UTC #186 properties feedback & recommendations

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

Participants **1**
1. UCD **2**
1.1 Stability policy for ID_Compat_Math_(Start|Continue) [#492] 2
1.2 Soft_Dotted English Phonotypic Alphabet letters [#496] 3
1.3 U+1D26A: HALF FILLED or HALF-FILLED? [#503] 4
1.4 Defective decision 184-C19 on two right triangles [#504] 5
1.5 kEH_Core classification of U+13096 [#505] 6
2. Characters **7**
2.1 L2/25-151R Proposal on two alternate Katakana letters [#494] 7
2.2 Comments on property assignments of some characters accepted for Unicode 18, accidentally omitted from L2/24-224 [#501] 7
3. Proposed new scripts & characters **9**
4. Line Break **11**
4.1 Proposed updates to Unicode line breaking [#488] 11
5. Security **12**
5.1 Document the relationship between XML Name/Nmtoken, UAX31-R2 immutable identifiers, and UAX31-R1 default identifiers [#175] 12
5.2 IdentifierType.txt: Group by values=sets → code point order [#447] 13
5.3 Proposal for Reclassifying the Balinese Script: Response to Unicode [#489] 14
5.4 Proposal for Reclassification of Buginese Script from ID_Type=Excluded [#491] 15
6. Confusables **16**
6.1 High priority confusables data based on feedback from David Corbett [#458] 16
6.2 Confusables data for Dandas [#468] 18
6.3 Non-NFD confusables to remove or correct [#486] 19
7. Other **21**
7.1 Unicode Link Detection and Formatting [#507] 21
7.2 UTS #18 not up-to-date [#499] 22

Participants

The following people have contributed to this document:

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

1. UCD

1.1 Stability policy for ID_Compat_Math_(Start|Continue) [#492]

Recommended UTC actions

1. **Consensus**: The UTC recommends to the officers the following new entries in the Property Value Stability policy, under Identifiers: For property ID_Compat_Math_Start, “Once a character is ID_Compat_Math_Start, it must continue to be so in all future versions.”; for property ID_Compat_Math_Continue, “Once a character is ID_Compat_Math_Continue, it must continue to be so in all future versions.”. The applicable Unicode versions are 15.1.0+ for both entries.

PAG input

These properties are meant to complement XID_Start and XID_Continue, and should be subject to the same stability requirement. We did not stabilize them at creation to allow us to fix egregious errors initially. We now have more than three years of deployed experience in a major compiler (Clang) as an extension enabled by default for a major programming language (C++), and the C++ standard is looking to adopt these properties in its identifier definition for C++29 (as a defect report against all previous versions, as had previously been done with the switch to default identifiers, as implementers want to maintain only one version of identifier lexing). These sets are based on attested usage in a corpus of C++, Swift, and Julia programs on GitHub, see L2/22-102. The time has come to make these properties stable.

Note that these identifiers are Not_XID and therefore excluded from the General Security Profile for identifiers; security-sensitive code bases would be expected to disallow them.

Note that the proposed stability policy allows for expansion (but not contraction) of these sets.

Peter Bindels (NL expert at WG 21) brought this matter to our attention.

1.2 Soft_Dotted English Phonotypic Alphabet letters [#496]

Recommended UTC actions

1. **No Action**: PAG recommends no action; the data files have been prepared as described in this section.

Feedback

From Kirk Miller via review comment on

- https://github.com/unicode-org/unicodetools/pull/1035

for characters for the English Phonotypic Alphabet.

Kirk Miller:

> Do pull requests need to specify when characters should be added the PropList file for the soft_dotted property? Three of the characters here do: U+1DF6F, U+1DF70 and U+1DF71.

Robin Leroy:

> We had reported on the properties of these characters in L2/25-087, pp. 14 sq., but nobody had spotted the soft dotted issue […].

Background information / discussion

```
None
1DF6F;LATIN SMALL LETTER PHONOTYPIC DIPHTHONG AI;Ll;0;L;;;;;N;;;1DF6E;;1DF6E
1DF70;LATIN SMALL LETTER I WITH PIGTAIL AT BOTTOM;Ll;0;L;;;;;N;;;;;
1DF71;LATIN SMALL LETTER STRETCHED I;Ll;0;L;;;;;N;;;;;
```

The proposal L2/24-277 did include:

The following characters have to get the “soft-dotted” property:

```
None
U+1DF6F LATIN SMALL LETTER PHONOTYPIC DIPHTHONG AI
U+1DF70 LATIN SMALL LETTER I WITH PIGTAIL AT BOTTOM
U+1DF71 LATIN SMALL LETTER STRETCHED I
```

Contrary to what was stated in L2/25-087, pp. 14 sq., the properties of these three characters are not like these of ɑ nor like those of ɬ, but instead like those of ʝ (for U+1DF6F LATIN SMALL LETTER PHONOTYPIC DIPHTHONG AI, which is part of a case pair like ʝʖ) or those of ɻ (for the other two, which have no uppercase counterpart, and ignoring ɻ’s Do_Not_Emit sequence).

