L2/25-228

UTC #185 properties feedback & recommendations

Markus Scherer & Josh Hadley / Unicode properties & algorithms group, 2025-oct-22

Participants

The following people have contributed to this document:

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

1. Core Spec

1.1 Write a proposal about character deprecation criteria [#274]

Recommended UTC actions

1. **No Action**: PAG recommends no action. Applying the Deprecated property should be done on a case-by-case basis, and there are newer properties that provide more information. See the background section below.

Replacement for UTC Action Item

B.14.7 Feedback on Properties [Pournader, L2/19-054]

[158-A100 Action Item for Roozbeh Pournader: Write a proposal about deprecation criteria. Also consider feedback from Charlotte Buff Oct 15 06:30:35 CDT 2018 in L2/19-010.

The original feedback was in document L2/18-301

Background information / discussion

This feedback has lingered a bit and some things have changed elsewhere in the standard since then.

There are two issues raised.

- The deprecated property isn't sufficient to cover all cases of characters that should not be used.
- The deprecated property is otherwise not consistently applied.

To better cover the former we now have the Do_Not_Emit data. With that, the set of deprecated character splits in two categories:

1. those for which there is a systematic preferred substitution (Do_Not_Emit_Type=Deprecated),
2. those for which there isn't, either because the preferred one is context-dependent (U+2329 and U+232A), or because they are not actually representable as characters (often describing things that can be achieved either by higher-level protocols or by changing the text).

\p{Deprecated} grouped by do_not_emit_type

The second set is clearly something that only the Deprecated property can handle. The first set is only a subset of what should not be emitted and does not appear to follow any easily discoverable principles.

For example, deprecation does not correlate with the script property. While it might seem to be more justified to deprecate a character if it is in Latin like U+0149 than if it is in Sharada, as the former is more likely to cause widespread issues, a look at \p{do_not_emit_type=/./} code points grouped by Deprecated shows that U+0140 is Latin and not deprecated.

Deprecation is seen as a blunt edged sword of last resort. It was originally conceived of as a way to handle what essentially are defects in the encoding while also maintaining an absolute stability of characters, once encoded. Over time, it became clear that there are a wider range of characters for which there exist strongly preferred alternates and these are now more systematically covered with the Do_Not_Emit property.

Rather than systematizing the application of the Deprecated property, it is felt useful to explicitly treat it as a case-by-case judgment call, while relying on other properties to carry the more general information on non-preferred characters and their preferred alternates.

2. UCD

2.1 Request to change Upper case of ß (00DF) to ẞ (1E9E) [#469]

Recommended UTC actions

1. **Action Item** for Markus Scherer, PAG: In SpecialCasing.txt, near the mapping for U+00DF sharp s, add a brief note about details in the core spec, and link to the core spec 5.18.2 “German sharp s” section. For Unicode 18.0. See L2/25-228 item 2.1.
2. **Action Item** for Markus Scherer, PAG: In the core spec 5.18.2 “German sharp s” section, clarify that “default case” pairings and mappings refer to Unicode Standard case mappings in the absence of language-specific or otherwise requested tailorings. For Unicode 18.0. See L2/25-228 item 2.1.

Feedback (verbatim)

Date/Time: Tue Aug 26 05:31:17 PST 2025
ReportID: ID20250826053117
Name: Roger Moser
Report Type: Error Report
Opt Subject: Upper case of ß (00DF) is ẞ (1E9E)

There is the error in http://www.unicode.org/Public/UCD/latest/ucd/UnicodeData.txt:

The upper case of LATIN SMALL LETTER SHARP S (00DF) is LATIN CAPITAL LETTER SHARP S (1E9E).

