

UTC #188 properties feedback & recommendations

Markus Scherer & Josh Hadley / [Unicode properties & algorithms group](#), 2026-jul-21

Participants	2
1. Core Spec	2
1.1 PRI #548: Conformance notes for D57h [#576]	2
2. UCD	5
2.1 Soft-dotted property for ! U+1D09 and ! U+1D4E [#493]	5
2.2 Unexpected Joining_Type for Samaritan (PRI-544) [#571]	7
2.3 Should U+16100 GURUNG KHEMA LETTER A have InSC=Consonant? (Yes.) [#575]	10
3. Characters	11
3.1 PAG review of U+1CED5 ALCHEMICAL SYMBOL FOR RETORT-2 [#579]	11
4. Proposed new scripts & characters	11
5. Bidi	16
5.1 UAX #9 Unicode Bidirectional Algorithm vs. digits (PRI-544) [#549]	16
5.2 Should U+1B42, U+1B3C, and U+1B3A have Bidi_Class=L like U+0CBF? (Yes.) [#574]	18
6. Normalization	21
6.1 Include U+11A7 in Normalization Tests (PRI-544) [#570]	21
7. Line Break	22
7.1 There should not be a line break between a CJK ideograph and a U+00B7 that follows it [#517]	22
7.2 New geometric shapes should be Line_Break=Alphabetic [#566]	24
8. Segmentation	25
8.1 There is no length limit for grapheme clusters (PRI-540) [#544]	25
8.2 Align UTS #51 & UAX #29 for flags and keycaps in ZWJ (PRI-543) [#545]	26
8.3 FYI: UTC-183 broke the Zanabazar square grapheme cluster 𐑦+𐑩=𐑦𐑩, but UTC-187 put it together again [#551]	28
8.4 Should the derivation of Indic_Conjunct_Break use Script_Extensions instead of Script? (Yes.) [#552]	29
8.5 Idea for changes to word boundary segmentation with hyphen-separated strings like UUID/GUID [#572]	31
8.6 Definition of "Grapheme extender" is incorrect (PRI-548) [#573]	32
9. Collation	33
9.1 UCA: remove Shift-Trimmed option [#553]	33
10. Regex	34
10.1 Typo in UTS18 [#564]	34
11. Security	34
11.1 Identifier_Type data for new Unicode 18.0 characters [#530]	34
11.2 Identifier_Type of Arabic and Hebrew combining marks [#555]	36
11.3 UTS39 needs to support ISO 15924 code HntI [#560]	37
11.4 Joiners vs combining marks in Arabic and similar scripts [#562]	38
12. Confusables	40
12.1 Confusables data from Google Summer of Code 2025 [#484]	40
12.2 confusables.txt and NFKC conflicts [#531]	43
12.3 Confusables data for U+2205 EMPTY SET [#548]	45

12.4 Additions to intentional.txt [#554].....	46
12.5 Confusables data for stretched Arabic math symbols [#556].....	47
12.6 Confusables data for new Unicode 18.0 characters [#557].....	48
12.7 Confusables data for Devanagari Sindhi implosives [#558].....	52
13. Emoji.....	53
13.1 Emoji_Presentation property for older characters (PRI-544) [#569].....	53
14. Other.....	55
14.1 UAX #24 needs to talk about Jpan/Kore/Hanb/HntI? [#561].....	55
15. Feedback not addressed in this document.....	55

Participants

The following people have contributed to this document:

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

1. Core Spec

1.1 PRI #548: Conformance notes for D57h [#576]

Recommended UTC actions

1. **Consensus:** Add notes to section 3.6.2 "Variation Selectors" in the Core spec to indicate that interpreting either misplaced variation selectors or unassigned variation sequences in any way not sanctioned by the Unicode Standard is not conformant. For Unicode 18.0. See [L2/26-154](#) item 1.1.
2. **Action Item** for Asmus Freytag, EDC: Add notes to section 3.6.2 "Variation Selectors" in the Core spec to indicate that interpreting either misplaced variation selectors or unassigned variation sequences in any way not sanctioned by the Unicode Standard is not conformant. For Unicode 18.0. See [L2/26-154](#) item 1.1.

Feedback (verbatim)

[PRI 548](#): Conformance notes for D57h

Date/Time: Wed June 24 09:00:03 PT 2026

ReportID: [ID20260624090003](#)

Name: Peter Constable

Report Type: Public Review Issue

The intent of UTC decision 187-C42 was to improve conformance language to mitigate against malicious use of variation selector characters, and new text in 3.6.2 helps. However, the current conformance note following D57h has limited effect for that goal:

"A conformant implementation must not interpret a misplaced variation selector as a request to modify the glyph shape of a preceding character."

This doesn't indicate that use of a misplaced variation selector *for any other purposed* is also non-conformant.

The paragraph preceding D57d does say,

"Variation sequences are not intended as a general extension mechanism for the character encoding."

The wording of that sentence is vague: someone might assert that an intentional use of a misplaced variation selector is not a *character encoding extension* but is some other type of information representation mechanism.

I suggest that the first note after D57h be revised as follows:

"Any use of a misplaced variation selector is non-conformant. A conformant implementation must not interpret a misplaced variation selector."

Related, it might be useful to add a statement somewhere regarding default ignorables along this line:

"Any use of byte sequences within text with an expectation that they will be interpreted as non-printing characters (hence invisible to users) but with a purpose of representing other information is not conformant text representation."

"Implementers are also advised that this technique can potentially be used as a security attack vector in applications. Any use of invisible characters for any purpose not explicitly defined in this standard is non-conformant."

Background information / discussion

PAG reviewed this feedback and agrees that the intended purpose of the conformance changes to Variation Selectors and Sequences were to prevent non-sanctioned use, which is a security risk. The request for UTC is to explicitly approve or reaffirm the suggested restrictions that go beyond graphical appearance.

We suggest the following amended language and disposition of the feedback:

- Add a note after D57d.

"A conformant implementation shall not interpret an unassigned variation sequence for any purpose not defined in this standard."

- Add a second note after D57h as follows:

"A conformant implementation shall not interpret a misplaced variation selector for any purpose not defined in this standard."

with the wording to be kept exactly parallel to the wording for the note 57d if the latter is edited in the process of editorial review.

PAG is working on a proposal to extend the tightened definition to other classes of default ignorable. The last two paragraphs of the feedback will be taken up in that context.

2. UCD

2.1 Soft-dotted property for Ꞥ U+1D09 and ꞥ U+1D4E [#493]

Recommended UTC actions

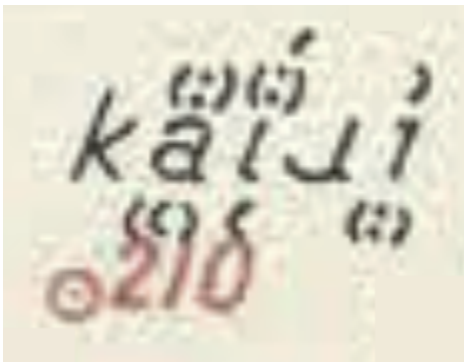
1. **No Action:** PAG recommends no action.

Feedback

From Kirk Miller:

The characters Ꞥ [U+1D09](#) LATIN SMALL LETTER TURNED I and ꞥ [U+1D4E](#) MODIFIER LETTER SMALL TURNED I should be added to PropList.txt. Unlike the other characters in that list, with these the dot is below and is replaced by a diacritic below. This soft-dotted behaviour is seen in dialectological atlases.

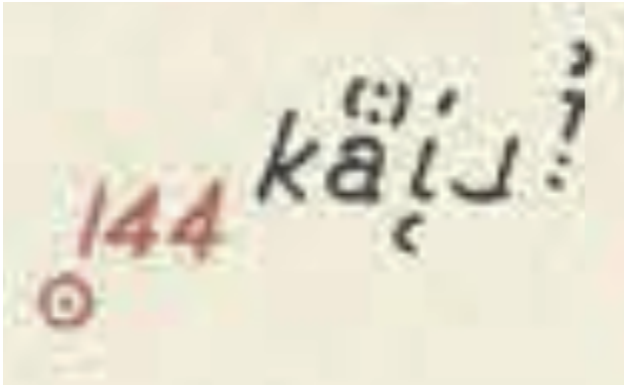
An example of 1D09 is,



Map 31, point 210 in Tomás Navarro Tomás et al. (1962) *Atlas lingüístico de la Península Ibérica*, vol. 1 Fonética. [Atlas lingüístico de la Península Ibérica, vol. 1 Fonética.](#)

The 3rd letter is an upright I with an open diacritic below and a colon diacritic, enclosed by parentheses, above. The last letter is a turned I with those same diacritics in reverse. In both cases the colon replaces the tittle.

An example of the modifier letter is,



Same map, point 144.

The colon is by far the most common diacritic used this way on these maps. Technically it could be substituted with a single dot on a hard-dotted I, but given the parallel between the upright and turned I's, their behaviour should be the same.

Other diacritics are rare, but in this example a tilde replaces the title of a turned nasal Ñ:



Map 66, point 205. The terminal shows this to be a turned I, but the open mark above confirms it.

PAG response

The group considered this feedback, and noted that the `Soft_Dotted` property is used in algorithms specified by *The Unicode Standard*, namely case transformations; the property is used in the definition of the context [After_Soft_Dotted](#), which is used in Lithuanian casing as defined in `SpecialCasing.txt`. This definition is tied to `Canonical_Combining_Class=230` (Above). Using `Soft_Dotted` for dots below would require changing the definition of these contexts; but the UTC no longer makes changes to the language-specific case mappings defined in `SpecialCasing`, deferring instead to CLDR. Thus even if there were otherwise a compelling use case for `Soft_Dotted` with dots below, this would be unacceptably destabilizing.

In addition, the proposal would require an implementation to know where the “soft” dot is, below or above, in order to decide if a combining mark should suppress the dot. Existing implementations that assume Soft_Dotted applies only to combining marks above the base would require changes, or else they would behave incorrectly.

The group considered the possibility of creating a new property similar to Soft_Dotted for dots below; this one would not need to be involved in casing. Two concerns were raised that led the group not to do so:

1. The proposal is for already-published characters; making them “soft dotted below” could change the behavior of existing text if people are using these characters with marks below.
2. The use cases for the soft dot for the characters are pretty obscure, and could be better solved by encoding new dotless characters. (The author of the feedback is already working on a proposal to encode such characters.)

2.2 Unexpected Joining_Type for Samaritan (PRI-544) [#571]

Recommended UTC actions

1. **Consensus:** Change the language and recommendations in ArabicShaping.txt to forcefully recommend that implementers use the derived files. For Unicode 18.0. See [L2/26-154](#) item 2.2.
2. **Action Item** for Roozbeh Pournader and Robin Leroy, PAG: Change the language and recommendations in ArabicShaping.txt to forcefully recommend that implementers use the derived files. For Unicode 18.0 See [L2/26-154](#) item 2.2.

Feedback (verbatim)

[PRI-544](#)

Date/Time: Thu June 11 19:30:21 PT 2026

ReportID: [ID20260611193021](#)

Name: Ned Holbrook

Report Type: Report Error in Publication/Data

Opt Subject: Unexpected Joining_Type for Samaritan

According to DerivedJoiningType.txt, Samaritan has characters with Joining_Type=Transparent but they do not appear in ArabicShaping.txt. How does this derivation occur?

Background information / discussion

Based on the following statement in [UAX #44](#):

The listing of `Joining_Type` in `DerivedJoiningType.txt` should be considered as definitive, because of the complexity of trying to derive the correct values directly from field 2 of `ArabicShaping.txt`.

replace the following text in `ArabicShaping.txt`:

None

```
# For an explicit listing of all characters of Joining_Type=T, see
# the derived property file DerivedJoiningType.txt.
# For an implementation that needs to parse for the values of
# Joining_Type, it is recommended to use DerivedJoiningType.txt
# instead of ArabicShaping.txt, to avoid the separate required step of
# calculating the set for Joining_Type=T based on General_Category values.
```

by the following more direct recommendation:

Implementers are strongly encouraged to use the derived property files `DerivedJoiningType.txt` and `DerivedJoiningGroup.txt` instead of the current file, to avoid the complication of having to calculate the `Joining_Type=T` values based on `General_Category`.

The revised recommendation should be made more prominent by moving it up in the file's header.

Similar "strong recommendation" should be given for using other derived property files.

Finally change the header comments to no longer say that `ArabicShaping.txt` **defines** `Joining_Type`, but instead note that it contributes to their definition in `DerivedJoiningType.txt`. See [UAX #44](#).

Change

None

```
# This file defines the Joining_Type and Joining_Group property
# values for Arabic, Syriac, N'Ko, Mandaic, and Manichaean positional
# shaping, repeating in machine readable form the information
# exemplified in various tables of The Unicode Standard core specification.
#
# This file also defines Joining_Type values for Mongolian, Phags-pa,
# Psalter Pahlavi, Sogdian, Old Uyghur, Chorasmian, and Adlam positional
# shaping, and Joining_Type and Joining_Group values for Hanifi Rohingya
# positional shaping, which are not listed in tables in the core
# specification.
```

to:

None

```
# This file lists the Joining_Type and Joining_Group property
# values for Arabic, Syriac, N'Ko, Mandaic, and Manichaean positional
# shaping, repeating in machine readable form the information
# exemplified in various tables of The Unicode Standard core specification.
#
# This file also lists the Joining_Type values for Mongolian, Phags-pa,
# Psalter Pahlavi, Sogdian, Old Uyghur, Chorasmian, and Adlam positional
# shaping, and Joining_Type and Joining_Group values for Hanifi Rohingya
# positional shaping, which are not listed in tables in the core
# specification.
#
# The full definition of these properties are found in the derived property
# files DerivedJoiningType.txt and DerivedJoiningGroup.txt.
```

The proposed language treats Joining Group in the same way as Joining Type even though the former doesn't have the same issues with algorithmic derivation. It seems easier to treat these in parallel rather than singling out one of them.

2.3 Should U+16100 GURUNG KHEMA LETTER A have InSC=Consonant? (Yes.) [#575]

Recommended UTC actions

1. **Consensus:** Change the Indic_Syllabic_Category of [U+16100](#) GURUNG KHEMA LETTER A from Vowel_Independent to Consonant. For Unicode Version 18.0. See [L2/24-203](#) and [L2/26-154](#) item 2.3.
2. **Action Item** for Robin Leroy, PAG: In UCD file IndicSyllabicCategory.txt, Change the Indic_Syllabic_Category of [U+16100](#) GURUNG KHEMA LETTER A from Vowel_Independent to Consonant. For Unicode Version 18.0. See [L2/24-203](#) and [L2/26-154](#) item 2.3.

