UTC #180 properties feedback & recommendations

Markus Scherer & Josh Hadley / Unicode properties & algorithms group, 2024-jul-16

Participants

The following people have contributed to this document:

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

1. UCD

1.1 More precisely define what Terminal_Punctuation means [#259]

Recommended UTC actions

1. **Consensus**: In UAX #44, clarify the description of Terminal_Punctation to mention that it is normally not part of the preceding word, as described in L2/24-162 item 1.1. For Unicode Version 16.0.
2. **Action Item** for Ken Whistler, PAG: In UAX #44, clarify the description of Terminal_Punctation to mention that it is normally not part of the preceding word, as described in L2/24-162 item 1.1. For Unicode Version 16.0.

PAG input

From Mark Davis, PAG

Terminal_Punctuation is an ill-defined property, which severely limits its usefulness. This came up in discussions of PAG issue “Are abbreviation marks Terminal_Punctuation? (No.)” (internally #246, see UTC-179 PAG report).

It cannot be "any punctuation that terminates anything, saying nothing about whether it is included in what it terminates", so it has to be far better specified than it is. There was some discussion of this in Background information / discussion in “Are abbreviation marks Terminal_Punctuation? (No.)”. That seems to say saying at least something about the terminating punctuation, than it is not included in the entity that it terminates. But that is clearly not true for !, since that is part of the sentence that it terminates!

We should be explicit about

1. the kinds of entities that are being terminated by a Terminal_Punctuation
2. whether or not the Terminal_Punctuation is included with an entity it terminates

For point 1, I suggest that we focus on sentences, clauses, and phrases. That is, a **Terminal_Punctuation** character generally terminates a sentence, clause, or phrase. Usage may vary by language, and by context.

For point 2, I suggest that we say that the **Terminal_Punctuation are** included in what they terminate. If there are punctuation in **Terminal_Punctuation** that would be excluded under this rule, we could consider whether or not it would be worth adding a **Terminal_Separator** property that would contain them.

Background information / discussion

It was noted in PAG discussion that part of the renewed interest in the Terminal_Punctuation property was the prospect of using it in a linkification algorithm in ICU-22657 and L2/24-122. For that application, what matters is not that terminal punctuation is part of what it terminates, whatever that may be. Neither is it obvious that this statement is true: figuring out which clause contains the commas delimiting subordinate clauses in German seems neither the UTC's business nor actually useful to algorithms; and word separators, which *are* Terminal_Punctation and delimit words, are not part of words.
Instead, the crucial point is that it is *not* generally part of the word preceding it. Incidentally, this is why we found that dedicated abbreviation marks are not Terminal_Punctuation in UTC-179-C21.

The description of Terminal_Punctation could clarify that as follows:

Property	Type	Status	Description
Terminal_Punctuation	B	I	Punctuation characters that generally mark the end of textual units. **Add:** These marks are not part of the word preceding them. A notable exception is the U+002E FULL STOP. Terminal_Punctuation characters may be part of some larger textual unit that they terminate.

1.2 UAX #31 shows different character for GREEK ANO TELEIA [#284]

Recommended UTC actions

1. **No Action**: PAG recommends no action: The issue has been corrected editorially.

Feedback (verbatim)

PRI-491

Date/Time: Mon Feb 05 21:59:46 CST 2024
ReportID: ID20240205215946
Name: Jules Bertholet
Report Type: Error Report
Opt Subject: Unicode® Standard Annex #31: Unicode Identifiers and Syntax

In Section 2.5 (https://www.unicode.org/reports/tr31/#Backward_Compatibility),
there is the line:

> U+0387 (·) GREEK ANO TELEIA

However, the character between the parentheses is not actually U+0387 (·); U+00B7 (·) MIDDLE DOT is erroneously used instead. (U+0387 NFC-decomposes to U+00B7.)

1.3 DoNotEmit wrong replacement for 0149 [#285]

Recommended UTC actions

1. **Consensus**: Change the line for U+0149 LATIN SMALL LETTER N PRECEDED BY APOSTROPHE in DoNotEmit.txt to: `0149; 2019 006E; Deprecated` — For Unicode 16.0. See L2/24-162 item 1.3.
2. **Action Item** for Roozbeh Pournader, PAG: Change the line for U+0149 LATIN SMALL LETTER N PRECEDED BY APOSTROPHE in DoNotEmit.txt to: `0149; 2019 006E; Deprecated`— For Unicode 16.0. See L2/24-162 item 1.3.

Feedback (verbatim)

PRI-489

Date/Time: Fri Jan 12 22:00:43 CST 2024
ReportID: ID20240112220043
Name: Ben Scarborough
Report Type: Public Review Issue
Opt Subject: 489

This is specifically a response to L2/24-021, the current draft of a new DoNotEmit.txt file meant for Unicode 16.0.

One line reads as follows:

0149; 02BC 006E; Deprecated # LATIN SMALL LETTER N PRECEDED BY APOSTROPHE; MODIFIER LETTER APOSTROPHE, LATIN SMALL LETTER N

The character in question, U+0149 LATIN SMALL LETTER N PRECEDED BY APOSTROPHE, has had the Deprecated property since Unicode 5.2. According to L2/08-287, the character was deprecated because its compatibility decomposition used the wrong apostrophe character—RIGHT SINGLE QUOTATION MARK is the preferred character for Afrikaans, not MODIFIER LETTER APOSTROPHE.

The line in DoNotEmit.txt should use the preferred string instead of U+0149's compatibility decomposition. The line should be changed to:

0149; 2019 006E; Deprecated # LATIN SMALL LETTER N PRECEDED BY APOSTROPHE; RIGHT SINGLE QUOTATION MARK, LATIN SMALL LETTER N

1.4 DoNotEmit.txt: discourage decomposed forms of some Cyrillic letters [#293]

Recommended UTC actions

1. **No Action**: PAG recommends no action: canonically equivalent sequences are out of scope for DoNotEmit.txt.

Feedback (verbatim)

Date/Time: Wed May 15 10:35:38 CDT 2024
ReportID: ID20240515103538
Name: Mikhail Merkuryev
Report Type: Public Review Issue
Opt Subject: DoNotEmit.txt

DoNotEmit.txt: Add to “Discouraged” or “Preferred spelling” decomposition of those Cyrillic letter known by me:
Ёё Йй Ўў
(Cyrillic capital/small letter Io, Cyrillic capital/small letter Short I, Cyrillic capital/small letter Short U)

e.g.
0415 0308 → 0401 # Cyrillic capital letter Ie + combining diaeresis → Cyrillic capital letter Io

Maybe others, but I don’t know.

Її (Cyrillic capital/small letter Yi) is tricky and IDK what to do: discouraged in decomposed form in modern Ukrainian text, but maybe allowed in Old Slavonic.

Rationale: most Cyrillic fonts do not lay combining marks properly, and common breve has other shape different from Cyrillic. And these four letters in modern shape are really distinct entities.

1.5 add Duployan to scx for several characters [#294]

Recommended UTC actions

1. **Consensus**: Add Duployan (Dupl) to the Script_Extensions of U+00B7 MIDDLE DOT, U+0307 COMBINING DOT ABOVE, U+0308 COMBINING DIAERESIS, U+030A COMBINING RING ABOVE, U+0323 COMBINING DOT BELOW, U+0324 COMBINING DIAERESIS BELOW, and U+2E3C STENOGRAPHIC FULL STOP. For Unicode 16.0. See L2/24-162 item 1.5 and PRI-502#ID20240522040217.
2. **Action Item** for Roozbeh Pournader, PAG: Add Duployan (Dupl) to the Script_Extensions of U+00B7 MIDDLE DOT, U+0307 COMBINING DOT ABOVE, U+0308 COMBINING DIAERESIS, U+030A COMBINING RING ABOVE, U+0323 COMBINING DOT BELOW, U+0324 COMBINING DIAERESIS BELOW, and U+2E3C STENOGRAPHIC FULL STOP. For Unicode Version 16.0. See L2/24-162 item 1.5.
3. **Action Item** for Rick McGowan, PAG: Ask Charlotte Buff for evidence for the use in Duployan shorthand of other combining diacritical marks. See L2/24-162 item 1.5.

