UTC #183 properties feedback & recommendations

Markus Scherer & Josh Hadley / Unicode properties & algorithms group, 2025-apr-16

Participants

The following people have contributed to this document:

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

1. UCD

1.1 UnicodeSet specification [#188]

Recommended UTC actions

1. **Consensus**: Authorize a Proposed Draft Unicode Technical Standard #61, Unicode Set Notation. This would not be a synchronized UTS.
2. **Action Item for** Robin Leroy, PAG: Prepare a Proposed Draft Unicode Technical Standard #61 based on document L2/25-127.
3. **Action Item for** Robin Leroy, PAG: Propose to ICU-TC the changes to the UnicodeSet grammar described in Sections 2.3 and 3 of PD UTS #61, and amend the draft grammar according to the decisions of ICU-TC.
4. **Action Item for** Mark Davis, PAG: Inform CLDR-TC of the PD UTS #61 and report to the UTC with a plan to coordinate the removal of the UnicodeSet specification from UTS #35 with the publication of UTS #61.
5. **Action Item for** Michelle Perham, UTC: Post a PRI for Proposed Draft UTS #61.

PAG input

From Robin Leroy, PAG.

The UTC gave us the following AI:

- UTC-176-A96 Action Item for Robin Leroy, Mark Davis, PAG: Review regular expression syntax throughout the Unicode Standard & Annexes, including how regex syntax used in the standard is defined and referenced, and provide recommendations to the UTC for Unicode 16.0. See L2/23-160 item 4.7 and ID20230621102117.

While we have a regex syntax defined in the core specification (Appendix A.2) and used there, the heavy usages of regular expressions, in, e.g., the segmentation algorithms, are really based on UnicodeSet syntax; this is understandable, since this is what we have actual implementations for.

UnicodeSet is defined in UTS #35; however, for its property sets, it refers to UTS #18. It is not quite honest in its reference; de facto CLDR has been using far more properties than the claimed UTS #18 Level 1 and RL2.5, including the Indic_Syllabic_Category which is not even in RL2.7. The specification diverges from implementation in other ways.

Besides its use in specifications, UnicodeSet syntax is heavily used in documents involved in the maintenance and discussion of our standards; via its heavily extended implementation in the online Unicode Utilities, it serves as our query language for the Unicode Character Database.

We do not want to standardize in a vacuum; that is how we got to the current state of the UnicodeSet specification. The changes that are appropriate for ICU should be approved by ICU-TC before we approve this specification. No changes are proposed that require changes in ICU4X, as ICU4X does not support character names and already supports bracketed-element in ranges. CLDR-TC, which owns the current UnicodeSet specification, such as it is, should also be involved.

1.2 Should the Hebrew nonspacing marks added in Unicode 4.1 be Diacritic? (Yes.) [#381]

Recommended UTC actions

1. **Consensus**: Assign the Diacritic property to U+05A2 HEBREW ACCENT ATNAH HAFUKH, U+05C5 HEBREW MARK LOWER DOT, and U+05C7 HEBREW POINT QAMATS QATAN. For Unicode 17.0. See L2/25-087 item 1.2.
2. **Action Item** for Robin Leroy, PAG: In UCD file PropList.txt, assign the Diacritic property to U+05A2 HEBREW ACCENT ATNAH HAFUKH, U+05C5 HEBREW MARK LOWER DOT, and U+05C7 HEBREW POINT QAMATS QATAN. For [target, e.g. Unicode 17.0]. See L2/25-087 item 1.2.

PAG input

From Robin Leroy, PAG:

While drafting data for UTC-184-C4 (HEAVY SHEVA), it came to my attention that Hebrew nonspacing marks have fairly regular property assignments: all are Diacritic, Points and puncta extraordinaria are also Alphabetic, cantillation marks are non-Alphabetic. The exception is that the three marks with Age=V4_1, U+05A2 HEBREW ACCENT ATNAH HAFUKH, U+05C5 HEBREW MARK LOWER DOT, and U+05C7 HEBREW POINT QAMATS QATAN, have Diacritic=No. Especially given the correlation with the Unicode version, this looks like an oversight. For instance, the contrast between QAMATS QATAN and QAMATS is a difference in vowel quality (see L2/24-150 p. 3), which should not matter for property assignments.

Background information / discussion

See https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5Cp%7Bsc%3DHebrew%7D%26%5Cp%7BMn%7D&g=dia+age&i=alpha+dia+ext for a list of Hebrew marks.
It was noted in L2/04-213 that the QAMATS vs. QAMATS QATAN situation is similar to the upcoming SHEVA vs. HEAVY SHEVA situation.

1.3 gc of Tolong Siki U+11DD9 SELA & U+11DDA HECAKA [#384]

Recommended UTC actions

1. **No Action**: PAG recommends no action; the difference in General_Category assignments between U+11DD9 and U+11DDA is intentional.

Feedback (verbatim)

Date/Time: Thu Feb 20 07:12:29 CST 2025
ReportID: ID20250220071229
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 514

U+11DD9 TOLONG SIKI SIGN SELA currently has General_Category=Modifier_Letter while U+11DDA TOLONG SIKI SIGN HECAKA has General_Category=Other_Letter, which may be unintentional considering that these two signs are very similar in nature. In the original proposal (L2/23-024) they are both categorised as Other_Letter, but I think that Modifier_Letter is probably the more appropriate property value for both because they are not “proper” letters of the alphabet.

Background information / discussion

PRI-514 Unicode 17.0 Alpha Review

This was mentioned in L2/24-064R2 §3 item 22:

> L2/23-024 Proposal to encode Tolong Siki
>
> - U+11DD9 (sign selā), being a vowel length mark, should be gc=Lm [Jan: rather than proposed Lo] (as well as Diacritic=Yes and Extender=Yes).
> - U+11DDA (sign hecakā) is fine as an Other Letter.
> - Other than that, this looks like a straightforward unicameral alphabetic script.

Elsewhere, Ken Whistler had written:

> I don't think it (hecakā) is a vowel length mark, as contrasted with selā, which is. The glottal stop seems to be a full segment, although perhaps in restricted distribution. See the discussion of Kurukh and Brahui phonology on Wikipedia, for instance, which shows length distinctions (and in some cases nasalization distinctions) for vowels, which when marked in the orthography can be considered vowel modifiers and/or extenders, etc. But I think it is safest to just leave the glottal stop as Lo and treat it as another letter.

1.4 DoNotEmit.txt typo “sequeences” [#385]

Recommended UTC actions

1. **No Action**: This has been fixed in the data file.

Feedback (verbatim)

Date/Time: Tue Feb 25 04:41:03 CST 2025
ReportID: ID20250225044103
Name: Ole Begemann
Report Type: Error Report
Opt Subject: DoNotEmit.txt in the UCD

The UCD file DoNotEmit.txt at
<https://www.unicode.org/Public/UCD/latest/ucd/DoNotEmit.txt> contains a spelling error in the initial comment block ("sequeences"):

```
# Preferred_Spelling:
# Miscellaneous characters and sequeences for which the Unicode Standard
# specifies a preferred spelling.
```

The error is still present in the current draft at
<https://www.unicode.org/Public/draft/ucd/DoNotEmit.txt> (as of 2025-02-25).

Is this something that can be fixed? Thanks!