But this information is missing. Therefore change

```
00DF;LATIN SMALL LETTER SHARP S;Ll;0;L;;;;;N;;;;;
```

to

```
00DF;LATIN SMALL LETTER SHARP S;Ll;0;L;;;;;N;;;1E9E;;1E9E
```

Document

L2/25-223 “Proposal to Update the Uppercase Mapping of LATIN SMALL LETTER SHARP S (U+00DF) to LATIN CAPITAL LETTER SHARP S (U+1E9E)” by Luca Rodenhäuser

This document also requests changing the Uppercase_Mapping of small sharp s, and describes why using the capital letter would be beneficial.

Background information / discussion

Several Unicode groups have been discussing how to handle the updated recommendation by the Rat für deutsche Rechtschreibung.

We have observed that while the recommendation by the German orthography council has changed, and there is some use of the capital sharp s character, the predominant uppercase form continues to be "SS", and even lowercase ß continues to be fairly frequently used in all-uppercase text.

The Case Pair Stability policy forbids changing the Unicode Standard’s Uppercase_Mapping of U+00DF as requested.

Core spec 5.18.2 “German sharp s” is an appropriate discussion of the issue. It has been changed in Unicode 17 from referring to “standard German orthography” to “customary German orthography”.

The Unicode CLDR (locale data) and ICU (libraries) projects use a wider variety of case mappings, but also follow customary usage. As long as customary usage uses "SS", these projects will continue to do so as well, at least by default (in the absence of other inputs).

The CLDR-TC and ICU-TC have been discussing parametric (runtime) options for users of our libraries to choose whether to uppercase to SS/ß/ẞ, but have not yet reached consensus on a good, acceptable, user-friendly way to do this in such low-level operations.

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.

- L2/25-159 document for [symbol] UAE Dirham sign [SEW #650]
  - Propertywise like the Saudi Riyal; in particular for the Line_Break property, lb=PR (Prefix_Numeric), rather than lb=PO (Postfix_Numeric) as suggested in L2/25-159 or lb=AL (Alphabetic) as suggested in L2/25-174. Currency symbols are always lb=PR or lb=PO.
  - Both lb=AL and lb=PO would have undesired effects for a prefix currency symbol, consider 通常価格:÷¥12,345 and 通常価格÷¥12,345 but 通常価格:×Y12,345 (lb=AL) and 通常価格×¢12,345 (lb=PO). For more on the effect of lb=PR and lb=PO see the comments on the properties of the Saudi Riyal, in L2/25-087 p. 16 [SEW #619]. A comparison of https://www.centralbank.ae/media/e4ebcgtb/the_guidelines_for_the_national_currency_symbol_uae_dirham_english.pdfslide 11 and https://www.sama.gov.sa/ar-sa/Currency/SRS/Documents/Guidelines.pdf slide 17 motivates the choice of lb=PR.
- WG2 N5330 document for Geometric shapes: Proposal to encode 17 geometric shapes [SEW #589]
  - The proposal seeks to add 16 new geometric characters—primarily found in mathematical and historical texts, especially in editions of Leibniz's writings—to the Unicode block 1F780 (Geometric Shapes Extended), with code points of 1F7DB and 1F7F1..1F7FF. The originally proposed 17th character, CIRCLE WITH DOUBLE VERTICAL LINE, was revised to be a variation sequence for the existing character 29B7. The properties of these new code points are propertywise consistent with others in the Geometric Shapes Extended block. These code points were previously reserved as Extended_Pictographic, but that property does not apply to these characters and has been removed as part of the implementation.
- L2/25-055 [SEW #657]
  - The PAG reviewed proposal L2/25-055 to add a kTangutNumeric property (which should now be named kTGT_Numeric). It recommends that the property be made Provisional rather than Informative as proposed, and that it have no effect on Numeric_Value derivation. The PAG otherwise has no objections to the proposed property.

4. Line Break

4.1 Chromium line breaking tailoring: alphanumeric HY ÷ numeric [#438]

Recommended UTC actions

1. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, Unicode Line Breaking Algorithm, consider mentioning the possibility of a (AL | HL | NU) HY ÷ NU tailoring for looser line breaking of URLs, mentioning pros and cons of such an approach. For Unicode 18.0. See L2/25-228 item 4.1.