PAG input

From Robin Leroy, PAG: This follows straightforwardly from the proposal [L2/24-203](#) accepted by [UTC-181-C44](#), Gurung Khema has a single character with InSC=Vowel_Independent which takes dependent vowels to form all independent vowels, as described in [the core specification](#).

It fell between the cracks as document [L2/24-203](#) was prepared between November of 2023 and August of 2024, before the Unicode 16 release, but accepted in January of 2025, after Unicode 16 had been released.

An invariant test has now been added as was promised in [L2/24-203](#).

3. Characters

3.1 PAG review of U+1CED5 ALCHEMICAL SYMBOL FOR RETORT-2 [#579]

Recommended UTC actions

1. **No Action:** Properties have been drafted as described.

PAG input

UTC decided:

[187-C12] Consensus: Provisionally assign 1 code point [U+1CED5](#) ALCHEMICAL SYMBOL FOR RETORT-2 in the Miscellaneous Symbols Supplement block, as described in [L2/25-131](#). [Ref: 3.2 in [L2/26-100](#)]

This is a second symbol for retort, propertywise like the first one [U+1CED4](#).

4. 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.

In the following, the term “propertywise” is a shorthand for “follows a similar pattern in character properties” where the comparison excludes properties that are necessarily distinct, such as Name or Age—the latter because the comparisons are to already-existing characters—, or ones where the differences are irrelevant for the behaviour of text, such as Block or `Unicode_1_Name`. The Properties and Algorithms Working Group uses a tool to compare tentative property assignments to those for characters that would be expected to have similar behavior and use and therefore similar property assignments.

- [L2/25-148](#): Proposal to encode two historical Persian letters used for ezafe — Ebrahim Byagowi and Roozbeh Pournader [#659]
 - Two new [non-joining Arabic letters](#), neither small nor high nor low; propertywise like [U+0621](#) ہ ARABIC LETTER HAMZA (mentioned in the proposal as a confusable).

- [L2/25-147](#): Unicode request for open q and x with stroke — Kirk Miller, et al [#654]
 - Three new letters in Latin Extended-G: one lowercase-only LATIN SMALL LETTER OPEN Q, propertywise like [U+1DF03](#) ɻ LATIN SMALL LETTER REVERSED K in the same block. A case pair (X WITH STROKE), propertywise like BARRED K in the same block, or like K k (K WITH STROKE) in Latin Extended-D.
- [L2/25-146](#): Unicode request for ɪ with retroflex hook — Kirk Miller [#644]
 - A new letter with retroflex hook in Latin Extended-G, propertywise like [U+1DF1B](#) ɹ LATIN SMALL LETTER O WITH RETROFLEX HOOK, in particular with a DoNotEmit entry for the corresponding letter ɪ followed by a combining retroflex hook, as proposed in [L2/25-146](#).
- [L2/25-145](#): Unicode request for wavy Y — Kirk Miller, Denis Moyogo Jacquerye [#655]
 - A new lowercase-only Latin letter (derived from two others, ʏ and ʎ, according to the sources cited in the proposal). Propertywise like the nearby ɸɣ LATIN SMALL LETTER FENG DIGRAPH WITH TRILL, or like its constituents ʏ LATIN SMALL LETTER TURNED H WITH FISHHOOK AND TAIL ʎ LATIN SMALL LETTER SQUAT REVERSED ESH (= long ɪ with left hook and tail).
- [L2/25-176](#): Unicode request to add capital U with left hook — Ismael Robles Haloui [#695]
 - An uppercase counterpart for the existing [U+AB52](#) ɹ LATIN SMALL LETTER U WITH LEFT HOOK, in the same block (Latin Extended-E). The new character and ɹ are propertywise like other case pairs in the Latin Extended blocks, e.g., [U+2C72](#) W and [U+2C73](#) w (W WITH HOOK).
- [L2/25-155](#): Unicode request for schwa with right hook — Kirk Miller, Neil Rees [#652]
 - Another lowercase-only Latin letter in Latin Extended-G, propertywise like [U+1DF25](#) ɹɹ LATIN SMALL LETTER D WITH MID-HEIGHT LEFT HOOK used in the same orthography for Malayalam.
- [L2/25-246](#): Proposal to encode Telugu Sign Spacing Candrabindu — Srinidhi A, Sridatta A [#696]
 - [U+0C70](#) TELUGU SIGN SPACING CANDRABINDU: Propertywise like the existing spacing candrabindus, namely Kannada ೞ ([U+0C80](#) KANNADA SIGN SPACING CANDRABINDU) and Devanagari ॠ ([U+A8F2](#) DEVANAGARI SIGN SPACING CANDRABINDU); in particular, InSC=Bindu as proposed, and lb=AL.
- [L2/25-245](#): Proposal to encode Candrabindu and Avagraha in Dogra — Srinidhi A, Sridatta A [#738]
 - A candrabindu for Dogra, propertywise like the Sharada candrabindu (or indeed like the Sharada inverted candrabindu, which looks more like it), in particular Alphabetic via Other_Alphabetic, and an avagraha, propertywise like the Sharada avagraha.
- [L2/25-244](#): Proposal to encode Combining Anusvara above in Tulu-Tigalari — Srinidhi A, Sridatta A [#736]
 - An anusvara above for Tulu-Tigalari, propertywise like the one in Grantha, in particular Alphabetic via Other_Alphabetic. Propertywise very much unlike the existing Tulu-Tigalari anusvara, as that one is a spacing mark with Indic_Positional_Category=Right as opposed to a nonspacing mark with Indic_Positional_Category=Top (with the associated differences in Bidi_Class and derived properties).

- [L2/25-121](#), [L2/25-091](#) for Devanagari: JIHVAMULIYA and UPADHMANIYA [#625]
 - New Devanagari *jihvāmūliya* and *upadhmānīya*, propertywise like the existing Vedic (Bengali) ones $\times$ & ω , but sc=scx=Devanagari rather than sc=Common, scx=Bengali; the Vedic ones remain sc=Common, as they might be usable for other scripts. Contrary to a statement made in the proposal, these do not have quite the same properties as the Kannada and Brahmi *jihvāmūliya* and *upadhmānīya*, as Devanagari and Bengali conjuncts are taken into account by grapheme segmentation, and Brahmi breaks at akshara boundaries: the new characters thus have InCB=Linker (see [UTC-187-C48](#)) as well as lb=Alphabetic as proposed, whereas their Kannada and Brahmi counterparts have InCB=None, and the Brahmi ones have lb=Aksara_Pibase.
- [L2/26-017](#): Alchemical symbol for calx — Kirk Miller [#771]
 - A second symbol for calx, propertywise like the first one ϵ .
- [L2/26-143](#): Leibnizian Ambiguous Signs (1DC00..1DC35) [#621]
 - Fifty-four operators derived from and used with [U+002B](#) + PLUS SIGN and [U+2212](#) – MINUS SIGN.
 - These new characters do not have the Pattern_Syntax property, so that they would not be programming language identifiers by default under [UAX #31](#)-R3c—one supposes few programmers are clamouring for these signs nobody has penned in the past three hundred years.
 - Propertywise like existing additive or subtractive operators, such as the common $\oplus$ CIRCLED PLUS, the uncommon $\dot{+}$ DOT PLUS, and the truly obscure $\dot{-}$ MINUS SIGN WITH COMMA ABOVE:
 - ea=Neutral, lb=Alphabetic whereas $\oplus$ has Ambiguous for both because of its East Asian legacy usage;
 - Even though some of these signs are asymmetric, they should not have Bidi_Mirrored=Yes, contrary to $\dot{-}$: their usage in right-to-left text would likely be citations of Leibniz, which should not be mirrored. These are also not relations whose converse is their mirror image, contrary to some of the Leibnizian characters discussed in [L2/26-096](#) p. 13 [#690].
 - Note that we are unaware of the usage of the existing $\dot{-}$ MINUS SIGN WITH COMMA ABOVE; a sort for it appears among the Monotype *Extraneous Sorts* shown on <https://ionathan.ch/2023/06/06/angzarr.html>, with $\dot{-}$, [now known to be azimuth](#); but that is a grab-bag of unusual things Monotype needed, so it may be difficult to determine its source.
 - Also propertywise like the obscure modern multiplicative ambiguous sign $\ast$ DIVISION TIMES and its Leibnizian counterpart [U+1CEF3](#) LEIBNIZIAN MULTIPLICATION-DIVISION SIGN (new in 18.0).
 - Unlike the existing $\pm$ and $\mp$ (which are semantically similar, indeed two of the new characters are used exactly with the meaning of $\pm$ and three with that of $\mp$), these signs are Line_Break=Alphabetic, rather than Line_Break=Prefix_Numeric, and Bidi_Class=Other_Neutral, rather than Bidi_Class=European_Terminator.

- The new characters are also Vertical_Orientation=Rotated by default (like plus, minus, and minus-plus) rather than Upright like plus-minus, in the absence of an East Asian typographical tradition for these signs.
 - They are also East_Asian_Width=Neutral, rather than Narrow like plus or Ambiguous like plus-minus, since there is no East Asian typographical tradition for these signs: Leibniz's ambiguous signs are not ambiguous (in terms of east asian width).
- Unlike [U+002B](#) + PLUS SIGN and [U+2212](#) – MINUS SIGN (with which these new signs would be used), these signs are Line_Break=Alphabetic, rather than Line_Break=Prefix_Numeric, and Bidi_Class=Other_Neutral, rather than Bidi_Class=European_Separator.
 - The plus and minus signs used to be bc=ET like plus-minus and minus-plus, but this was changed by decisions [UTC-98-M3](#) and [UTC-102-M1](#).
 - Assigning bc=ES for consistency with the plus and minus signs (which were used by Leibniz together with these new characters) was considered, but rejected, since any usage in right-to-left context is likely going to be either isolated or citing Leibniz, rather than doing new calculations using Leibniz's notation: it is vanishingly unlikely that someone would write a formula using +, realize they would need a – case (or two) and change the + to a Leibnizian ambiguous sign, so consistency would not help. Further, situations where bc=ES would have an effect would be extremely contrived (they require citations in a bc=R rather than bc=AL context, where the ambiguous sign occurs at the beginning of a citation but between numbers). The simpler bc=ON, consistent with most uncommon mathematical symbols, was preferred.
- [L2/26-142](#): 6 letterlike symbols (1DB1D..1DB21) [#691]
 - Six letterlike mathematical symbols, propertywise like ∃ (a quantifier used in game theory). They differ from ∫ (a rather more successful Leibnizian letterlike symbol) in not being Bidi_Mirrored, and in having Line_Break=Alphabetic and East_Asian_Width=Neutral rather than Ambiguous for both, since they have no tradition of use in East Asian typesetting.
- [WG2 5336 R4](#): misc scientific symbols (1DB22..1DB2B) [#692]
 - A maze of twisty little symbols, all alike in properties. These do not have the Pattern_Syntax property; it seems unlikely that they would be needed in default formal syntaxes.
 - They include two proportion signs, similar in usage to the existing :: PROPORTION; PROPORTION differs in that it has Ambiguous East_Asian_Width and Line_Break (vs. Neutral and Alphabetic for the new signs), owing to its presence in East Asian character sets.
 - The new symbols also include an infinity with dots above and below; besides the same ea and lb considerations that apply to PROPORTION, the new symbols differ in properties from ∞ INFINITY in that INFINITY is ID_Compat_Math_Start and _Continue. Absent evidence of a need to use the new symbols as identifiers

in scientific computing, they should not have the ID_Compat_Math_ properties. Further, INFINITY has Vertical_Orientation=Upright where the new characters have vo=Rotated. Rotated is the default for such characters without an East Asian typographical tradition. Note that related characters do not generally have the same Vertical_Orientation; consider [U+00B1](#) ± PLUS-MINUS SIGN (Upright) vs. [U+2212](#) − MINUS SIGN and [U+2212](#) ∓ MINUS-OR-PLUS SIGN (Rotated).

- The new symbols are most similar to ∷ GEOMETRIC PROPORTION, a different proportion symbol, and ÷ GEOMETRICALLY EQUAL TO, another symbol with dots above and below.
- [L2/26-141](#): 2 combining marks (20F1..20F2) [#825]
 - Combining triple solidus and saltire overlays, propertywise like the double solidus overlay, in particular Math via Other_Math.
- [L2/26-089](#): Proposal to encode the Belarusian Ruble sign to the UCS — National Bank of the Republic of Belarus [#819]
 - A new currency sign for a ruble, propertywise like most other currency signs, e.g., the RUBLE SIGN ₸. In particular, lb=PR, as the code point already was per [UTC-133-C26](#). For more on the effect of lb=PR and lb=PO see the comments on the properties of the Saudi Riyal, in [L2/25-087](#) p. 16.
- [L2/26-168](#): Additional Pushpika signs (113D9, 113DA) [#854]
 - Tailed counterparts of the existing Tulu-Tigalari pushpikas, propertywise like them.
- [L2/26-170](#): KANNADA SIGN COMBINING ANUSVARA ABOVE (0CF4) [#882]
 - An anusvara above for Kannada, propertywise like the one in Telugu, in particular Alphabetic via Other_Alphabetic. Propertywise very much unlike the existing Kannada anusvara, as that one is a spacing mark with Indic_Positional_Category=Right as opposed to a nonspacing mark with Indic_Positional_Category=Top (with the associated differences in Bidi_Class and derived properties).

5. Bidi

5.1 UAX #9 Unicode Bidirectional Algorithm vs. digits (PRI-544) [#549]

Recommended UTC actions

1. **Action Item** for Robin Leroy, PAG: Update the text of [UAX #9](#) as proposed in [L2/26-154](#) item 5.1. For Unicode 18.0. See [L2/26-154](#) item 5.1.

Feedback (verbatim)

[PRI-544](#)

Date/Time: Tue April 28 20:27:43 PT 2026

ReportID: [ID20260428202743](#)

Name: Vikki McDonough

Report Type: Report Error in Publication/Data

Opt Subject: UAX #9 Unicode Bidirectional Algorithm

UAX #9 (latest version <https://www.unicode.org/reports/tr9/tr9-51.html>) states, in part:

(section 1) "However, there are several scripts (such as Arabic or Hebrew) where the natural ordering of

horizontal text in display is from right to left. If all of the text has a uniform horizontal direction, then the ordering of the display text is unambiguous.

However, because these right-to-left scripts use digits that are written from left to right, the text is actually bidirectional: a mixture of right-to-left and left-to-right text."

(section 3.2, table 4)

None

Strong L Left-to-Right LRM, most alphabetic, syllabic, Han ideographs, non-European or non-Arabic digits, ...