1.3 U+1D26A: HALF FILLED or HALF-FILLED? [#503]

Recommended UTC actions

1. **Consensus**: Change the name of U+1D26A, approved for Unicode Version 18.0, from MUSICAL SYMBOL DIAMOND NOTEHEAD HALF FILLED to MUSICAL SYMBOL DIAMOND NOTEHEAD HALF-FILLED. For Unicode Version 18.0. See L2/26-006 item 1.3.
2. **Action Item** for Robin Leroy, PAG: In UCD file UnicodeData.txt and derived files, correct the name of U+1D26A to MUSICAL SYMBOL DIAMOND NOTEHEAD HALF-FILLED. For Unicode Version 18.0. See L2/26-006 item 1.3.

PAG input

From Robin Leroy and Ken Whistler, PAG.

Proposal L2/25-017 contains two lists of names, which differ in the name of U+1D26A: MUSICAL SYMBOL DIAMOND NOTEHEAD HALF FILLED (no hyphen) in the “Properties” section, MUSICAL SYMBOL DIAMOND NOTEHEAD HALF-FILLED (with a hyphen) in the “Characters” section. Decision UTC-182-C10 explicitly referenced the “Properties” section, and therefore approved the hyphenless name. The data files have been prepared accordingly.

However, the Project Editor of ISO/IEC 10646 used the names from the “Characters” section instead in Committee Draft 3, so one of the two drafts has to change for synchronization.

A look at both the caption of Figure 20 of the proposal and at the text by Gould shown in that figure shows that HALF-FILLED is correct, and HALF FILLED is a typo.

1.4 Defective decision 184-C19 on two right triangles [#504]

Recommended UTC actions

1. **Consensus**: In reference to decision 184-C19, the UTC asserts that the character U+1F7FD is named LOWER LEFT FLATTENED RIGHT TRIANGLE, and the character U+1F7FE is named LOWER RIGHT FLATTENED RIGHT TRIANGLE, as listed in Section 4 of WG 2 N5330, with reference glyphs as shown in L2/26-006 item 1.4.

PAG input

From Robin Leroy and Ken Whistler, PAG.

Proposal WG 2 N5330 provides two different lists of names, which swap the names of U+1F7FD and U+1F7FE. That is, the glyphs are swapped in the code chart on page 3 of the proposal; the list given in Section 3 of the proposal, which has both the names and glyphs swapped, is not used. Decision UTC-184-C19 did not specify which list was used; it is therefore ambiguous, and a new decision must be taken to clarify the situation; compare UTC-182-C17.

The Project Editor of ISO/IEC 10646 used the list in Section 3; the editor of the UCD used the list in Section 4. Neither seems obviously more correct than the other; the proposal resolves the ambiguity in favour of the draft UCD files.

Background

With the proposed change, the two characters are as follows:

Code Point	Glyph	Name
1F7FD		LOWER LEFT FLATTENED RIGHT TRIANGLE
1F7FE		LOWER RIGHT FLATTENED RIGHT TRIANGLE

1.5 kEH_Core classification of U+13096 [#505]

Recommended UTC actions

1. **Action Item** for Michel Suignard, SAH: In Unicode Standard Annex #57, Unicode Egyptian Hieroglyph Database, reword Section 3.5 and Section 3.7 to clarify the intended purpose, scope, and principles of the kEH_Core and kEH_AltSeq properties. Document exceptional cases such as U+13096 EGYPTIAN HIEROGLYPH D031. For Unicode Version 18.0. See L2/26-006 item 1.5.

PAG input

From Charlotte Buff, PAG

U+13096 EGYPTIAN HIEROGLYPH D031 is currently the only character with a non-null kEH_AltSeq property value that isn't also classified as kEH_Core=Legacy; instead it is kEH_Core=Core. While this was originally assumed to be a simple clerical error, further discussion with the SEW and Michel Suignard instead revealed inconsistencies between the current wording of UAX #57 (Unicode Egyptian Hieroglyph Database) and how the two properties in question are actually used in practice.

The issue of which hieroglyphs get to be classified as Core or Legacy (or neither) is complicated and also to some extent arbitrary. There is no universal agreement among experts. Section 3.5 of UAX #57 currently states that Legacy signs »may be present in fonts for legacy reasons, but that their usage is discouraged« whereas the Core set »is the recommended set for Egyptologists and should be implemented in widely used fonts«. This would imply the signs in the Legacy set to be essentially deprecated and of a lower priority for font designers.

However, this is too strong a statement to make. Legacy hieroglyphs may still have some use in specialised contexts that are difficult to predict, and in fact many Egyptologists continue to use some of these signs as if they were Core. Furthermore, as Legacy hieroglyphs are a subset of the immensely important Gardiner list, any font that does not include them can in some sense be considered incomplete.

The kEH_AltSeq property is similarly not to be understood as simply an Egyptian-specific version of DoNotEmit. These alternate sequences are of importance when dealing with complex quadrat layouts involving insertions and overlays, but they are not meant to be absolute replacements in every context. In particular, Egyptologists generally treat U+13096 EGYPTIAN HIEROGLYPH D031 as its own distinct entity even though structurally it can be analysed as a combination of other signs.

