
>
> Therefore, I would like to suggest allowing breaks before the en dash character (U+2013), perhaps by moving the character from HH to B2 along with em dash.
>
> Regards,
> Léane Grasser
>
> PS. I tried to search this mailing list's archive up to 2014 and couldn't find a discussion regarding this very topic. > Sorry in advance if it's already been discussed.

Background information / discussion

This usage of the EN DASH is common, and has – since Unicode 4 – been documented in the core specification:

> U+2014 — EM DASH is used to make a break—like this—in the flow of a sentence. (Some typographers prefer to use U+2013 – EN DASH set off with spaces – like this – to make the same kind of break.)

We should not require WORD JOINER shenanigans for such common use cases.

Rule LB12a and the Line_Break class of EN DASH have changed recently, but deliberately not in a way that affects the applicability of LB12a: EN DASH used to be lb=BA, and LB12a did not mention HH (because this class did not exist). See L2/24-224 Section 6.1 and UTC decision 181-C35.

The change to rule LB12a which created the unwanted break opportunity between an EN DASH and a following NO-BREAK SPACE was in Unicode Version 5.1. See UTC-110-C17, L2/07-028, and https://www.unicode.org/notes/tn54/alba-3.html#p401.17.

However, the introduction of class HH makes a fix easy. The rationale for LB12a mentions

> Allowing a break after BA or HY matches widespread implementation practice and supports a common way of handling special line breaking of explicit hyphens, such as in Polish and Portuguese.

So BA was here for hyphens (and in particular HYPHEN) specifically, not for EN DASH. It was also there for SOFT HYPHEN; we should move that one to HH, with HYPHEN. HYPHEN is the archetypal Unambiguous_Hyphen (HH); there is no reason to break between the remaining BAs and NBSP.

Further, EN DASH is not used as a hyphen; it was simply caught in the initial derivation of lb=HH based on its General_Category. Reverting it back to BA, together with the change to LB12a, would fix the issue. Looking at UTC-181-C53, the FIGURE DASH also sticks out like a sore thumb; it is not a hyphen either.

As for *allowing* a break before an EN DASH that follows a space as described in the feedback, it is already allowed (LB18 comes before LB21).

Treating EN DASH exactly like EM DASH for line breaking, *i.e.*, moving it to class B2, would interfere with its other uses, for instance, in ranges: there should not be a line break after the digit 4 in 1914–1918; fortunately the use of spaces when en dashes are used for—wait, how do you say *incise* in English—these things allows us to accommodate this usage without creating new problems.

5. Segmentation

5.1 InCB=Linker needs to be extended for new characters [#516]

Recommended UTC actions

1. **Consensus**: Change Rule GB9c in UAX #29 to `\p{InCB=Linker} \p{InCB=Extend}* × \p{InCB=Consonant}`. For Unicode 18.0. See L2/26-096 item 5.1.
2. **Consensus**: Change the derivation of InCB=Linker in UAX #44 to include InSC=Consonant_With_Stacker. For Unicode 18.0. See L2/26-096 item 5.1.
3. **Action Item** for Josh Hadley, PAG: Update UAX #29 to reflect the changed Rule GB9c. For Unicode 18.0. See L2/26-096 item 5.1.
4. **Action Item** for Ken Whistler, PAG: Update UAX #44 to reflect the changed derivation of InCB=Linker. For Unicode 18.0. See L2/26-096 item 5.1.
5. **Action Item** for Robin Leroy, PAG: Update UCD data files to reflect the changed derivation of InCB=Linker and the change to GB9c. For Unicode 18.0. See L2/26-096 item 5.1.

PAG input

From Roozbeh Pournader, PAG:

Two new characters provisionally assigned at UTC186, U+11B0B DEVANAGARI SIGN JIHVAMULIYA and U+11B0C DEVANAGARI SIGN
UPADHMANIYA, have InSC=Consonant_With_Stacker and sc=Deva (see L2/25-121).
Consonant_With_Stacker means that they form conjuncts with the following consonant without needing a virama in between: for example, <JIHVAMULIYA, KA> forms a conjunct. These conjuncts need to be one (extended) grapheme cluster, so we need to make sure there is no grapheme break in the above sequence. At the same time, if any of the two new characters is followed by a non-Consonant, such as a space or punctuation, there should be a grapheme break between them. In other words, they act like a Consonant+Virama.