Feedback (verbatim)

PRI-502

Date/Time: Wed May 22 04:02:17 CDT 2024
ReportID: ID20240522040217
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 502

I propose adding Duployan (Dupl) to the Script_Extensions for the following code points based on annotations in the names list for the Duployan block, the contents of UTN #37, “Duployan Shorthand”, and the original encoding proposal for Duployan, L2/10-272r2:

U+00B7 MIDDLE DOT
U+0300 COMBINING GRAVE ACCENT
U+0301 COMBINING ACUTE ACCENT
U+0304 COMBINING MACRON
U+0306 COMBINING BREVE
U+0307 COMBINING DOT ABOVE
U+0308 COMBINING DIAERESIS
U+030A COMBINING RING ABOVE
U+0323 COMBINING DOT BELOW
U+0324 COMBINING DIAERESIS BELOW
U+0331 COMBINING MACRON BELOW
U+2E3C STENOGRAPHIC FULL STOP

Duployan for Romanian also makes use of U+00B0 DEGREE SIGN in numerical contexts, though as this character is in common use in a variety of writing systems and has no explicit Script_Extensions as of now there would likely be little benefit to specifically listing just Duployan.

Background information / discussion

Roozbeh noted the following:
I can confirm that the following are explicitly mentioned in the NamesList, UTN #37, and L2/10-272r2: U+00B7, U+0307, U+0308, U+030A, U+0323, U+0324, U+2E3C. I also agree that the degree sign is too common to add an scx list for. The rest seem to be only ambiguously refered to, in a list ending in "&c":

> Combining diacritical marks on vowels.
>
> Several Duployan orthographies use combining diacritical marks to distinguish vowels. These diacritics include acute, grave, breve, macron, under macron, over dot, under dot, diaeresis, under diaeresis, &c. They can appear on orienting vowels, circle vowels, and nasal vowels (On, and An). Although there are several vowel letters with marks included in the allocation, these are not decomposable as a combining sequence, as the diacritic marks change position along with their "base" orienting vowel. Combining diacritics indicate vowels with diacritics that consistently appear above or below the base character, no
> matter the adjacent joining characters.

1.6 DoNotEmit U+13217 EGYPTIAN HIEROGLYPH N035A [#295]

Recommended UTC actions

1. **Consensus**: Remove Egyptian hieroglyphs from DoNotEmit.txt. For Unicode Version 16.0. See L2/24-162 item 1.6.

Feedback (verbatim)

PRI-502

Date/Time: Thu May 23 09:34:28 CDT 2024
ReportID: ID20240523093428
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 502

DoNotEmit.txt currently includes the following line:

13217; 13216 13430 13216 13430 13216; Precomposed_Hieroglyph # EGYPTIAN HIEROGLYPH N035A; EGYPTIAN HIEROGLYPH N035, EGYPTIAN HIEROGLYPH VERTICAL JOINER, EGYPTIAN HIEROGLYPH N035, EGYPTIAN HIEROGLYPH VERTICAL JOINER, EGYPTIAN HIEROGLYPH N035

However, section 11.4.3 of the core spec specifically states:

»For example, U+13217 □ EGYPTIAN HIEROGLYPH N035A apparently could be represented by the sequence <13216, 13430, 13216, 13430, 13216>. However, this compound sign is considered a single entity in Ancient Egyptian by Egyptologists, because the compound sign conveys a function that is not covered by the meaning of its individual parts. As a result, the atomic character U+13217 should be used.«

I do not know which representation is actually the preferred one, so either this DoNotEmit entry or this section of the core spec should be removed.

Background information / discussion

The Properties and Algorithms Working Group reviewed this feedback and found that the data in DoNotEmit.txt 16.0β is consistent with the text of the Unicode 15.1 core specification, but not with the text of the Unicode 16.0β core specification referenced by the feedback.

The issue was referred to the Script Encoding Working Group.
The Script Encoding Working Group found that the Unicode 16.0β core specification text correctly reflects the current guidance from Egyptologists. However, this guidance has evolved in the recent past: in Unicode 14.0 the recommendation was systematically in favour of the compound signs, in 15.0 it was systematically in favour of the sequences, and now it is a more nuanced approach. While the guidance appears to be converging, it is not stable enough to be included in DoNotEmit.txt; further, the examples listed in the core specification are far from comprehensive. Recommendations informed and stabilized by usage, together with a general review of relations between hieroglyphs as described by the Unikemet database, would be needed to produce useful DoNotEmit data for hieroglyphs. For now, we should not pretend that we have usable data there.

The change has been made in the working draft of the UCD, so no action item needs to be recorded.

1.7 scx=Hani for two new strokes 31E4..31E5 [#296]

Recommended UTC actions

1. **No Action**: The recommended changes have been made.

Feedback (verbatim)

PRI-502

Date/Time: Thu May 23 09:57:09 CDT 2024
ReportID: ID20240523095709
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 502

The two new CJK strokes, U+31E4 CJK STROKE HXG and U+31E5 CJK STROKE SZP, currently have no explicit Script_Extensions. They should be given the property value “Hani” like all the other CJK strokes (U+31C0..U+31E3).

Background information / discussion

Since these characters are new in Unicode 16.0, we do not need a UTC decision to correct the property assignments. The changes recommended by Charlotte Buff have been made in the UCD. No further action is required, except to thank Charlotte Buff for her vigilance.

1.8 Should Unikemet Normative and Informative properties be listed in Property(Value)Aliases? (Yes.) [#298]

Recommended UTC actions

1. **Action Item** for Robin Leroy, PAG: Add the Unikemet Normative and Informative properties to PropertyAliases.txt and PropertyValueAliases.txt, namely kEH_Cat, kEH_Desc, kEH_HG, kEH_IFAO, kEH_JSesh, kEH_NoMirror, and kEH_NoRotate (as well as kEH_Core if the UTC does not follow the recommendation of the Script Encoding Working Group to change its status to Provisional). For Unicode Version 16.0. See L2/24-162 item 1.8.

PAG input

From Robin Leroy, PAG: UAX #57 defines some Normative and Informative properties; contrary to the Provisional properties, these are subject to the Property Stability policy. In the 16.0β data files, they are not listed in PropertyAliases.txt and PropertyValueAliases.txt. Just like the Normative and Informative properties from UAX #38, they should be listed there; in particular, this would make it easy to enforce the stability policy (properties should not be removed from these files), and would also simplify handling the properties in the tooling (no need to put them in the Extra files).

The change has been made in the working draft of the UCD, excluding kEH_Core in anticipation of a recommendation from the Script Encoding Working Group (internally “Egyptian Hieroglyphs: 3-tier property system for Ancient Egyptian hieroglyphs”, internal SEW issue 496). We are requesting that an action item be recorded regardless, so to ensure that the files correctly reflect the status of the properties after UTC-180.

Background information / discussion

Any Informative or Normative property must have property aliases, and property value aliases if appropriate. For certain non-property properties, it might be beneficial to elevate them to Provisional properties so that they can be clearly referenced with a name (as opposed to referencing by such-and-such field in file such-and-such).

1.9 Is gc=Lo *really* right for the two CHINESE SMALL ER? (No.) [#299]

Recommended UTC actions

1. **Note**: The provisionally assigned characters U+16FF2 CHINESE SMALL SIMPLIFIED ER and U+16FF3 CHINESE SMALL TRADITIONAL ER will have General_Category Modifier_Letter (Lm).

PAG input

From Robin Leroy, PAG:

In “PAG review of L2/23-284 Proposal to encode two small form CJK characters for Chinese” (internally #233, reported to UTC-179 in L2/24-064R2 §2.8) we had discussed the GC of the provisionally assigned U+16FF2 CHINESE SMALL SIMPLIFIED ER and U+16FF3 CHINESE SMALL TRADITIONAL ER. The possibility of gc=Lm had been considered, but rejected, among other reasons because we hesitated to start picking out some ideographs and giving them gc=Lm based on function.

However, it was pointed out that we already have characters that are gc=Lm and script=Hani, e.g., the iteration mark U+3005. The new characters behave like U+3005 in line breaking (lb=NS), and are also non-diacritic extenders; likewise they do not stand on their own as ideographs. Given this precedent, gc=Lm seems more appropriate.

Background information / discussion

It was noted in discussion that currently, `\p{Hani}&\p{gc=Lo}` is equal to the scope of the Unihan database (which can also be characterized as those characters with a kRSUnicode value, as those characters with a kTotalStrokes value, or those characters with at least one IRG source). If these new characters were gc=Lo, they would be the first exception to this relation. However, this relation seems like a product of happenstance rather than a fundamental statement about gc=Lm, gc=Lo, or the scope of various properties in the Unihan database. This relation therefore was not determinant in the recommendation to make these characters gc=Lm. In particular, it is possible that some of the characters proposed in L2/24-125 (CJK Abbreviations) would be exceptions to this relation between General_Category, Script, and the Unihan scope.