For the time being, kEH_Core property value assignments should remain as they are and instead the relevant sections of UAX #57 be overhauled. This does not preclude any hieroglyphs being recategorised in the future once the principles behind Core status have stabilised. It would also be useful to explicitly document anomalies and edge cases such as U+13096 in UAX #57 going forward.

2. Characters

2.1 L2/25-151R Proposal on two alternate Katakana letters [#494]

Recommended UTC actions

1. **No Action**: PAG recommends no action: The data files have been prepared.

Document

L2/25-151R *Proposal on two alternate Katakana letters* proposes alternate katakana letters U+1B127 & U+1B128.

Background information / discussion

While they are named ALTERNATE rather than ARCHAIC on the advice of the CJK group, these characters otherwise have the same properties as neighbouring archaic Katakana, e.g., U+1B122 □ KATAKANA LETTER ARCHAIC WU; in particular, lb=ID, as well as ea=W and vo=U mentioned in the proposal.

2.2 Comments on property assignments of some characters accepted for Unicode 18, accidentally omitted from L2/24-224 [#501]

Recommended UTC actions

1. **No Action**: PAG recommends no action; the data files have been prepared as described in this section.

PAG input

During the preparation of L2/24-224, the PAG report to UTC-181, the contents for several items in Section 4, Proposed new scripts & characters, were accidentally omitted from the report.

They are instead included here so that the rationale behind the property assignments may be reviewed by the UTC and archived in its document register.

Note: 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.

L2/24-234 Unicode request for barred letters (#510)

The proposed barred capital AKMNV are propertywise to the proposed small barred akmnv as ƙ (with stroke, not barred) is to ƙ. In particular, this means LATIN CAPITAL LETTER BARRED M and LATIN CAPITAL LETTER BARRED V should have gc=Lu, not gc=Ll as proposed in L2/24-234 (presumably a typo).

The other proposed small (and small capital) letters (barred ɔɛɛgɧhɦɱŋɹɷχ, y with low stroke) are propertywise like other lowercase Latin letters with no uppercase counterparts, such as neighbouring U+1DF01 ᵽ LATIN SMALL LETTER REVERSED SCRIPT G from in the same Latin Extended-G block.

The proposed LATIN LETTER GLOTTAL STOP WITH DOUBLE STROKE should have the same properties as other variants of the glottal stop, such as U+02A1 ʡ LATIN LETTER GLOTTAL STOP WITH STROKE, or nearby U+1DF0E ᴞ LATIN LETTER INVERTED GLOTTAL STOP WITH CURL. This means that it should have General_Category=Lowercase_Letter, not General_Category=Other_Letter as the basic ʔ LATIN LETTER GLOTTAL STOP; the General_Category of ʔ is a result of its disunification from a case pair, but there is no such requirement with the ʔ with two strokes; the General_Category should therefore be assigned consistently with the ʔ with one stroke ʡ.

The proposed modifier letters are related in the expected way to their non-modifier counterparts, ðđgɧ and proposed ɹ with stroke.

L2/24-231 Unicode request for modifier small capital P (#554)

Propertywise to ᴘ as ˪ is to ʟ, in particular Other_Lowercase. If PAG #315 is accepted, the proposed character should be Diacritic.

L2/24-232 Unicode request for compound tone diacritics III (#528)

Combining marks above and below, propertywise like other combining marks above (like the doubled circumflex) and below (like the wiggly line) in the same block. In particular, Diacritic.

3. Proposed new scripts & characters

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

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

- L2/25-165 document for Tangut rên¹ [SEW #678]
  - Another Tangut ideograph, propertywise like the others.
- L2/25-175 document for Dagesh ḥazaq [SEW #688]
  - Propertywise like the existing HEBREW POINT DAGESH OR MAPIQ, in particular Alphabetic and Diacritic like all of the other Hebrew points (contrast the accents which are typically non-Alphabetic Diacritic), and CCC21 (which is only used for DAGESH OR MAPIQ).
- L2/24-178 PROPOSAL TO ENCODE SIXTEEN QURANIC ARABIC CHARACTERS -- Rikza F. Sh. [SEW #497]
  - The small baseline letters should be propertywise like U+0888 ◌ ARABIC RAISED ROUND DOT; this means that they should be gc=Sk rather than gc=Lo as proposed.
  - The northeast pointing arrowhead above is propertywise like U+0657 ARABIC INVERTED DAMMA.
  - The circles above are propertywise like U+06E0 ARABIC SMALL HIGH UPRIGHT RECTANGULAR ZERO.
  - The small high noons with fatha and damma and the small high heh initial form are propertywise like U+06E2 ARABIC SMALL HIGH MEEM ISOLATED FORM. They are *not* like U+06E8 ARABIC SMALL HIGH NOON, since that one is MCM.
  - The small high word kabbir is propertywise like U+08DE ARABIC SMALL HIGH WORD QIF.
  - The arrowheads below are propertywise like U+0656 ARABIC SUBSCRIPT ALEF.
  - The small low upright rectangular zero, square below, and filled square below are propertywise like U+08D1 ARABIC LARGE CIRCLE BELOW.
  - The small low noons with fatha and damma are propertywise like U+08D3 ARABIC SMALL LOW WAW. They are *not* like U+08D9 ARABIC SMALL LOW NOON WITH KASRA, which is not MCM.
  - The Diacritic and Alphabetic properties have been assigned according to the above correspondences. However, the assignments of the existing characters are currently under review; when they get updated, the new characters will likely follow according to these correspondences.
- L2/25-122R document for Maldivian Rufiyaa Sign [SEW #632]
  - Propertywise like the Saudi Riyal; in particular for the Line_Break property, lb=PR (Prefix_Numeric), rather than lb=PO (Postfix_Numeric). 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 (SEW issue #619)
  - Note the comments about Directionality and the choice of ET in the proposal document.
- L2/25-164 document for Jurchen Small Script characters [SEW #675]
  - Added 5 provisionally assigned code points U+18CD6..U+18CDA for characters used in Jurchen Small Script as described in L2/25-164. No significant differences for propertywise tests when compared to similar characters such as U+18CD5.