PAG input

From Robin Leroy, PAG.

While working on aligning Chromium's linebreaking with UAX #14, I found that it has an interesting tailoring introduced in 2011 (addressing https://bugs.webkit.org/show_bug.cgi?id=20677) (see the current code) which allows for breaks after the hyphen-minus in [a-zA-Z0-9] - [0-9], with the rationale that these can occur in long URLs. These breaks are disallowed by UAX #14 rule LB25 HY × NU.

We should not change the UAX #14 default to incorporate this, as the rule HY × NU has been there since 1999, and it is not hard to find cases where a break in a sequence (alphanumeric, hyphen-minus, numeric) would be undesired; consider, e.g., F-5, MiG-28, ISO/IEC 10646-1, etc.

It might be useful to mention this possible tailoring in UAX #14 for looser line breaking, perhaps suggesting a less ASCII-specific way to express it (e.g., (AL | HL | NU) HY ÷ NU). (Cf. CSS line-break:loose, which is defined to allow breaks between ID and HH characters relevant in CJK typography—implemented as ID ÷ HH in the upcoming version of ICU.)

4.2 Request: Do not break numbers from letters, when divided with unresolved hyphen [#464]

Recommended UTC actions

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

Feedback (verbatim)

Date/Time: Sat Aug 16 06:44:46 PT 2025
ReportID: ID20250816064446
Name: Mikhail Merkuryev
Report Type: General Feedback
Subject: TR14: More discussion about my rule

Do not break numbers from letters, when divided with unresolved hyphen

AL HY × (NU IS PR)

(NU IS PO) HY × AL

In Russian/Ukrainian, there are lots of places with number+hyphen+text:

2-й = 2nd

ВАЗ-2101 = VAZ 2101, car model

Терминатор-2 = Terminator 2, film

9-этажка = 9-storey block

Actually was bugged with such browsers’ behaviour while editing some wikis.

Need more discussion, maybe just write as a note rather than strict rule.

And just a note (no strict rule): in Russian/Ukrainian some short particles are written with a hyphen, but morphological analysis is needed whether you can break: давай-ка (let’s, breaking is discouraged); да-с (yes milord, breaking is impossible).

4.3 TR14: multiple chars GL→WJ [#474]

Recommended UTC actions

1. **No Action**: PAG recommends no action. There is no contradiction in the algorithm; no real-life example of problematic line breaks were provided.

Feedback (verbatim)

Date/Time: Sun Aug 24 09:53:16 PT 2025
ReportID: ID20250824095316
Name: Mikhail Merkuryev
Report Type: General Feedback
Opt Subject: TR14: multiple chars GL→WJ

WJ is for places where Unicode usually breaks, and we should absolutely disable this feature.
GL is for places where Unicode should not break after, but in some soft circumstances it can break before.

This division into WJ and GL contradicts with another rule: do not break before marks.

So I suggest moving Mark/any (M*) + Non-tailorable/glue (GL) → WJ. Do not break before, do not break after.

Examples:

035C combining double breve below
FE20 combining ligature left half
16FE4 Khitan small script filler

Background information / discussion

GL behaves differently from WJ in that it allows a break before it if it follows a space, a hyphen, or a character with lb=BA; the break opportunities before GL can be tailored. See https://www.unicode.org/reports/tr14/#LB12a.
One of the main reasons for that is the soft hyphen + non-breaking hyphen construct used in Polish and Portuguese, as described in https://www.unicode.org/reports/tr14/#Hyphen.

Some characters with General_Category=Mark have Line_Break=GL. These include double diacritics and left half diacritics, as well as some more unusual conjoiners, e.g., the Brahmi number joiner.