It was also noted that the same arguments that were made in L2/24-064R2 §2.8 against Diacritic (behaviour as a phonetic complement, usage standing in for another syllable—another ideograph—that has become an [ə̸]) can be made against gc=Lm. However, the gc classification is less specific than the Diacritic property; see, *e.g.*, the non-Diacritic vowel modifier letters in the Phonetic Extensions Supplement, used in some transcriptions of diphthongs (L2/04-132 §E.4.1). Since that General_Category assignment makes these characters more consistent with others that behave similarly, there was no sustained objection to Lm.

1.10 The description of Hex_Digit is incorrect [#304]

Recommended UTC actions

1. **Action Item** for Ken Whistler, PAG: In Unicode Standard Annex #44, Unicode Character Database, correct the description of property Hex_Digit as mentioned in L2/24-162 item 1.10, to mention that it only includes those compatibility equivalents to ASCII hex digits that have Decomposition_Type=Wide. For Unicode Version 16.0. See L2/24-162 item 1.10.

PAG input

From Robin Leroy, PAG. UAX #44 reads:

Property	Type	Status	Description
Hex_Digit	B	I	Characters commonly used for the representation of hexadecimal numbers, plus their compatibility equivalents.

That description would include 331 characters in Unicode Version 15.1: [0０𝟎𝟘𝟢𝟬𝟶🯰 1１𝟏𝟙𝟣𝟭𝟷🯱 2２𝟐𝟚𝟤𝟮𝟸🯲 3３𝟑𝟛𝟥𝟯𝟹🯳 4４𝟒𝟜𝟦𝟰𝟺🯴 5５𝟓𝟝𝟧𝟱𝟻🯵 6６𝟔𝟞𝟨𝟲𝟼🯶 7７𝟕𝟟𝟩𝟳𝟽🯷 8８𝟖𝟠𝟪𝟴𝟾🯸 9９𝟗𝟡𝟫𝟵𝟿🯹 ⓪⁰₀ ①¹₁ ②²₂ ③³₃ ④⁴₄ ⑤⁵₅ ⑥⁶₆ ⑦⁷₇ ⑧⁸₈ ⑨⁹₉ aａ𝐚𝑎𝒂𝒶𝓪𝔞𝕒𝖆𝖺𝗮𝘢𝙖𝚊ⓐᵃªₐA Ａ𝐀𝐴𝑨𝒜𝓐𝔄𝔸𝕬𝖠𝗔𝘈𝘼𝙰Ⓐᴬ🄰 bｂ𝐛𝑏𝒃𝒷𝓫𝔟𝕓𝖇𝖻𝗯𝘣𝙗𝚋ⓑᵇBＢℬ 𝐁𝐵𝑩𝓑𝔅𝔹𝕭𝖡𝗕𝘉𝘽𝙱Ⓑᴮ🄱 c ｃ 𝐜𝑐𝒄𝒸𝓬𝔠𝕔𝖈𝖼𝗰𝘤𝙘𝚌ⓒᶜCＣ ℂℭ𝐂𝐶𝑪𝒞𝓒𝕮𝖢𝗖𝘊𝘾𝙲Ⓒ🄫🅭🄲 dｄ ⅆ𝐝𝑑𝒅𝒹𝓭𝔡𝕕𝖉𝖽𝗱𝘥𝙙𝚍ⓓᵈD Ｄ ⅅ𝐃𝐷𝑫𝒟𝓓𝔇𝔻𝕯𝖣𝗗𝘋𝘿𝙳Ⓓᴰ🄳 eｅℯⅇ𝐞𝑒𝒆𝓮𝔢𝕖𝖊𝖾𝗲𝘦𝙚𝚎ⓔᵉₑE Ｅℰ𝐄𝐸𝑬𝓔𝔈𝔼𝕰𝖤𝗘𝘌𝙀𝙴Ⓔᴱ🄴 fｆ𝐟𝑓𝒇𝒻𝓯𝔣𝕗𝖋𝖿𝗳𝘧𝙛𝚏ⓕᶠFＦℱ 𝐅𝐹𝑭𝓕𝔉𝔽𝕱𝖥𝗙𝘍𝙁𝙵Ⓕꟳ🄵], but \p{Hex_Digit} only contains 44: [0０ 1１ 2２ 3３ 4４ 5５ 6６ 7７ 8８ 9９ aａAＡ bｂBＢ cｃCＣ dｄDＤ eｅEＥ fｆFＦ].

The description should be amended as follows:

Property	Type	Status	Description
Hex_Digit	B	I	Characters commonly used for the representation of hexadecimal numbers, plus their compatibility equivalents **Add:** with Decomposition_Type=Wide.

Background information / discussion

The current description was slightly more accurate in Unicode Version 3.1: \p{U3.1:Hex_Digit} has 94 characters, [0０𝟎𝟘𝟢𝟬𝟶 1１𝟏𝟙𝟣𝟭𝟷 2２𝟐𝟚𝟤𝟮𝟸 3３𝟑𝟛𝟥𝟯𝟹 4４𝟒𝟜𝟦𝟰𝟺 5５𝟓𝟝𝟧𝟱𝟻 6６𝟔𝟞𝟨𝟲𝟼 7７𝟕𝟟𝟩𝟳𝟽 8８𝟖𝟠𝟪𝟴𝟾 9９𝟗𝟡𝟫𝟵𝟿 aａAＡ bｂBＢ cｃCＣ dｄDＤ eｅEＥ fｆFＦ]. However even then it did not include all dt=Font compatibility equivalents of ASCII hex digits, only those of the decimal digits; nor did it include other decomposition types (Circle, Compat, Sub, and Super).

Hex_Digit was limited to dt=Wide in Unicode Version 3.2, by decision UTC-86-M13:

[86-M13] Motion: Approve the proplist file (L2/01-082) as amended in the meeting with:

- removal of math alphanumeric digits from hex digits,

- addition of ASCII hex digits as a category,
- making non-character, bidi, ASCII hex digits, joiner, and two whitespace categories normative with the names to be decided later, and
- have two whitespace categories that differ only by inclusion of ZWSP.

Moved by Mark Davis, seconded by Asmus Freytag

11 for (Apple, Basis, Compaq, HP, IBM, Justsystem, Progress, RLG, Sun, Sybase, Unisys)
0 against
1 abstain (Microsoft)

1.11 DoNotEmit.txt and Kashmiri [#305]

Recommended UTC actions

1. **Action Item** for Roozbeh Pournader, PAG: Remove the three DoNotEmit entries for Hamza_Forms mentioned in feedback ID20240625045929. For Unicode 16.0. See L2/24-162 item 1.11.

Feedback (verbatim)

PRI-502

Date/Time: Tue Jun 25 04:59:29 CDT 2024
ReportID: ID20240625045929
Name: Richard Ishida
Report Type: Public Review Issue [PAG]
Opt Subject: 502

The Do Not Emit data file contains the following lines.

```
Unset
---
  ;ٔب; Hamza_Form # ARABIC LETTER BEH, ARABIC HAMZA ABOVE; ARABIC LETTER BEH WITH HAMZA ABOVE
ځ ;ٔج; Hamza_Form # ARABIC LETTER HAH, ARABIC HAMZA ABOVE; ARABIC LETTER HAH WITH HAMZA ABOVE
ڑ ;ٔر; Hamza_Form # ARABIC LETTER REH, ARABIC HAMZA ABOVE; ARABIC LETTER REH WITH HAMZA ABOVE
---
```

These mappings are valid for orthographies that use the atomic character as a letter of the alphabet, but they are not appropriate for Kashmiri, which uses the hamza as a vowel diacritic, not as an ijam.

See
https://r12a.github.io/scripts/arab/ks.html#non_canonical
https://r12a.github.io/scripts/arab/homographs.html#nehomographs

Although the hamza is not a tashkil, the distinction made here follows the logic in the standard related to ijam vs tashkil usage. See https://r12a.github.io/scripts/arab/homographs.html#ijam_tashkil

Having special rules for just a few, arbitrary combinations of hamza and base in Kashmiri is likely not only to lead to inconsistency in encoding, leading to failures in searching and other operations, but it is also a recipe for confusion for users. Note that all other uses of the vowel hamza above a base character in Kashmiri have no corresponding ijam (and if there's a possibility that atomic characters for these pairings may be created for other languages in the future this adds further complexity).

It seems to me that one solution to this would be to add some sort of qualification, by language, for these entries. Or perhaps it would be helpful to make these combinations canonically equivalent and remove them from Do Not Emit. Users would then be able to type the items either way, and end up with compatible text.