- WG2 N5344R document for Seal Script [SEW #221]
    - Propertywise like other ideographic scripts, e.g., Tangut, in particular Ideographic=Yes as noted in the proposal. The range was already defaulted to East_Asian_Width=W and Line_Break=ID, so algorithms depending on these properties should work out of the box. Of the script-specific properties, the THX source and radical have complete coverage; the other three sources together cover all but three of the characters.
- L2/25-211 document for Proto-Cuneiform: Revised proposal [SEW #211]
    - The ideograms (characters with GC=Lo, U+12690..U+12BE7) have the same properties as the cuneiform signs (e.g., U+12000 𒀀 CUNEIFORM SIGN A), except for their script.
    - The numeric signs have the same properties as proto-cuneiform numerals proposed in L2/24-210R and approved for Unicode 18 in the Archaic Cuneiform Numerals block, e.g., the N02 series (U+125BE sqq.). As for other cuneiform and proto-cuneiform numerals, the Numeric_Value property reflects the multiplicity of the sign, not the actual number represented by the sign, if any.
    - Note that contrary to the UnicodeData file attached to the proposal L2/25-211, the numeric signs should not all have an incorrect Numeric_Value=2, but instead should have a numeric value matching their name.
    - Note also that contrary to the characters in the Archaic Cuneiform Numerals block, gaps in attestations for the numeric signs in the Proto-Cuneiform block are not encoded, so that, e.g., U+12BF6..U+12BFE have numeric values 1..5, 8..10, 12; this is because the signs in the Archaic Cuneiform Numerals block are part of clearly understood metrological systems, so that signs that happen to be unattested can be securely reconstructed and can be expected to be found eventually, whereas the signs in the Proto-Cuneiform block are not part of well-understood notations.
- WG2 N5331 document for Symbols: Proposal to encode 10 mathematical symbols [SEW #590]
    - Added 8 provisionally assigned code points U+1CEF6..U+1CEFD for sector and angle symbols used in historic mathematical works by G. W. Leibniz as described in L2/25-232R. No significant differences for propertywise tests when compared to U+29A1 SPHERICAL ANGLE OPENING UP.

4. Line Break

4.1 Proposed updates to Unicode line breaking [#488]

Recommended UTC actions

1. **No Action**: PAG recommends no action.
2. **Note**: PAG would also like to state that because of the extreme sensitivity of implementations of Line Breaking, we need very strong evidence, including compelling use cases and analysis, before making changes. See L2/26-006 item 4.1.

Document

L2/25-261 "Proposed updates to Unicode line breaking" by Kent Karlsson.

The document proposes to "improve the (automatic) line breaking behaviour in some very common cases."

Background information / discussion

In general, the document does not cite "real world cases" of infelicitous line breaking, or even any examples.

It proposes the removal of a non-tailorable line break rule; these are frequently hard-coded in implementations, and often interact with implementation-specific special cases; changes to this part of the algorithm should only be done with extremely good reasons and analysis, and should consider feedback from implementers. For background on class NL see UTC-94-M2 and L2/03-071 §2.

The proposed additional rules for QU would effectively revert the handling of that class to Unicode 15.1 (the proposed rule, placed before LB15a, would have the same effect as the old LB19 after LB18). The changes made in Unicode 16 were motivated by user complaints relayed by major implementers, illustrated by real-world examples, and included an analysis of the lack of detrimental effect on users of Pf as opening and Pi as closing quotation marks, see UTC-179-C28 and L2/24-064 §5.5. The L2/25-261 proposal does not explain what concrete issues would be solved by reverting to the Unicode 15.1 behaviour, let alone why that would outweigh the degradation of Simplified Chinese line breaking.

5. Security

5.1 Document the relationship between XML Name/Nmtoken, UAX31-R2 immutable identifiers, and UAX31-R1 default identifiers [#175]

Recommended UTC actions

1. **No Action**: PAG recommends no further action: a note has been added in Proposed Update UAX #31.