1.5 Should kMandarin, kUnihanCore2020, and kTotalStrokes be listed in PropertyAliases.txt? (Yes.) [#390]

Recommended UTC actions

1. **Consensus**: Add the Informative properties kMandarin, kTotalStrokes, and kUnihanCore2020 to PropertyAliases.txt, with respective short aliases cjkMandarin, cjkTotalStrokes, and cjkUnihanCore2020. For Unicode Version 17.0. See L2/25-087 item 1.5.
2. **Action Item** for Robin Leroy, PAG: In UCD file PropertyAliases.txt, add the Informative properties kMandarin, kTotalStrokes, and kUnihanCore2020, with respective short aliases cjkMandarin, cjkTotalStrokes, and cjkUnihanCore2020. For Unicode Version 17.0. See L2/25-087 item 1.5.

PAG input

From Robin Leroy, PAG.

There are seven Informative Unihan properties: kAccountingNumeric, kMandarin, kOtherNumeric, kPrimaryNumeric, kRSUnicode, kTotalStrokes, and kUnihanCore2020.

Of those seven, only four are listed in PropertyAliases.txt:

```
cjkAccountingNumeric     ; kAccountingNumeric
cjkOtherNumeric          ; kOtherNumeric
cjkPrimaryNumeric        ; kPrimaryNumeric

cjkRSUnicode             ; kRSUnicode                 ; Unicode_Radical_Stroke; URS
```

This inconsistency makes it difficult to consistently handle properties that are subject to full UTC oversight (Normative and Informative properties), and creates uncertainty as to the meaning of the stability policy: https://www.unicode.org/policies/stability_policy.html#Alias_Stability states that « Property aliases, once defined in PropertyAliases.txt, will never be removed, nor will their precise spelling be changed. ». Does that mean that three of the Informative properties could actually have their aliases removed?

Background information / discussion

kMandarin and kTotalStrokes were made Informative in Unicode 6.1.0 by decision UTC-126-C3. See also the Proposed Update https://www.unicode.org/reports/tr38/tr38-10.html and UTC-131-C4. kUnihanCore2020 was created Informative by UTC-161-C23.

In contrast the four Informative properties that are already in PropertyAliases were already Informative in the first version of UAX #38, in Unicode 5.1.0, see https://www.unicode.org/reports/tr38/tr38-5.html.

1.6 CaseFolding.txt comment about MASSE/Maße: request to change to FUSS/Fuß [#392]

Recommended UTC actions

1. **Action Item** for Markus Scherer, PAG: In CaseFolding.txt, change the full case folding example from "MASSE"/"Maße" to "FUSS"/"Fuß". For Unicode 17.0. See L2/25-087 item 1.6.

Feedback (verbatim)

Date/Time: Thu Mar 20 08:32:35 CDT 2025
ReportID: ID20250320083235
Name: Dieter Niebel
Report Type: Website Problem
Opt Subject: CaseFolding Example

The Introduction to https://unicode.org/Public/UCD/latest/ucd/CaseFolding.txt reads:

```
#...
# The data supports both implementations that require simple case foldings
# (where string lengths don't change), and implementations that allow full case folding
# (where string lengths may grow). Note that where they can be supported, the
# full case foldings are superior: for example, they allow "MASSE" and "Maße" to match.
#...
```

A match of "MASSE" and "Maße" may not be desirable since "Masse" (engl. "mass") is distinct from "Maße" (engl. "measures"). To save the example a pair such as "FUSS" and "Fuß" could be used.

Regards

Background information / discussion

The current example is not wrong, but because "MASSE" could be one of two words with different pronunciations, it could be distracting.

- MASSE is ambiguous: Pronounced with a long vowel, it is the traditional (and still commonly used, and Unicode default) uppercase form of "Maße", while with a short vowel it is the uppercase of "Masse".
- Note that in Switzerland, ß is not used at all, so both words are spelled the same.
- FUSS and Fuß are the same word. (English "foot")
- None of this matters for the purpose of what the file says: The full case foldings allow each pair to match.

1.7 Tai Yo signs ccc=230 vs. 226 or 0 [#395]

Recommended UTC actions

1. **No change**: PAG recommends no data change: ccc=230 is appropriate for these combining marks.
2. **Action Item for** Ken Whistler, PAG: Add a paragraph about ccc=220 below UAX #44 Table 15. Canonical_Combining_Class Values explaining that some symbols can be above-or-below, similar to explaining ccc=224. For example, some ccc=230 musical symbols can also be above-or-below. For Unicode 17.0. See L2/25-087 item 1.7.

Feedback (verbatim)

Date/Time: Sat Feb 22 05:14:03 CST 2025
ReportID: ID20250222051403
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 514

The five combining marks of the Tai Yo script currently have Canonical_Combining_Class=Above (230):

```
U+1E6E3 TAI YO SIGN UE
U+1E6E6 TAI YO SIGN AU
U+1E6EE TAI YO SIGN AY
U+1E6EF TAI YO SIGN ANG
U+1E6F5 TAI YO SIGN OM
```

However, these signs sit above their base only when Tai Yo text is turned on its side to fit into a horizontal layout; when written vertically
like normal they sit on the right of their base. As such, they should be changed to CCC=Right (226).

The model case for this is U+18A9 MONGOLIAN LETTER ALI GALI DAGALGA which has CCC=Above_Left (228) based on its position in vertical layouts
only. When Mongolian is written horizontally, U+18A9 sits *below* left instead.

Another option would be to change the Tai Yo signs to CCC=0 since they’re all positioned on the same side anyway, but this might cause problems
if the tone marks mentioned in section 4.3 of L2/22-289r, which sit to the left of their base, are ever encoded in the future and users have to
guess whether to enter the vowel/final or the tone mark first; both orders would look identical since the two types of marks don’t interact
typographically but would not be canonically equivalent. I therefore recommend against this approach.

Date/Time: Mon Feb 24 05:47:44 CST 2025
ReportID: ID20250224054744
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 514

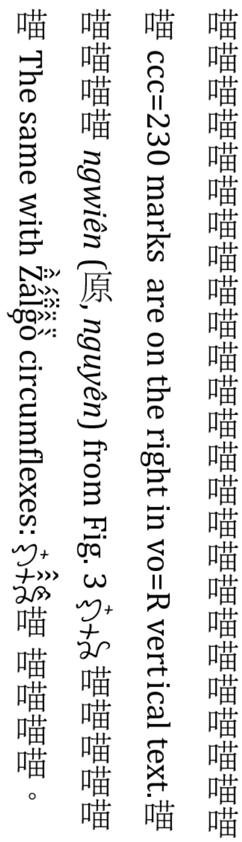
Regarding my previous feedback on the CCC values of Tai Yo combining marks: I have discovered that the question of whether to assign positional
combining classes based on vertical orientation isn't as straightforward, as the Old Uyghur script is also predominantly written vertically, but
its CCC values reflect the position of marks in a horizontal layout. I still think that the CCC values I proposed for Tai Yo, i.e. following the
Mongolian model, are the better option because Old Uyghur is not a living script. Old Uyghur text will almost always be found embedded in horizontal
layouts and therefore flipped on its side, which makes the CCC assignments based on horizontal orientation reasonable. The same is not true for Tai
Yo which has an active user community expecting the characters' properties to reflect their most common usage.

Background information / discussion

The PAG noted that Canonical_Combining_Class should be assigned according to the desired interaction with characters of the same CCCs (which do not reorder in normalization and stack in layout) and of different CCCs (which reorder in normalization and do not stack in layout). The name of the CCC is ultimately just a helpful identifier, rather than an absolute statement about positioning; see also musical combining marks, where marks below can be above—but what matters is that marks that would go on the same side of the same note have the same CCC.

On a script whose letters have Vertical_Orientation=R, in vertical text, combining marks above appear to the right. (Combining marks should follow the rotation of the grapheme cluster; see UAX #50, Section 3.2.1; but even in implementations that do not know about that, the marks above that are used with vo=R scripts are vo=R.)