2. Characters

2.1 YANGQIN SIGN SLOW TWO through FOUR [#286]

Recommended UTC actions

1. **Consensus**: Change the name of the provisionally assigned characters U+16FF4, U+16FF5, and U+16FF6 from YANGQIN SIGN SLOW TWO, YANGQIN SIGN SLOW THREE, and YANGQIN SIGN SLOW FOUR to YANGQIN SIGN SLOW ONE BEAT, YANGQIN SIGN SLOW THREE HALF BEATS, and YANGQIN SIGN SLOW TWO BEATS.
2. **Action Item** for Ken Whistler, UTC: Update the Pipeline to change the name of the provisionally assigned characters U+16FF4, U+16FF5, and U+16FF6 from YANGQIN SIGN SLOW TWO, YANGQIN SIGN SLOW THREE, and YANGQIN SIGN SLOW FOUR to YANGQIN SIGN SLOW ONE BEAT, YANGQIN SIGN SLOW THREE HALF BEATS, and YANGQIN SIGN SLOW TWO BEATS. See L2/24-162 item 2.1.
3. **Note**: U+16FF4, U+16FF5, and U+16FF6 will have General_Category Letter_Number (Nl).

Document

- L2/24-071R by Eiso Chan: Proposal to encode three stable extended Suzhou Numeral–like letters for Cantonese Music
    - I propose three extended Suzhou Numeral–like letters in this document, which are used for Cantonese Music (广东音乐/粤乐).
- Code points: 16FF4..16FF6
- CJK recommendations doc and section: L2/24-067 §38

Background information / discussion

The PAG noted that the proposed General_Category=Other_Letter (Lo) for these characters could be highly disruptive, as these would be the first `Hani Other_Letter`s to without Unihan data, and would add exceptions to the relation between the Ideographic and Unified_Ideograph properties. General_Category=Other_Symbol (So) was considered; while less egregious, it too would add create inconsistencies with existing characters, including the Yangqin sign slow one (half-beat), which is unified with the existing gc=Nl Suzhou numeral six (〤). General_Category=Letter_Number (Nl) seems preferable. Since that requires a numeric value, it was suggested to make that value the number of beats, rather than half-beats, and to make the character names more explicit.

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.

- L2/24-106 Request to encode DOT ABOVE characters for the ORIYA/ODIA script in the UCS -- Evans [SEW #454]
    - Propertywise like the neighbouring ORIYA SIGN OVERLINE; one of the non-`Alphabetic InSC=Dependent_Vowel`s.
- L2/24-107 Revised Proposal to encode three Armenian superscript characters - Hossep Dolatian [SEW #432]
    - The UnicodeData.txt lines in L2/24-107 incorrectly have U+055B and U+055C instead of U+058B and U+058C; this has been corrected in the draft UCD changes.
    - Like existing modifier small letters in other cased scripts, these are Other_Lowercase and Diacritic and have a compatibility Decomposition_Mapping of type `<super>`.
- L2/24-105 Unicode request for double caron - Miller [SEW #297]
    - A diacritic above, like the ones from SEW #298, with the same properties.
- L2/24-131 Proposal to encode Arabic Crown Letters (حروف التاج) -- Goregaokar, Hosny, Yang, Hasan [SEW #353]
    - A mark above, a number of letters. First occurrence of jt=L in Arabic, but that does not have any special implications for our algorithms.
    - Collation is interesting.
    - The crowned letters (whether encoded atomically or using the combining mark) should be primary-equal after their bareheaded counterparts.
    - Since this was meant as a case distinction, tertiary differences make sense; but these would be confusing relative to other relative distinctions among Arabic characters, and would run into technical hurdles in the sifter. Secondary differences (like the ḥarakāt) should work fine for foreseeable use cases (mainly fuzzy string search based on primary collation weights).
- L2/24-130R Unicode request for modifier voiceless implosive letters -- Miller [SEW #442]
    - More Latin modifier letters, propertywise like the existing ones; Other_Lowercase, Diacritic, `<super>`-decomposing to the corresponding small letters. The one with a retroflex hook has a DoNotEmit entry.

4. Normalization

4.1 document NFC_QC=N = Full_Composition_Exclusion [#310]

Recommended UTC actions

1. **Action Item** for Markus Scherer, PAG: Document in UAX #15 and in UAX #44 that NFC_QC=N = Full_Composition_Exclusion. For Unicode 16.0. See L2/24-162 item 4.1.

Feedback (verbatim)

Date/Time: Tue May 28 13:45:01 CDT 2024
ReportID: ID20240528134501
Name: Alexei Chimendez
Report Type: Public Review Issue
Opt Subject: 506

The derived property NFC_Quick_Check=N is defined as characters that "cannot (ever) occur in [Normalization Form C]" (UAX15 §9, UAX44 §5.7.5). The definition of NFC (full decomposition followed by full recomposition) guarantees that the only characters that cannot occur in NFC are those with the property Full_Composition_Exclusion=Y, which will only appear in their decomposed form.

These two sets are, by definition, always equal. Because this is not immediately obvious, I think it would be of help to implementers to add this fact as a note in UAX15; or alternatively in some other relevant place, such as UAX44, Table 9.

Such an annotation can be useful, for example, to people implementing normalization in environments where memory usage is a concern, since they can choose to implement a check for the property Comp_Ex=Y as a lookup in the same table used for NFC_QC=N, or vice versa.

Background information / discussion

Makes sense. ICU does rely on this equivalence, and uses it in its implementation of Full_Composition_Exclusion.

4.2 UAX #15 “transform text according to the Stream-Safe Text Format” [#311]

Recommended UTC actions

1. **Action Item** for Markus Scherer, PAG: Make editorial changes to conformance clause UAX15-C4 along the lines of L2/24-162 item 4.2. For Unicode 16.0. See L2/24-162 item 4.2.

Feedback (verbatim)

Date/Time: Tue May 28 13:40:01 CDT 2024
ReportID: ID20240528134001
Name: Alexei Chimendez
Report Type: Public Review Issue
Opt Subject: 506

The conformance clause UAX15-C4 states:

> A process that purports to transform text according to the Stream-Safe Text Format must do so in accordance with the specifications in this annex.

This should be:

> [...] transform text *into* the Stream-Safe Text Format [...]

(which is done "according to" a process.)

Background information / discussion

Modified suggestion: This would be improved by writing the conformance clause as "... according to the Stream-Safe Text Process..." and then linking that to
https://www.unicode.org/reports/tr15/tr15-55.html#UAX15-D4
instead of to Section 13, per se. D4 defines the *process*, which is what one wants to claim conformance to, and the process then incorporates the D3 definition of the *format*.

5. Line Break

5.1 Linebreak property value of U+00AD SOFT HYPHEN [#273]

Recommended UTC actions

1. **No Action**: PAG recommends no action: the current behavior is reasonable, and note that changing the default semantics of a common format control character after twenty-five years would be disruptive to implementers; contra L2/19-041, major implementations follow the default semantics for this character.

Replacement for UTC Action Item

B.11.5.1.3 Linebreak property value of U+00AD SOFT HYPHEN [Muller, L2/19-041]

[158-A50] Action Item for Andy Heninger: Follow up with Eric Muller about document L2/19-041.

Background information / discussion

2024-apr: We still have

```
Unset
00AD          ; BA # Cf       SOFT HYPHEN
```

The long-standing linebreak property of a widely-used character created precisely for interaction with line breaking should be out of scope, as it would be sure to be highly disruptive to implementers (in particular, the specifics of the special-casing required of implementers could change).

[long-standing: 25 years: this character has had lb=BA since the first version of UAX #14.]

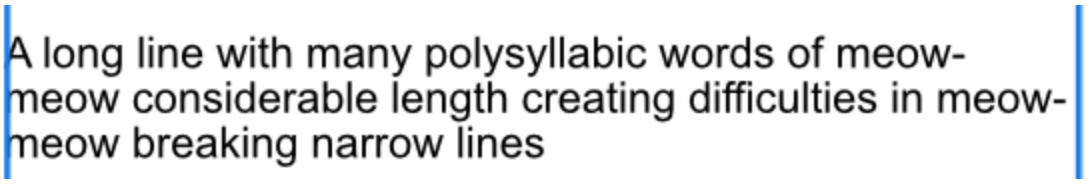
The document states

> Many layout engines that determine their linebreak opportunities using a stock implementation of UAX#14 either tailor it or post-process the result to ignore this linebreak opportunity. I have seen at least three independent implementations doing this, and I believe that Chromium does that too.