PAG input

From Robin Leroy, PAG, and from discussion with CLDR-TC/Keyboards SC.

Summary

Name and Nmtoken are fairly common immutable identifier systems; using them instead of UAX31-R2 with no profile can make sense if you get them for free as part of a syntax which is mostly XML-based.
This came up recently [editor's note: in August 2023] in a Unicode standard, see https://github.com/unicode-org/cldr/pull/3226#issuecomment-1699420565.
However, we should document the relationship between the two (effectively spelling out the conformance claim to UAX31-R2 you could make if you use those definitions).

Mark Davis had done some work documenting the relation between Name and UAX31-R2, but never published it.

Robin Leroy looked at the relation between UAX31-R1 and Nmtoken in https://github.com/unicode-org/cldr/pull/3226#issuecomment-1699420565: four XID_Continue characters are not NameChar, it could be useful to document that:

	Code point	Name
ª	U+00AA	FEMININE ORDINAL INDICATOR
µ	U+00B5	MICRO SIGN
º	U+00BA	MASCULINE ORDINAL INDICATOR
‿	U+2054	INVERTED UNDERTIE

Robin Leroy have drafted a note to that effect in the proposed update for UAX #31.

5.2 IdentifierType.txt: Group by values=sets → code point order [#447]

Recommended UTC actions

1. **No Action**: PAG recommends no action; the order of lines in IdentifierType.txt has changed as described. See L2/26-006 item 5.2.

PAG input

From Markus Scherer, PAG

The UTS #39 data file IdentifierType.txt groups data by values, but many values are **sets** with more than one Identifier_Type. This means that adding a type to the set for a character (or removing a type) moves the data for that character far across the file.

I propose that we instead show the data in this file in code point order.

We have had a similar discussion and conclusion for UCD ScriptExtensions.txt: A Script_Extensions value is a *set* of Script values. We made this kind of change there for Unicode 16. See

UTC #179 discussion of L2/24-064
Section 2.10 Should ScriptExtensions.txt be grouped by value? (No.)

[179-N1] Note: The UCD file ScriptExtensions.txt is changing the order of lines (but not the format) from grouping by value to simple code point order. The 16.0 beta PRI will include a version of 15.1 scx data for comparison with 16.0 data.

5.3 Proposal for Reclassifying the Balinese Script: Response to Unicode [#489]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The additional information confirms again that Identifier_Type=Limited_Use is correct.

Document

Previous:

- L2/25-218 Proposal for Reclassifying the Balinese Script
- L2/25-219 PAG assessment: Proposal for Reclassifying the Balinese Script

New:

- L2/25-269 by PANDI (.id Registry) et al. (2025-oct-27)

PANDI responds to several points in the L2/25-219 PAG assessment.

From the submission email:

Thank you very much for the detailed assessment from the PAG and the time taken to carefully review our proposal for the reclassification of the Balinese script.

I am sharing with you our response document, which addresses the points raised in your message, including further elaboration and supporting evidence as requested.

In addition, we would like to inform you that there is one more contributor's name to be added from the Balinese script community:
Ida Bagus Gede Sarasvananda - Udayana University.

We greatly appreciate the opportunity to engage in this process and look forward to your feedback. Please don’t hesitate to let us know if there is anything further we can clarify or provide.

Background information / discussion

The additional information confirms that the *encoding* of the Balinese script in 2006 was appropriate, and Balinese letters are *allowed* in Unicode default identifiers, but for the purpose of security-sensitive identifiers, Identifier_Type=Limited_Use is the correct classification.

Once the Balinese script is actually in “everyday common use” as explained in L2/24-019, this may be revisited.

5.4 Proposal for Reclassification of Buginese Script from ID_Type=Excluded [#491]

Recommended UTC actions

1. **No Action**: PAG recommends no action. In the context of security-sensitive identifiers and the Identifier_Type property, the available evidence confirms that Identifier_Type=Exclusion is appropriate.

Document

L2/25-273 by PANDI (.id Registry) et al.

From the submission email:

I would like to formally submit a new proposal for the reclassification of the Buginese script under the Identifier_Type property in Unicode Standard Annex #31 (UAX #31).

In this submission, we propose to elevate the classification of Buginese from “Excluded” to “Recommended”, based on an updated body of evidence from active community use, educational integration, and digital efforts. However, we fully recognize the high standard set for “Recommended” status, and we deeply appreciate the rationale and assessment you provided in the previous review.

With that in mind, while our current proposal is framed toward “Recommended,” we would also respectfully welcome and appreciate consideration for reclassification to “Limited_Use” if the supporting evidence is not yet deemed sufficient for the full “Recommended” level. Our main intent is to ensure that the Buginese script is acknowledged for its growing presence and revitalization, and that it can be included in relevant technical contexts moving forward.

From section III. Introduction:

... the current status of Buginese script in Unicode remains classified as “Excluded”, indicating that global support is still restricted and not yet fully integrated across major operating systems and digital platforms ...