Unfortunately, this behavior does not seem achievable using the existing GB9c rule. Fortunately, it is achievable using the alternative GB9c rule proposed in PAG issue #432 “Fixing GB9c rule in UAX #29”: `\p{InCB=Linker} \p{InCB=Extend}* × \p{InCB=Consonant}`. We would simply need to make the two new characters InCB=Linker.

Background

The proposed rule is the result of follow-up discussion on the following action item:

- [177-A100] Action Item for Manish Goregaokar, Robin Leroy, PAG: Contact Josh Hadley and/or Mark Davis to understand why the current GB9c has a consonant at the beginning. Consider creating a document describing the rationale for the change from the “virama-sequence” rule proposed in L2/18-147 to the rule GB9c proposed in L2/23-079. (Ref. Section 19 of L2/23-238).

At a PAG meeting on 2025-05-29, several questions about the current rule `\p{InCB=Consonant} [ \p{InCB=Extend} \p{InCB=Linker} ]* \p{InCB=Linker} [ \p{InCB=Extend} \p{InCB=Linker} ]* × \p{InCB=Consonant}` were raised whose answer seem worth reporting:

> Why are multiple Linker characters allowed between the Consonants?
> Why are Extend characters beyond nuktas and joiners allowed between the Consonants?

For the second question, part of that is canonical equivalence. But also, any unexpected gcb=Extend character here is degenerate, so we don’t care. And trying to be too narrow in our characterization of what we expect here has repeatedly led us to actual problems.

> Why is the first Consonant necessary? (This was UTC Action Item 177-A100 that led to this discussion.)

Here we have a real nondegenerate problem, because that first consonant prevented Norbert from extending GB9c to some scripts.

Part of this might be that the rules were developed in the ICU framework, which do not express a × X rule directly, but as a regex including the stuff being stuck to the X (so `.` X in the simplest case), which leads you naturally to consider what that `.` may be.
In line, word, and sentence breaking, where we have treat-as rules to deal with combining marks, we do not want to start such a rule with a combining mark, and so any leading `.` is actually restricted to something that is not a combining mark. We suspect that this kind of concern is why, out of habit, Andy Heninger may have steered away from making GB9c a `. $Linker $Consonant`.

It appears that we can harmlessly chop off the leading consonant in GB9c, thus `\p{InCB=Linker} [ \p{InCB=Extend} \p{InCB=Linker} ]* × \p{InCB=Consonant}`.
And that would then be equivalent to a straightforward `\p{InCB=Linker} \p{InCB=Extend}* × \p{InCB=Consonant}`.

This rule is in turn of the form of the GB9d proposed in L2/23-141, so it should be possible to incorporate that feedback by changing the derivation of InCB, rather than by adding yet another rule.

The revised rule also fixes some Malayalam clusters with ZWNJ, some of which are listed in https://www.unicode.org/versions/Unicode17.0.0/core-spec/chapter-12/#G706410.

6. IDNA

6.1 Idna2008.txt typo CONTEXT0 [#519]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The typo has been fixed in data files for Unicode 18.

Feedback (verbatim)

Date/Time: Thu Jan 12 16:07:38 PT 2026
ReportID: ID20260112160738
Name: Meghan Denny
Report Type: Report Error in Publication/Data
Opt Subject: typo in idna/Idna2008.txt comment

https://www.unicode.org/Public/17.0.0/idna/Idna2008.txt contains the following comment:

```
None
# Field 1: IDNA2008_Category, consisting of one of these values

#            "PVALID"     - Protocol valid (generally Letters, Digits and Hyphen)

#            "CONTEXTJ"   - Join control

#            "CONTEXT0"   - Other code points requiring context

#            "DISALLOWED" - The code point is not allowed in IDNA2008

#            "UNASSIGNED" - The code point is not assigned in this version

"CONTEXT0" should be "CONTEXTO" in the next release as that would reflect the data
accurately.
```

7. Collation

7.1 re-align the DUCET & CLDR: Additional contractions for Tibetan [#269]

Recommended UTC actions

1. **Consensus**: In the UCA default sort order, add the Tibetan contractions needed to make the DUCET fully well-formed, as described in UTS #10 version 17 section 6.7, Tibetan and Well-Formedness of DUCET, like in CLDR. For Unicode 18.0. See L2/26-096 item 7.1.
2. **Action Item** for Ken Whistler, PAG: In the UCA default sort order, add the Tibetan contractions needed to make the DUCET fully well-formed, as described in UTS #10 version 17 section 6.7, Tibetan and Well-Formedness of DUCET, like in CLDR. For Unicode 18.0. See L2/26-096 item 7.1.
3. **Action Item** for Markus Scherer, PAG: In UTS #10, document that the DUCET is fully well-formed including the Tibetan contractions described in UTS #10 version 17 section 6.7, Tibetan and Well-Formedness of DUCET, like in CLDR. For Unicode 18.0. See L2/26-096 item 7.1.