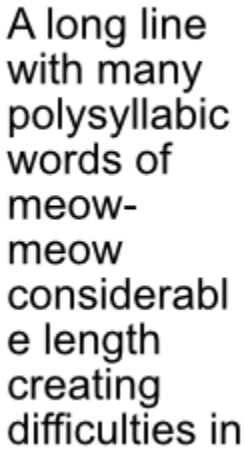
At least in Chrome, the use of U+00AD does create a line break opportunity, which is used, and which does result in the appearance of a hyphen glyph when it is used: the text consisting of fifty-eight occurrences of the word « meow », separated by U+00AD, displays as follows:

meowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeow-
meowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeow-
meowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeowmeow

There are plenty of other implementations that break at a SHY, even when "automatic hyphenation" is turned off. Asmus provided an example of a popular inexpensive publishing suite:



The SHY is used as a linebreak opportunity even when other words **only** receive emergency linebreaks.



5.2 Follow-up on UTC-172-A66 and UTC-172-A100 [#291]

Recommended UTC actions

1. **Consensus**: Approve the change of the Line_Break property of the Creative Commons symbols U+1F10D..U+1F10F, U+1F16D..U+1F16F, and U+1F1AD from Line_Break=Ideographic (ID) to Line_Break=Alphabetic (AL), and of the phonetic closing brackets U+2E56, U+2E58, U+2E5A, and U+2E5C from Line_Break=Close_Punctuation (CL) to Line_Break=Close_Parenthesis (CP). For Unicode 16.0". See L2/24-162 item 5.2.

These changes have been made for Unicode 16.0 before the start of the beta. No further action needed.

PAG input

From Robin Leroy, for Ken Whistler, PAG

Ken Whistler has the following action items:

[172-A66] Action Item for Ken Whistler, PAG: Consider Line_Break values for Creative Commons symbols; for a future version of the Unicode Standard. See L2/22-124 item UCD11.

[172-A100] Action Item for Ken Whistler, PAG: Investigate whether to change the Line_Break values for U+2E55..U+2E5C to match those for the ASCII square brackets; for a future version of the Unicode Standard. See L2/22-124 item Seg6.

These stem from public feedback
https://www.unicode.org/review/pri453/feedback.html#:~:text=Fri%20Jun%2024%2009:56:01%20CDT%20202

2 (Charlotte Buff) and https://www.unicode.org/review/pri446/feedback.html#:~:text=Fri%20Jun%203%2019:49:05%20CDT%202022 (David Corbett).

Ken considered the relevant part of the feedback from Charlotte Buff and investigated the feedback from David Corbett, and agreed that both of the suggested sets of changes make sense.

These changes have been made in 16.0β data and in Proposed Update UAX #14, so no actions need to be recorded to make the changes; however a decision must be recorded to approve them, per https://www.unicode.org/reports/tr44/#Allowed_Changes.

Background information / discussion

The feedback from Charlotte Buff was:

```
Unset
The following characters in the Enclosed Alphanumeric Supplement, Alchemical
Symbols, Geometric Shapes Extended, and Supplemental Arrows-C blocks
currently are Line_Break=Ideographic (ID):

       Intellectual property rights symbols:
              U+1F10D..U+1F10F    CIRCLED ZERO WITH SLASH..CIRCLED DOLLAR SIGN WITH OVERLAID
BACKSLASH
              U+1F16D..U+1F16F    CIRCLED CC..CIRCLED HUMAN FIGURE
              U+1F1AD                    MASK WORK SYMBOL

       Astronomical and astrological symbols:
              U+1F774..U+1F776    LOT OF FORTUNE..LUNAR ECLIPSE
              U+1F77B..U+1F77F    HAUMEA..ORCUS

       Go stone markers (compare ⚆, ⚇, ⚈, ⚉):
              U+1F7D5..U+1F7D8    CIRCLED TRIANGLE..NEGATIVE CIRCLED SQUARE

       Star symbol (compare ☆,  ,   etc.):
              U+1F7D9                    NINE POINTED WHITE STAR

       Arithmetic symbol dingbat (compare ➕, ➖, ✖, ➗):
              U+1F7F0                    HEAVY EQUALS SIGN

       Arrows for legacy computing:
              U+1F8B0..U+1F8B1    ARROW POINTING UPWARDS THEN NORTH WEST..ARROW POINTING
RIGHTWARDS THEN CURVING SOUTH WEST


A more appropriate line break value for them would be Alphabetic (AL) as a
matter of consistency, because all comparable characters are categorised as
Alphabetic (or Ambiguous in a few cases) as well. The Creative Commons
symbols in particular would benefit from this change because several of
them are often used in sequence.
```

```
In fact, it would be a good idea to likewise set the default line break
value for unassigned code points in these four blocks to Alphabetic since
the encoding of Ideographic characters in these ranges seems to be the
exception rather than the norm.
```

The feedback from David Corbett was:

```
Unset
L2/21-042 gives examples of U+2E55..U+2E5C within words, just like how
U+0029 is used in “(s)he”. It is central to these characters’ purpose to
appear within words, so it is likely that their line breaking works the
same as for U+0029. The closing characters U+2E56, U+2E58, U+2E5A, and
U+2E5C should therefore have Line_Break=Close_Parenthesis.
```

5.3 L2/24-143 changes to East_Asian_Width property [#300]

Recommended UTC actions

1. **Action Item** for Asmus Freytag, PAG: Review the suggested East_Asian_Width property value changes in L2/24-143. For Unicode 17.0. See L2/24-162 item 5.3.

The editorial change has been made already.

Document

L2/24-143 by Jules Bertholet
Proposal to change the East_Asian_Width property of various characters, and the derivation of Line_Break=Ambiguous

Background information / discussion

We are recommending the following editorial change for clarification on this question; there is no hard-and-fast derivation between East_Asian_Width and Line_Break. This change has been made.

> Make a two-word change to UAX #11 ED7, changing "By extension, they also do not occur in East Asian typography" to "By extension, they also do not tend to occur in East Asian typography".

5.4 Line_Break of U+19DA NEW TAI LUE THAM DIGIT ONE [#302]

Recommended UTC actions

1. **Consensus**: Change the Line_Break assignment of U+19DA NEW TAI LUE THAM DIGIT ONE from Line_Break=Complex Context to Line_Break=Numeric. For Unicode 16.0. See L2/24-162 item 5.4.
2. **Action Item** for Robin Leroy, PAG: Change the Line_Break assignment of U+19DA NEW TAI LUE THAM DIGIT ONE from Line_Break=Complex Context to Line_Break=Numeric in LineBreak.txt. For Unicode 16.0. See L2/24-162 item 5.4.

Feedback (verbatim)

PRI-502
Date/Time: Wed Jun 19 09:19:47 CDT 2024
ReportID: ID20240619091947
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 502

Currently, U+19DA NEW TAI LUE THAM DIGIT ONE has Line_Break=Complex_Context while all the other digit characters of the New Tai Lue script (U+19D0..U+19D9) have Line_Break=Numeric. For consistency, I propose changing U+19DA to Line_Break=Numeric as well.

5.5 UAX #14 - AI Line_Break class description omissions [#303]

Recommended UTC actions

1. **Consensus**: Change the Line_Break property of U+2150..U+2153, U+2156..U+215A, and U+215C..U+215D (VULGAR FRACTION ONE SEVENTH, ONE NINTH, ONE TENTH, ONE THIRD, TWO FIFTHS, THREE FIFTHS, FOUR FIFTHS, ONE SIXTH, FIVE SIXTHS, THREE EIGHTHS, and FIVE EIGHTHS) to lb=AI. For Unicode 16.0. See L2/24-162 item 5.5.
2. **Action Item** for Robin Leroy, PAG: change the Line_Break property of U+2150..U+2153, U+2156..U+215A, and U+215C..U+215D (VULGAR FRACTION ONE SEVENTH, ONE NINTH, ONE TENTH, ONE THIRD, TWO FIFTHS, THREE FIFTHS, FOUR FIFTHS, ONE SIXTH, FIVE SIXTHS, THREE EIGHTHS, and FIVE EIGHTHS) to lb=AI. For Unicode 16.0. See L2/24-162 item 5.5.
3. **Action Item** for Robin Leroy, PAG: In UAX #14, update the description of Line_Break class AI to mention that all vulgar fractions have this value. For Unicode 16.0. See L2/24-162 item 5.5.