(and similar statements elsewhere in the document)

Background information / discussion

This request is similar to the one for Balinese (L2/25-218, PAG assessment in L2/25-219).

The Buginese language is spoken by several million people, and the Buginese/Lontara script is historically important and sees some contemporary use.

The “Proposal for Reclassifying the Buginese Script” provides evidence for use of the script in historical texts, public signage, teaching materials, and decorations. It points to a single website with a small number of pages which uses the Buginese script in body text. (All dated contents is from 2025-jun-01..03.) There is no evidence of common, or even robust-but-limited, everyday-type online usage.

In comparison with the Balinese script, online usage of the Buginese script is even less prevalent. In our assessment, in the context of UAX #31 “Unicode Identifiers and Syntax” and Buginese is appropriately listed among “Excluded” scripts.

Note that the request document and email suggest that the Identifier_Type=Exclusion classification hinders adoption. This is a misunderstanding: The use of a script in security-sensitive contexts is not a prerequisite for general adoption. Instead, a recommendation for use in such contexts naturally has to *follow* “widespread everyday common use” as noted in the criteria document L2/24-019.

6. Confusables

6.1 High priority confusables data based on feedback from David Corbett [#458]

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:
I analyzed the confusable suggestions from David Corbett, proposed in 2018 and 2019 and submitted through the old web form, for cases where both sides of proposed confusability had Identifier_Status=Allowed. These are the highest priority confusables because they can create confusability among allowed identifiers. I further added a few confusable pairs based on that data to make the data cleaner to incorporate and review. Here is the proposed additions (CJK confusables are handled in a separe issue):

Combining marks:
U+030B ˝ ≈ <U+0301 U+0301> ˊ

Thaana:
U+07A6 ˊ ≈ U+0301 ˊ
U+07A7 ˋ ≈ <U+07A6 U+07A6> ˊ
U+07A8 ˎ ≈ U+0317 ˎ
U+07A9 ˏ ≈ <U+07A8 U+07A8> ˎ
U+07AA ˆ ≈ U+0350 ˆ
U+07AB ˇ ≈ <U+07AA U+07AA> ˆ
U+07AC ˘ ≈ U+1DFE ˘
U+07AD ˙ ≈ <U+07AC U+07AC> ˘
U+07AE ˚ ≈ <U+07AC U+07AA> ˚
U+07B0 ˚ ≈ U+030A ˚

Sinhala:
U+0DDB ෙෙ◌ ≈ <U+0DD9 U+0DD9> ෙෙ◌
U+0DF2 ◌ාා ≈ <U+0DD8 U+0DD8> ◌ාා

Thai:
U+0E3A ◌ฺ ≈ U+0323 ◌̣

Lao:
U+0EC1 ◌ແ ≈ <U+0EC0 U+0EC0> ◌ແ

Tibetan:
U+0F7E ◌ཾ ≈ U+030A ◌̊

Myanmar:
U+1022 ◌ဢ ≈ <U+1075 U+102C> ◌ဢ

Khmer:
U+17A1 ◌ឡ ≈ <U+1791 U+17D2 U+1794> ◌ឡ
U+17B0 ◌ឰ ≈ <U+1796 U+17D2 U+1792> ◌ឰ
U+17BE ◌ើ ≈ <U+17C1 U+17B8> ◌ើ
U+17C4 ◌ោ ≈ <U+17C1 U+17B6> ◌ោ
U+17C7 ◌ះ ≈ U+0983 ◌ঃ

Grantha (used in Tamil):
U+11303 ◌𑌃 ≈ U+0983 ◌ঃ

Data

030B ; 0301 0301
07A6 ; 0301
07A7 ; 07A6 07A6
07A8 ; 0317
07A9 ; 07A8 07A8
07AA ; 0350
07AB ; 07AA 07AA
07AC ; 1DFE
07AD ; 07AC 07AC
07AE ; 07AC 07AA
07B0 ; 030A
0DDB ; 0DD9 0DD9
0DF2 ; 0DD8 0DD8
0E3A ; 0323
0EC1 ; 0EC0 0EC0
0F7E ; 030A
1022 ; 1075 102C
17A1 ; 1791 17D2 1794
17B0 ; 1796 17D2 1792
17BE ; 17C1 17B8
17C4 ; 17C1 17B6
17C7 ; 0983
11303 ; 0983

6.2 Confusables data for Dandas [#468]

Recommended UTC actions

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

Confusables source

CLDR uses | as a syntax character. Occasionally submitters confuse that with a native character, indicating confusability. While CLDR tooling catches a number of those, a new category popped up: dandas (or similar characters). The CLDR-TC is requesting those to be added to the confusable source data.