PAG input

From Markus Scherer, PAG

As documented in UTS #10 version 17 section 6.7, Tibetan and Well-Formedness of DUCET,
the DUCET is missing Additional contractions for Tibetan:
Two to fulfill well-formedness condition 5, and eight more to preserve the default order for Tibetan.
CLDR has been adding these contractions for a number of years.

This omission was due to a limitation in the DUCET generator tool. Ken Whistler has removed this limitation and added the Tibetan contraction mappings to the allkeys.txt data file.

7.2 re-align the DUCET & CLDR: U+FFFE with minimal weights [#270]

Recommended UTC actions

1. **Consensus**: In the UCA default sort order, give U+FFFE minimal non-zero weights, with special processing, as specified in CLDR. For Unicode 18.0. See L2/26-096 item 7.2.
2. **Action Item** for Ken Whistler, PAG: In the UCA default sort order data file allkeys.txt, give U+FFFE a non-zero primary weight lower than that of all other code points. For Unicode 18.0. See L2/26-096 item 7.2.
3. **Action Item** for Markus Scherer, PAG: In UTS #10, document that U+FFFE maps to a non-zero weight lower than that of all other code points at least on primary and identical levels, and potentially on all levels, with special processing, as specified in CLDR. For Unicode 18.0. See L2/26-096 item 7.2.

PAG input

From Markus Scherer, PAG

The DUCET has no explicit mapping for U+FFFE, sorting it among unassigned code points.

CLDR tailors U+FFFE:

> This code point produces a CE [collation element] with minimal, unique weights on primary and identical levels.
> ...
> This allows for Merging Sort Keys within code point space.
> For example, when sorting names in a database, a sortable string can be formed with *last_name* + '\uFFFE' + *first_name*. These strings would sort properly, without ever comparing the last part of a last name with the first part of another first name.

Besides the special collation element with a non-zero primary weight, this requires a small amount of special processing:

- This collation element (CE) must not be treated as variable despite its low but non-zero primary weight.
- On the identical level, a non-zero minimal weight must be emitted as well.
- On intermediate levels, the collation element weights need not be minimal.
- Backwards secondary level sorting must be modified.
- See the CLDR spec for details.

I propose adding a special collation element mapping for U+FFFE to the DUCET, and specifying the special handling in UTS #10, for the same U+FFFE behavior as in CLDR.

7.3 re-align the DUCET & CLDR: U+FFFF with maximal primary weight [#271]

Recommended UTC actions

1. **Consensus**: In the UCA default sort order, give U+FFFF a primary weight higher than that of all other code points. For Unicode 18.0. See L2/26-096 item 7.3.
2. **Action Item** for Ken Whistler, PAG: In the UCA default sort order data file allkeys.txt, give U+FFFF a primary weight higher than that of all other code points. For Unicode 18.0. See L2/26-096 item 7.3.
3. **Action Item** for Markus Scherer, PAG: In UTS #10, document that U+FFFF has a primary weight higher than that of all other code points. For Unicode 18.0. See L2/26-096 item 7.3.

PAG input

From Markus Scherer, PAG

The DUCET has no explicit mapping for U+FFFF, sorting it among unassigned code points.

CLDR tailors U+FFFF:

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

I propose adding a special collation element mapping for U+FFFF to the DUCET, making it sort this code point like CLDR does.

7.4 Collation of "barred closed omega" [#523]

Recommended UTC actions

1. **No Action**: PAG recommends no action.

Feedback (verbatim)

Date/Time: Tue Feb 10 05:15:43 PT 2026
ReportID: ID20260210051543
Name: Ismael RH
Report Type: Report Error in Publication/Data
Opt Subject: Collation of "barred closed omega"

Dear staff,

The character "closed omega" is collated as a variant of "o" in IPA Extensions (lowercase) as well as in Latin Extended-F (modifier lowercase). Additional EPA variants (closed omega with long stem; turned closed omega) are also collated as such in the provisional order for EPA letters in Latin Extended-G.

In light of this, I would like to request UTC to place "barred closed omega" after "barred eng" so that it is likewise treated as a variant of "o" rather than of "w", in consistency with the rest of encoded (or futurely encoded) barred omegas.