Feedback

PRI-490
Date/Time: Sun Jun 09 06:54:59 CDT 2024
ReportID: ID20240609065459
Name: Jules Bertholet
Report Type: Public Review Issue
Opt Subject: 494

The description of the AI Line_Break class claims:

> As updated, the AI line breaking class includes all characters with East Asian Width A that are outside the range U+0000..U+1FFF, plus the following characters:
>
> 24EA CIRCLED DIGIT ZERO
> 2780..2793 DINGBAT CIRCLED SANS-SERIF DIGIT ONE..DINGBAT NEGATIVE CIRCLED SANS-SERIF NUMBER TEN

However, the class actually includes several more characters:
https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5Cp%7BLine_Break%3DAI%7D-%5Cp%7BEast_Asian_Width%3DAmbiguous%7D

There is no clear rhyme or reason to these. Notably, U+2155 VULGAR FRACTION ONE FIFTH is included, but many other fractions are not, and U+2574 BOX DRAWINGS LIGHT LEFT is included, but other light box-drawing characters are not.

Background information / discussion

The group investigated this issue and found that the vulgar fractions indeed all behave language-dependently in line breaking in some common word processing software, so that in order to align with industry practice they should be lb=AI.

Historical note

The current lb value of U+2155 (which is lb=AI as it should be, despite not being ea=A) is a consequence of an interaction between a change to East_Asian_Width and the decoupling of lb=AI from ea=A.

A look at https://util.unicode.org/UnicodeJsps/character.jsp?a=2155&history=assigned shows that it used to have East_Asian_Width=Ambiguous in Unicode Version 3.1 (thus got lb=AI), but that this was changed in Unicode Version 3.2.

The relevant decision is UTC-90-C26:

> [90-C26] Consensus: Update Unicode Standard Annex #11 East Asian Width based on changes proposed in document L2/02-078. Post as a final UAX as part of Unicode 3.2.

The document in question reads:

> (a) Vulgar Fraction
> 2155;A # VULGAR FRACTION ONE FIFTH
> I could not find this in any of the sets we used to generate the EA width assignment. This looks like a mistake in generating our tables. It's at the end of a range, so it might be a 'one too much' error.
> Suggested action: A → N

The decoupling of lb=AI from ea=A was decided by UTC-97-C21:

> [97-C21] Consensus: Decouple East Asian Width properties from from Linebreak properites. [L2/03-419]

5.6 Update LB10 and LB21a for implementability [#307]

Recommended UTC actions

1. **Consensus**: Approve the changes to LB10 (to treat all properties of lb=CM characters unaffected by LB9 as those of U+0041 A, rather than just Line_Break) and to LB21a (to exclude characters that have both Line_Break=Break_After and East_Asian_Width Fullwidth, Halfwidth, or Wide from participating in the rule) described in L2/24-162 item 5.6. For Unicode Version 16.0.
2. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, change rules LB10 and LB21a as described in L2/24-162 item 5.6. For Unicode Version 16.0.
3. **Action Item** for Robin Leroy, PAG: In LineBreakTest.txt, change rules LB10 and LB21a as described in L2/24-162 item 5.6. For Unicode Version 16.0.

PAG input

From Robin Leroy, PAG: while integrating Unicode 16β line breaking changes, ICU ran into some complex implementation issues.
These stem from interactions between LB19 and both LB10 and LB21a, as their contexts end up overlapping in ways that are difficult to handle in the ICU state machine builder language.

These difficulties are easily fixed by small changes to UAX #14 in a way that:

- for LB10:
  - has no effect except in degenerate cases, in which it only retains pre-16 behaviour anyway,
  - simplifies a broad range of implementation strategies;
- for LB21a, improves the behaviour of the algorithm in a corner case (fixing what is effectively a bug).

Since, without this change, it is likely that ICU would be unable to produce an untailored implementation of Unicode 16.0 line breaking in ICU 76 (whereas convergence had been a major goal of the changes made in Unicode 16.0 and in ICU 76), and since the change seems like an improvement on its own merits, I propose making the change now, even though we are quite late in the process.

Proposed change to LB10

The proposed change to LB10 is to say that all properties, not just Line_Break, are those of U+0041 A for a CM that was unaffected by LB9:

> Treat any remaining CM or ZWJ as if it were had the properties of U+0041 A LATIN CAPITAL LETTER A, that is, Line_Break=AL, General_Category=Lu, East_Asian_Width=Na, Extended_Pictographic=N.

The only change in behaviour resulting from that change involves an $EastAsian CM that is at the beginning of the line, or follows a space or zero-width space, and precedes a quotation mark itself followed by an East Asian character. In that case, with the above change, the behaviour is the same as in Unicode 15.1 and earlier, whereas with the 16.0β text, an additional break may be permitted on one side of the quotation mark. This is an obscure, rare, and arguably degenerate case, and we would not be degrading the behaviour compared to Unicode 15.1 anyway. Note that it has long been obsolete for a combining mark to be in such a place, see Sections 9.2 of UAX #14 and 7.2 of The Unicode Standard.

This greatly simplifies implementations; besides ICU's state machine, which would otherwise need rules with very long distance context to deal with the CM properly in LB19a, it also helps with other kinds of implementations:

- For a state-machine based implementation, this means that only one CM state is needed, instead of CM&EastAsian and CM-EastAsian which decay to AL&EastAsian and AL-EastAsian. Since some state machine implementations (notably, ICU4X) hardcode the handling of the CM state, avoiding splitting that state is beneficial.
- An implementation that implements LB10 by actually changing the character in some buffer or variable also does not need to adjust its logic to pick a character that lies in lb=AL but retains the right East_Asian_Width. This approach to implementing LB10 was the one I used in the demo I had written in late 2022 for PRI-472. It is also used by the « old monkey » reference implementation originally written 21 years ago by Andy Heninger and which ICU uses to test its production implementation. It is therefore likely the case of many naïve implementations of UAX #14.
  In the ICU old monkeys, the change necessary for 16.0β (unneeded with this proposal) was from

```
Unset
    if (fCM->contains(*posChar)) {
      *posChar = u'A';
    }
```

- to

```
Unset
    if (fCM->contains(*posChar)) {
      switch (u_getIntPropertyValue(*posChar, UCHAR_EAST_ASIAN_WIDTH)) {
      case U_EA_WIDE:
          *posChar = u'♈';
          break;
      case U_EA_NEUTRAL:
          *posChar = u'ᴬ';
          break;
      case U_EA_AMBIGUOUS:
          *posChar = u'Ⓐ';
          break;
      default:
          puts("Unexpected ea value for lb=CM");
          std::terminate();
      }
  }
```

- Nothing intractable, but not completely trivial, especially if you don't know that the introduction of LB19 suddenly requires it.

Proposed change to LB21a

The proposed change to LB21a is to exclude EastAsian (that is [\p{ea=F}\p{ea=W}\p{ea=H}]) from BA in that rule:

> HL (HY | [BA - $EastAsian]) × [^HL]

The only BA & $EastAsian character is U+3000 IDEOGRAPHIC SPACE.
The goal of this rule is to prevent a break after the hyphen in Hebrew + Hyphen + non-Hebrew—because a hyphen is used to separate the prefix -ו (and) from a word written in another script—; BA is used to catch HYPHEN as well as HEBREW PUNCTUATION MAQAF, but IDEOGRAPHIC SPACE is clearly out of scope for the underlying intent[1].
The effect of the proposed change is to allow breaks after an ideographic space separating Hebrew from non-Hebrew, which seems like something that would actually happen, and where a break would be expected.

While it would certainly make sense in the future to exclude more lb=BA characters from LB21a, such as all the spaces (or to define the hyphens by inclusion like in LB20a), we propose this minimal change for 16.0 to avoid further unexpected interactions. Note that because lb=BA and $EastAsian are used separately in the 16.0 rules, implementations that compute a single partition of the code space need to have a `BA & $EastAsian` class already to implement 16.0, so the change does not require them to refine their partition any further. A PAG issue has been opened to track future work on LB21a.

Background information / discussion

The UAX #14 changes approved by UTC-179 had been prototyped in ICU, and extensively tested using the « old monkey tests », which throw random strings at both a naïve UAX #14-like implementation, and the efficient state machine shipped by ICU.

The changes had been tested both individually and together on millions of strings. However, this failed to catch the issues, for two reasons:

1. The strings were poorly distributed; the line breaking class was chosen uniformly at random, and the character chosen uniformly at random within that class; the East_Asian_Width was not taken into account. ea=W is rare in \p{lb=CM} (4,5‰) and \p{lb=BA} (3,7‰), so comparatively few test cases involving these combinations of properties were generated.
2. The random number generator was a 32-bit linear congruential generator; on 500-character strings, each character being picked with two random numbers, the generator cycled over a couple million strings, rendering further testing meaningless.