As noted in the encoding proposal and by the submitter, some users of the script experiment with adding combining marks to indicate tone. If they were to do so with existing combining marks above, or apply combining marks above to Tai Yo letters for any other reason, these would show up to the right in vertical text (as illustrated by the adjacent figure, produced using Microsoft Word's support for East Asian vertical text layout), and should therefore stack with the rime marks; they should therefore have the same CCC.



1.8 Other=ISO_Comment [#396]

Recommended UTC actions

1. **Action Item** for Ken Whistler, PAG: In UAX #44, note that an alias for the General_Category grouping Other matches an alias for the property ISO_Comment under UAX44-LM3. For Unicode 17.0. See L2/25-087 item 1.8.

PAG input

From Robin Leroy, PAG

The tools used to generate the Unicode Character database and associated data files were recently updated to implement the Unicode 6.0 change to UAX44-LM3 (ignoring prefix "is").

As the tools are currently unable to distinguish \p{X=} from \p{X}, this caused \p{C} to be interpreted as \p{ISO_Comment=}. (\p{C} is meant to be equivalent to \p{General_Category=Other}, the set of code points whose General_Category is in the Other grouping, while \p{ISO_Comment=} is the set of code points whose ISO_Comment is the empty string).

While the Unicode tools should be able to distinguish \p{X=} from \p{X}, this seems like a treacherous enough issue that it is worth calling out in UAX #44. We propose updating the last paragraph of https://www.unicode.org/reports/tr44/#Matching_Symbolic as follows:

> Implementations sometimes use other syntactic constructs that interact with loose matching. For example, the property matching expression \p{L} may be defaulted to refer to the Unicode General_Category property: \p{General_Category=L}. General_Category aliases do not match binary property values under UAX44-LM3, so this does not conflict with a syntax such as \p{Whitespace} for binary properties. Care must be taken to avoid matching such constructs against other types of properties, since the alias isc of the Miscellaneous property ISO_Comment matches (when “is” is ignored) the alias C for the General_Category grouping Other. For more information about the use of property values in regular expressions and other environments, see Section 1.2, Properties, in Unicode Technical Standard #18, "Unicode Regular Expressions" [UTS18].

Background information / discussion

On the 6.0 change to LM3, see https://www.unicode.org/reports/tr44/tr44-6.html#UAX44-LM3, the proposed update https://www.unicode.org/reports/tr44/tr44-5.html#UAX44-LM3, UTC-120-A106, and L2/09-248.

Besides Other=C=isc=ISO_Comment, there are three more collisions between the General_Category value namespace and the non-binary property namespace (not related to the 6.0 change):

- Format=Cf=cf=Case_Folding;
- Cased_Letter=LC=lc=Lowercase_Mapping;
- Currency_Symbol=Sc=sc=Script.

1.9 PD-UAX #60 script-specific property names [#399]

Recommended UTC actions

1. **Consensus**: In TangutSources.txt, change the tag kRSTUnicode to kTGT_RSUnicode. For Unicode 17.0. See L2/25-087 item 1.9.
2. **Action Item** for Michel Suignard, Ken Whistler, PAG: In TangutSources.txt, change the tag kRSTUnicode to kTGT_RSUnicode. For Unicode 17.0. See L2/25-087 item 1.9.
3. **Consensus**: In NushuSources.txt, change the tag kSrc_NushuDuben to kNSHU_DubenSrc, and change the tag kReading to kNSHU_Reading. For Unicode 17.0. See L2/25-087 item 1.9.
4. **Action Item** for Michel Suignard, Ken Whistler, PAG: In NushuSources.txt, change the tag kSrc_NushuDuben to kNSHU_DubenSrc, and change the tag kReading to kNSHU_Reading. For Unicode 17.0. See L2/25-087 item 1.9.
5. **Action Item** for Ken Whistler, PAG: In UAX #44, adjust the names of fields in TangutSources.txt and in NushuSources.txt, according to L2/25-087 item 1.9. For Unicode 17.0.
6. **Action Item** for Michel Suignard, PAG: In PD-UAX #60, rename kTANG_MergedSrc to kTGT_MergedSrc, and kTANG_RSUnicode to kTGT_RSUnicode. For Unicode 18.0. See L2/25-087 item 1.9.

Feedback (verbatim)

PRI-520 Proposed Draft UAX #60, Data for Non Han Ideographic Scripts

Date/Time: Fri Feb 28 12:49:10 CST 2025
ReportID: ID20250228124910
Name: Kenneth W Whistler
Report Type: Public Review Issue
Opt Subject: 520

This proposed draft suggests, among other things, that script-specific property names for Jurchen, Tangut, and Nushu be regularized, with the same kind of 4-character script prefix added to all of them, parallel to the way Unihan properties use a "CJK" prefix. In general, for completely new properties, as for Jurchen, this is fine. However, for the existing Tangut and Nushu data files, this is a very disruptive suggestion that would create discontinuities in the data and break tooling.

In particular, the suggested changes for Tangut are very bad. The existing data uses kTGT_MergedSrc and kRSTUnicode. These are already unique identifiers, and despite the Tangut properties not yet formally being documented as UCD properties per se, they are already widely treated as extended UCD properties. Changing the identifiers would require the introduction of legacy aliases, create a discontinuity, and would break tooling in unicodetools, Unibook, and elsewhere.

Less would break for the Nushu changes. But there is no need whatsoever to change kSrc_NushuDuben, which is already clearly identified and distinct. The only value deserving a potential change is kReading, which is too ambiguous. However, even for that one, given the fact that software exists that has already used that identifier, any new change would probably require keeping "kReading" around as a legacy alias, anyway, obviating the need for this change in the first place.

I suggest all four of these Tangut and Nushu properties be simply reverted to what we have had in the files for years.

Background information / discussion

From PAG discussion:

- Sources should be normative
- Data fields and provisional properties can be dropped, changed, renamed, …
- We should not invent distinct aliases unnecessarily
- We should not rename existing things that are ok. Do rename kReading.
- A common prefix for related fields/properties is useful, especially for distinguishing from Unihan properties
- Want “Src” and “RS” somewhere in the name where appropriate
- Going forward, a prefix of 'k' + 4-letter script code makes sense
- kTGT_MergedSrc — keep as is → change PD-UAX #60
- kRSTUnicode — change to kTGT_RSUnicode → change PD-UAX #60
- kSrc_NushuDuben — change to kNSHU_DubenSrc
- kReading — change to kNSHU_Reading

2. Characters

2.1 Proposal to encode Katakana Digraph Yori [#366]

Recommended UTC actions

None from PAG. PAG reviewed the proposal and is fine with the SAH recommendation.

Document

L2/24-279 "Proposal to Encode One Kana Ligature" by CheonHyeong Sim

Proposal to encode KATAKANA DIGRAPH YORI in the Kana Extended-A block.

Background information / discussion

The proposed yori is to ヨリ (yo-ri) as コト (koto) is to コト (ko-to) as far as properties are concerned; as was noted in discussion, ヨリはコトと同じ (‘yori is the same as koto’).

3. Proposed new scripts & characters

PAG members reviewed the following proposals, provided feedback to SAH, and the feedback has been addressed.

No further recommended actions from our side.