Yours truly,
Ismael

Background information / discussion

The way characters and strings are sorted for end users is determined by the collation algorithm (UTS #10) and its DUCET default sort order (plus optional tailorings). The sort order is not generally tied to the code point order.

The order in the code chart (e.g., Unicode 18 Latin Extended-G: barred v, barred closed omega, barred chi) has no influence on where these characters are inserted in the DUCET sort order.

U+1DF4F LATIN SMALL LETTER BARRED ENG and U+1DF53 LATIN SMALL LETTER BARRED CLOSED OMEGA are both new in Unicode 18, see the proposal L2/24-234. As of the time of this feedback, Unicode has not yet published collation data for Unicode 18 characters.

In the DUCET, barred o and closed omega sort near but primary-after letter o. Letter eng sorts near but primary-after letter n. New characters will sort near related ones, with barred letters after non-barred variants.

This will result in “"barred closed omega" [...] treated as a variant of "o" rather than of "w"” as requested. Note that this order cannot be achieved by inserting “"barred closed omega" after "barred eng"”.

8. Security

8.1 UTS #39 assumes casefolding means lowercasing [#456]

Recommended UTC actions

1. **Action Item** for Josh Hadley, PAG: In UTS #39 section 4 remove the sentence “To reduce security risks, it is advised that identifiers use casefolded forms, thus eliminating uppercase variants where possible.”. For Unicode 18.0. See L2/26-096 item 8.1.

PAG input

From Roozbeh Pournader, PAG:

UTS #39, in its section 4 Confusable Detection, says: "To reduce security risks, it is advised that identifiers use casefolded forms, thus eliminating uppercase variants where possible."

This sentence is probably from the era where case-folding was always towards lowercase, before Unicode encoded lowercase Cherokee letters. (In Cherokee, case-folding goes towards the uppercase.)

We have two options: edit the sentence so that it's also correct for Cherokee, or remove the sentence altogether.

Considering that this sentence is in the confusables section and

- for case-sensitive identifiers, the advice is bad
- the quality of the Unicode confusables data is no different for lowercase and uppercase characters (and there are quite a few confusables among lowercase letters too)
- it would need a lot more context

it may make sense to remove the sentence altogether.

8.2 Ambiguity in UTS #39 (PRI-540) [#541]

Recommended UTC actions

1. **Action Item** for Markus Scherer, PAG: In UTS #39 section 5.4, under “Check for unlikely sequences of combining marks”, change items a and b to list the General_Category values and also refer to core spec definition D53 Nonspacing mark. For Unicode 18.0. See L2/26-096 item 8.2.

Feedback (verbatim)

PRI-540
Date/Time: Mon Mar 23 14:23:05 PT 2026
ReportID: ID20260323142305
Name: Karl Williamson
Report Type: Report Error in Publication/Data
Opt Subject: Ambiguity in UTS 39

It says

Forbid sequences of the same nonspacing mark.

Forbid sequences of more than 4 nonspacing marks (gc=Mn or gc=Me).

I believe the traditional definition of nonspacing mark is gc=Mn. The

second line effectively redefines that definition to include enclosing

marks. Does that redefinition apply to the line above, or just to the

second line? It seems to me that sequences of the same enclosing mark

would be suspicious.

The document should be clarified, but in the meantime I don't know the answer, and I'm writing code that depends on it. So please tell me.

Background information / discussion

The quoted text is in UTS #39 section 5.4 under “Check for unlikely sequences of combining marks:”.

Core spec chapter 3 section 3.6.1 Combining Character Sequences definition D53 defines “*Nonspacing mark*: A combining character with the General Category of Nonspacing Mark (Mn) or Enclosing Mark (Me).”

8.3 Change Identifier_Type for U+028C LATIN SMALL LETTER TURNED V from Technical to Uncommon_Use [#542]

Recommended UTC actions

1. **Consensus**: Change Identifier_Type for U+028C LATIN SMALL LETTER TURNED V from Technical to Technical Uncommon_Use. For Unicode 18.0. See L2/26-096 item 8.3.
2. **Action Item** for Markus Scherer, PAG: Change Identifier_Type for U+028C LATIN SMALL LETTER TURNED V from Technical to Technical Uncommon_Use. For Unicode 18.0. See L2/26-096 item 8.3.

PAG input

From Asmus Freytag, PAG

U+028C LATIN SMALL LETTER TURNED V is documented in orthographies for Algonquin and Wendat, and appears extensively in community materials. Canadian domains are interested in supporting it to better support languages used by First Nations. Its orthographic use is documented, for example, in this dictionary, where the character is used over 1,000 times (search limit).