R Right-to-Left RLM, Hebrew alphabet, and related punctuation

However, not all RTL scripts have LTR digits, and not all non-European-non-Arabic digits have strong left-to-right directionality. Several RTL scripts, both current (N'Ko, Mende Kikakui, Adlam) and historical (Old North and South Arabian, Imperial Aramaic, Manichaean, Parthian and Pahlavi, Palmyrene),

have digits with strong right-to-left directionality, but the current wording of these sections of UAX #9 falsely implies that all RTL scripts use LTR digits, that all non-European-non-Arabic digits have strong left-to-right directionality, and that no digits have strong right-to-left directionality.

Background information / discussion

Introduction

Section 1, being the introduction, only has the task to explain what bidirectionality is and to give some common examples. It is not the place to cover all nuances.

Proposed Text changes:

Original:

However, because these right-to-left scripts use digits that are written from left to right, the text is actually bidirectional: a mixture of right-to-left and left-to-right text. In addition to digits, embedded words from English and other scripts are also written from left to right, also producing bidirectional text. Without a clear specification, ambiguities can arise in determining the ordering of the displayed characters when the horizontal direction of the text is not uniform.

....

The bidirectional types associated with numbers are called weak types, and characters of those types are called weak directional characters.

Changed:

Whenever right-to-left scripts, for example, use digits that are written from left to right, the text is actually bidirectional: a mixture of right-to-left and left-to-right text, even in the same script. In addition to digits, embedded words from English and other scripts written from left to right, also produce bidirectional text.

Without a clear specification, ambiguities can arise in determining the ordering of the displayed characters when the horizontal direction of the text is not uniform. This specification provides for unambiguous handling of bidirectional text, no matter what issue gives rise to the bidirectionality.

The bidirectional types associated with most numbers are called weak types, and characters of those types are called weak directional characters. The digits for some scripts are strong characters, that's why these types are assigned on a per character basis and not simply inferred from the other character properties.

These two tweaks make clear that Section 1 largely uses the most typical cases to introduce the algorithm and the need for it, but does not intend to duplicate script information, whether from script intros or the data file.

Table 3.2

Table 3.2 is more detailed, but not intended to be exhaustive. I believe a relatively small fix can be made to prevent anyone from overinterpreting:

(section 3.2, table 4) (changes bolded)

Strong L Left-to-Right LRM, most alphabetic, syllabic, Han ideographs, **most** non-European or non-Arabic digits, ...

R Right-to-Left RLM, Hebrew alphabet, **and characters from other Right-To-Left scripts not assigned AL, as well as related punctuation or digits depending on the script.**

5.2 Should U+1B42, U+1B3C, and U+1B3A have Bidi_Class=L like U+0CBF? (Yes.) [#574]

Recommended UTC actions

1. **Consensus:** Change the Bidi_Class of [U+1B42](#) ꦱ BALINESE VOWEL SIGN PEPET, [U+1B3C](#) ꦱ BALINESE VOWEL SIGN LA LENGA, and [U+1B3A](#) ꦱ BALINESE VOWEL SIGN RA REPA from Nonspacing_Mark (NSM) to Left_To_Right (L). For Unicode Version 18.0. See [L2/26-154](#) item 5.2.
2. **Action Item** for Robin Leroy, PAG: In UCD file UnicodeData.txt and derived files, change the Bidi_Class of [U+1B42](#), [U+1B3C](#), and [U+1B3A](#) from Nonspacing_Mark (NSM) to Left_To_Right (L). For Unicode Version 18.0. See [L2/26-154](#) item 5.2.
3. **Note:** In Unicode Versions 5.0 through 17.0, the Bidi_Class values of [U+1B42](#) ꦱ BALINESE VOWEL SIGN PEPET, [U+1B3C](#) ꦱ BALINESE VOWEL SIGN LA LENGA, and [U+1B3A](#) ꦱ BALINESE VOWEL SIGN RA REPA (all Bidi_Class=Nonspacing_Mark) were inconsistent with those of [U+1B43](#) ꦱ BALINESE VOWEL SIGN PEPET TEDUNG, [U+1B3D](#) ꦱ BALINESE VOWEL SIGN LA LENGA TEDUNG, and [U+1B3B](#) ꦱ BALINESE VOWEL SIGN RA REPA TEDUNG (all Bidi_Class=Left_To_Right), contradicting the Property Value Stability Policy “The property values for the bidirectional properties Bidi_Class and Bidi_Mirrored preserve canonical equivalence”.

PAG input

From Robin Leroy, PAG, spotted while working on the properties for Sirmauri:

For consistency of the UBA with canonical equivalence, the nonspacing marks [U+0CBF](#) ೀ KANNADA VOWEL SIGN I and [U+0CC6](#) ೀ KANNADA VOWEL SIGN E, which appear initially in canonical decompositions of spacing marks with bc=L (namely vowel signs II ೀ, EE ೀ, AI ೀ, O ೀ, and OO ೀ, see the table below), have bc=L rather than NSM since this decision:

[93-C25] Consensus: Change the bidi property of code points [U+0CBF](#) and [U+0CC6](#) from NSM to L. [\[L2/02-194\]](#)

The same should hold for the Balinese vowel signs [U+1B42](#) ꦺ PEPET which appears at the beginning of the decomposition of [U+1B43](#) ꦺꦴ PEPET TEDUNG, [U+1B3C](#) ꦺꦴꦭꦤꦒ LA LENGA which appears at the beginning of the decomposition of [U+1B3D](#) ꦺꦴꦭꦤꦒꦠꦺꦴꦤꦒ LA LENGA TEDUNG, and [U+1B3A](#) ꦺꦴꦂꦫꦺꦴꦥ RA REPA which appears at the beginning of the decomposition of [U+1B3B](#) ꦺꦴꦂꦫꦺꦴꦥꦠꦺꦴꦤꦒ RA REPA TEDUNG, see the table below. These Balinese characters were added in 5.0, whereas decision [UTC-93-C25](#) affected the 4.0 assignments, so they were not caught at the time of [UTC-93](#). Apparently nobody added an invariant test that would have caught that. We have an invariant test that catches inconsistent non-NSM classes in decompositions, and non-initial bc=NSM in a decomposition do not matter for equivalence because they inherit the classes of preceding non-NSM characters, but initial NSM in decompositions inherit the bidi class of whatever precedes so should be guarded against.

Note that this means that the following stability policy has been violated since Version 5.0:

The property values for the bidirectional properties Bidi_Class and Bidi_Mirrored preserve canonical equivalence.

This can result in changes to the display of, e.g., Arabic text discussing Balinese vowels, but is unlikely to have an effect on most text, including any orthographic Balinese.

Arabic for « the vowel ꦺ. » and « the vowel ꦺꦴ » in NFC and NFD, as currently rendered in Chrome on Windows.

NFC	NFD
حرف العلة ة.	حرف العلة ة.
حرف العلة ه.	حرف العلة ه.

An invariant test will be added so that this does not happen again.

Relevant Kannada decompositions, nonspacing marks bold, KANNADA(VOWEL SIGN)? omitted from the names:

character	decomposition mapping	canonical decomposition if different
U+0CC0 ೆ II	U+0CBF ೆ I U+0CD5 ೆ LENGTH MARK	
U+0CC7 ೆ EE	U+0CC6 ೆ E U+0CD5 ೆ LENGTH MARK	
U+0CC8 ೆ AI	U+0CC6 ೆ E U+0CD6 ೆ AI LENGTH MARK	
U+0CCA ೆ O	U+0CC6 ೆ E U+0CC2 ೆ UU	
U+00CCB ೆ OO	U+0CCA ೆ O U+0CD5 ೆ LENGTH MARK	U+0CC6 ೆ E U+0CC2 ೆ UU U+0CD5 ೆ LENGTH MARK

Relevant Balinese decompositions, nonspacing marks bold, BALINESE VOWEL SIGN omitted from the names:

character	decomposition mapping
U+1B43 ೆ PEPET TEDUNG	U+1B42 ೆ PEPET U+1B35 ೆ TEDUNG
U+1B3D ೆ LA LENGA TEDUNG	U+1B3C ೆ LA LENGA U+1B35 ೆ TEDUNG
U+1B3B ೆ RA REPA TEDUNG	U+1B3A ೆ RA REPA U+1B35 ೆ TEDUNG

6. Normalization

6.1 Include U+11A7 in Normalization Tests (PRI-544) [#570]

Recommended UTC actions

1. **No Action:** PAG recommends no action. The test case has been added to the test data file.

Feedback (verbatim)

[PRI-544](#)

ReportID: [ID20260606121235](#)

Name: Diego Frias

Report Type: Report Error in Publication/Data

Opt Subject: Include [U+11A7](#) in Normalization Tests

0 Proposal

[U+11A7](#) (HANGUL JUNGSEONG O-YAE) is a frequent source of bugs in implementations of canonical composition. I propose to include a test in the Unicode Normalization Forms Test Data File (<https://www.unicode.org/Public/17.0.0/ucd/NormalizationTest.txt>) that targets this character.

1 Rationale

The Unicode core specification chapter 3

(<https://unicode.org/versions/Unicode17.0.0/core-spec/chapter-3/#G60469>) recommends setting the following constants

TBase = 0x11A7

TCount = 28

Almost all real-world implementations of canonical composition I have seen use these constants. The document also makes the following note:

TBase is set to one less than the beginning of the range of trailing consonants, which starts at [U+11A8](#). TCount is set to one more than the number of trailing consonants relevant to the decomposition algorithm: $(11C216 - 11A816 + 1) + 1$

However, many implementations miss this note and end up treating [U+11A7](#) as if it were a trailing consonant. Here are implementations of canonical composition that have had this bug:

Postgres

(<https://www.postgresql.org/message-id/B92ED640-7D4A-4505-B09F-3548F58CBB16%40dzfrias.de.v>)

utf8proc

(<https://github.com/JuliaStrings/utf8proc/commit/0260ba56c81e5ef6f06c0804034a36284bcb8710>)

unicode-rs

(<https://github.com/unicode-rs/unicode-normalization/blob/576ae0b1407dd14854876c93f1a348df0c19dffe/src/normalize.rs#L218>)

2 Details

I propose the following line be added to the normalization test file, in Part0 (the Specific Cases section):

AC00 11A7;AC00 11A7;1100 1161 11A7;AC00 11A7;1100 1161 11A7; # (가ㄴᄇᆞ; 가ㄴᄇᆞ; 가; 가ㄴᄇᆞ; 가;)
HANGUL SYLLABLE GA, HANGUL JUNGSEONG O-YAE

This test should go after the other specific Hangul cases tests (after line 68 for version 17.0.0).

7. Line Break

7.1 There should not be a line break between a CJK ideograph and a U+00B7 that follows it [#517]

Recommended UTC actions

1. **Consensus:** Change the Line_Break assignments of [U+00B7](#) · MIDDLE DOT from Ambiguous (AI) to Infix_Numeric (IS) and [U+0387](#) · GREEK ANO TELEIA from Alphabetic (AL) to Infix_Numeric (IS). For Unicode Version 18.0. See [L2/26-154](#) item 7.1.
2. **Action Item** for Robin Leroy, PAG: In UCD file LineBreak.txt and derived files, change the Line_Break assignments of [U+00B7](#) · MIDDLE DOT from Ambiguous (AI) to Infix_Numeric (IS) and [U+0387](#) · GREEK ANO TELEIA from Alphabetic (AL) to Infix_Numeric (IS). For Unicode Version 18.0. See [L2/26-154](#) item 7.1.
3. **Action Item** for Robin Leroy, PAG: In UCD file LineBreakTest.txt and derived files, add test cases for Chinese usage of [U+00B7](#). For Unicode Version 18.0. See [L2/26-154](#) item 7.1.

Feedback

From ICU4X-TC <https://github.com/unicode-org/icu4x/issues/7500>, in turn based on user feedback.

Background information / discussion

Simplified Chinese typography makes use of an interpunct (间隔号), see GB/T 15834—2011 §4.14, which may not appear at the beginning of the line:

5.1.7 间隔号标在需要隔开的项目之间,占半个字位置,上下居中,不出现在一行之首。

GB/T 15834 is silent on the encoding of the interpunct. In traditional Chinese, CNS 11643 maps the character 1-2126 to [U+2027](#) HYPHENATION POINT, see <https://www.cns11643.gov.tw/wordView.jsp?ID=74022>; CLDR's punctuation exemplars accordingly

have [U+2027](#) for `blo|gd|yue|zh-Hant`; note the lack of `zh=zh-Hans-CN`. [U+2027](#) has `lb=BA`, which indeed has the effect of prohibiting a break between a preceding ideograph and the interpunct.

The W3C Requirements for Chinese Text Layout (CLREQ, <https://www.w3.org/TR/clreq/#id86>) recommend the use of [U+00B7](#) MIDDLE DOT:

Due to the fact that Big5 code does not give a detailed definition of interpuncts, sometimes [U+FF0E](#) FULLWIDTH FULL STOP [。], [U+2027](#) HYPHENATION POINT [·] or [U+2022](#) BULLET [•] are used. It is recommended to use [U+00B7](#) MIDDLE DOT [·] altogether. [U+30FB](#) KATAKANA MIDDLE DOT [・] is from JIS code, and is therefore not recommended.

过去因大五码未有详细的语意定义，所以有时混用[U+FF0E](#) FULLWIDTH FULL STOP [。]、[U+2027](#) HYPHENATION POINT [·]、[U+2022](#) BULLET [•]等字符作为间隔号的例子，建议使用[U+00B7](#) MIDDLE DOT [·]。而[U+30FB](#) KATAKANA MIDDLE DOT [・]来自日文JIS编码，并非中文编码，不建议使用。

過去因大五碼未有詳細的語意定義，所以時有混用[U+FF0E](#) FULLWIDTH FULL STOP [。]、[U+2027](#) HYPHENATION POINT [·]、[U+2022](#) BULLET [•]等字元作為間隔號的例子，建議使用[U+00B7](#) MIDDLE DOT [·]。而[U+30FB](#) KATAKANA MIDDLE DOT [・]來自日文JIS編碼，並非中文編碼，不建議使用。

Accordingly CLDR punctuation exemplars have [U+00B7](#) for `gd|i|ko|sc|vec|yue|yue-Hans|zh|zh-Hant`.

As reported to the PAG, the properties of [U+00B7](#) (`lb=AI`) are inappropriate for that usage: the rules allow for a break between an ideograph and [U+00B7](#). `lb=BA` like [U+2027](#) or `LB=NS` like [U+30FB](#) would both work for Chinese usage, but would be inappropriate for other usages, as it would create a break after the *punct volat* in, e.g., Catalan *Paral·lel*. Note that another high profile usage of [U+00B7](#) is the Greek *άνω τελεία* (functionally a semicolon).

The assignment `lb=AI` was never appropriate; while the character has a shared ancestry that justifies `East_Asian_Width=Ambiguous`, it never behaves like an ideograph for line breaking.