- L2/24-272 Unicode request for lezh with curl -- Miller [SEW #586]
  - Another phonetic letter with curl, propertywise like both ɮ lezh and ʒ ezh with curl.
- L2/24-136 Draft Fifth Revised Proposal to encode characters for the English Phonotypic [SEW #456]
  - Punctuation:
    - Two wiggly exclamation marks, U+2E60 and U+2E61, used sentence-finally (see L2/24-277 pp. 11 & 36). Propertywise most like U+2E2E ؟ REVERSED QUESTION MARK (used at the end of rhetorical questions, see L2/07-004 p. 3), in particular Line_Break=EX (like U+0021 ! EXCLAMATION MARK, as noted in the proposal), Sentence_Terminal, Terminal_Punctuation, East_Asian_Width=Neutral. Differs from U+0021 in East_Asian_Width (the ASCII one is Narrow, whereas the new one has no history of usage in East Asian typography and is thus Neutral).
    - Two parentheses with middle ring, U+2E62 and U+2E63, propertywise like the double parentheses ⦅⦆ as noted in the proposal, p. 11. In particular, the right parenthesis is Line_Break=Close_Punctuation rather than Line_Break=Close_Parenthesis; the latter is used for parentheses that are expected to be used word-internally, as in (s)he. L2/24-277 shows that these characters are used to enclose words spelled in non-phonotypic English orthography; there is no evidence of word-internal usage. Note that absent evidence of word-internal usage in alphabetic scripts, we assign Line_Break=Close_Punctation, as we would otherwise prohibit line breaks that are desired in CJK contexts.
  - Letters[1]:
    - An uppercase counterpart for U+0277 ɷ LATIN SMALL LETTER CLOSED OMEGA, making this another IPA Extensions to Latin Extended D case pair like ɬ Ɬ L WITH BELT.
    - Eleven new Latin case pairs; these are propertywise like ƀ Ɓ VOLAPUK AE, which is part of one of the versions of the EPA. These are the first Latin case pairs outside the BMP[2], but there are plenty of case pairs in the SMP in other scripts already:
      1. PHONOTYPIC A WITH SWASH,
      2. PHONOTYPIC ROUNDTOP A,
      3. REVERSED SCRUPLE,
      4. PHONOTYPIC DIPHTHONG AI,
      5. O WITH CURL,
      6. CLOSED OMEGA WITH LONG STEM,
      7. TURNED CLOSED OMEGA,
      8. PHONOTYPIC TH,
      9. U WITH HOOK TAIL,
      10. U WITH NOTCH AT BOTTOM,
      11. REVERSED ENLARGED SMALL U/REVERSED U.
    - Two new lowercase letters with no uppercase counterparts (U+1DF70 LATIN SMALL LETTER I WITH PIGTAIL AT BOTTOM and U+1DF71 LATIN SMALL LETTER STRETCHED I), propertywise like U+1DF03 Ʞ LATIN SMALL LETTER REVERSED K.

        - Two new uppercase letters with no lowercase counterparts (U+1DF80 LATIN CAPITAL LETTER A WITH TOPBAR and U+1DF81 LATIN CAPITAL LETTER E WITH BENT TOPBAR).
          These are proposed with General_Category=Uppercase_Letter. This is not, in itself, new: the mathematical uppercase alphabets have plenty of those, and we also have U+03D2 ϒ GREEK UPSILON WITH HOOK SYMBOL. However, both the mathematical alphabets and ϒ are compatibility decomposable, and have the Math property; uppercase-only letters have so far been a math-only thing. Instead the PAG agreed to assign General_Category=Other_Letter, similar to existing letters with uppercase glyphs and no lowercase counterparts, *e.g.*, U+A7FB ꟻ LATIN EPIGRAPHIC LETTER REVERSED F.
          If at a later date these characters were found to have lowercase counterparts, they could be changed to General_Category=Uppercase_Letter. The lowercase would need to casefold to the uppercase, as has happened before.

[1] There are some typos in the case mappings in the UnicodeData.txt lines given in the proposal for 1DF6A (should lowercase to 1DF6B, not 1DF62) and 1DF6B (should titlecase to 1DF6A, not 1DF77), and in the General_Category for 1DF6D LATIN SMALL LETTER REVERSED SCRUPLE (should be Ll, not Lu).
[2] Note that they are wholly in the SMP; we do not allow case pairs to cross plane boundaries.

- L2/24-274 Proposal to encode Hebrew Point Sheva Na -- Jonathan Mosesson [SEW #537]
    - Propertywise like the existing HEBREW POINT SHEVA, in particular Alphabetic and Diacritic like all (but one, probably erroneously, see unicode-org/properties#381) of the other Hebrew points (contrast the accents which are typically non-Alphabetic Diacritic), and CCC10 (which is only used for SHEVA).
- L2/24-273 Latin: Character additions for Initial Teaching Alphabet [SEW #224]
    - Thirteen lowercase Latin letters with no uppercase counterparts; propertywise like U+1D6B ᵫ LATIN SMALL LETTER UE (an already-encoded letter of the ITA).
- L2/25-022 Devanagari letter Sindhi DDDA [SEW #592]
    - Yet another Devanagari DA, propertywise like existing ones (द DA, ड DDA, ड़ DDDA). Not like U+095C ड़ DDDHA, which has a decomposition (with a nukta) and Composition_Exclusion.
- L2/25-016 RFE letters [SEW #561]
    - Three more lowercase-only Latin letters with their modifier counterparts, propertywise like many other such pairs, *e.g.*, ʍʷ. In particular, the modifier letters are Diacritic and Lowercase (via Other_Lowercase). The naming of the modifier letters is atypical (SUPERSCRIPT instead of MODIFIER), but this has been brought to the attention of the script encoding working group by the proposal author.
- L2/24-233 Unicode request for additional Baroque ornament — Gavin Jared Bala, Kirk Miller [SEW #533]
    - Another ornament, propertywise like others cited in the proposal in co-occurrences with the proposed one: 𝆕, 𝆔 (used in the sequence 𝆔𝆕 for the *pincé*, see L2/25-015 p. 4). In particular, Line_Break=Alphabetic, Vertical_Orientation=Upright.
- L2/25-017 Miscellaneous musical symbols [SEW #560]
    - A maze of twisty little musical symbols, all alike as far as properties are concerned. This includes a bar, crescendos, noteheads, fermate, pedal marks, clefs, and an arpeggio, all with the same properties as existing ones, *e.g.*, 𝄁, 𝆒, 𝅝, 𝄐, 𝆮, 𝄞, 𝄒; in particular, Line_Break=Alphabetic and Vertical_Orientation=Upright.

- L2/25-064 Response to feedback on the buzz roll in L2/24-213 -- Gavin Jared Bala, Kirk Miller [SEW #608]
    - With this change, the character is now no longer propertywise like existing tremoli; instead it is like the existing stems (both the normal stem and the Sprechgesang stem). Besides the ccc and gc changes listed in the proposal, this means that it is no longer Diacritic, that it goes from sc=Inherited to sc=Common, and that it becomes Other_Grapheme_Extend to remain gcb=Extend.
- L2/25-061 Unicode request for compound stress mark -- Miller [SEW #613]
    - The proposed character (used for primary-or-secondary stress) should have the same properties as the existing characters ˈ and ˌ used for primary and secondary stress. In particular, lb=BB (an unusual line breaking class specifically designed for these kinds of dictionary characters, see https://www.unicode.org/reports/tr14/#BB), and Diacritic. Notably, this means that it should have gc=Lm, rather than gc=Sk as proposed.
- L2/25-036 One Small Kana Letter [SEW #612]
    - A small kana, propertywise like existing small kanas, e.g., ャ. In particular, besides the properties listed in the proposal, ea=W.
- L2/25-018 Musical bowing symbols [SEW #598]
    - More combining stems, propertywise like the existing combining stems. In particular, gcb=Extend via Other_Grapheme_Extend, vo=U, ccc=ATAR[1].

[1] no relation to the Atelier technique aéronautique de Rickenbach.