Kitci anicinabe (LS), kitci aʌiciʌape (PIK), sômis (KIT),
kitizī (KZ), kitci anicinabe (LP), kichi aya (TFN),
ketedizidjik (ABL), kitci AʌicʌApe (WAH): Aîné(e) / elder.

Widigemagan / odikweman (LS), ʌi8itike makaʌ (PIK), gôkôm / ni gôkôm (KIT),
wīdigemāgan (KZ), odikweman, (his wife)-nidikwem, (my wife)-kidikwem,
(your wife) (LP), wīdigemaganikwe (TFN), ni dikwem, ni widigemagin (ABL),
ʌikokom (WAH): Épouse / wife.

Its current Identifier_Type is "Technical". This may be correct for the non-orthographic use of this character, but seems mismatched for its documented orthographic role in these Indigenous language systems.

Given that there is active interest to support this character in domain names, it would be appropriate to change the Identifier_Type to "Uncommon_Use".

9. Confusables

9.1 Mid-priority confusables data based on suggestions from David Corbett [#460]

Recommended UTC actions

1. **No Action**: This does not require any UTC action; the Confusables data files are updated.

Confusables source and data

L2/26-086 from Roozbeh Pournader

Doc intro:

This document extracts lengthy source and data listings from a large collection of confusables originally suggested by David Corbett, subsequently edited and updated by Roozbeh Pournader, for consideration by the Properties and Algorithms Working group in updating Unicode data files.

9.2 Confusables data for U+1D9F [#500]

Recommended UTC actions

1. **No Action**: This does not require any UTC action; the Confusables data files are updated.

Confusables source

From Roozbeh Pournader, PAG:

It appears that there is a codepoint typo in the confusables prototype value for U+1D9F. confusables.txt currently says:

1D9F ; 1D4B ; MA # (ᵌ → ᵋ) MODIFIER LETTER SMALL REVERSED OPEN E → MODIFIER LETTER SMALL OPEN E

The correct mapping should be to U+1D4C ᵌ.

Data

Remove:
1D9F ; 1D4B
Add:
1D9F ; 1D4C

9.3 Suppressed confusables data for compatibility equivalent sequences [#528]

Recommended UTC actions

1. **No Action**: This does not require any UTC action; the Confusables data files are updated.

Confusables source

From Roozbeh Pournader, PAG:

The confusables generator code in unicodetools contains a condition that drops sequences from its source data that are *compatibility* equivalent. This does not make sense, since the algorithm that determines confusability only normalizes text according to NFD, and ignores compatibility decompositions. I removed the condition, and it resulted in various additions, some undesirable. After going through all the additions and removing undesirable ones, the following additions that were added due to existing confusable pairs in the source files seem reasonable.