Both of these issues have been addressed: the old monkey tests now:

1. construct a maximally coarse partition of the code space such that no rule references a set that refines the partition (in particular, the singleton [\N{DOTTED CIRCLE}] is one of the sets in the partition), and from that partition uniformly at random, rather than from the Line_Break partition;
2. use the ranlux48 PRNG, whose cycle length is approximately 2^{576}.

After these changes to the old monkeys, and with rules LB10 and LB21a of the old monkeys amended as proposed here, the ICU implementation was successfully tested with upward of 624 million 500-character strings.

ICU complexity

The motivation for filing this issue was intractable complexity in the ICU rules. See various places these had gone trying to strictly implement 16.0β:

- https://github.com/unicode-org/icu/blob/7e27e41a887dba7bdc2c9f413897dfc6345a62a2/icu4c/source/data/brkitr/rules/line.txt#L277-L332 (Did not quite work, unclear whether it could have been made to work. The rules were long enough and hard enough to derive that spotting errors took millions of monkey tests.)
- a different approach https://github.com/unicode-org/icu/blob/f63de5e633d1d7a44a56eee75967b7fba2245d63/icu4c/source/data/brkitr/rules/line.txt#L283-L338 (Conceptually simpler, this specific set of rules was known not to work.)
- https://github.com/eggrobin/icu/blob/94ab34d42a33d30920d9197f2cd9166758dd3d1c/icu4c/source/data/brkitr/rules/line.txt#L297-L351 (Continuation of the one above, should in principle have been workable eventually, but as the last few rules were added took half an hour to compile to an automaton and then just crashed the rule compiler).

Compare the rules for LB19 and LB19a with the proposed changes: https://github.com/unicode-org/icu/blob/ec3e6f63c82ebcf55d23cf332b105ca78d68f249/icu4c/source/data/brkitr/rules/line.txt#L277-L286.

5.7 lb=ID: no more unassigned in CJK Unified Ideographs & CJK Unified Ideographs Extension A [#312]

Recommended UTC actions

1. **Action Item** for Markus Scherer, PAG: In UAX #14 documentation of lb=ID, replace the complete list of blocks and ranges that default to that value with one example and a reference to DerivedLineBreak.txt. For Unicode 16.0. See L2/24-162 item 5.7.

Feedback (verbatim)

Date/Time: Mon Jul 01 15:13:58 CDT 2024
ReportID: ID20240701151358
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 490

The description of the line break class Ideographic states that unassigned code points in the CJK Unified Ideographs and CJK Unified Ideographs Extension A blocks default to lb=ID. However, said blocks no longer contain any unassigned code points.

Background information / discussion

https://www.unicode.org/reports/tr14/proposed.html#ID

6. Segmentation

6.1 Word boundaries and line breaks [#280]

Recommended UTC actions

1. **No Action:** The proposed changes have been incorporated.

Document

L2/24-043 by Norbert Lindenberg in response to action item UTC-176-A101 “For UAX #29, propose improved wording for the last paragraph of the introduction of Section 4; for Unicode 16.0. See document L2/23-160 item 4.12.”

Background information / discussion

The proposed text changes have been incorporated into the proposed update of UAX #29 during the early part of the Unicode 16.0 beta.

6.2 grapheme cluster boundaries vs. canonical equivalence: Kannada vowel signs etc. [#287]

Recommended UTC actions

1. **Consensus**: Assign the Other_Grapheme_Extend property to U+0CC0, U+0CC7, U+0CC8, and U+0CCA, U+0CCB (Kannada vowel signs II, EE, AI, O, and OO), as well as U+1B3B, U+1B3D, and U+1B43 (Balinese vowel signs ra repa tedung, la lenga tedung, and pepet tedung), thereby changing their Grapheme_Cluster_Break property from SpacingMark to Extend, ensuring consistency of legacy grapheme clusters with canonical equivalence. For Unicode 16.0. See L2/24-162 item 6.2, PRI-494#ID20240422114157, and PRI-494#ID20240422120150.
2. **Action Item** for Robin Leroy, PAG: Assign the Other_Grapheme_Extend property to U+0CC0, U+0CC7, and U+0CC8, and U+0CCA, U+0CCB (Kannada vowel signs II, EE, AI, O, and OO), as well as U+1B3B, U+1B3D, and U+1B43 (Balinese vowel signs ra repa tedung, la lenga tedung, and pepet tedung), thereby changing their Grapheme_Cluster_Break property from SpacingMark to Extend. For Unicode 16.0. See L2/24-162 item 6.2.

Feedback (verbatim)

PRI-494

Date/Time: Mon Apr 22 11:41:57 CDT 2024
ReportID: ID20240422114157
Name: Jules Bertholet
Report Type: Error Report
Opt Subject: PropList.txt

UAX 29 (http://unicode.org/reports/tr29/) says the following:

> The default rules have been written so that they can be applied directly to non-NFD text and yield equivalent results [versus applying to NFD text].

In support of this aim, it later says the following about legacy grapheme clusters:

> The continuing characters include nonspacing marks, the Join_Controls (U+200C ZERO WIDTH NON-JOINER and U+200D ZERO WIDTH JOINER) used in Indic languages, and a few spacing combining marks to ensure canonical equivalence.

However, this property (that grapheme cluster boundaries are closed under canonical equivalence) currently does not hold. U+0CC0 KANNADA VOWEL SIGN II has `Grapheme_Cluster_Break=SpacingMark`, but it NFD decomposes to two characters (U+0CBF KANNADA VOWEL SIGN I and U+0CD5 KANNADA LENGTH MARK) which both have `Grapheme_Cluster_Break=Extend`. To correct this error, U+0CC0 should be given the property `Other_Grapheme_Extend` in `PropList.txt`.

Date/Time: Mon Apr 22 12:01:50 CDT 2024
ReportID: ID20240422120150
Name: Jules Bertholet
Report Type: Error Report
Opt Subject: PropList.txt

Amending my previous report

A few moments ago, I submitted an error report about the `Grapheme_Cluster_Break` property of U+0CC0. I would like to amend this report to note the following other characters which are also affected:

- U+0CC7
- U+0CC8
- U+0CCA
- U+0CCB
- U+1B3B
- U+1B3D
- U+1B43

Background information / discussion

It was noted in discussion that a similar issue with Kannada vowel signs occurred regarding consistency of the UBA with canonical equivalence; this was resolved by decision UTC-93-C25.

Regarding the issue at hand in grapheme cluster segmentation, an invariant test was added, which found exactly the characters reported across these two feedback items, namely:

```
Unset
# Canonical decomposition preserves the initial GCB, except for LV and LVT.

In [\p{dt=canonical}-[\p{gcb=LV}\p{gcb=LVT}]], gcb * (take 1) * dm = gcb

**** START Test Failure 1 ****
### Got unexpected property values: 7

0CC0   ; Extend≠SpacingMark       #            (ೀ)   KANNADA VOWEL SIGN II
0CC7..0CC8    ; Extend≠SpacingMark       #      [2]    (ೇ..ೈ)      KANNADA VOWEL SIGN
EE..KANNADA VOWEL SIGN AI
0CCA   ; Extend≠SpacingMark       #            (ೊ)   KANNADA VOWEL SIGN O
1B3B   ; Extend≠SpacingMark       #            ( )    BALINESE VOWEL SIGN RA REPA TEDUNG
1B3D   ; Extend≠SpacingMark       #            ( )    BALINESE VOWEL SIGN LA LENGA TEDUNG
1B43   ; Extend≠SpacingMark       #            ( )    BALINESE VOWEL SIGN PEPET TEDUNG
```

6.3 more Sentence_Break changes after AI 179-A113 [#292]

Recommended UTC actions

1. **Consensus**: Update Table 4, Sentence_Break Property Values of UAX #29, categorizing Semicolons (U+003B, U+FE14, U+FE54, and U+FF1B) and Greek Question Mark (U+037E) as SContinue, Coptic punctuation (U+2CF9..U+2CFB) as STerm, and Vertical Forms punctuation (U+FE10..U+FE19) in the same categories as their compatibility equivalents, and make corresponding changes in SentenceBreakProperty.txt. For Unicode Version 16.0. See L2/24-162 item 6.3. This decision supersedes consensus UTC-179-C33.

PAG recommends no further action; the UCD and UAX #29 have been updated to reflect the modified proposal.