The submitter is alluding to infelicitous break opportunities such as the one after the hyphen-minus in —͜a (em dash, combining double breve below, latin small letter a). There is no contradiction *per se*: rule LB9 does not apply because the combining double breve below is not lb=CM, rule LB12a does not apply because the em dash is lb=BA, and so the algorithm allows this break. While the break is certainly awkward, no evidence has been presented that such constructs are actually used with any regularity, and so no change is warranted. Note that if one needs to prevent the break in an extraordinary occurrence of such a construct, WJ can be used after the em dash.

In any case, it would be inappropriate to use class lb=WJ for this; the classes for the overriding controls lb=WJ and lb=ZW should remain singletons; these are special characters overriding the algorithm, whose behaviour cannot be tailored in any way, and which are likely to be specially handled in implementations. Entangling them with ordinary characters, whose behaviour gets refined over time, would be problematic for the maintainability of the standard and of its implementations.

4.4 Incorrect rule LB15b description [#475]

Recommended UTC actions

1. **No Action**: PAG recommends no action; the rule works as intended and as described in UAX #14.

Feedback (verbatim)

Date/Time: Tue Sep 16 15:49:04 PT 2025
ReportID: ID20250916154904
Name: Nikolay Sivov
Report Type: Error Report
Opt Subject: Incorrect rule LB15b description

Current description at https://www.unicode.org/reports/tr14/tr14-55.html#LB15b says:

Do not break before an unresolved final punctuation that lies at the end of the line, before a space, before a prohibited break,
or before an unresolved quotation mark, even after spaces.

The part "even after spaces" should be removed I believe. It's not reflected in the rule grammar and looks like it was copied from
LB15a description.

Background information / discussion

This is incorrect; we are before LB18, so any « don’t break before » rule means « even after spaces ».

4.5 LineBreak: inconsistent handling of future emojis [#476]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The algorithm is working as intended. There is more to unassigned code points than the possibility of being an emoji affected by skin colour, and different predictions about unassigned code points affect different rules.

Feedback (verbatim, except list of line numbers truncated)

Date/Time: Mon Sep 29 04:14:29 PT 2025
ReportID: ID20250929041429
Name: Bruno Haible
Report Type: Report Error in Publication/Data
Opt Subject: LineBreak: inconsistent handling of future emojis

UAX #14 has a rule (LB30b) that treats unassigned characters in the "future" emoji base range `[\p{Extended_Pictographic}&\p{Cn}]` like emoji base characters (EB):
https://www.unicode.org/reports/tr14/#LB30b
This makes perfect sense.

The Extended_Pictographic property is specified in the file ArchiveVersions/17.0.0/ucd/emoji/emoji-data.txt. It includes some characters listed as "". Among these, there are:

- U+1FFFD
- U+1F8FF

(There are more of them, but I'm picking these two for the sake of the discussion.)
This is perfectly consistent with ArchiveVersions/17.0.0/ucd/UnicodeData.txt: this file does not list U+1F02C nor U+1F8FF as assigned characters; hence they are unassigned.

Now, the file ArchiveVersions/17.0.0/ucd/auxiliary/LineBreakTest.txt treats these two sample future emoji base characters differently:

- In LineBreakTest.txt line 1527 there is a test case that allows a line break between U+0023 and U+1FFFD. This makes perfect sense: When U+1FFFD might be assigned to an emoji base in the future, it will behave like a character with line breaking property EB or ID. It makes sense to allow a line break there.
- In LineBreakTest.txt line 1647 there is a test case that *disallows* a line break between U+0023 and U+1F8FF. This makes no sense: When U+1F8FF might be assigned to an emoji base in the future, it *should* behave like a character with line breaking property EB or ID. Thus it *should* allow a line break there.

In this line the justification is rule LB28. But this rule https://www.unicode.org/reports/tr14/#LB28 says "Do not break between alphabetics". While unassigned characters in general (line breaking property XX) are mapped to AL by rule LB1 https://www.unicode.org/reports/tr14/#LB1, this rationale does not hold for future emoji base characters: it *should* behave like EB or ID, not AL.