- L2/25-062 Proposal to encode 5 historic mathematical operators -- Mayer et al. [SEW #559]
    - Five mathematical operators, propertywise like most existing mathematical operators, in particular like U+22C7 ⋇ DIVISION TIMES, the modern equivalent of LEIBNIZIAN MULTIPLICATION-DIVISION SIGN. They differ in properties from the ordinary × and ÷, as these existing basic operators are saddled with complex considerations of compatibility with East Asian typesetting: ea=Ambiguous, lb=Ambiguous. In contrast, these signs are not typographically ambiguous (even the ambiguous operator!), and have ea=R, lb=AL.
- L2/25-075 Saudi Riyal Sign [SEW #619]
    - Another currency sign, propertywise like existing ones, e.g., U+20B9 ₹ INDIAN RUPEE SIGN.
    - In particular, Line_Break=PR, not PO. Note that the difference between PR and PO is nowadays only relevant in an East Asian context; see Section 3 of document L2/05-292, adopted by decision 105-C37, has useful background here. The point of picking lb=PR or lb=PO is to allow a break, as in お白湯は÷¥500頂戴しております。 or 上記税込価格にサービス料として10％÷を頂戴いたします. Since the currency sign is meant to be prefix in LTR text, it would likely be prefix in CJK text as currency signs usually are (with exceptions such as ¢), so that we would expect お白湯は÷SAR sign13頂戴しております。 and the character should be lb=PR like ¥.

4. Regex

4.1 Add - to PROP_VALUE syntax in UTS18 [#393]

Recommended UTC actions

1. **Action Item** for Robin Leroy, Mark Davis, PAG: Consider feedback ID20250129082145 and L2/25-087 item 4.1 in light of the proposed standard for Unicode Set Notation, and present a proposed update of UTS #18 to the UTC once the proposed standard for UnicodeSet Notation has reached the Draft stage.

Feedback (verbatim)

Date/Time: Wed Jan 20, 2025
ReportID: ID20250129082145
Name: Owen Shepherd
Report Type: Feedback
Opt Subject: UTC Doc: Add - to PROP_VALUE syntax in TS18

```
The syntax for unicode property specifications in regular expressions, according
to TS18, looks like this:
CHARACTER_CLASS
:= '\' [pP] '{' PROP_SPEC '}'
:= '[:' COMPLEMENT? PROP_SPEC ':]'
PROP_SPEC := PROP_NAME (RELATION PROP_VALUE)?
PROP_NAME := ID_CHAR+
ID_CHAR := [A-Za-z0-9\ \-_]
RELATION := "=" | "≠" | "!="
PROP_VALUE := LITERAL*
LITERAL
:= ESCAPE (SYNTAX_CHAR | SPECIAL_CHAR)
:= NON_SYNTAX_CHAR
:= HEX
:= "\\N{" ID_CHAR+ "}"
ESCAPE := "\\"
SYNTAX_CHAR := [\- \[ \] \{ \} / \\ \^ |]
SPECIAL_CHAR := [abcefnrtu]
NON_SYNTAX_CHAR := [^SYNTAX_CHAR]
HEX
:= "\\u" HEX_CHAR{4}
:= "\\u{" SP? CODEPOINT (SP CODEPOINT)* SP? "}"
HEX_CHAR := [0-9A-Fa-f]
CODEPOINT := "10" HEX_CHAR{4} | HEX_CHAR{1,5}
SP := " "+
COMPLEMENT := "^"
Narrowing in on the specification of PROP_VALUE, we see that it’s comprised
of 'LITERAL*'.
None of the productions of 'LITERAL' admit the ‘-’ character. This means
that regex syntax such as '\p{name=ZERO WIDTH NO-BREAK SPACE}', and
```

```
'\p{name=surrogate-D800}', (both used as examples in this document), are
invalid.
I propose altering the production of 'PROP_VALUE' to:
PROP_VALUE := (LITERAL | '-')*
```

Background information / discussion

The submitter is correct about the defect in the cited syntax. However, this is best addressed not by the proposed fix, but by a more systematic approach.

In particular, the example syntax in UTS #18 is a strange beast, and we may not want to invest too much in it. Right now it has the problem of being the closest thing we have to a specification of the actual syntax used in UnicodeSet, via this sentence in UTS #35:

> Unicode property sets are defined as described in UTS #18: Unicode Regular Expressions [UTS18], Level 1 and RL2.5, including the syntax where given.

As such, in addition to serving as an illustrative syntax, it is also trying its best to define UnicodeSet property query syntax, to the detriment of the readability and coherence of UTS #18, which, as a standard, is about defining features, not syntax.

Once we actually have a UnicodeSet standard — and we are working on one now (see 1.1 in this document) — we could significantly tone down the UTS #18 specification of the example syntax, so that it can actually be limited to an illustrative example; things such as the dual `[:` and `\p{` syntaxes have no business being in an example. If more advanced examples are needed, UTS #18 could also reference the UnicodeSet standard and give examples in UnicodeSet syntax.

5. Security

5.1 Change Identifier_Type for 40 characters to Recommended [#361]

Recommended UTC actions

1. **Consensus**: Change Identifier_Type for 40 characters listed in L2/25-032R to Recommended. For Unicode 17.0. See L2/25-087 item 5.1.
2. **Action Item** for Josh Hadley, PAG: Change Identifier_Type for 40 characters listed in L2/25-032R to Recommended. For Unicode 17.0. See L2/25-087 item 5.1.

PAG input

Related to Action item 161-A58 for Mark Davis, Michel Suignard, Karan Miśra, Asmus Freytag, PAG

> Review the list of characters in section six (6) of L2/19-329R and suggest changes to identifier type.

an adjustment to **Recommended** is proposed for 40 alphabetic characters that are part of ICANN's Second Level Reference Label Generation Rules (RefLGR) and/or the Root Zone LGR (RZ-LGR), but do not have Identifier_Type Recommended.

Recommendation

These characters are listed in L2/25-032R and should be given Identifier_Type Recommended in light of the extensive research documenting that they are used for languages with a robust enough infrastructure and use to qualify for inclusion in the DNS Root Zone LGR and Reference LGRs for the Second Level respectively.

Background information / discussion

The current Identifier_Type of these is as follows:

```
Uncommon_Use 0192 0199 01B4 01DD 024D 0253-0254 0256-0257 025B 0263 0268-0269 0272 0289 0292 0DA6
Obsolete 068E
Technical 028B 0DA6 17CB-17CD 17D0
Uppercase equivalents 0191 0198 01B2 01B3 018E 024C 0181 0186 0189-018A 0190 0194 0196-0197 019D
0244 01B7
```

The analysis assumes case pairs should always be treated the same, so the uppercase characters are not broken out by their original Identifier_Type, but they should also be updated to Recommended.

The full list of characters and documentation, including the modern languages these are used with and the links to specific source documents is found in document L2/25-032R.

For convenience in reviewing here's a link to the UnicodeSet for the above.

5.2 Improve IDN Security Profile For Identifiers in UTS #39 [#362]

Recommended UTC actions

1. **Action Item** for Mark Davis, PAG: Add text based on L2/25-087 item 5.2 into UTS #39. For Unicode 17.0.

PAG input

From Asmus Freytag, PAG

In the context of reviewing UTS #39 for other action items, I noticed the following issue:

Issue

The current section "IDN Security Profiles for Identifiers" is cursory and ignores much of the recent work done by IETF and ICANN in this area over the last decade.