Making it `lb=IS` (infix numeric separator), like the semicolon, the full stop, or the comma, would fix the issue without harming usages such as the *punct volat* (since there is no break between `IS` and `AL` per `LB29` nor between `AL` and `IS` per `LB15d`) or the *άνω τελεία* (which is like a semicolon). In theory, there is a case for that on its own merits: the middle dot has been used as a decimal separator, see, e.g., https://en.wikipedia.org/wiki/Decimal_separator#English-speaking_countries. The difference from `AI` (most often `=AL` in practice by `LB1`) isn't very great; this would prevent the break before the dollar sign in `(4·)$`.

The *άνω τελεία* [U+0387](#) should follow, both for consistency under canonical equivalence and because `lb=IS` is more appropriate for its specific function (it is like a semicolon).

7.2 New geometric shapes should be Line_Break=Alphabetic [#566]

Recommended UTC actions

1. **Consensus:** Change the Line_Break assignment of code points [U+1F7DB](#) and [U+1F7F1..U+1F7FF](#) from Ideographic (ID) to Alphabetic (AL). For Unicode 18.0. See [L2/26-154](#) item 7.2.
2. **Action Item** for Robin Leroy, PAG: In LineBreak.txt, change the Line_Break assignment of code points [U+1F7DB](#) and [U+1F7F1..U+1F7FF](#) from Ideographic (ID) to Alphabetic (AL). For Unicode 18.0. See [L2/26-154](#) item 7.2.

PAG input

From Charlotte Buff, PAG:

In the Beta data for Unicode 18.0, sixteen characters for mathematical use that are to be added to the *Geometric Shapes Extended* block based on document [N5330](#) – namely [U+1F7DB](#), [U+1F7F1..U+1F7FF](#) – currently have the property value `Line_Break=Ideographic`. This should be `Line_Break=Alphabetic` as a matter of consistency. Comparing these additions to existing characters:

- All characters in the *Geometric Shapes* block are `lb=AL` or `lb=AI`.
- All characters in the *Geometric Shapes Extended* block that were encoded prior to Unicode 11.0 are `lb=AL`. Most of the characters with `lb=ID` in that block are emoji.
- Nearly all math symbols (`gc=Sm`) are `lb=AL` or `lb=AI`. The only math symbols with `lb=ID` are compatibility forms derived from East Asian standards.

Furthermore, N5330 shows some of these symbols being used in such a manner that ideographic line break behaviour would cause infelicitous break opportunities. For instance, [U+1F7FF](#) RHOMBUS may be succeeded by four Latin letters defining its vertices (see illustration on page 8); this entire sequence should ideally be kept together on one line.

N5330 did not provide any direct guidance on preferred line break behaviour. These new characters were made `lb=ID` due to false analogy with [U+1F7D9](#) ☀ NINE POINTED WHITE STAR. A better comparandum would be [U+25B1](#) WHITE PARALLELOGRAM.

8. Segmentation

8.1 There is no length limit for grapheme clusters (PRI-540) [#544]

Recommended UTC actions

1. **Action Item** for Josh Hadley, PAG: In Unicode Standard Annex #29, add a note after the second paragraph of Section 3 pointing out that character count limits based on grapheme cluster breaking are not data size limits. For Unicode Version 18.0. See [L2/26-154](#) item 8.1.

Feedback (verbatim)

[PRI-540](#)

Date/Time: Fri Feb 27 05:21:53 PT 2026

ReportID: [ID20260227052153](#)

Name: Yuya Hamada

Report Type: Report Error in Publication/Data

Opt Subject: There is no limit codepoint in grapheme cluster

Ref: <https://unicode-org.atlassian.net/browse/ICU-23302>

Grapheme cluster is no limit codepoint in grapheme cluster.

That means grapheme cluster can include many codepoints.

However, Computer resources is limited, so computer can crash(Denial of Service) when many codepoints but 1 grapheme cluster.

I think include sentence "grapheme cluster max codepoint is limited" in UAX#29.

Max codepoint is suitable value is 32 because there is that sentence in UAX#15 -D3

[UAX #15](#)-D3. Stream-Safe Text Format: A Unicode string is said to be in Stream-Safe Text Format if it would not contain any sequences of non-starters longer than 30 characters in length when normalized to NFKD.

<https://unicode.org/reports/tr15/#UAX15-D3>

Therefore, I suggest to max codepoint is 32 in UAX#29.

Regards

Background information / discussion

The use case given by the submitter in the discussion in on the ICU ticket, in <https://unicode-org.atlassian.net/browse/ICU-23302?focusedCommentId=195532>, is limiting the total data size of a form submission. Grapheme cluster count is not suited for this purpose anymore than word count would be; both are a limit on human-perceived size for reasonable text.

For the purposes of limiting data size, a limit on data size (in the appropriate encoding) should be used. As noted in discussion on the ICU ticket, a limit on user-perceived character count implemented by GCB can be used together with a limit on data size, and is used in practice in the industry, with the AT Protocol limiting posts to 300 extended grapheme clusters as well as 3000 bytes:

<https://github.com/bluesky-social/atproto/blob/0193467463a1a49c0c7f17d129c07b687a1f5057/lexicons/app/bsky/feed/post.json#L13-L18>.

[UAX #29](#) already mentions *"character" counting for text size limits* in the second paragraph of Section 3. We could add a note after this paragraph that such a limit is not a data size limit, and that data size should be checked separately.

Note: A text size limit based on a count of grapheme clusters is a user-perceived limit, just like a word count; it is not a data size limit. There is no limit on the size of a grapheme cluster as defined in this section. When limits on data size are desirable, they should be used together with word or character count limit. For instance, a form might simultaneously restrict submissions to 300 “characters” defined by a count of grapheme clusters and to 3000 UTF-8 bytes to ensure a bound on storage size.

No need is demonstrated by the submitter for limiting the size of any individual grapheme cluster, let alone for a mechanism similar to stream-safe normalization, so there is no need to discuss such a possibility even as a tailoring. In particular, the submitter does not appear to be performing grapheme cluster breaking on a stream.

8.2 Align UTS #51 & UAX #29 for flags and keycaps in ZWJ (PRI-543) [#545]

Recommended UTC actions

1. **Action Item** for Josh Hadley, PAG: In Unicode Standard Annex #29, in Section 3.1.1, Grapheme Cluster Boundary Rules, correct the note about emoji sequences as described in [L2/26-154](#) item 8.2. For Unicode Version 18.0.

Feedback (verbatim)

[PRI-543](#) PU-[UTS #51](#) Emoji

Date/Time: Fri April 17 11:53:25 PT 2026

ReportID: [ID20260417115325](#)

Name: Kuen-Bang Hou (Favonia)

Report Type: Report Error in Publication/Data

Opt Subject: Align UTS#51 & UAX#29 for flags and keycaps in ZWJ

[UTS #51](#) and [UAX #29](#) appear to be misaligned here.

In UTS #51, ED-15a allows `emoji_flag_sequence` and `emoji_keycap_sequence` to participate in emoji ZWJ sequences via `emoji_core_sequence`, and ED-17 treats the resulting sequences as emoji sequences. UTS #51 further states that all emoji sequences are single grapheme clusters. UAX #29 likewise states that each emoji sequence is a single grapheme cluster.

However, the corresponding default boundary rules in UAX #29 still rely on `Extended_Pictographic`. In particular, GB11 specifies `×` (no boundary) only for the pattern `Extended_Pictographic Extend* ZWJ × Extended_Pictographic`, while emoji flag sequences are handled separately by GB12/GB13, and keycap-based cases are not brought under the same `Extended_Pictographic`-based framework.

As a result, some sequences now admitted by UTS #51 are not consistently treated as single grapheme clusters by the default segmentation rules in UAX #29. A previously discussed remedy was to change the `Extended_Pictographic` assignments of the keycap bases and `Regional_Indicator` characters, but no action was taken ([PRI-453](#)). Whether or not that remedy is adopted, I urge the committee to resolve this misalignment between the two specifications.

Background information / discussion

ZWJ sequences involving flags were deliberately made well-formed and valid by this decision:

[[171-C27](#)] Consensus: Accept proposed changes to UTS #51, based on document [L2/22-094](#) and Section 2 of document [L2/22-066R](#), for Unicode version 15.0.

and document [L2/22-094R](#) includes a hypothetical example of such a ZWJ sequence.

At the same time, absent evidence frequent of use of such sequences that would motivate a change to the rules of GCB (always an onerous change since they are independently implemented all over the place, including in standard libraries of programming languages), there is no need for them to be treated as single extended grapheme clusters. Conformance requirement <https://unicode.org/reports/tr51/#C2> only involves support for editing (which treatment as single grapheme clusters is a part of) for RGI sequences. Section 1.5 of [UTS #51](#) has no requirement nor even recommendation of support for editing of non-RGI emoji sequences, only a permission.

However, this note in [UAX #29](#) is incorrect:

- Each *emoji sequence* is a single grapheme cluster. See definition ED-17 in Unicode Technical Standard #51, "Unicode Emoji" [[UAX51](#)].

It should be corrected as follows:

- Each *emoji sequence* in the *RGI emoji set* is a single grapheme cluster. See definitions ED-17 and ED-27 in Unicode Technical Standard #51, "Unicode Emoji" [[UAX51](#)]. Grapheme clusters can therefore be used to implement support for editing of RGI emoji sequences, as defined by conformance clause C2 of [[UAX51](#)].
- Some valid emoji sequences that are not recommended for general interchange comprise multiple grapheme clusters.

8.3 FYI: UTC-183 broke the Zanabazar square grapheme cluster 𐰇+𐰆=𐰇, but UTC-187 put it together again [#551]

Recommended UTC actions

1. **No Action:** PAG recommends no action: the regression introduced in Unicode Version 17.0 was fixed by decisions [UTC-187-C47](#) and [UTC-187-48](#).

PAG input

From Robin Leroy, PAG:

In Berkeley, UTC decided

[187-C47] Consensus: Change Rule GB9c in UAX #29 to $\backslash p\{\text{InCB=Linker}\} \backslash p\{\text{InCB=Extend}\}^* \times \backslash p\{\text{InCB=Consonant}\}$. For Unicode Version 18.0. See [L2/26-096](#) item 5.1.

[187-C48] Consensus: Change the derivation of InCB=Linker in UAX #44 to include InSC=Consonant_With_Stacker. For Unicode Version 18.0. See [L2/26-096](#) item 5.1.

I have an action item

[187-A135] 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 Version 18.0. See [L2/26-096](#) item 5.1.

We forgot to record an action item for me to add realistic test cases affected by these changes to GraphemeClusterBreak.txt, but I did it nonetheless.

One of the InSC=Consonant_With_Stacker characters is [U+11A3A](#) 𐰇 ZANABAZAR SQUARE CLUSTER-INITIAL LETTER RA, and one example cluster with it given in the encoding proposal

[L2/14-024](#), p. 15 is 𑌕 (with 𑌕 KA).

Interestingly, this used to be one extended grapheme cluster, and still is in most implementations: until Unicode Version 17.0, 𑌕 was InSC=Consonant_Prefixed and therefore Grapheme_Cluster_Break=Prepend.

The 17.0 change to Consonant_With_Stacker (whence gcb=Other) was decided at [UTC-183](#):

[183-C15] Consensus: Update the Indic_Syllabic_Category of [U+11A3A](#) ZANABAZAR SQUARE CLUSTER-INITIAL LETTER RA to Consonant_With_Stacker and its Indic_Positional_Category to NA (not applicable) as described in [L2/25-119](#), for Unicode Version 17.0. [Ref. 3.2 in [L2/25-091R](#)]

Document [L2/25-119](#) did not discuss grapheme cluster breaking.

Note that the change to the Soyombo cluster-initial RA had no effect on grapheme cluster breaking, as its new InSC Consonant_Preceding_Repha still lands it in gcb=Prepend.

With the decisions from [UTC-187](#), 𑌕 is one extended grapheme cluster again (now by rule GB9c rather than GB9b).

This transient regression is unfortunate, as GCB is widely implemented at a fairly low level; for example, it is the indexable unit in Swift strings, and it is used to determine padding in C++ std::format.

8.4 Should the derivation of Indic_Conjunct_Break use Script_Extensions instead of Script? (Yes.) [#552]

Recommended UTC actions

1. **Consensus:** In Unicode Standard Annex #44, Unicode Character Database, update the derivation of the Indic_Conjunct_Break property to use the Script_Extensions (scx) rather than the Script (sc) property. For Unicode Version 18.0. See [L2/26-154](#) item 8.4.
2. **Action Item** for Ken Whistler, PAG: In Unicode Standard Annex #44, Unicode Character Database, update the derivation of the Indic_Conjunct_Break property to use the Script_Extensions (scx) rather than the Script (sc) property; include a note that $\backslash p\{scx=S\}$ means the set of characters that have S as one of their Script_Extensions values. For Unicode Version 18.0. See [L2/26-154](#) item 8.4.
3. **Action Item** for Robin Leroy, PAG: Update UCD data files to reflect the changed derivation of Indic_Conjunct_Break. For Unicode Version 18.0. See [L2/26-154](#) item 8.4.

PAG input

From Robin Leroy, PAG:

In Berkeley, UTC decided

[187-C47] Consensus: Change Rule GB9c in UAX #29 to `\p{InCB=Linker}\p{InCB=Extend}* × \p{InCB=Consonant}`. For Unicode Version 18.0. See [L2/26-096](#) item 5.1.

[187-C48] Consensus: Change the derivation of `InCB=Linker` in UAX #44 to include `InSC=Consonant_With_Stacker`. For Unicode Version 18.0. See [L2/26-096](#) item 5.1.

I have an action item

[187-A135] 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 Version 18.0. See [L2/26-096](#) item 5.1.
Start with a line like "From (names), PAG" or similar.

We forgot to record an action item for me to add realistic test cases affected by these changes to `GraphemeClusterBreak.txt`, but I did it nonetheless; specifically, I added `×` and `ω` based on <https://unicode.org/versions/Unicode17.0.0/core-spec/chapter-12/#G723286>, and I would have expected them to become one extended grapheme cluster with these changes, based on the discussion in this section of the core specification (with the ZWNJed versions remaining two extended grapheme clusters). That is however not the case; while the `jihvamuliya` and `upadhmaniya` are `InSC=Consonant_With_Stacker`, they are not in the set *conjunctLinkingScripts* defined in Section 5.3.1 of [UAX #44](#), https://www.unicode.org/reports/tr44/tr44-36.html#Derivation_InCB. The reason is that while Bengali is one of those scripts (specifically, it is one of the *viramaScripts*), `×` and `ω` have `Script=Common`, and `scx=Bengali Devanagari` (though they will not stay Devanagari for long, see [UTC-186-A48](#) and [UTC-187-A35](#)).