- The test cases with character U+1F8FF in the following lines of LineBreakTest.txt have the same problem:
  267
  269
  [... see the original feedback...]
  17381
  19037

Background information / discussion

The line breaking algorithm indeed takes into account the prediction of future emoji status (in the form of Extended_Pictographic). But it is not its goal to treat potential future emoji exactly as they will be treated if encoded as emoji; nor is emoji status the only thing that the property assignments try to predict.

The purpose of LB30b is to avoid line breaks inside of emoji, which are particularly bad (they expose normally invisible technicalities of emoji encoding to users). It is extended to potential future emoji to avoid these especially bad line breaks for new emoji on old systems. (In fact, Extended_Pictographic is really acting as potential future emoji with skin colour here; most emoji will turn out to be lb=ID, not lb=EB, and will allow a break before a subsequent EM.)

Missing a line break opportunity that would be allowed if the emoji turns out to be an lb=ID emoji is nowhere near as bad as breaking in the middle of an emoji, and so for other rules, predictions other than the slim chance of being an emoji with skin colour take precedence; in this case, U+1FFFD is predicted to be lb=ID, and U+1F8FF is predicted behave more like lb=AL.

Note that Extended_Pictographic really means that it is *possible* that the character becomes an emoji, not that it is likely; and it is even less likely that it becomes an emoji that is affected by skin colour. Out of 72 code points that were unassigned and Extended_Pictographic in Unicode 14 and have now been assigned, only 34 have been assigned to emoji, and of those, only two, namely 🫷 and 🫸, take skin tone modifiers, i.e., are Line_Break=Emoji_Base. It is a prediction of a future assignment of Line_Break=Emoji_Base with lots of false positives, but the breaks forbidden by LB30b are so bad that we take this abundance of caution.

5. Security

5.1 Proposal for Reclassification of Balinese Script (Limited_Use → Recommended) [#446]

Recommended UTC actions

1. **No Action**: PAG recommends no action. In the context of Identifier_Type, the available evidence confirms that Limited_Use is appropriate.

Document

L2/25-218 by PANDI (.id Registry) et al.

From the submission email:

... I'm pleased to submit our proposal to reclassify the **Balinese script** from *Limited_Use* to *Recommended* under UAX #31.

The proposal follows the structure and evidence guidelines outlined in document L2/24-019, including:

- Community and government usage across education, signage, and administration
- EGIDS vitality (Level 2 – Provincial)
- Technical readiness (Unicode fonts, keyboards, digital dictionaries)
- Evidence of online usage and community engagement
- Confusable character analysis and mitigation notes

Background information / discussion

PAG has reviewed the request, and responded to the submitters in a timely fashion with the contents of L2/25-219 PAG assessment: Proposal for Reclassifying the Balinese Script

6. Confusables

6.1 UTS #39 refers to the stabilized UTR #36 for gatekeeper-confusable strings [#414]

Recommended UTC actions

1. **Action Item** for Markus Scherer, PAG: In UTS #39, Unicode Security Mechanisms, delete the second paragraph of Section 4 (“Collection of data for detecting gatekeeper-confusable strings […]”), and update the first paragraph to remove the reference to UTR #36. For Unicode 18.0. See L2/25-228 item 6.1.

PAG input

From Roozbeh Pournader, PAG

UTS #39 talks about "gatekeeper-confusable strings" in its Section 4, referring to UTR #36 for its definition. But since we are stabilizing UTR #36, we probably want to move the definition to either UTS #39 or somewhere else where we can maintain it.

Background information / discussion

This is just one of several types of confusables that we are not collecting: things to slip past gatekeepers; but there are also semantic confusables, underlining vs. macron below, confusable with uppercase, italics (Cyrillic *m* with Latin *m*), etc.