Recommendation

The section should be updated by adding the following text:

> RFC7940 specifies a machine-readable format, for expressing profiles for IDNs with specific features that support security against spoofing. Such profiles are known as "Label Generation Rules" (LGR) and are typically defined for a script or language, with features that allow support of multiple LGRs for the same zone.
>
> Using the LGR format, implementations can:
>
> 1. **Select a repertoire of characters or enumerated character sequences.** The selection of the set of characters can take into account the intersection between IDNA2008 allowed values and Identifier_Status resp. Identifier_Type.
> 2. **Limit certain characters to one or more of an enumerated set of sequences.** That's useful to limit diacritics to contexts where they are expected, or to exclude standalone use of some characters only found in certain sequences.
> 3. **Provide a context rule**. A context rule is an anchored regular expression describing either a required or prohibited context for a character or sequence inside a label. This is useful for the kind of restrictions on Indic scripts, so you can stay within safe boundaries of what rendering engines can render in a distinct way. One application is to implement Limited Contexts for Joining Controls.
> 4. **Make two characters / sequences "blocked" variants of each other**. Either one is allowed, but the first label registered blocks the label differing only by a variant character or sequence at that position.
> 5. **Use a "whole label evaluation" rule to validate an identifier.** A WLE rule is a regular expression describing a pattern either for a valid or an invalid label. This can be used, for example, to prevent digit set mixing in the same label, or making sure a label is consistent in the choice of a regional alternative for a character.

The LGR mechanism, as defined in RFC 7940, allows you to present all of these restrictions in a machine readable form which you can then convert into whatever works best for validating and resolving competing identifier registrations in your system (or symbol table). The format was designed with IDNA2008 in mind and is used in the normative definition of IDN profiles for the DNS Root Zone. However, the concepts carry over to other types of identifiers. The regular expressions can be specified in terms of Unicode properties, or custom attributes assigned to the characters in the LGR.

The variant mechanism is most effective in mitigating homographs or semantic synonyms, but for general confusables detection, a second layer of processing, specifically addressing confusables, may be appropriate.

Rationale

Rather than a relatively weak and hand-waving pointer to repertoire restrictions, a text passage like this can establish a suitable correlation between up-to-date identifier profile mechanisms and this Unicode Technical Standard.

Arguably, the RFC also implements something that Unicode so far has not, which is a language / data format to express an identifier profile in a machine readable format. Describing the elements of such a format seems useful, even though there's no suggestion here that Unicode should define its own at this point.

Background information / discussion

For examples of published profiles in the RFC7940 format for several dozen scripts and multiple languages, see ICANN: Internationalized Domain Names, in particular
RZ-LGR-5 Overview and Summary
as well as
Reference Label Generation Rules (LGR) for
the Second Level: Overview and Summary and here.

5.3 Factors for determining Identifier_Type [#363]

Recommended UTC actions

1. **Consensus**: Adopt the proposed goals and factors for assigning Identifier_Type values from L2/25-069. See L2/25-087 item 5.3.
2. **Action Item** for Markus Scherer, PAG: Document for PAG data maintainers the goals and factors for assigning Identifier_Type values from L2/25-069. See L2/25-087 item 5.3.
3. **Consensus**: Incorporate proposed language further defining identifier types and their usage into UTS #39. For Unicode 17.0. See L2/25-087 item 5.3.
4. **Action Item** for Markus Scherer, PAG: Incorporate proposed language further defining identifier types and their usage into UTS #39. For Unicode 17.0. See L2/25-087 item 5.3.

PAG input

In reference to:
172-A64 for Asmus Freytag, PAG:

> For UTS #39, look at publicly available data from the ICANN root zone LGR project to look at adjustments to the Uncommon_Use property and further clarifications of its definition.

In UTS #39 we define "Recommended" scripts as those that are in "widespread everyday common use:" This concept should carry through when defining character subsets inside these scripts, but that needs a bit more precision.

This feedback proposes a written documentation for the goals and factors for assigning Identifier_Type values other than those that derive from other character properties or combinations thereof. Note that a key recommendation is to switch the default for newly assigned characters to Uncommon_Use -- unless positive evidence to the contrary can be documented.

Document

After it is approved, document L2/25-069 "Factors used in determining the Identifier_Type of characters" should be archived where it is accessible to those making decisions on Identifier_Type.

Language proposed for UTS #39

> The following identifier types may be assigned singly or in combination. The values are based on best available information and may be updated when new information becomes available. Multiple classifications may be possible, particularly where a character is of one type in a commonly used writing system and of different type in another context.
>
> - **Uncommon_Use** focuses on usage of the character in orthographies for living languages. Uncommon_Use can also represent the absence of confirmed or credible data on a level of usage that would correspond to "common everyday use" for an orthography in widespread use. Uncommon_Use is the default for newly encoded characters for all scripts other than Excluded scripts, unless a sufficient level of usage can be confirmed for one or more specific orthographies at the time of encoding.

- **Obsolete** focuses on the degree to which a character is in common modern use. If a writing system or orthography using a character have fallen out of use, or a character is no longer used in a given context, it is marked Obsolete. A character can become obsolete in the context of a writing system; it is not required that the entire writing system has fallen out of use. A character may be Obsolete in a widely used writing system, but also part of an orthography where it is in Uncommon_Use.
- **Technical** focuses on the purpose of use. If a character is limited to particular types of texts or forms part of a notation without concurring everyday use, then it would be appropriate to categorize it as Technical. Technical uses can comprise for liturgical purposes, poetry, phonetic notation and so on. A character may have a common technical use, but also be used in one or more orthographies at a level that is marked by Uncommon_Use, or it could be a Obsolete as a character for general use.
- **Inclusion** focuses on punctuation or characters that look like punctuation and that should not be automatically included in identifiers but may be appropriate in specific identifier environments. Usually the reason is that the character can be confused with syntax characters in the given environment.

The other Identifier Types are largely, if not fully determined by a character's other property values and are therefore automatically assigned.

5.4 Reclassify Identifier_Type for characters that were misclassified [#364]

Recommended UTC actions

1. **Consensus**: Change Identifier_Type of 392 characters marked Proposed:Uncommon_Use, Proposed:Technical or Proposed:Obsolete in document L2/25-034R4 to Uncommon_Use, Technical, or Obsolete, respectively. For Unicode 17.0. See L2/25-087 item 5.4.
2. **Action Item** for Josh Hadley, PAG: Change Identifier_Type of 392 characters marked Proposed:Uncommon_Use, Proposed:Technical or Proposed:Obsolete in document L2/25-034R4 to Uncommon_Use, Technical or Obsolete, respectively. For Unicode 17.0. See L2/25-087 item 5.4.
3. **Action Item** for Markus Scherer, PAG: Add proposed language on “Combining Marks / Needed for NFD” from document L2/25-034R4 to UTS #39. For Unicode 17.0. See L2/25-087 item 5.4.
4. **Action Item**: for Markus Scherer, Add proposed language from document L2/25-034R4 on the challenges of using Tibetan in an identifier context to UAX #31. For Unicode 17.0. See L2/25-087 item 5.4.
5. **Consensus**: Change Identifier_Type of 753 characters from document L2/25-033R4 to their proposed new values. For Unicode 17.0. See L2/25-087 item 5.4.
6. **Action Item** for Josh Hadley, PAG: Change Identifier_Type of 753 characters from document L2/25-033R4 to their proposed new values. For Unicode 17.0. See L2/25-087 item 5.4.

Document

Two L2 documents have been filed for this issue:
L2/25-033R4, Characters excluded from both MSR and Reference LGR but allowed in UTS#39
L2/25-034R4, Characters recommended in both UTS#39 and in MSR but excluded from the Root Zone or Reference LGR

This is in response to the following action items
161-A58 Mark Davis, Michel Suignard, Karan Miśra, Asmus Freytag, PAG
Review the list of characters in section six (6) of L2/19-329R and suggest changes to identifier type.