Data

```
00BC ; 006C 002F 0034 # VULGAR FRACTION ONE QUARTER → LATIN SMALL LETTER L, SOLIDUS, DIGIT FOUR
00BD ; 006C 002F 0032 # VULGAR FRACTION ONE HALF → LATIN SMALL LETTER L, SOLIDUS, DIGIT TWO
00BE ; 0033 002F 0034 # VULGAR FRACTION THREE QUARTERS → DIGIT THREE, SOLIDUS, DIGIT FOUR
2153 ; 006C 002F 0033 # VULGAR FRACTION ONE THIRD → LATIN SMALL LETTER L, SOLIDUS, DIGIT THREE
2154 ; 0032 002F 0033 # VULGAR FRACTION TWO THIRDS → DIGIT TWO, SOLIDUS, DIGIT THREE
2155 ; 006C 002F 0035 # VULGAR FRACTION ONE FIFTH → LATIN SMALL LETTER L, SOLIDUS, DIGIT FIVE
2156 ; 0032 002F 0035 # VULGAR FRACTION TWO FIFTHS → DIGIT TWO, SOLIDUS, DIGIT FIVE
2157 ; 0033 002F 0035 # VULGAR FRACTION THREE FIFTHS → DIGIT THREE, SOLIDUS, DIGIT FIVE
2158 ; 0034 002F 0035 # VULGAR FRACTION FOUR FIFTHS → DIGIT FOUR, SOLIDUS, DIGIT FIVE
2159 ; 006C 002F 0036 # VULGAR FRACTION ONE SIXTH → LATIN SMALL LETTER L, SOLIDUS, DIGIT SIX
215A ; 0035 002F 0036 # VULGAR FRACTION FIVE SIXTHS → DIGIT FIVE, SOLIDUS, DIGIT SIX
215B ; 006C 002F 0038 # VULGAR FRACTION ONE EIGHTH → LATIN SMALL LETTER L, SOLIDUS, DIGIT EIGHT
215C ; 0033 002F 0038 # VULGAR FRACTION THREE EIGHTHS → DIGIT THREE, SOLIDUS, DIGIT EIGHT
215D ; 0035 002F 0038 # VULGAR FRACTION FIVE EIGHTHS → DIGIT FIVE, SOLIDUS, DIGIT EIGHT
215E ; 0037 002F 0038 # VULGAR FRACTION SEVEN EIGHTHS → DIGIT SEVEN, SOLIDUS, DIGIT EIGHT
215F ; 006C 002F # FRACTION NUMERATOR ONE → LATIN SMALL LETTER L, SOLIDUS

3192 ; 30FC # ( ㆒ → ー ) IDEOGRAPHIC ANNOTATION ONE MARK → KATAKANA-HIRAGANA PROLONGED SOUND MARK
3196 ; 4E0A # ( ㆖ → 上 ) IDEOGRAPHIC ANNOTATION TOP MARK → CJK UNIFIED IDEOGRAPH-4E0A

FB05 ; 0066 0074 # LATIN SMALL LIGATURE LONG S T → LATIN SMALL LETTER F, LATIN SMALL LETTER T

FCF2 ; 005F 0301 10D27 # ARABIC LIGATURE SHADDA WITH FATHA MEDIAL FORM → LOW LINE, COMBINING ACUTE
ACCENT, HANIFI ROHINGYA SIGN TASSI
FCF3 ; 005F 0313 10D27 # ARABIC LIGATURE SHADDA WITH DAMMA MEDIAL FORM → LOW LINE, COMBINING COMMA
ABOVE, HANIFI ROHINGYA SIGN TASSI
FCF4 ; 005F 0650 10D27 # ARABIC LIGATURE SHADDA WITH KASRA MEDIAL FORM → LOW LINE, ARABIC KASRA,
HANIFI ROHINGYA SIGN TASSI
FE71 ; 005F 0301 0301 # ARABIC TATWEEL WITH FATHATAN ABOVE → LOW LINE, COMBINING ACUTE ACCENT,
COMBINING ACUTE ACCENT
FE77 ; 005F 0301 # ARABIC FATHA MEDIAL FORM → LOW LINE, COMBINING ACUTE ACCENT
FE79 ; 005F 0313 # ARABIC DAMMA MEDIAL FORM → LOW LINE, COMBINING COMMA ABOVE
FE7B ; 005F 0650 # ARABIC KASRA MEDIAL FORM → LOW LINE, ARABIC KASRA
```

```
FE7D ; 005F 10D27 # ARABIC SHADDA MEDIAL FORM → LOW LINE, HANIFI ROHINGYA SIGN TASSI
FE7F ; 005F 030A # ARABIC SUKUN MEDIAL FORM → LOW LINE, COMBINING RING ABOVE

FE11 ; 3001 # PRESENTATION FORM FOR VERTICAL IDEOGRAPHIC COMMA → IDEOGRAPHIC COMMA
FE13 ; 003A # PRESENTATION FORM FOR VERTICAL COLON → COLON
FE14 ; 003B # PRESENTATION FORM FOR VERTICAL SEMICOLON → SEMICOLON
FE15 ; 0021 # PRESENTATION FORM FOR VERTICAL EXCLAMATION MARK → EXCLAMATION MARK
FE16 ; 003F # PRESENTATION FORM FOR VERTICAL QUESTION MARK → QUESTION MARK

1F12B ; 00A9 # CIRCLED ITALIC LATIN CAPITAL LETTER C → COPYRIGHT SIGN
1F12C ; 00AE # CIRCLED ITALIC LATIN CAPITAL LETTER R → REGISTERED SIGN
```

9.4 Confusables data for Cyrillic Extended-D [#529]

Recommended UTC actions

1. **No Action**: This does not require any UTC action; the Confusables data files are updated.