Feedback (verbatim)

Date/Time: Thu May 09 10:36:08 CDT 2024
ReportID: ID20240509103608
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: 494

Action item 179-A113 says to categorize semicolons as Sentence_Break = SContinue. https://github.com/unicode-org/unicodetools/pull/812 modifies U+1364 ETHIOPIC SEMICOLON, U+A6F6 BAMUM SEMICOLON, and U+1DA89 SIGNWRITING SEMICOLON accordingly. Those three scripts also have commas and colons, which still have Sentence_Break = Other. If those scripts' semicolons are recategorized to match ASCII, so should their commas and colons; if there is not yet any evidence supporting changing their commas and colons, there probably isn't any for their semicolons either, so their semicolons should not be recategorized.

Lisu, Medefaidrin, Mongolian, Newa, and Vai don't have semicolons, but they do have commas or colons. Should those be recategorized too? I wouldn't assume so just from their character names, but maybe.

6.4 East Asian Auto Spacing [#306]

Recommended UTC actions

1. **Action Item** for Koji Ishii, Markus Scherer, PAG: Prepare a working draft UTR for East Asian Auto Spacing based on L2/24-057R in collaboration with PAG. See L2/24-162 item 6.4.

Document

L2/24-057R EAST ASIAN AUTO SPACING (Proposal) by Koji Ishii, Yasuo Kida, Fuqiao Xue (2024-jul-01) supersedes L2/24-057, L2/23-283

From the doc intro:

East Asian text usually consists of multiple scripts [...]. East Asian established typography conventions define that a thin spacing between East Asian scripts and other scripts improves the readability. [...]

While detailed rules of the spacing can vary across documents, it is important that the choice made by an author for a specific document be clearly established, so that a rendering system can display what the author intended. It is also important that this choice be established independently of the font resources, as the rendering systems may have to use other fonts than those intended or specified in the document. [...]

This report describes a Unicode character property which can serve as a stable default rule of inserting the spacing for the purpose of reliable document interchange.

Background information / discussion

We noticed that this is already being done outside the web platform (MS Word seems to perform auto-spacing without inserting space characters), therefore seems appropriate for the UTC.

The doc suggests using ZWSP if you don't want any space; but it adds a line breaking opportunity which may not be desirable. Authors should take this into consideration and review boundary conditions here vs. UAX #14.

7. IDNA

7.1 UTS #46 disallowed_STD3_valid and disallowed_STD3_mapped [#282]

Recommended UTC actions

1. **Consensus**: In UTS #46, handle UseSTD3ASCIIRules only in the Validity Criteria. In the IDNA Mapping Table, replace Status disallowed_STD3_valid with valid and disallowed_STD3_mapped with mapped. Note edge cases of labels that are no longer unnecessarily disallowed. For Unicode 16.0. See L2/24-162 item 7.1.
2. **Action Item** for Markus Scherer, PAG: In UTS #46, handle UseSTD3ASCIIRules only in the Validity Criteria. In the IDNA Mapping Table, replace Status disallowed_STD3_valid with valid and disallowed_STD3_mapped with mapped. Note edge cases of labels that are no longer unnecessarily disallowed. For Unicode 16.0. See L2/24-162 item 7.1.

Feedback (verbatim)

Date/Time: Tue Apr 02 10:47:44 CDT 2024
ReportID: ID20240402104744
Name: Henri Sivonen
Report Type: Error Report
Opt Subject: UTS 46

When implementing UTS 46, the most time-consuming wrong path was trying to design data structures for UTS 46 data assuming that the data needs to have distinct data entries for disallowed_STD3_valid and disallowed_STD3_mapped before discovering that these can be handled as valid and mapped with an ASCII deny list applied afterwards.

I suggest refactoring the spec so that:

1) disallowed_STD3_valid and disallowed_STD3_mapped become simply valid and mapped in the data and the spec says when to apply an ASCII deny list 2) instead of a boolean UseSTD3ASCIIRules the algorithm would take an ASCII deny list.

UTS 46 itself could define an STD3 ASCII deny list and the WHATWG URL Standard could use forbidden domain code point https://url.spec.whatwg.org/#forbidden-domain-code-point as an ASCII deny list parameter to UTS 46.

It would probably appropriate to make informative remarks that a) putting ASCII letters, digits, or hyphen on the deny list would break things and b) in the validation phase, the ASCII period can be put on the deny list to handle that validity constraint as part of the ASCII deny list check.

Background information / discussion

We have carefully compared the existing (Unicode 15.1) UTS #46 algorithm with the suggested approach of checking for valid ASCII characters after the Map and Normalize steps.

We found subtle behavior differences, but we have concluded that they are unnecessary, and that the simpler post-processing check is

1. more consistent with Unicode 15.1 changes of disallowed_STD3_valid characters (removing the only three non-ASCII characters)
2. consistent with how UTS #46 has documented using a custom set of valid ASCII characters
3. avoids unnecessarily disallowing certain labels that contain disallowed_STD3_mapped characters but which do not contain non-LDH ASCII characters when the mappings are applied
4. easier to implement (simpler data format & data lookup)
5. matches the implementation of a widely used UTS #46 implementation (in ICU)

Therefore, we agree with Henri's suggested algorithm simplification, except as an actual algorithm change with observable changes in behavior, rather than documenting it as equivalent to the current algorithm.

These proposed changes are marked with cyan background in the proposed update of UTS #46 for Unicode 16.0: https://www.unicode.org/reports/tr46/tr46-32.html

The proposed changes in behavior will affect users of UseSTD3ASCIIRules=true. (Assuming a conformant implementation.) See details below.

The behavior does not change for UseSTD3ASCIIRules=false. This includes implementations that use a custom set of valid ASCII characters according to section 4.1.1 UseSTD3ASCIIRules.

Before Unicode 15.1, U+2260 (≠), U+226E (≮), and U+226F (≯) were disallowed_STD3_valid.

Note that when UseSTD3ASCIIRules=true *the data lookup (UTS #46 section 5) resolves* disallowed_STD3_valid and disallowed_STD3_mapped to disallowed, *ignoring the mappings in the latter case*.

Examples for UseSTD3ASCIIRules=true behavior changes:

- Example for a label which continues to fail the Validity Criteria despite the change in Processing:
  In Unicode 15.1, input label "⑷" was unchanged in Processing and failed the Validity Criteria. (U+2477 disallowed_STD3_mapped was resolved to disallowed, and its mapping was not applied.)
  With the proposed changes, "⑷" would be Mapped to "(4)", which would still fail the Validity Criteria, except if a custom set of valid ASCII characters is used that includes the parentheses.
- Example for a label which newly passes the Validity Criteria due to the change in Processing:
  In Unicode 15.1, input label "\uFF1D\u0338" (fullwidth equals + combining solidus overlay) was unchanged in Processing and failed the Validity Criteria. (U+FF1D disallowed_STD3_mapped was resolved to disallowed, and its mapping was not applied.)
  With the proposed changes, "\uFF1D\u0338" would be Mapped to "\u003D\u0338" and Normalized to "\u2260" (not equal to), which is valid.

7.2 UTS #46: clarify whether it is an error for Punycode decoding to not yield any non-ASCII output [#283]

Recommended UTC actions

No Action: PAG recommends no new action because this is covered by existing action item UTC-165-A48 which has been done for Unicode 16.0.

Feedback (verbatim)

Date/Time: Tue Apr 02 10:50:47 CDT 2024
ReportID: ID20240402105047
Name: Henri Sivonen
Report Type: Error Report
Opt Subject: UTS 46

It seems to me that in practice it should be considered an error for Punycode decoding not to yield any non-ASCII output (both Firefox and Safari treat this as an error). However, I don't see any spec text to that effect either in UTS 46 itself or in RFC 3492. I suggest adding an item under "Processing" step 4 'If the label starts with "xn--":' between current items 1 and 2: "If the label ends with U+002D HYPHEN-MINUS, record that there was an error, and continue with the next label."

This would catch both the case where the hyphen is the last hyphen of "xn--" and Punycode decoding would have no output at all and the case where there are no Punycode digits after the delimiter, which means not producing any non-ASCII output.

Notably in Firefox and Safari, https://xn--unicode-.org/ is in error and not equivalent to https://unicode.org/ and https://unicode.org.xn--/ is in error and not equivalent to https://unicode.org./ .

Background information / discussion

This is covered by UTC-165-A48 "Update UTS #46 to validate ACE label edge cases, see L2/20-240 item F7. For Unicode 14. (Retargeted for 15.1.)" which has been done for Unicode 16.0. See https://www.unicode.org/reports/tr46/tr46-32.html#ProcessingStepPunycode

1. In fact, when this rule was added in Unicode 6.1 (see UTC-125-A99, L2/11-141R), U+3000 was *not* lb=BA, but lb=ID; it was changed to lb=BA in Unicode 6.3 (UTC-132-C29). ↩