161-A59 Mark Davis, Michel Suignard, Karan Miśra, Asmus Freytag, PAG
Review the list of characters in section seven (7) of L2/19-329R and suggest changes to identifier type.

The original classification of Identifier_Type appears to have been too lenient, often appearing to default to Recommended where the actual usage of a character could not be ascertained easily. This submission is in response to a pair of action items to review available external research published in the IDN community and to propose an adjustment of Identifier_Type to reflect the information on usage of affected characters.

Recommendation

The Identifier_Type for a set of 386 characters from document L2/25-034R4 should be changed to Uncommon_Use, and 1 character should be changed to Technical, and 5 characters should be changed to Obsolete, based on the fact that the expert teams charged with reviewing them for the ICANN Root Zone LGR and Reference LGR for the Second Level could not come up with evidence that they are used in common everyday writing, even for minority languages in reasonably widespread use. Consequently, they declined to include them in the respective LGRs. (The specific breakdown of proposed assignments is found in L2/25-034R4). This list was adjusted in review by PAG experts.

753 characters from document L2/25-033R4 should **not** be considered Recommended, based on an independent analysis that has found indications that they should have been assigned Identifier_Type Uncommon_Use, Obsolete, Technical or Inclusion instead, based on information available at the time of encoding, such as proposal documents. Where they are already assigned Identifier_Type Inclusion, no change is recommended. After review, some characters that were Recommended are retained, reducing the number of proposed changes. (The specific breakdown of proposed assignments, adjusted in review by PAG experts, is found in L2/25-033R4).

Uppercase and lowercase characters should have the same Identifier_Type. This should be enforced with an invariance test.

The Tibetan script has not been fully vetted for Identifier purposes. It is recommended to add a warning about that until the Identifier_Type properties are in a state befitting *default identifiers*. The warning should also cover the specific challenge created by Tibetan stacking. (See document L2/25-034R4 for proposed language)

Background

There are more than a thousand non-Han characters with Identifier_Type Recommended that should be reclassified because they appear to fail reasonable criteria for being needed in identifiers.[1] They come in two sets. For one set, an independent analysis MSR has found indications that they should have been considered Uncommon_Use, Obsolete or Technical based on information available at the time of encoding. That set is discussed in document L2/25-033R3.

The second set contains characters that were tentatively included in the MSR but upon further review by local expert teams from the RZ-LGR project were found to not be needed for any language or minority language in reasonably widespread use. That determination started with the EGIDS classification as a proxy but made further adjustments in expert review. The set of these characters was amended in review by PAG experts; they

are listed in document L2/25-033R4 and should be considered Uncommon_Use unless other supporting evidence is found and documented.

This initial analysis cited was carried out for the purposes of defining the repertoire for IDN Top Level Domain names for the DNS Root Zone. There are some restrictions that are specific to the Root Zone, such as a prohibition on digits, so a follow-on effort determined how to relax these restrictions in a manner appropriate for the needs of Second-Level Domains. This resulted in the Second-Level Reference Label Generation Rules RefLGR. The characters listed in document L2/25-034R4 are those that were not added to the RefLGR, for lack of evidence of their use in everyday common writing for any language or minority language in vigorous and reasonably widespread use.

The implication here is that any character not included in the Reference LGR for lack of documented or identifiable usage should be considered Obsoleted or Technical, where that can be established, and otherwise considered Uncommon_Use for Unicode's default identifiers—until such time as independent evidence to the contrary is produced. Until then, in lack of a demonstrated use case, it seems not helpful to continue to suggest that these characters should be supported as recommended. This also applies to some of the sets of native digits, where local experts considered them obsolete for the purpose of identifiers.

Arriving at a precise cutoff for Uncommon_Use and the other identifier types is difficult because there is no single source or perfect information on the use of writing systems, and the details of such use are changing over time. Accordingly, this document suggests that the UTC should consider the published results of the cited research as one of the better sources of information available and only deviate from it on the basis of even better information.

All decisions for the classification of characters in MSR, or inclusion in RZ-LGR and RefLGR are documented and sourced on the character level; the same is not true for Unicode's classification, so it is not easily possible to verify any of the decisions that underlie the classification published in UTS #39. By first making the alignment proposed here, and then carefully documenting deviations, a positive side effect might be that the classification overall becomes more transparent and reviewable.

The original analysis being carried out in the context of IDNs for the DNS Root Zone did not consider uppercase characters or digits. However, the proposed re-classification should apply equally to both parts of a case pair, and the proposals therefore encompass the uppercase equivalents. Likewise, any native digit sets that were explicitly excluded from the RefLGR because they were not found to be in modern common use are also included in the set proposed for Identifier_Type Uncommon_Use.

Defaults for characters newly assigned for Unicode 17.0.0 and after. Newly encoded characters for existing scripts should not be given Identifier_Type Recommended by default, as most often, scripts are supplemented with additional Technical or Obsolete characters. For the occasional modern characters, unless they are intended for support of a particularly vigorous living language with well established orthography in widespread use in the community, they should be considered Uncommon_Use by default. The same should be retroactively applied to the few characters added to the recommended scripts in the last few Unicode versions since the cutoff for the MSR work (as documented in the latest MSR version).

[1] The alignment is closer in the opposite direction. There are only 38 characters that are part of the Root Zone or Reference LGR but not recommended for default identifiers. They are discussed in a separate feedback.

5.5 Bopomofo should be classed "Limited_Use" for default identifiers [#371]

Recommended UTC actions

1. **Consensus**: Change Bopomofo to "Limited Use" in UAX #31 because the script is limited to educational use. For Unicode 17.0. See L2/25-087 item 5.5.
2. **Action Item** for Asmus Freytag, PAG: Move Bopomofo from Table 5 Recommended scripts to Table 7 Limited Use scripts in UAX #31 and note the reason for the change is because the script is limited to educational use. Update Identifier_Status and Identifier_Type properties accordingly. For Unicode 17.0. See L2/25-087 item 5.5.

PAG input

Bopomofo is currently listed in: https://www.unicode.org/reports/tr31/#Table_Recommended_Scripts

Bopomofo is undoubtedly widely used, but it isn't used for the full range of "everyday" uses, being mainly relegated to educational purposes.[1] For any subset of characters, such a narrow scope of use would mean that they are considered "Technical", but we don't have that status for whole scripts. We should consider moving Bopomofo to the list of "Limited_Use" scripts.

Recommendation:

Change Bopomofo to "Limited_Use". This change is predicated on the restricted type of use for the Bopomofo script compared to all the general use scripts that are Recommended. While this affects the Identifier_Status and Identifier_Type properties, it does not prevent, for example, the support of Bopomofo in important classes of identifiers, such as IDN for which the change in listing merely ratifies the existing disparity in the number of supporting domains.

Background information / discussion

In both the DNS Root Zone and the Second Level, the Root Zone Label Generation Rules (RZ-LGR) or Reference LGRs[2] published by ICANN with active participation of the communities where Bopomofo is used, the script is excluded from IDNs.

In the repository of IDN Practices at IANA,[3] Bopomofo is supported exactly for as many domains as Tifinagh, Sundanese, Ol Chiki and many other Limited_Use scripts, while Chinese is listed as supported by more than thirty times as many domains. (Note, this repository covers gTLDs, not ccTLDs). The exact match in the number of supporting domains is a strong indicator that registries for a certain small number of zones simply decided to support all such scripts systematically, rather than singling out any based on perceived requirements or demand.