Confusables source

From Roozbeh Pournader, PAG:

There seems to be no confusables data for the Cyrillic Extended-D block, while a lot of them are similar to superscript or subscript Latin letters. See the proposed data below for confusable pairs:

Data

```
A69D ; 1D47 # MODIFIER LETTER CYRILLIC SOFT SIGN ~ MODIFIER LETTER SMALL B
1E030 ; 1D43 # MODIFIER LETTER CYRILLIC SMALL A ~ MODIFIER LETTER SMALL A
1E032 ; 10784 # MODIFIER LETTER CYRILLIC SMALL VE ~ MODIFIER LETTER SMALL CAPITAL B
1E035 ; 1D49 # MODIFIER LETTER CYRILLIC SMALL IE ~ MODIFIER LETTER SMALL E
1E037 ; 1D9F # MODIFIER LETTER CYRILLIC SMALL ZE ~ MODIFIER LETTER SMALL REVERSED OPEN E
1E039 ; 1D37 # MODIFIER LETTER CYRILLIC SMALL KA ~ MODIFIER LETTER CAPITAL K
1E03C ; 1D52 # MODIFIER LETTER CYRILLIC SMALL O ~ MODIFIER LETTER SMALL O
1E03E ; 1D56 # MODIFIER LETTER CYRILLIC SMALL ER ~ MODIFIER LETTER SMALL P
1E03F ; 1D9C # MODIFIER LETTER CYRILLIC SMALL ES ~ MODIFIER LETTER SMALL C
1E040 ; 1D40 # MODIFIER LETTER CYRILLIC SMALL TE ~ MODIFIER LETTER CAPITAL T
1E041 ; 02B8 # MODIFIER LETTER CYRILLIC SMALL U ~ MODIFIER LETTER SMALL Y
1E042 ; 1DB2 # MODIFIER LETTER CYRILLIC SMALL EF ~ MODIFIER LETTER SMALL PHI
1E043 ; 02E3 # MODIFIER LETTER CYRILLIC SMALL HA ~ MODIFIER LETTER SMALL X
1E047 ; A69D 1DA6 # MODIFIER LETTER CYRILLIC SMALL YERU ~ MODIFIER LETTER CYRILLIC SOFT SIGN,
MODIFIER LETTER SMALL CAPITAL I
1E04B ; 1D4A # MODIFIER LETTER CYRILLIC SMALL SCHWA ~ MODIFIER LETTER SMALL SCHWA
1E04C ; 2071 # MODIFIER LETTER CYRILLIC SMALL BYELORUSSIAN-UKRAINIAN I ~ SUPERSCRIPT LATIN SMALL
LETTER I
1E04D ; 02B2 # MODIFIER LETTER CYRILLIC SMALL JE ~ MODIFIER LETTER SMALL J
1E04E ; 1DB1 # MODIFIER LETTER CYRILLIC SMALL BARRED O ~ MODIFIER LETTER SMALL BARRED O
1E04F ; 107B2 # MODIFIER LETTER CYRILLIC SMALL STRAIGHT U ~ MODIFIER LETTER SMALL CAPITAL Y
1E050 ; 1D35 # MODIFIER LETTER CYRILLIC SMALL PALOCHKA ~ MODIFIER LETTER CAPITAL I
1E051 ; 2090 # CYRILLIC SUBSCRIPT SMALL LETTER A ~ LATIN SUBSCRIPT SMALL LETTER A
1E056 ; 2091 # CYRILLIC SUBSCRIPT SMALL LETTER IE ~ LATIN SUBSCRIPT SMALL LETTER E
1E05C ; 2092 # CYRILLIC SUBSCRIPT SMALL LETTER O ~ LATIN SUBSCRIPT SMALL LETTER O
1E061 ; 2093 # CYRILLIC SUBSCRIPT SMALL LETTER HA ~ LATIN SUBSCRIPT SMALL LETTER X
1E068 ; 1D62 # CYRILLIC SUBSCRIPT SMALL LETTER BYELORUSSIAN-UKRAINIAN I ~ LATIN SUBSCRIPT SMALL
LETTER I
```

```
1E069 ; 209B # YRILLIC SUBSCRIPT SMALL LETTER DZE ~ LATIN SUBSCRIPT SMALL LETTER S
1E06B ; 1D9C 0327 # MODIFIER LETTER CYRILLIC SMALL ES WITH DESCENDER ~ MODIFIER LETTER SMALL C,
COMBINING CEDILLA
1E06C ; A69C 1DA6 # MODIFIER LETTER CYRILLIC SMALL YERU WITH BACK YER ~ MODIFIER LETTER CYRILLIC
HARD SIGN, MODIFIER LETTER SMALL CAPITAL I
1E08F ; 0365 # COMBINING CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I ~ COMBINING LATIN SALL
LETTER I
```

10. Math

10.1 Missing variation sequence (PRI-533) [#538]

Recommended UTC actions

1. **Action Item** for John Wilcock, PAG: Add the variation sequence `29B7 + VS1` “CIRCLED PARALLEL with parallel lines touching the circle” to UTR #25 Table 10 “Variants of Mathematical Symbols using VS1”. See L2/26-096 item 10.1.