8.5 Idea for changes to word boundary segmentation with hyphen-separated strings like UUID/GUID [#572]

Recommended UTC actions

1. **Action Item** for Asmus Freytag, PAG: Update UAX #29 to clarify wording around the use of hyphens in special text types as described in [L2/26-154](#) item 8.5. For Unicode 18.0. See [L2/26-154](#) item 8.5.

Feedback

From Daniel White via email directly to Josh Hadley

One challenge I have with the word boundary rules is that many text editors have double click to indicate select a word. However, there are times when I would like to copy a UUID/GUID or other string with hyphens. Since hyphens can indicate a word boundary, this breaks the UX and I have to resort to manual highlighting. Changing the format may hurt readability or parsing in other contexts.

My idea - could there be a word boundary start and end markers? For example, double clicking the value be the complete uuid without the range delimiters, instead of one of the segments.

[BEGIN_WORD_RANGE]019ec42f-93a6-7fb0-9cbd-56ba92d7845f[/END_WORD_RANGE]

Background information / discussion

Default Word Breaking is a compromise that cannot address all contexts or types of textual material. Additional strategies outside the word breaking algorithm, or specific tailorings may be needed. This is discussed in UAX #29 in a series of notes. However, editorially, it would be more useful to elevate that discussion into a new section, 4.2. Non-Default Word Breaking. Note also the new specification on link detection in UTS #58.

Specifically:

Insert a new 4.2 before the existing 4.2, named "Non-Default Word Breaking" and populate it with the contents of the "Notes". Related issues can be grouped by headers:

Opening para for 4.3 = first bullet

4.2.1 Hyphens (I would use the plural, because we talk about more than 002D)

4.2.2 Apostrophe

4.2.3 Colon

4.2.4 Hebrew

4.3.5 Other (should include format characters)

Then add a brief pointer to UUIDs, URLs and Phone numbers, etc etc. to "Hyphen" and or "Colon" or add the discussion of piggybacking off linkification into "Other".

8.6 Definition of "Grapheme extender" is incorrect (PRI-548) [#573]

Recommended UTC actions

1. **Action Item** for Robin Leroy, EDC: Clean up bullet points of D59 of section 3.6.2 of the Core Spec. For Unicode 18.0. See [L2/26-154](#) item 8.6.

Feedback (verbatim)

[PRI-548](#)

Date/Time: Mon June 22 15:50:26 PT 2026

ReportID: [ID20260622155026](#)

Name: Jules Bertholet

Report Type: Public Review Issue

Opt Subject: Definition of "Grapheme extender" is incorrect

D59 in §3.6.3 of the Core Specification says:

D59 Grapheme extender: A character with the property Grapheme_Extend.

Grapheme extender characters consist of all nonspacing marks, ZERO WIDTH JOINER, ZERO WIDTH NON-JOINER, [U+FF9E](#) HALFWIDTH KATAKANA VOICED SOUND MARK, [U+FF9F](#) HALFWIDTH KATAKANA SEMI-VOICED SOUND MARK, and a small number of spacing marks.

A grapheme extender can be conceived of primarily as the kind of nonspacing graphical mark that is applied above or below another spacing character.

ZERO WIDTH JOINER and ZERO WIDTH NON-JOINER are formally defined to be grapheme extenders so that their presence does not break up a sequence of other grapheme extenders.

[...]

However, this definition does not match the actual property assignments: [U+200D](#) ZERO WIDTH JOINER does not have the Grapheme_Extend property in DerivedCoreProperties.txt.

I believe this should be corrected not by changing the spec, but instead by changing the property assignments

in [DerivedCoreProperties.txt](#) to make [U+200D](#) be Grapheme_Extend, and changing the derivation of Grapheme_Cluster_Break=Extend to exclude 200D.

Background information / discussion

Remove ZWJ from bullet points for D59 in section 3.6.2. Strengthen references to UAX #29.

9. Collation

9.1 UCA: remove Shift-Trimmed option [#553]

Recommended UTC actions

1. **Consensus:** In the UCA spec, remove the Variable Weighting option Shift-Trimmed. For Unicode 18.0. See [L2/26-154](#) item 9.1.
2. **Action Item** for Markus Scherer, PAG: In the UCA spec, remove the Variable Weighting option Shift-Trimmed. For Unicode 18.0. See [L2/26-154](#) item 9.1.
3. **Action Item** for Mark Davis, PAG: Create a CLDR ticket to deprecate the name of the option. See [L2/26-154](#) item 9.1.

PAG input

From Markus Scherer, PAG:

[UCA section 4 Variable Weighting](#) defines this option:

- **Shift-Trimmed:** This option is the same as **Shifted**, except that all trailing FFFFs are trimmed from the sort key. This could be used to emulate POSIX behavior, but is otherwise not recommended.

This option

- is defined but not recommended
- makes the only mention in this spec of POSIX (without a reference link)
- violates well-formedness condition [WF1](#) (“no collation element can have a zero weight at Level N and a non-zero weight at Level N-1”)
- does not have conformance test data
- is not implemented in ICU

and we (Ken Whistler and I) believe that this is an unnecessary complication in this spec.

10. Regex

10.1 Typo in UTS18 [#564]

Recommended UTC actions

1. **Action Item** for Mark Davis, Robin Leroy, PAG: Fix the regular expression referenced in [L2/26-154](#) item 10.1 in [UTS #18](#) in a future version.

PAG input

From Roozbeh Pournader, PAG:

[UTS #18](#), Section 2.2.1, contains the following regular expression:

```
[p\{RGI_Emoji\}--[a-z👉\q{ch|sch|🇧🇪|🇩🇪|🇪🇪 }]]
```

The `p\` at the beginning should be replaced by `\p`.

11. Security

11.1 Identifier_Type data for new Unicode 18.0 characters [#530]

Recommended UTC actions

1. **No Action:** PAG recommends no action. The data files are (or will soon be) updated.

PAG input

From Roozbeh Pournader, PAG:

The following are the proposed Identifier_Type values for the new letters in Unicode 18.0. Only one character, [U+11B0A](#) DEVANAGARI LETTER ALTERNATE DDDA, is "Recommended" for identifiers, based on evidence that it is used in modern Sindhi orthography. The rest are restricted, and classified according to their usage based on proposals leading to their encoding.

The status of one character, [U+2B81E](#) CJK UNIFIED IDEOGRAPH-2B81E, could not be determined by the author. It is a disunification of [U+6B25](#) 𠂔 CJK UNIFIED IDEOGRAPH-6B25, which has Identifier_Type=Recommended, but it was unclear if the new character should be Recommended or Uncommon_Use. Ken Lunde, the chair of the CJK & Unihan Working Group, recommended Uncommon_Use.

0558 ; Not_NFKC # otherwise would be Technical

058B..058C ; Not_NFKC # otherwise would be Technical

05C8..05C9 ; Technical

0B53..0B54 ; Uncommon_Use
1ADE..1ADF ; Technical
1AEC..1AF0 ; Technical
208F ; Technical
209D..209F ; Not_NFKC # otherwise would be Technical
A7DD ; Technical Obsolete
A7E2 ; Obsolete
AB6C..AB6D ; Technical
107BB..107BF ; Not_NFKC # otherwise would be Technical
10EC9..10ECF ; Technical
10ED9..10EEE ; Obsolete
10EF0..10EF8 ; Technical
10EF9 ; Obsolete
11B0A ; Recommended
11DF0 ; Obsolete
11DF1 ; Technical
1B123..1B126 ; Not_NFKC # otherwise would be Obsolete
1B127..1B128 ; Obsolete
1B168 ; Technical
1DF1F..1DF24 ; Technical
1DF2B..1DF59 ; Technical
1DF5A..1DF66 ; Uncommon_Use
1DF67 ; Technical
1DF68..1DF81 ; Technical Obsolete
1DF90..1DF96 ; Technical Obsolete # except 1DF95 is Not_NFKC
1DFCD..1DFFF ; Not_NFKC # otherwise would be Technical
2B81E ; Uncommon_Use

11.2 Identifier_Type of Arabic and Hebrew combining marks [#555]

Recommended UTC actions

1. **Consensus:** Update the Identifier_Type of Arabic and Hebrew combining marks per [L2/26-154](#) item 11.2. For Unicode 18.0.
2. **Action Item** for Roozbeh Pournader and Josh Hadley, PAG: Update IdentifierType.txt to reflect the new Identifier_Type values of Arabic and Hebrew combining marks per [L2/26-154](#) item 11.2. For Unicode 18.0.

PAG input

From Roozbeh Pournader, PAG:

We have been struggling to add clean Identifier_Type values for some Unicode 18.0 characters; the tooling was causing some unintended additions. I investigated the tooling, and the source of the additions appear to be these two regular expressions in an input file:

None

```
[[[:sc=Hebr:]]&[:gc=Mn:]] ; uncommon-use # Hebrew - Cantillation marks  
[[[:sc=Arab:]]&[:gc=Mn:]]-[\u064B-\u0655\u00010EFB]] ; uncommon-use # Arabic nonspacing marks (some exceptions)
```

We can hack our way around this by adding more exceptions to the Arabic and Hebrew lists, but looking at other Arabic and Hebrew combining marks tells me that their Identifier_Type values are far from desirable. I have examined them carefully and proposed proper Identifier_Type values for them below, so we can remove these two regular expressions.

These are my proposed Identifier_Types for Hebrew and Arabic combining marks:

None

```
# Hebrew  
0591..05AF; Technical # cantillation marks  
05B0..05BD; Uncommon_Use # points  
05BF; Technical Uncommon_Use # rafe  
05C1..05C2; Uncommon_Use # shin and sin dots  
05C4..05C5; Technical # puncta extraordinaria  
05C7; Technical # qamats qatan  
05C8..05C9; Technical # sheva na mudgash and dagesh hazaq mudgash  
FB1E; Uncommon_Use # Judeo-Spanish varika  
  
# Arabic  
0610..0614; Uncommon_Use # combining honorifics  
0615; Technical # quranic mark  
0616; Obsolete # used in early Persian  
0617..061A; Technical # quranic marks
```

```
0656..065B; Uncommon_Use # various marks
065C; Uncommon_Use Technical # used both in African languages and as quranic mark
065D..065F; Uncommon_Use # various marks
06D6..06DC; Technical # quranic marks
06DF..06E4; Technical # quranic marks
06E7..06E8; Technical # quranic marks
06EA..06ED; Technical # quranic marks
0897; Uncommon_Use # pepet
0898..089F; Technical # quranic marks
08CA..08E1; Technical # quranic marks
08E3..08EF; Uncommon_Use # various marks
08F0..08F3; Technical # quranic marks
08F4..08F7; Uncommon_Use # various marks
08F8; Uncommon_Use Technical # used both in African languages and as quranic mark
08F9..08FA; Uncommon_Use # various marks
08FB; Uncommon_Use Technical # used both in African languages and as quranic mark
08FC..08FF; Uncommon_Use # various marks
10ECB..10ECF; Technical # quranic marks
10EF0..10EF8; Technical # quranic marks
10EF9; Obsolete # combining crown
10EFA; Obsolete # used in Old Sindhi
10EFB..10EFF; Technical # quranic marks
```

11.3 UTS39 needs to support ISO 15924 code HntI [#560]

Recommended UTC actions

1. **Consensus:** In Unicode Standard Annex #39, Unicode Security Mechanisms, take HntI into account in the Mixed-script detection algorithm. For Unicode Version 18.0. See [L2/26-154](#) item 11.3.
2. **Action Item** for Josh Hadley and Roozbeh Pournader, PAG: Add HntI to Section 5.1 of [UTS #39](#). For Unicode 18.0. See [L2/26-154](#) item 11.3.

PAG input

From Roozbeh Pournader, PAG:

[UTS #39](#), in its [Section 5.1, Mixed-Script Detection](#), contains an algorithm for assigning an augmented script set to a character. It has rules about all multi-script ISO 15924 codes, except for the new HntI code, added in 2025. The algorithm needs to be modified to add HntI (Hant+Latn).

Here are the changes proposed, with additions in ***bold italic***:

Entries for the writing systems containing multiple scripts — Hanb (Han with Bopomofo), ***Hntl*** (***Traditional Han with Latin***), Jpan (Japanese), and Kore (Korean) — are added according to the following rules.

1. If Script_Extensions contains Hani (Han), add Hanb, ***Hntl***, Jpan, and Kore.
2. If Script_Extensions contains Hira (Hiragana), add Jpan.
3. If Script_Extensions contains Kana (Katakana), add Jpan.
4. If Script_Extensions contains Hang (Hangul), add Kore.
5. If Script_Extensions contains Bopo (Bopomofo), add Hanb.
6. ***If Script_Extensions contains Latn (Latin), add Hntl.***

Additionally, the note in the same section about the term “single-script string” needs to be updated to say “five” scripts and Hntl to the set.

Also, Table 1a needs to be updated to add Hntl in various places, where Latn and Hani are mentioned.

Further changes in the same section, as well as Section 5.2, may also be necessary.

Note that this has implications on Restriction Levels of some strings, as defined in [Section 5.2](#) of [UTS #39](#). For example, the Restriction level of the string "X射线" ("X-ray" in Simplified Chinese) was "Highly Restrictive" before, but after this change, it will be "Single Script".

11.4 Joiners vs combining marks in Arabic and similar scripts [#562]

Recommended UTC actions

1. **Consensus:** Change the regular expression in Section 3.1.1.1 of [UTS #39](#) from `/\$LJ \$T* ZWNJ \$T* \$RJ/` to `/\$LJ \$T* ZWNJ \$RJ/`, for Unicode 18.0. See [L2/26-154](#) item 11.4.
2. **Action item** for Josh Hadley and Roozbeh Pournader, PAG: Change the regular expression in Section 3.1.1.1 of [UTS #39](#) from `/\$LJ \$T* ZWNJ \$T* \$RJ/` to `/\$LJ \$T* ZWNJ \$RJ/`, for Unicode 18.0. See [L2/26-154](#) item 11.4.
3. **Action item** for Roozbeh Pournader, EDC: Add text to the Core Specification to recommend that combining marks should appear before joiners in Arabic and similar scripts. For Unicode 18.0. See [L2/26-154](#) item 11.4.

PAG input

From Roozbeh Pournader, PAG:

I was reviewing [UTS #39](#), when I noticed that it provides this regular expression in [Section 3.1.1.1](#) for when ZWNJ may be allowed in identifiers in Arabic script and similarly joining scripts:

```
/\$LJ \$T* ZWNJ \$T* \$RJ/
```

Ignoring the letters, the regular expression is allowing `Joining_Type=Transparent` characters on both sides of the ZWNJ! This made me realize that we don't specify any preference in the standard about the way Arabic combining marks are supposed to interact with joiners.

For example, the following three sequences, no two of which are canonically equivalent, are displayed identically or almost identically in most environments on the latest version of Android (Android 16):

لَّب
لَّب
لَّب

(The sequences are `<lam, shadda, ZWNJ, fatha, beh>`, `<lam, shadda, fatha, ZWNJ, beh>`, and `<lam, ZWNJ, shadda, fatha, beh>`.)