Data

[ı ɩ ɪ | ׀ ו □ □ □ □] ; |
[ǁ ‖ ॥ ∥ ‖ ॥ □ □ □ □] ; ||

```
None
# NOTE: all characters confusable to the same character, such as U+007C, are also confusable
with one another.
# Mark's original list
0964 ; 007C # DEVANAGARI DANDA
104A ; 007C # MYANMAR SIGN LITTLE SECTION
A8CE ; 007C # SAURASHTRA DANDA
11047 ; 007C # BRAHMI DANDA
110C0 ; 007C # KAITHI DANDA
11141 ; 007C # CHAKMA DANDA
111C5 ; 007C # SHARADA DANDA
1144B ; 007C # NEWA DANDA
11641 ; 007C # MODI DANDA
11C41 ; 007C # BHAIKSUKI DANDA
# Roozbeh's additions
AA5D ; 007C # CHAM PUNCTUATION DANDA
113D4 ; 007C # TULU-TIGALARI DANDA
115C5 ; 007C # SIDDHAM SEPARATOR BAR
16D63 ; 007C # KIRAT RAI VOWEL SIGN AA

# Mark's original list
0965 ; 007C 007C # DEVANAGARI DOUBLE DANDA
104B ; 007C 007C # MYANMAR SIGN SECTION
A8CF ; 007C 007C # SAURASHTRA DOUBLE DANDA
11048 ; 007C 007C # BRAHMI DOUBLE DANDA
110C1 ; 007C 007C # KAITHI DOUBLE DANDA
11142 ; 007C 007C # CHAKMA DOUBLE DANDA
111C6 ; 007C 007C # SHARADA DOUBLE DANDA
1144C ; 007C 007C # NEWA DOUBLE DANDA
11642 ; 007C 007C # MODI DOUBLE DANDA
11C42 ; 007C 007C # BHAIKSUKI DOUBLE DANDA
```

```
# Roozbeh's additions
113D5 ; 007C 007C # TULU-TIGALARI DOUBLE DANDA
```

6.3 Non-NFD confusables to remove or correct [#486]

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:
In a recent discussion with Ken Lunde, the chair of the CJK Working Group, he noted that there are a lot of CJK Compatibility Ideographs mistakenly listed in the confusables.txt data file. These data lines will simply be ignored in the algorithm in UTS #39 that determines if two strings are confusable, so including them serves no purpose and would simply result in confusion.

I wrote a script that finds all the existing data in confusables.txt that are not in NFD form, and thus should either be removed or corrected. Here is a list as of October 20, 2025 (the CJK Compatibility Ideographs are abbreviated).

Note: All these removals have absolutely no effect on the results of the confusability algorithm. The first two lines, U+0226 Ȧ vs U+00C5 Å, and U+0227 ȧ vs U+00E5 å are not confusable at this time, because the combining dot above and the combining ring above are not considered confusable. The rest of the data pairs proposed to be removed are confusable at this time, because the first part either decomposes to the second part or decomposes to a string that is already confusable with the second part.

Removals

U+0226 Ȧ ; U+00C5 Å # both parts are non-NFD, remove
U+0227 ȧ ; U+00E5 å # both parts are non-NFD, remove
U+0340 ̀ ; U+0300 ̀ # first part is non-NFD, remove
U+0341 ́ ; U+0301 ́ # first part is non-NFD, remove
U+0343 ̓ ; U+0313 ̓ # first part is non-NFD, remove
U+0374 ʹ ; U+0027 ' # first part is non-NFD, remove
U+037E ; ; U+003B ; # first part is non-NFD, remove
U+0387 · ; U+00B7 · # first part is non-NFD, remove
U+0B94 ஔ ; U+0B92 U+0BB3 ஔ # first part is non-NFD, remove
U+1FBE ι ; U+0069 i # first part is non-NFD, remove
U+1FEF ` ; U+0027 ' # first part is non-NFD, remove
U+1FFD ´ ; U+0027 ' # first part is non-NFD, remove
U+2000 ; U+0020 # first part is non-NFD, remove
U+2001 ; U+0020 # first part is non-NFD, remove
U+2126 Ω ; U+03A9 Ω # first part is non-NFD, remove
U+212A K ; U+004B K # first part is non-NFD, remove

U+2329 〈 ; U+276C ❬ # first part is non-NFD, remove
U+232A 〉 ; U+276D ❭ # first part is non-NFD, remove
U+F900 豈 ; U+8C48 豈 # first part is non-NFD, remove
[...]
U+FAD9 □ ; U+9F8E 龍 # first part is non-NFD, remove
U+2F800 □ ; U+4E3D 丽 # first part is non-NFD, remove
[...]
U+2FA1D □ ; U+2A600 □ # first part is non-NFD, remove

Correction
U+2251 ≑ ; U+003D U+0307 U+0323 ≒̣ # second part is non-NFD, replace the second part with U+003D U+0323 U+0307

7. Other

7.1 Unicode Link Detection and Formatting [#507]

Recommended UTC actions

1. **Consensus**: Advance UTS #58 version 17.0 to an approved UTS, based on the pre-UTC-186 snapshot L2 doc (equivalent to revision 1 draft 7), for publication after UTC-186, and to be synchronized with Unicode versions starting with version 18.0. Publish the UTS #58 version 17.0 data files in https://www.unicode.org/Public/17.0.0/linkification. See L2/26-006 item 7.1.
2. **Action Item** for Mark Davis, PAG: Publish UTS #58 version 17.0 based on the pre-UTC-186 snapshot L2 doc (equivalent to revision 1 draft 7), with appropriate editorial adjustments. See L2/26-006 item 7.1.
3. **Action Item** for Markus Scherer, PAG: Publish the data files for UTS #58 version 17.0 in https://www.unicode.org/Public/17.0.0/linkification. See L2/26-006 item 7.1.