Feedback (verbatim)

PRI-533 PU-UTR #25, Unicode Support for Mathematics

Date/Time: Tue Mar 10 10:21:38 PT 2026
ReportID: ID20260310102138
Name: Charlotte Buff
Report Type: PRI feedback
Opt Subject: PRI 533: Missing variation sequence

Table 10, “Variants of Mathematical Symbols using VS1”, does not list the relatively new variation sequence `<29B7, FE00>` CIRCLED PARALLEL with parallel lines touching the circle, which was added specifically for use in mathematical notation.

Background

The sequence was proposed in L2/25-126 (see usages p. 9).

11. Other

11.1 Feedback on UTS #58 post UTC #185 [#512]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The feedback was taken into account before UTC approved and published UTS #58 version 17.0.

Feedback

PRI-509 Draft UTS #58, Unicode Link Detection and Formatting: URLs and Email Addresses

The feedback is long, so it is just linked here.

- ID20260109222039
- ID20260109222219

Background information / discussion

ID20260109222039

https://www.unicode.org/review/pri509/feedback.html#ID20260109222039

We were aware of these documents before (Asmus pointed them out). However, on the home page (https://uasg.tech/) we find the following. We should not be using or referencing an abandoned project.

> This site has been archived as of November 2025 and is no longer being updated.
> Please see the more detailed archive note for additional information. For current information and ongoing updates on Universal Acceptance, please visit https://icann.org/ua.

Other points:

- The licensing situation is also murky.
- Aside from that, the test data doesn't appear to have PQF (Path/Query/Fragment) tests, which are the primary focus of UTS #58 and the test data, not domain names. We have made changes in UTS #58 to make that clearer.
- As for BIDI, we cannot structurally address it (eg, using RLMs and LRMs to disambiguate would actually change the structure of the text, is not in scope).
    - We could recommend a few things (but without conformance criteria), where feasible:
        - Use bidi isolation to avoid the link text "bleeding" into the surrounding text.
        - Use the approach in https://unicode.org/reports/tr9/#HL4Example3 for inside the linkified text.
    - Note: In UAX #9, the UTC might consider recommending the above for URL display in general, rather than having https://unicode.org/reports/tr9/#HL4Example3 just be an example.
- The draft RFCs are out of scope, since they are recommendations for disallowing certain kinds of email addresses. That is, they are connected with the creation of email addresses on the server side. And

some of their recommendations (eg "阿Q正传@dømi.fo") are also beyond the scope. However, some guidance has been added to the Security Considerations in UTS #58.

ID20260109222219

https://www.unicode.org/review/pri509/feedback.html#ID20260109222219

Most of these are copyedit fixes, and are fixed in UTS #58.

Some notable others in post-feedback revisions.

- A clarification of the focus has been moved to Section 1.4 Focus, and expanded.
- A reference is added for UnicodeSet
- Clarified that in the detection algorithms, the return value is a part of (start, end) offsets.
- Clarified that the quoted local-part for email addresses is out of scope.

11.2 Advance UTS #61 to Draft status [#537]

Recommended UTC actions

1. **Consensus**: Advance the Proposed Draft Unicode Technical Standard #61, Unicode Set Notation, to Draft status. See L2/26-096 item 11.2.
2. **Action Item** for Robin Leroy, PAG: Prepare Draft Unicode Technical Standard #61, and post for public review. See L2/26-096 item 11.2.

PAG input

From Robin Leroy, PAG: The ICU-TC has reviewed PD UTS #61, recommended changes to it which have been made in the latest draft, and has approved changes to the ICU implementation (for ICU version 79) to align with the Proposed Draft standard as amended; these have been implemented in the C++ version, and will soon be ported to Java.

Based on this feedback and implementation experience, it seems appropriate to advance UTS #61 to Draft status.

Recall, from https://www.unicode.org/reports/about-reports.html, that

> A Draft Unicode Technical Report (DUTR) has the basic structure and content required for a new technical report, but has not yet received final approval for publication.

whereas

> A Proposed Draft Unicode Technical Report (PDUTR) is in an early stage of development.