I don't think rendering engines or fonts necessarily need to distinguish how these three sequences are displayed, but we need to have some advice in the Core Spec about which way is the preferred representation of such text and also restrict the regular expression in [UTS #39](#).

(Fortunately, the confusability algorithm in [UTS #39](#) marks the three strings as confusable with each other because it removes the ZWNJ from the strings' skeleton before comparing them.)

This is not a theoretical case either: such strings actually happen in Persian texts. For example, in the word صُمِّبْكُمْ (an Arabic loanword that literally means “deaf and mute” but practically means “absolutely silent” in modern Persian), the first *meem* needs to be final (thus needing a ZWNJ) as well as accented with a *shadda* and a *dammatan* (to help the uninitiated pronounce the word properly).

My recommendation is going with the combining marks first. There are two main reasons:

1. Semantically, the ZWNJ is separating two subwords, and the combining marks are pronounced with the first subword.
2. Rendering engines may have problems with optimally placing the combining marks over the letter if a ZWNJ intervenes. (This is visible in Windows' Notepad for example.)

I propose that we add a paragraph to the Core Spec specifying that in such sequences, we recommend that the combining marks occur before the ZWJ/ZWNJ. I also propose that in [UTS #39](#), we change the regular expression to say `/\$LJ \$T* ZWNJ \$RJ/`, removing the transparent characters after the ZWNJ.

12. Confusables

12.1 Confusables data from Google Summer of Code 2025 [#484]

Recommended UTC actions

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

Confusables source

In the summer of 2025, a Google Summer of Code project to detect confusable pairs was done by Om Doiphode and supervised by Frank Yung-Fong Tang and Shane Carr. The report is available at <https://gist.github.com/Om-Doiphode/d2cb46f25c61e5d8a0dd133bfe5b074c>.

First phase of data based on a partial review by Roozbeh Pournader follows. Unfortunately there are too many false positives in the original data, needing very careful review. More pairs will be proposed based on further review of the remaining data.

Data

0024 ; 0586 # \$ DOLLAR SIGN ≈ Ɑ ARMENIAN SMALL LETTER FEH

0027 ; 055B # ' APOSTROPHE ≈ ʹ ARMENIAN EMPHASIS MARK

0027 ; A78B # ' APOSTROPHE ≈ ' LATIN CAPITAL LETTER SALTILLO

0028 ; 10141 # (LEFT PARENTHESIS ≈ Ϸ GREEK ACROPHONIC ATTIC ONE HALF

0029 ; 10140 #) RIGHT PARENTHESIS ≈ ϸ GREEK ACROPHONIC ATTIC ONE QUARTER

002B ; 1540 # + PLUS SIGN ≈ + CANADIAN SYLLABICS WEST-CREE Y

002D ; 174D # - HYPHEN-MINUS ≈ – BUHID LETTER RA

0031 ; 16D0 # 1 DIGIT ONE ≈ ᚠ RUNIC LETTER SHORT-TWIG-TYR T

0032 ; 1616 # 2 DIGIT TWO ≈ ᚡ CANADIAN SYLLABICS CARRIER JO

0033 ; 0545 # 3 DIGIT THREE ≈ Յ ARMENIAN CAPITAL LETTER YI

0033 ; 1702 # 3 DIGIT THREE ≈ 3 TAGALOG LETTER U

0033 ; 1C95 # 3 DIGIT THREE ≈ 3 GEORGIAN MTAVRULI CAPITAL LETTER VIN

0034 ; AB9E # 4 DIGIT FOUR ≈ 4 CHEROKEE SMALL LETTER SE

0036 ; ABBE # 6 DIGIT SIX ≈ 6 CHEROKEE SMALL LETTER WV

0037 ; 172A # 7 DIGIT SEVEN ≈ 7 HANUNOO LETTER BA

0037 ; 1C84 # 7 DIGIT SEVEN ≈ 7 CYRILLIC SMALL LETTER TALL TE

0039 ; A76F # 9 DIGIT NINE ≈ 9 LATIN SMALL LETTER CON

003A ; 1393 # : COLON ≈ : ETHIOPIC TONAL MARK SHORT RIKRIK

003A ; 1361 # : COLON ≈ : ETHIOPIC WORDSPACE

003A ; 1015B # : COLON ≈ : GREEK ACROPHONIC EPIDAUREAN TWO

004E ; 11ABE # N LATIN CAPITAL LETTER N ≈ N CANADIAN SYLLABICS SPO

004F ; 1CBF # O LATIN CAPITAL LETTER O ≈ O GEORGIAN MTAVRULI CAPITAL LETTER LABIAL

SIGN

0053 ; 10BD # S LATIN CAPITAL LETTER S ≈ S GEORGIAN CAPITAL LETTER CHAR
0053 ; 10FD # S LATIN CAPITAL LETTER S ≈ S GEORGIAN LETTER AEN
0053 ; 1CBD # S LATIN CAPITAL LETTER S ≈ S GEORGIAN MTAVRULI CAPITAL LETTER AEN
005A ; 10CD # Z LATIN CAPITAL LETTER Z ≈ Z GEORGIAN CAPITAL LETTER AEN
005A ; 11ABC # Z LATIN CAPITAL LETTER Z ≈ Z CANADIAN SYLLABICS SPE
005A ; 2C8C # Z LATIN CAPITAL LETTER Z ≈ Z COPTIC CAPITAL LETTER ZATA

005E ; 18D4 # ^ CIRCUMFLEX ACCENT ≈ ^ CANADIAN SYLLABICS OJIBWAY P
0060 ; 2CFF # ` GRAVE ACCENT ≈ ` COPTIC MORPHOLOGICAL DIVIDER

0068 ; 10B9 # h LATIN SMALL LETTER H ≈ h GEORGIAN CAPITAL LETTER CHIN
0068 ; 04BA # h LATIN SMALL LETTER H ≈ h CYRILLIC CAPITAL LETTER SHHA
006C ; A781 # l LATIN SMALL LETTER L ≈ l LATIN SMALL LETTER TURNED L
006C ; A7FE # l LATIN SMALL LETTER L ≈ l LATIN EPIGRAPHIC LETTER l LONGA
0077 ; ABA4 # w LATIN SMALL LETTER W ≈ w CHEROKEE SMALL LETTER TA
007A ; 2C8D # z LATIN SMALL LETTER Z ≈ z COPTIC SMALL LETTER ZATA
007A ; 2D2D # z LATIN SMALL LETTER Z ≈ z GEORGIAN SMALL LETTER AEN

00AF ; 1428 # ¯ MACRON ≈ ¯ CANADIAN SYLLABICS FINAL SHORT HORIZONTAL STROKE
00B0 ; 00BA # ° DEGREE SIGN ≈ ° MASCULINE ORDINAL INDICATOR
00B0 ; 1424 # ° DEGREE SIGN ≈ ° CANADIAN SYLLABICS FINAL RING
00B0 ; 18DE # ° DEGREE SIGN ≈ ° CANADIAN SYLLABICS FINAL SMALL RING
00B2 ; 14BE # ² SUPERScript TWO ≈ ² CANADIAN SYLLABICS ATHAPASCAN M
00B9 ; 18F3 # ¹ SUPERScript ONE ≈ ¹ CANADIAN SYLLABICS BEAVER DENE L

00D7 ; 1763 # × MULTIPLICATION SIGN ≈ × TAGBANWA LETTER KA

00DE ; 16A6 # Þ LATIN CAPITAL LETTER THORN ≈ Þ Runic LETTER THURISAZ THURS THORN
00DF ; A7D7 # ß LATIN SMALL LETTER SHARP S ≈ ß LATIN SMALL LETTER MIDDLE SCOTS S

00F7 ; 1365 # ÷ DIVISION SIGN ≈ ÷ ETHIOPIC COLON

0142 ; 16C5 # ł LATIN SMALL LETTER L WITH STROKE ≈ ł Runic LETTER LONG-BRANCH-AR
AE

017F ; 1017E # ſ LATIN SMALL LETTER LONG S ≈ ſ GREEK THREE OBOLS SIGN

018B ; 018C # đ LATIN CAPITAL LETTER D WITH TOPBAR ≈ đ LATIN SMALL LETTER D WITH
TOPBAR

018E ; 1D228 # Ǝ LATIN CAPITAL LETTER REVERSED E ≈ Ǝ GREEK INSTRUMENTAL NOTATION
SYMBOL-18

0192 ; 0284 # f LATIN SMALL LETTER F WITH HOOK ≈ f LATIN SMALL LETTER DOTLESS J
WITH STROKE AND HOOK

01A7 ; 10C7 # Ɔ LATIN CAPITAL LETTER TONE TWO ≈ Ɔ GEORGIAN CAPITAL LETTER YN

01A7 ; 1CB7 # Ɔ LATIN CAPITAL LETTER TONE TWO ≈ Ɔ GEORGIAN MTAVRULI CAPITAL
LETTER YN

01AE ; A68C # Ƨ LATIN CAPITAL LETTER T WITH RETROFLEX HOOK ≈ Ҁ CYRILLIC CAPITAL LETTER TWE

0239 ; 03C6 # ꝥ LATIN SMALL LETTER QP DIGRAPH ≈ ϕ GREEK SMALL LETTER PHI

0249 ; 025F # ȶ LATIN SMALL LETTER J WITH STROKE ≈ ȵ LATIN SMALL LETTER DOTLESS J WITH STROKE (would appear identical if an accent above is added)

0251 ; AB64 # α LATIN SMALL LETTER ALPHA ≈ ɒ LATIN SMALL LETTER INVERTED ALPHA

02D9 ; 18DF # ˙ DOT ABOVE ≈ ˙ CANADIAN SYLLABICS FINAL RAISED DOT

02ED ; 167F # ˆ MODIFIER LETTER UNASPIRATED ≈ ˆ CANADIAN SYLLABICS BLACKFOOT W

0304 ; 1752 # ˘ COMBINING MACRON ≈ ̄ BUHID VOWEL SIGN I

03B8 ; 1340 # θ GREEK SMALL LETTER THETA ≈ ፬ ETHIOPIC SYLLABLE TZA

0416 ; 1D219 # Ж CYRILLIC CAPITAL LETTER ZHE ≈ Χ GREEK VOCAL NOTATION SYMBOL-51

043E ; 1C82 # о CYRILLIC SMALL LETTER O ≈ 0 CYRILLIC SMALL LETTER NARROW O

0441 ; 1C83 # с CYRILLIC SMALL LETTER ES ≈ с CYRILLIC SMALL LETTER WIDE ES

14BC ; A7F2 # ˆ CANADIAN SYLLABICS WEST-CREE M ≈ ˆ MODIFIER LETTER CAPITAL C

1EFA ; 0049 004C # ll LATIN CAPITAL LETTER MIDDLE-WELSH LL ≈ I LATIN CAPITAL LETTER I, L LATIN CAPITAL LETTER L

207C ; 167F # ˆ SUPERScript EQUALS SIGN ≈ ˆ CANADIAN SYLLABICS BLACKFOOT W

A742 ; 10182 # Ƨ LATIN CAPITAL LETTER K WITH DIAGONAL STROKE ≈ Ƨ GREEK KYATHOS BASE SIGN

10171 ; 16C9 # ʝ GREEK ACROPHONIC THESPIAN ONE THOUSAND ≈ ʝ Runic LETTER ALGIZ EOLHX

12.2 confusables.txt and NFKC conflicts [#531]

Recommended UTC actions

1. **Action Item** for Roozbeh Pournader, PAG: Add a clarification note to UTS #39 about casefolding and normalization in relation to confusables. For a future version of the standard. See [L2/26-154](#) item 12.2.

Feedback (verbatim)

Date/Time: Sun Feb 22 08:48:37 PT 2026

ReportID: [ID20260222084837](#)

Name: Paul Wood FRSA

Report Type: Report Error in Publication/Data

Opt Subject: confusables.txt and NFKC conflicts (31 entries)

31 entries in confusables.txt (UTS #39) map a source character to a different Latin letter or digit than NFKC normalization (UTR #15) produces for that same character.

For example, [U+017F](#) LATIN SMALL LETTER LONG S is mapped to "f" in confusables.txt, but NFKC normalization maps it to "s". Applications that run NFKC before confusable detection — as recommended by IDNA (UTS #46), ENS, and GitHub — will never reach these 31 confusable entries, and if stage order were reversed, the confusable mapping would produce incorrect results.

The full list of conflicts:

None

U+017F	Long S	→ TR39: f, NFKC: s
U+1CCDE	Outlined Capital I	→ TR39: l, NFKC: i
U+2110	Script Capital I	→ TR39: l, NFKC: i
U+2111	Fraktur Capital I	→ TR39: l, NFKC: i
U+2160	Roman Numeral One	→ TR39: l, NFKC: i
U+FF29	Fullwidth Capital I	→ TR39: l, NFKC: i
U+1D408	Math Bold Capital I	→ TR39: l, NFKC: i
U+1D43C	Math Italic Capital I	→ TR39: l, NFKC: i
U+1D470	Math Bold Italic Cap I	→ TR39: l, NFKC: i
U+1D4D8	Math Bold Script Cap I	→ TR39: l, NFKC: i
U+1D540	Math Double-Struck I	→ TR39: l, NFKC: i
U+1D574	Math Bold Fraktur I	→ TR39: l, NFKC: i
U+1D5A8	Math Sans-Serif I	→ TR39: l, NFKC: i
U+1D5DC	Math Sans Bold I	→ TR39: l, NFKC: i
U+1D610	Math Sans Italic I	→ TR39: l, NFKC: i
U+1D644	Math Sans Bold Italic I	→ TR39: l, NFKC: i
U+1D678	Math Monospace I	→ TR39: l, NFKC: i
U+1CCF0	Outlined Digit Zero	→ TR39: o, NFKC: 0
U+1D7CE	Math Bold Digit Zero	→ TR39: o, NFKC: 0
U+1D7D8	Math Double-Struck 0	→ TR39: o, NFKC: 0

```
U+1D7E2 Math Sans-Serif 0      → TR39: o, NFKC: 0
U+1D7EC Math Sans Bold 0       → TR39: o, NFKC: 0
U+1D7F6 Math Monospace 0       → TR39: o, NFKC: 0
U+1FBF0 Segmented Digit Zero   → TR39: o, NFKC: 0
U+1CCF1 Outlined Digit One     → TR39: l, NFKC: 1
U+1D7CF Math Bold Digit One    → TR39: l, NFKC: 1
U+1D7D9 Math Double-Struck 1   → TR39: l, NFKC: 1
U+1D7E3 Math Sans-Serif 1     → TR39: l, NFKC: 1
U+1D7ED Math Sans Bold 1      → TR39: l, NFKC: 1
U+1D7F7 Math Monospace 1      → TR39: l, NFKC: 1
U+1FBF1 Segmented Digit One    → TR39: l, NFKC: 1
```

This is not a request to change the confusables.txt mappings, which are correct visual assessments. It is a request to document in UTS #39 that applications applying NFKC normalization before confusable detection should filter confusables.txt entries against NFKC to avoid dead code and potential incorrect mappings if pipeline order is changed.