Document

L2/26-052 by Mark Davis & Markus Scherer. Snapshot of UTS #58 revision 1 draft 7.

This draft Unicode Technical Standard specifies a standard mechanism for detecting URLs embedded in plain text — in particular, detecting URLs containing non-ASCII characters. It also defines the minimally necessary escaping of non-ASCII code points in the Path, Query, and Fragment portions of a URL that aligns with the mechanism for detecting URLs.

Linkification is the process of adding links to URLs in plain text input, such as in emails, text messaging, or video meeting chats. The first step in this process is link detection, which is determining the boundaries of spans of text that contain URLs. That substring can then have a link applied to it in output text. The functions that perform these operations are called a linkifier and link detector, respectively.

The specifications for a URL don’t specify how to handle link detection, since they are only concerned with the structure in isolation, not when it is embedded within flowing text. The lack of a clear specification for link detection also causes many implementations to overuse percent escaping for non-ASCII characters when converting URLs into plain text.

Background information / discussion

See PRI-509

7.2 UTS #18 not up-to-date [#499]

Recommended UTC actions

1. **No Action**: No UTC Action is required. This has been fixed by making the correct version available from the "Latest Version" link https://www.unicode.org/reports/tr18/, completing the publication of Version 25 of UTS #18.

Feedback (verbatim)

Date/Time: Mon Nov 10 13:13:21 PST 2025
ReportID: ID20251110131321
Name: Michel Mariani
Report Type: Report Error in Publication/Data
Opt Subject: UTS #18 not up-to-date

The Unicode page about Unicode Regular Expressions (UTS #18), dated February 8, 2022, has never been updated since then.

https://www.unicode.org/reports/tr18/

The four new Unicode properties: IDS_Unary_Operator, ID_Compat_Math_Start, ID_Compat_Math_Continue, and NFKC_Simple_Casefold, introduced in Unicode 15.1, are only listed later in the Proposed Update page, dated May 11, 2023.

https://www.unicode.org/reports/tr18/proposed.html

It is unclear whether this document is a reference for third parties, but not having it kept up-to-date would explain why all implementations of regular expressions in JavaScript in navigators such as Safari or Firefox, or in the Electron framework based on Chromium, or even in the NPM module regexpu-core, appear to have no support for those four new properties: for instance, while the regex `\p{IDS_Binary_Operator}/u` is just fine, `\p{IDS_Unary_Operator}/u` triggers an "Invalid regular expression" error...

Background information / discussion

At the start of 2026:

- The "Latest Version" https://www.unicode.org/reports/tr18/
- is the same as https://www.unicode.org/reports/tr18/tr18-23.html dated 2022-02-08
- while the "Latest Proposed Update" https://www.unicode.org/reports/tr18/proposed.html
- is https://www.unicode.org/reports/tr18/tr18-24.html dated 2023-05-11
- but there is also https://www.unicode.org/reports/tr18/tr18-25.html dated 2025-01-16
- which is not linked from any other page.

It looks like something went wrong when Version 25 was published and the file for version 25 was not property copied over https://www.unicode.org/reports/tr18/, leaving it in a "never published" state. There is no issue with the content of any of the files.

Publication of version 25 was authorized by

- [176-C28] Consensus: UTC authorizes the release of PU UTS #18 as amended in discussion at UTC #176
- [176-A106] Action Item for Mark Davis, PAG: Update PU UTS #18 and prepare for publication.

The consensus was based on the version 24 snapshot L2/23-171, plus discussion during UTC #176.
Version 25 also includes changes for

- the earlier [172-A87] Action Item for Mark Davis, PAG: Produce proposed updates of UTS #51 and UTS #18 that contain the changes outlined in document L2/22-160, for future versions of these standards (incl. UTS #51 version 15.1 or 16). See document L2/22-124 item Emoji1. Discuss the emoji-test.txt filename and the Overqualified_Emoji value name.
- and same-meeting [176-A90] Action Item for Mark Davis, PAG: Add Indic_Conjunct_Break to the list of Full Properties in Section 2.7 of UTS #18, Unicode Regular Expressions, for a future revision of that UTS. See document L2/23-160 item 4.3.
- as well as the clarifications
- [181-A143] Action Item for Mark Davis, PAG: In Unicode Technical Standard #18, Unicode Regular Expressions, clarify the impact of changing property values on regular expressions. See L2/24-224 item 6.1.
- [181-A149] Action Item for Mark Davis, PAG: In UTS #18, change the discussion of Any/Assigned/ASCII to clarify that these are not General_Category values. See L2/24-224 item 8.1.

Given the UTC consensus and action items, we conclude that the incomplete publication of UTS #18 version 25 was a clerical error or oversight, and are completing the publication in January of 2026.