All other recommended scripts being supported by between ten to thirty times as many domains speaks for a systematic difference in how the domain name industry views these scripts and therefore supports the case made here that Bopomofo is currently not classified correctly and should never have been "Recommended" for default identifiers.

[1] https://en.wikipedia.org/wiki/Bopomofo#Modern_use

[2] In the Root Zone, all "Recommended" scripts are eligible, with the exception of Bopomofo. Note that the Reference LGR for the Second Level does include some scripts that are "Limited_Use" based on community request, demonstrating that a change in listing would not be a permanent bar against *endorsed* use for Second Level domains, should a community request it.

[3] The IANA repository documents existing practice, predating the creation and publication of Reference LGRs. The latter were instituted to help insure a modicum of consistency and minimal standards for the support of scripts and languages. While the former were developed in private by the various registries, the latter follow a process that involves community input and public review.

5.6 UAX #31 Table 4 Excluded Scripts: update for Unicode 17 [#389]

Recommended UTC actions

1. **Consensus**: Add Sidetic, Tolong Siki, Chisoi, Beria Erfe, and Tai Yo to UAX #31 Table 4 Excluded Scripts. For Unicode 17.0. See L2/25-087 item 5.6.
2. **Action Item** for Markus Scherer, PAG: Add Sidetic, Tolong Siki, Chisoi, Beria Erfe, and Tai Yo to UAX #31 Table 4 Excluded Scripts. For Unicode 17.0. See L2/25-087 item 5.6.

PAG input

From Roozbeh Pournader, PAG

We need to update UAX #31 Table 4 Excluded Scripts to add the five new scripts in Unicode 17: Sidetic, Tolong Siki, Chisoi, Beria Erfe, and Tai Yo

6. Other

6.1 PRI-509 Linkification feedback before UTC-183 [#394]

Recommended UTC actions

1. **Action Item** for Roozbeh Pournader, PAG: Review linkification feedback ID20250326055934 from Ebrahim Byagowi to see if changes are needed. See L2/25-087 item 6.1.
2. **No further action**: All feedback has been reviewed, and one typo has been fixed. PAG recommends no further action as a result of public feedback.

Feedback (verbatim)

Several items of feedback are listed below, with response comments from PAG discussion interleaved:

Date/Time: Fri Dec 06 21:31:48 CST 2024
ReportID: ID20241206213148
Name: Dennis Tan
Report Type: Public Review Issue
Opt Subject: 509

In section 3 (Link Detection Algorithm), subsection "Initiation", the document uses the following reference for "top-level domains" https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains. Perhaps it would be better suited to use a more authoritative source and one that is updated regularly — e.g., https://www.iana.org/domains/root/db. The wiki page doesn't even list the IDN top-level domains.

Mark Davis:
This had already been fixed before the January UTC.

Date/Time: Wed Dec 18 02:41:23 CST 2024
ReportID: ID20241218024123
Name: Hank Nussbacher
Report Type: Public Review Issue
Opt Subject: 509

In section 7 - https://www.unicode.org/reports/tr58/#test-data - might i suggest that you include test data for bidirectional content for Linkification - like from Arabic or Hebrew?

Mark Davis:
This is not necessary; the algorithm is applied in memory order. For best display of URLs with BIDI characters, people should see https://unicode.org/reports/tr9/#HL4Example3, but that is orthogonal to this draft specification. We do mention that in the https://www.unicode.org/reports/tr58/#security section.

Date/Time: Mon Jan 20 08:04:59 CST 2025
ReportID: ID20250120080459
Name: Arnt Gulbrandsen
Report Type: Public Review Issue
Opt Subject: 509

Hi,

I have compared the UTS58 draft with a few linkifiers.

One omission I saw is that another linkifier tolerates and ignores U+00AD (soft hyphen). The commit message is terser than terse, but hints that someone sends text of the form "foo example.com/foo/bar bar" with a soft hyphen after the full stop and/or slashes.

It's not clear to me that this is worth bothering with. Your call.

Mark Davis:
This had already been fixed before the January UTC.

Date/Time: Mon Feb 10 11:17:13 CST 2025
ReportID: ID20250210111713
Name: Jules Bertholet
Report Type: Public Review Issue
Opt Subject: 509

In addition to being used ⌜like⌝ ⌞this⌟,
the half brackets, and possibly also the half parentheses,
can also be used ⌜like⌟ ⌞this⌝
(imitating the East Asian corner brackets).
The Link_Paired_Opener property should be array-valued to reflect this.

Mark Davis:
This runs into the same problem as in https://www.unicode.org/reports/tr58/#review-issues “Quotation Marks”, so it can’t be added.

Date/Time: Wed Mar 26 05:59:34 CDT 2025
ReportID: ID20250326055934
Name: Ebrahim Byagowi
Report Type: Public Review Issue
Opt Subject: 509

I like to provide a quick drive by comment about https://www.unicode.org/reports/tr58/tr58-2.html

I think the lack of standard recommendation on how URLs should actually be displayed has caused https://issuetracker.google.com/issues/40665886 and essentially breaking Persian text in URL bars and misrendering of Emoji skin tones using ZWJ in Chrome as described on the tracker.

The same situation exists in Safari but URLs are hidden there for the most part but if one tries to edit a URL containing ZWNJ things go wrong by double encoding already encoded ZWNJ in the URL like https://phabricator.wikimedia.org/F58924232 (unfortunately this isn't always reproducible in Safari but is annoying enough and comes from the same root ZWNJ being displayed by its code)

I'll understand that you may consider these as browsers bugs but after seeing https://www.unicode.org/reports/tr58/tr58-2.html and the lengthy discussion I had in Chromium's bug tracker which made the developers sure understand what is going on https://issuetracker.google.com/issues/40665886
I felt if it were some official recommendation things could go more smoothly.

Mark Davis:
For best display of URLs with BIDI characters, people should see https://unicode.org/reports/tr9/#HL4Example3, but that is orthogonal to this draft specification. We do mention that in the https://www.unicode.org/reports/tr58/#security section.

Date/Time: Fri Mar 28 11:53:15 CDT 2025
ReportID: ID20250328115315
Name: cketti
Report Type: Public Review Issue
Opt Subject: 509

Step 4.7. of the termination algorithm currently reads "If LT == Open", but it should be "If LT == Close". (Step 4.6. handles "LT == Open")

Mark Davis:
Fixed in the newest draft.

Received after the PRI closing date:

Date/Time: Mon Apr 07 06:31:16 CDT 2025
ReportID: ID20250407063116
Name: Arnt Gulbrandsen
Report Type: Public Review Issue
Opt Subject: 509

Hi,

I ran across a bug today that I think points out a relevant problem in UTS58: A user expected 普遍适用测试。我爱你 to be linkified as
普遍适用测试。我爱你 (note the changing dot).

Chrome and some other web browsers map "。" to "." in domains when you hit enter after typing/pasting into the address bar. I do feel
that at least U+06D4 and U+3002 ought to be mapped to the ASCII dot in UTS58 since it's such a common mistake. ("。" and "." are even
on the same key on the Chinese keyboards I've seen.)

I mention U+06D4 and U+3002 because I've seen those mistakenly used in "domain names" in the course of my work. U+FF61 and others
might also be used mistakenly in theory, but I haven't seen that.

Markus Scherer:
UTS #58 defers to a WhatWG spec for identifying and parsing the early parts of a URL, including the domain name.
UTS #46 does provide a mapping table which maps U+3002 to U+002E. Same for U+FF0E & U+FF61.
U+06D4 is the ARABIC FULL STOP and does not look like a dot. UTS #46 does not map it to U+002E.