Detailed writeup: <https://paultendo.github.io/posts/unicode-confusables-nfkc-conflict/>

Reproducing script:

<https://github.com/paultendo/namespace-guard/blob/main/scripts/generate-confusables.ts>

12.3 Confusables data for U+2205 EMPTY SET [#548]

Recommended UTC actions

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

Feedback

From Shane Carr, ICU4X-TC:

As observed in https://github.com/tc39/proposal-amount/pull/102#discussion_r3133298432

We currently have 2 different skeletons for these characters:

Code Point	Character	Skeleton
U+2300	∅	∅ (U+2205)
U+2205	∅	∅ (U+2205)
U+00F8	ø	ø (U+006F U+0338)
U+AB3E	ø	ø (U+006F U+0338)
U+00D8	Ø	Ø (U+006F U+0338)
U+2D41	Ø	Ø (U+006F U+0338)

Maybe these classes should be merged or better split into uppercase O and lowercase O.

Background information / discussion

The original report is not completely correct. The uppercase and lowercase LATIN LETTER O WITH STROKE are not in the same confusability class, and the uppercase letter's skeleton is <[U+004F](#), [U+0338](#)>.

Still, since the empty set glyph is usually close to an uppercase Ø ([U+00D8](#)), the best solution appears to be bringing [U+2300](#) ø and [U+2205](#) ∅ into that class.

This will be done by adding the following pair to the confusable sources:

2205; 00D8

12.4 Additions to intentional.txt [#554]

Recommended UTC actions

- **No Action:** This does not require any UTC action; the data file was updated.

PAG input

From Roozbeh Pournader, PAG:

A recent change in Unicode data tooling exposed various data lines that were expected to have been added to [UTS #39](#)'s data file intentional.txt, but were never actually added. These lines exist in the unicodetools source file [confusables-intentional.txt](#), but used to be filtered out due to the pairs being *compatibility* equivalent (see PAG issue #528). Considering that the suppressions do not appear to have been intentional, we should consider adding them to intentional.txt. The data additions follow:

0043 ; 216D # (C ~ C) LATIN CAPITAL LETTER C ~ ROMAN NUMERAL ONE HUNDRED
0044 ; 216E # (D ~ D) LATIN CAPITAL LETTER D ~ ROMAN NUMERAL FIVE HUNDRED
0049 ; 2160 # (I ~ I) LATIN CAPITAL LETTER I ~ ROMAN NUMERAL ONE
004C ; 216C # (L ~ L) LATIN CAPITAL LETTER L ~ ROMAN NUMERAL FIFTY
004D ; 216F # (M ~ M) LATIN CAPITAL LETTER M ~ ROMAN NUMERAL ONE THOUSAND
0056 ; 2164 # (V ~ V) LATIN CAPITAL LETTER V ~ ROMAN NUMERAL FIVE
0058 ; 2169 # (X ~ X) LATIN CAPITAL LETTER X ~ ROMAN NUMERAL TEN
0063 ; 217D # (c ~ c) LATIN SMALL LETTER C ~ SMALL ROMAN NUMERAL ONE HUNDRED
0064 ; 217E # (d ~ d) LATIN SMALL LETTER D ~ SMALL ROMAN NUMERAL FIVE HUNDRED
0069 ; 2170 # (i ~ i) LATIN SMALL LETTER I ~ SMALL ROMAN NUMERAL ONE
006C ; 217C # (l ~ l) LATIN SMALL LETTER L ~ SMALL ROMAN NUMERAL FIFTY
006D ; 217F # (m ~ m) LATIN SMALL LETTER M ~ SMALL ROMAN NUMERAL ONE THOUSAND
0076 ; 2174 # (v ~ v) LATIN SMALL LETTER V ~ SMALL ROMAN NUMERAL FIVE
0078 ; 2179 # (x ~ x) LATIN SMALL LETTER X ~ SMALL ROMAN NUMERAL TEN
05D0 ; 2135 # (א ~ א) HEBREW LETTER ALEF ~ ALEF SYMBOL
05D1 ; 2136 # (ב ~ ב) HEBREW LETTER BET ~ BET SYMBOL

05D2 ; 2137 # (׀ ~ ׀) HEBREW LETTER GIMEL ~ GIMEL SYMBOL
05D3 ; 2138 # (׀ ~ ׀) HEBREW LETTER DALET ~ DALET SYMBOL

Note: The intentional.txt file is not used by any known Unicode algorithm, and its whole documentation is: "**Intentional Confusable Mappings**: A selection of characters whose glyphs in any particular typeface would probably be designed to be identical in shape when using a harmonized typeface design."

12.5 Confusables data for stretched Arabic math symbols [#556]

Recommended UTC actions

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

Confusables source

The "stretched symbols" ([U+1EE61..U+1EE7E](#)) of the [Arabic Mathematical Alphabetic Symbols block](#), have a compatibility decomposition to normal Arabic letters, but are indeed confusable with a sequence of the letter they decompose to followed by an alef. Currently, they are marked as confusable with the letter they decompose to. These should be replaced with the sequence of the letter plus alef.

For example [U+1EE61](#) ﺏ ARABIC MATHEMATICAL STRETCHED BEH is listed as confusable with [U+0628](#) ﺏ ARABIC LETTER BEH. It should instead be listed as confusable with ﺏ (<[U+0628](#) ARABIC LETTER BEH, [U+0627](#) ARABIC LETTER ALEF>). Same with the 22 other stretched symbols.

(This would need a small change in the code that generates confusables.txt and other confusability data files.)

Data

1EE61 ; 0628 0627
1EE62 ; 062C 0627
1EE64 ; 0647 0627
1EE67 ; 062D 0627
1EE68 ; 0637 0627
1EE69 ; 064A 0627
1EE6A ; 0643 0627
1EE6C ; 0645 0627
1EE6D ; 0646 0627
1EE6E ; 0633 0627
1EE6F ; 0639 0627
1EE70 ; 0641 0627
1EE71 ; 0635 0627
1EE72 ; 0642 0627

1EE74 ; 0634 0627
1EE75 ; 062A 0627
1EE76 ; 062B 0627
1EE77 ; 062E 0627
1EE79 ; 0636 0627
1EE7A ; 0638 0627
1EE7B ; 063A 0627
1EE7C ; 066E 0627
1EE7E ; 06A1 0627

12.6 Confusables data for new Unicode 18.0 characters [#557]

Recommended UTC actions

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

Confusables source

Roosbeh Pournader investigated the new Unicode 18.0 characters and came up with the following proposed list. Some other confusables for new Unicode 18.0 were already done as action items from the UTC.

Data

05C8 ; 05B0 # HEBREW POINT SHEVA NA MUDGASH; HEBREW POINT SHEVA
05C9 ; 05BC # HEBREW POINT DAGESH HAZAQ MUDGASH; HEBREW POINT DAGESH OR
MAPIQ
0B53 ; 0307 # ORIYA SIGN DOT ABOVE; COMBINING DOT ABOVE
0B54 ; 0308 # ORIYA SIGN DOUBLE DOT ABOVE; COMBINING DIAERESIS
208F ; 00A6 # MODIFIER LETTER HIGH AND LOW VERTICAL LINE; BROKEN BAR
2E62 ; FD3E # LEFT PARENTHESIS WITH MIDDLE RING; ORNATE LEFT PARENTHESIS
2E63 ; FD3F # RIGHT PARENTHESIS WITH MIDDLE RING; ORNATE RIGHT PARENTHESIS
AB6C ; 218A # LATIN CAPITAL LETTER SCRIPT R; TURNED DIGIT TWO
107BD ; A7F2 # MODIFIER LETTER STRETCHED C; MODIFIER LETTER CAPITAL C
10ECE ; 030A # ARABIC SMALL CIRCLE ABOVE; COMBINING RING ABOVE
10EF3 ; 06E8 064E # ARABIC SMALL HIGH NOON WITH FATHA; ARABIC SMALL HIGH NOON,
ARABIC FATHA
10EF4 ; 0650 10EFB # ARABIC SMALL LOW NOON WITH FATHA; ARABIC KASRA, ARABIC
SMALL LOW NOON
10EF5 ; 06E8 064F # ARABIC SMALL HIGH NOON WITH DAMMA; ARABIC SMALL HIGH NOON,
ARABIC DAMMA
11B0A ; 0926 0952 # DEVANAGARI LETTER ALTERNATE DDDA; DEVANAGARI LETTER DA,
DEVANAGARI STRESS SIGN ANUDATTA
11DF0 ; 030A # BENGALI SIGN COMBINING ANUSVARA ABOVE; COMBINING RING ABOVE

12562 ; 2022 # CUNEIFORM NUMERIC SIGN ONE N14; BULLET
 16D80 ; 218A # CHISOI LETTER A; TURNED DIGIT TWO
 16D85 ; 0662 # CHISOI LETTER TA; ARABIC-INDIC DIGIT TWO
 16D86 ; 004C # CHISOI LETTER E; LATIN CAPITAL LETTER L
 16D88 ; 004D # CHISOI LETTER NA; LATIN CAPITAL LETTER M
 16D8D ; 0055 # CHISOI LETTER HA; LATIN CAPITAL LETTER U
 16D98 ; 030A # CHISOI SIGN ANUSVARA; COMBINING RING ABOVE
 16D9C ; 0983 # CHISOI LETTER JARAH; BENGALI SIGN VISARGA
 16D9D ; 094D # CHISOI SIGN SISO; DEVANAGARI SIGN VIRAMA
 16DA0 ; 0030 # CHISOI DIGIT ZERO; DIGIT ZERO
 16DA5 ; 0053 # CHISOI DIGIT FIVE; LATIN CAPITAL LETTER S
 16DA7 ; 0050 # CHISOI DIGIT SEVEN; LATIN CAPITAL LETTER P
 16DA8 ; 0055 # CHISOI DIGIT EIGHT; LATIN CAPITAL LETTER U
 1CEF4 ; 0662 # LEIBNIZIAN FRACTION REDUCTION SIGN-1; ARABIC-INDIC DIGIT TWO
 1CEFC ; 0056 # ANGLE OPENING UP; LATIN CAPITAL LETTER V
 1D127 ; 0317 # MUSICAL SYMBOL COMBINING STRESS; COMBINING ACUTE ACCENT BELOW
 1D128 ; 032E # MUSICAL SYMBOL COMBINING UNSTRESS; COMBINING BREVE BELOW
 1D1F7 ; 2643 # MUSICAL SYMBOL DIGIT TWO WITH STROKE; JUPITER
 1D1FE ; 0037 # MUSICAL SYMBOL DIGIT SEVEN WITH LOW STROKE; DIGIT SEVEN
 1D260 ; 266D 266D 266D # MUSICAL SYMBOL TRIPLE FLAT; MUSIC FLAT SIGN, MUSIC FLAT SIGN, MUSIC FLAT SIGN
 1D262 ; 0965 # MUSICAL SYMBOL HEAVY DOUBLE BARLINE; DEVANAGARI DOUBLE DANDA
 1DB00 ; 2293 # LEIBNIZIAN EQUALS SIGN; SQUARE CAP
 1DF1F ; 0064 00F0 # LATIN SMALL LETTER D-ETH DIGRAPH; LATIN SMALL LETTER D, LATIN SMALL LETTER ETH
 1DF20 ; 0064 026E # LATIN SMALL LETTER D-LEZH DIGRAPH; LATIN SMALL LETTER D, LATIN SMALL LETTER LEZH
 1DF22 ; 0074 026C # LATIN SMALL LETTER TL DIGRAPH WITH BELT; LATIN SMALL LETTER T, LATIN SMALL LETTER L WITH BELT
 1DF23 ; 0074 A78E # LATIN SMALL LETTER TL DIGRAPH WITH RETROFLEX HOOK AND BELT; LATIN SMALL LETTER T, LATIN SMALL LETTER L WITH RETROFLEX HOOK AND BELT
 1DF24 ; 0074 03B8 # LATIN SMALL LETTER T-THETA DIGRAPH; LATIN SMALL LETTER T, GREEK SMALL LETTER THETA
 1DF2B ; 0064 0293 # LATIN SMALL LETTER DEZH DIGRAPH WITH CURL; LATIN SMALL LETTER D, LATIN SMALL LETTER EZH WITH CURL
 1DF2C ; 0074 0286 # LATIN SMALL LETTER TESH DIGRAPH WITH CURL; LATIN SMALL LETTER T, LATIN SMALL LETTER ESH WITH CURL
 1DF2D ; 0257 0321 # LATIN SMALL LETTER D WITH HOOK AND PALATAL HOOK; LATIN SMALL LETTER D WITH HOOK, COMBINING PALATALIZED HOOK BELOW
 1DF2E ; 02A3 0321 # LATIN SMALL LETTER DZ DIGRAPH WITH PALATAL HOOK; LATIN SMALL LETTER DZ DIGRAPH, COMBINING PALATALIZED HOOK BELOW
 1DF2F ; 00F0 0321 # LATIN SMALL LETTER ETH WITH PALATAL HOOK; LATIN SMALL LETTER ETH, COMBINING PALATALIZED HOOK BELOW
 1DF30 ; 0262 0321 # LATIN LETTER SMALL CAPITAL G WITH PALATAL HOOK; LATIN LETTER

SMALL CAPITAL G, COMBINING PALATALIZED HOOK BELOW
 1DF31 ; 0263 0321 # LATIN SMALL LETTER GAMMA WITH PALATAL HOOK; LATIN SMALL LETTER GAMMA, COMBINING PALATALIZED HOOK BELOW
 1DF32 ; 0127 0321 # LATIN SMALL LETTER H WITH STROKE AND PALATAL HOOK; LATIN SMALL LETTER H WITH STROKE, COMBINING PALATALIZED HOOK BELOW
 1DF33 ; 0278 0321 # LATIN SMALL LETTER PHI WITH PALATAL HOOK; LATIN SMALL LETTER PHI, COMBINING PALATALIZED HOOK BELOW
 1DF34 ; 0071 0321 # LATIN SMALL LETTER Q WITH PALATAL HOOK; LATIN SMALL LETTER Q, COMBINING PALATALIZED HOOK BELOW
 1DF35 ; 0280 0321 # LATIN LETTER SMALL CAPITAL R WITH PALATAL HOOK; LATIN LETTER SMALL CAPITAL R, COMBINING PALATALIZED HOOK BELOW
 1DF36 ; 0281 0321 # LATIN LETTER SMALL CAPITAL INVERTED R WITH PALATAL HOOK; LATIN LETTER SMALL CAPITAL INVERTED R, COMBINING PALATALIZED HOOK BELOW
 1DF37 ; 027D 0321 # LATIN SMALL LETTER R WITH TAIL AND PALATAL HOOK; LATIN SMALL LETTER R WITH TAIL, COMBINING PALATALIZED HOOK BELOW
 1DF38 ; 0074 1D8A # LATIN SMALL LETTER TS DIGRAPH WITH PALATAL HOOK; LATIN SMALL LETTER T, LATIN SMALL LETTER S WITH PALATAL HOOK
 1DF39 ; 028B 0321 # LATIN SMALL LETTER V WITH HOOK AND PALATAL HOOK; LATIN SMALL LETTER V WITH HOOK, COMBINING PALATALIZED HOOK BELOW
 1DF3A ; 0295 0321 # LATIN LETTER PHARYNGEAL VOICED FRICATIVE WITH PALATAL HOOK; LATIN LETTER PHARYNGEAL VOICED FRICATIVE, COMBINING PALATALIZED HOOK BELOW
 1DF3B ; 03B2 0321 # LATIN SMALL LETTER BETA WITH PALATAL HOOK; GREEK SMALL LETTER BETA, COMBINING PALATALIZED HOOK BELOW
 1DF3C ; 03B8 0321 # LATIN SMALL LETTER THETA WITH PALATAL HOOK; GREEK SMALL LETTER THETA, COMBINING PALATALIZED HOOK BELOW
 1DF3D ; 03C7 0321 # LATIN SMALL LETTER CHI WITH PALATAL HOOK; GREEK SMALL LETTER CHI, COMBINING PALATALIZED HOOK BELOW
 1DF3E ; 0265 0335 # LATIN SMALL LETTER BARRED TURNED H; LATIN SMALL LETTER TURNED H, COMBINING SHORT STROKE OVERLAY
 1DF3F ; 0077 0335 # LATIN SMALL LETTER BARRED W; LATIN SMALL LETTER W, COMBINING SHORT STROKE OVERLAY
 1DF40 ; 0041 0335 # LATIN CAPITAL LETTER BARRED A; LATIN CAPITAL LETTER A, COMBINING SHORT STROKE OVERLAY
 1DF41 ; 0061 0335 # LATIN SMALL LETTER BARRED A; LATIN SMALL LETTER A, COMBINING SHORT STROKE OVERLAY
 1DF42 ; 0254 0335 # LATIN SMALL LETTER BARRED OPEN O; LATIN SMALL LETTER OPEN O, COMBINING SHORT STROKE OVERLAY
 1DF43 ; 1D07 0335 # LATIN LETTER SMALL CAPITAL BARRED E; LATIN LETTER SMALL CAPITAL E, COMBINING SHORT STROKE OVERLAY
 1DF44 ; 025B 0335 # LATIN SMALL LETTER BARRED OPEN E; LATIN SMALL LETTER OPEN E, COMBINING SHORT STROKE OVERLAY
 1DF45 ; 1D83 0335 # LATIN SMALL LETTER G WITH STROKE AND PALATAL HOOK; LATIN SMALL LETTER G WITH PALATAL HOOK, COMBINING SHORT STROKE OVERLAY
 1DF46 ; 0068 0335 # LATIN SMALL LETTER BARRED H; LATIN SMALL LETTER H, COMBINING

SHORT STROKE OVERLAY

1DF47 ; 0266 0335 # LATIN SMALL LETTER BARRED H WITH HOOK; LATIN SMALL LETTER H WITH HOOK, COMBINING SHORT STROKE OVERLAY

1DF48 ; 004B 0335 # LATIN CAPITAL LETTER BARRED K; LATIN CAPITAL LETTER K, COMBINING SHORT STROKE OVERLAY

1DF49 ; 006B 0335 # LATIN SMALL LETTER BARRED K; LATIN SMALL LETTER K, COMBINING SHORT STROKE OVERLAY

1DF4A ; 004D 0335 # LATIN CAPITAL LETTER BARRED M; LATIN CAPITAL LETTER M, COMBINING SHORT STROKE OVERLAY

1DF4B ; 006D 0335 # LATIN SMALL LETTER BARRED M; LATIN SMALL LETTER M, COMBINING SHORT STROKE OVERLAY

1DF4C ; 0271 0335 # LATIN SMALL LETTER BARRED M WITH HOOK; LATIN SMALL LETTER M WITH HOOK, COMBINING SHORT STROKE OVERLAY

1DF4D ; 004E 0335 # LATIN CAPITAL LETTER BARRED N; LATIN CAPITAL LETTER N, COMBINING SHORT STROKE OVERLAY

1DF4E ; 006E 0335 # LATIN SMALL LETTER BARRED N; LATIN SMALL LETTER N, COMBINING SHORT STROKE OVERLAY

1DF4F ; 014B 0335 # LATIN SMALL LETTER BARRED ENG; LATIN SMALL LETTER ENG, COMBINING SHORT STROKE OVERLAY

1DF50 ; 0279 0335 # LATIN SMALL LETTER TURNED R WITH STROKE; LATIN SMALL LETTER TURNED R, COMBINING SHORT STROKE OVERLAY

1DF51 ; 0056 0335 # LATIN CAPITAL LETTER BARRED V; LATIN CAPITAL LETTER V, COMBINING SHORT STROKE OVERLAY

1DF52 ; 0076 0335 # LATIN SMALL LETTER BARRED V; LATIN SMALL LETTER V, COMBINING SHORT STROKE OVERLAY

1DF53 ; 0277 0335 # LATIN SMALL LETTER BARRED CLOSED OMEGA; LATIN SMALL LETTER CLOSED OMEGA, COMBINING SHORT STROKE OVERLAY

1DF54 ; 03C7 0335 # LATIN SMALL LETTER BARRED CHI; GREEK SMALL LETTER CHI, COMBINING SHORT STROKE OVERLAY

1DF55 ; 024F # LATIN SMALL LETTER Y WITH LOW STROKE; LATIN SMALL LETTER Y WITH STROKE

1DF5A ; 0061 # LATIN SMALL LETTER HALF A; LATIN SMALL LETTER A

1DF5B ; 0297 0068 # LATIN SMALL LETTER STRETCHED C-H DIGRAPH; LATIN LETTER STRETCHED C, LATIN SMALL LETTER H

1DF5C ; 03F5 03F5 # LATIN SMALL LETTER LUNATE EE; GREEK LUNATE EPSILON SYMBOL, GREEK LUNATE EPSILON SYMBOL

1DF5D ; 0069 0065 # LATIN SMALL LETTER IE; LATIN SMALL LETTER I, LATIN SMALL LETTER E

1DF5E ; 006F 0069 # LATIN SMALL LETTER OI DIGRAPH; LATIN SMALL LETTER O, LATIN SMALL LETTER I

1DF5F ; 006F 0075 # LATIN SMALL LETTER OU DIGRAPH; LATIN SMALL LETTER O, LATIN SMALL LETTER U

1DF62 ; 0283 0068 # LATIN SMALL LETTER ESH-H DIGRAPH; LATIN SMALL LETTER ESH, LATIN SMALL LETTER H

1DF63 ; 0288 0068 # LATIN SMALL LETTER STRETCHED T-H DIGRAPH; LATIN SMALL LETTER

T WITH RETROFLEX HOOK, LATIN SMALL LETTER H

1DF64 ; 0074 0068 # LATIN SMALL LETTER REVERSED T-H DIGRAPH; LATIN SMALL LETTER T, LATIN SMALL LETTER H

1DF65 ; 0077 0068 # LATIN SMALL LETTER WH DIGRAPH; LATIN SMALL LETTER W, LATIN SMALL LETTER H

1DF67 ; 006C 0293 # LATIN SMALL LETTER LEZH WITH CURL; LATIN SMALL LETTER L, LATIN SMALL LETTER EZH WITH CURL

1DF69 ; 1D8F # LATIN SMALL LETTER PHONOTYPIC A WITH SWASH; LATIN SMALL LETTER A WITH RETROFLEX HOOK

1DF6A ; 0041 # LATIN CAPITAL LETTER PHONOTYPIC ROUNDTOP A; LATIN CAPITAL LETTER A

1DF6C ; 0404 # LATIN CAPITAL LETTER REVERSED SCRUPLE; CYRILLIC CAPITAL LETTER UKRAINIAN IE

1DF6D ; 03F5 # LATIN SMALL LETTER REVERSED SCRUPLE; GREEK LUNATE EPSILON SYMBOL

1DF6E ; 0197 # LATIN CAPITAL LETTER PHONOTYPIC DIPHTHONG AI; LATIN CAPITAL LETTER I WITH STROKE

1DF70 ; 0279 # LATIN SMALL LETTER I WITH PIGTAIL AT BOTTOM; LATIN SMALL LETTER TURNED R

1DF7B ; 1D99 # LATIN SMALL LETTER U WITH HOOK TAIL; LATIN SMALL LETTER U WITH RETROFLEX HOOK

1DF7D ; 026F # LATIN SMALL LETTER U WITH NOTCH AT BOTTOM; LATIN SMALL LETTER TURNED M

1DF81 ; 0045 # LATIN CAPITAL LETTER E WITH BENT TOPBAR; LATIN CAPITAL LETTER E

12.7 Confusables data for Devanagari Sindhi implosives [#558]

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 are four Sindhi implosive letters in the Devanagari block that were created by extending a main Devanagari letter with an underline. There is also an underline-like character in the Devanagari block, at [U+0952](#) DEVANAGARI STRESS SIGN ANUDATTA. Confusable pairs should be added for these.

Data

097B ; 0917 0952 # ङ vs ङ

097C ; 091C 0952 # ञ vs ञ

097E ; 0921 0952 # ण vs ण

097F ; 092C 0952 # ण vs ण

13. Emoji

13.1 Emoji_Presentation property for older characters (PRI-544) [#569]

Recommended UTC actions

1. **Action Item** for Ned Holbrook, ESR: Edit the text of [UTS #51](#) to improve guidance on the usage of emoji presentation selectors and the `Emoji_Presentation` property based on [L2/26-154](#) item 13.1. For Unicode 18.0. See [L2/26-154](#) item 13.1.

Feedback (verbatim)

[PRI-544](#)

Date/Time: Thu June 04 19:22:01 PT 2026

ReportID: [ID20260604192201](#)

Name: Property Correction

Report Type: Report Error in Publication/Data

Opt Subject: Emoji_Presentation property for older characters

Hello,

Please correct `Emoji_Presentation` property to No for all characters stabilized before Unicode 6.1, when the variation selectors became available, that were not initially encoded as emoji characters. Having some set to Yes causes text in spreadsheets that were composed before Unicode 6.1 to display with inappropriate color.

As document authors could not have entered the variation selectors before they were standardized, it is critical to have characters that predate the variation selectors have `Emoji_Presentation` set to No.

Thank you for your consideration.

Background information / discussion

From Charlotte Buff, PAG:

Like many aspects of Unicode Emoji, the `Emoji_Presentation` property was originally conceived to be much more descriptive in nature, documenting existing behaviour found in key implementations to aid interoperability. As such, some characters dating as far back as Unicode 1.1 were given the property value `Yes` under the assumption that this information would be useful to emoji-specific environments (text messages, chats etc.) without affecting general text.

Alas, emoji have now become so ubiquitous that *any* application dealing with Unicode text needs to be able to account for their peculiarities and the easiest way to go about this is by following [UTS #51](#) and its associated data. This has led to many existing documents retroactively changing appearance

as the pre–Unicode 6.0 characters they had been using as if they were any other symbols were now explicitly classified as emoji by default, with all that entails.

While it is true that the appearance of any Unicode character can never be guaranteed, it must be noted that emoji presentation is a rather disruptive status to be bestowed upon an established character. The innately colourful glyphs of emoji make them impervious to many important text style settings, effectively taking control away from the author in a way that was impossible to prepare for. This goes beyond a mere change in visual form; it could even be argued that this constitutes a violation of character identity.

The adverse consequences of this initial assignment of `Emoji_Presentation` values are immensely regrettable and the damage that has been done is undeniably real. However, this data has also been stable for over a decade now – with the exception of the introduction of said property, only freshly encoded characters have ever been made emoji-style by default – and we recommend against making any sweeping changes.

The number of old documents affected by the original change in emoji status is large but fixed, whereas the amount of newly created text data containing emoji is ever growing. Support for the variation selector mechanism is patchy in terms of both emission and interpretation, and so the `Emoji_Presentation` property remains a valuable tool to foster baseline interoperability between different platforms and applications (even if adherence to the property is oftentimes loose in practice as well).

We do, however, see room for improvement in the documentation of this issue:

1. History must not repeat itself. As previously mentioned, after the creation of the `Emoji_Presentation` property no already assigned character has been changed from `No` to `Yes` ever again. It would be useful to explicitly state in [UTS #51](#) that this initial blunder has remained and will remain the only one of its kind. The reverse case, going from `Yes` to `No`, has never occurred and should also be avoided for the sake of stability, though we consider the prospect of an emoji becoming a regular text symbol somewhat less harmful in that regard.
2. Vendors should be advised to *always* emit emoji and text presentation selectors where appropriate (i.e. in accordance with `emoji-presentation-sequences.txt`), even if the character's default behaviour as indicated by its `Emoji_Presentation` value already matches the intended style. Relying on both mechanisms in tandem maximises the odds of correct interpretation in all contexts.
3. Add guidance for handling pre-emojification documents. If a certain document is known to have been composed before the concept of emoji presentation was formally introduced and therefore could not have taken it into consideration, it would be advisable to treat that entire document as text style regardless of character properties and other factors. This would mitigate the problems discussed to some extent.

14. Other

14.1 UAX #24 needs to talk about Jpan/Kore/Hanb/Hntl? [#561]

Recommended UTC actions

1. **No Action:** [UAX #24](#) has been updated with a discussion of script aliases.

PAG input

From Roozbeh Pournader, PAG:

[UTS #39](#) includes a reference to [UAX #24](#) saying "For more information on the Script_Extensions property and Jpan, Kore, and Hanb, see UAX #24, Unicode Script Property [[UAX #24](#)]." But there is no explicit mention of Jpan, Kore, and Hanb (or the new Hntl) in [UAX #24](#)!

We should update [UAX #24](#) to discuss such script aliases.

15. Feedback not addressed in this document

The following public feedback was received, but is not covered in this report. PAG plans to address these for UTC #189.

1. [ID20260417172618](#) — Jules Bertholet — Georgian_Extended has incorrect titlecase mappings [#578]
2. [ID20260712083815](#) — Philippe Verdy — Draft UTS #61 editorial changes [#583]