6.2 Confusables data for Devanagari UE and UUE [#449]

Recommended UTC actions

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

Confusables source

From Roozbeh Pournader, PAG:
The two Devanagari dependent vowels U+0956 ◌ॖ and U+0957 ◌ॗ look identical to a single breve below (U+032E ◌̮) and a double breve below. But such data is missing from `confusables.txt`.

Data

0956 ; 032E
0957 ; 032E 032E

6.3 Wrong confusables data for U+302C IDEOGRAPHIC DEPARTING TONE MARK [#451]

Recommended UTC actions

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

Confusables source

From Roozbeh Pournader, PAG:
The pair U+302C ◌〬 IDEOGRAPHIC DEPARTING TONE MARK and U+0309 ◌̉ COMBINING HOOK ABOVE are listed in `confusables.txt` by mistake. The first is a top-right ring shape, while the second is a centered hook above, looking like a semi-ring at best. U+302C ◌〬 should instead be made confusable with U+309A ◌゚ COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK, which is another top-right ring shape.

Data

Remove:
302C ; 0309
Add:
302C ; 309A

6.4 Confusables data for U+2800 BRAILLE PATTERN BLANK [#452]

Recommended UTC actions

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

Confusables source

From Mateusz Naściszewski, May 2022, through the old confusable submission form:
U+2800 BRAILLE PATTERN BLANK is confusable with a space.

Data

2800 ; 0020

6.5 Confusables data for U+00A1 INVERTED EXCLAMATION MARK [#453]

Recommended UTC actions

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

Confusables source

From dennis@..., July 2021, through the old confusable submission form:
U+00A1 ¡ INVERTED EXCLAMATION MARK is confusable with U+0069 i LATIN SMALL LETTER I

PAG comments: These are indeed confusable, and ¡ is commonly used across the internet to replace i either in a playful way or to go around simple filters.

Data

00A1 ; 0069

6.6 Post-2020 confusables data from the old form [#454]

Recommended UTC actions

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

Confusables source

Here are some confusable information from submissions to the old confusable forms. This covers all submissions made in 2020 and later, vetted by Roozbeh Pournader.

From David Corbett, January 2020, Bopomofo, Kana, and CJK, with Latin additions by Roozbeh Pournader:

U+3109 ㄉ BOPOMOFO LETTER D ≈ U+206A3 □ CJK UNIFIED IDEOGRAPH-206A3

U+310A ㄊ BOPOMOFO LETTER T ≈ U+20AD3 𠫓 CJK UNIFIED IDEOGRAPH-20AD3

U+310D ㄍ BOPOMOFO LETTER G ≈ U+5DDC 巜 CJK UNIFIED IDEOGRAPH-5DDC

U+310E ㄎ BOPOMOFO LETTER K ≈ U+4E02 丂 CJK UNIFIED IDEOGRAPH-4E02

U+3110 ㄐ BOPOMOFO LETTER J ≈ U+4E29 丩 CJK UNIFIED IDEOGRAPH-4E29

U+3111 ㄑ BOPOMOFO LETTER Q ≈ U+21FE8 𡿨 CJK UNIFIED IDEOGRAPH-21FE8

U+3112 ㄒ BOPOMOFO LETTER X ≈ U+4E05 丅 CJK UNIFIED IDEOGRAPH-4E05 ≈ U+0054 T LATIN CAPITAL LETTER T

U+3113 ㄓ BOPOMOFO LETTER ZH ≈ U+37A2 㞢 CJK UNIFIED IDEOGRAPH-37A2

U+3115 ㄕ BOPOMOFO LETTER SH ≈ U+21C23 □ CJK UNIFIED IDEOGRAPH-21C23

U+311A ㄚ BOPOMOFO LETTER A ≈ U+4E2B 丫 CJK UNIFIED IDEOGRAPH-4E2B ≈ U+0059 Y LATIN CAPITAL LETTER Y

U+311E ㄞ BOPOMOFO LETTER AI ≈ U+20005 □ CJK UNIFIED IDEOGRAPH-20005

U+3128 ㄨ BOPOMOFO LETTER U ≈ U+3405 㐅 CJK UNIFIED IDEOGRAPH-3405

U+312D ㄭ BOPOMOFO LETTER IH ≈ U+5E00 帀 CJK UNIFIED IDEOGRAPH-5E00

U+31A4 ㆤ BOPOMOFO LETTER EE ≈ U+30BB セ KATAKANA LETTER SE

U+5DFF 巿 CJK UNIFIED IDEOGRAPH-5DFF ≈ U+5E02 市 CJK UNIFIED IDEOGRAPH-5E02

U+66F0 曰 CJK UNIFIED IDEOGRAPH-66F0 ≈ U+65E5 日 CJK UNIFIED IDEOGRAPH-65E5

U+672B 末 CJK UNIFIED IDEOGRAPH-672B ≈ U+672A 未 CJK UNIFIED IDEOGRAPH-672A

U+2011E □ CJK UNIFIED IDEOGRAPH-2011E ≈ U+4E8C 二 CJK UNIFIED IDEOGRAPH-4E8C

U+299B4 □ CJK UNIFIED IDEOGRAPH-299B4 ≈ U+29994 □ CJK UNIFIED IDEOGRAPH-29994

Miscellaneous:

U+3147 ㅇ HANGUL LETTER IEUNG ≈ U+006F o LATIN SMALL LETTER O (from crackthrough@, December 2020)

Data

3109 ; 206A3
310A ; 20AD3
310D ; 5DDC
310E ; 4E02
3110 ; 4E29
3111 ; 21FE8
3112 ; 4E05
3112 ; 0054
3113 ; 37A2
3115 ; 21C23
311A ; 4E2B
311A ; 0059
311E ; 20005
3128 ; 3405
312D ; 5E00
31A4 ; 30BB
5DFF ; 5E02
66F0 ; 65E5
672B ; 672A
2011E ; 4E8C
299B4 ; 29994
3147 ; 006F

6.7 Confusables data for Indic abbreviation marks [#455]

Recommended UTC actions

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

Confusables source

From Roozbeh Pournader, PAG:
There has been a lot of Indic abbreviation signs added to Unicode, which mostly lack confusable data. In one specific case, the confusable data is wrong. Generally, Indic abbreviation marks should be dividided into a hollow class (confusable with U+00B0 ° DEGREE SIGN) and a filled class (confusable with U+00B7 · MIDDLE DOT).

This is the wrong existing data that should be removed, since the Sharada one is filled but the Devanagari one is hollow:

111C7 ; 0970 # (□ → ∘) SHARADA ABBREVIATION SIGN → DEVANAGARI ABBREVIATION SIGN
(originally reported by David Corbett through the old confusables form, December 2018)

Here is the hollow class:
U+0970 ∘ Devanagari
U+09FD ∘ Bengali
U+0AF0 ∘ Gujarati
U+110BB ∘ Kaithi
U+1123D ∘ Khojki
U+1144F ∘ Newa
U+114C6 □ Tirhuta
U+11643 □ Modi
U+1183B □ Dogra
U+1E5FF □ Ol Onal

This is the filled class:
U+0A76 · Gurmukhi
U+11174 □ Mahajani
U+111C7 □ Sharada
U+116B9 □ Takri

Data

Remove:
111C7 ; 0970

Add:
0970 ; 00B0
09FD ; 0970
0AF0 ; 0970
110BB ; 0970

1123D ; 0970
1144F ; 0970
114C6 ; 0970
11643 ; 0970
1183B ; 0970
1E5FF ; 0970

0A76 ; 00B7
11174 ; 0A76
111C7 ; 0A76
116B9 ; 0A76

6.8 Confusables data to remove based on feedback from David Corbett [#457]

Recommended UTC actions

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

Confusables source

The following confusable pairs are mistakes and should be corrected. The corrections come from contributions by David Corbett in 2018 and 2019. Comments are from Roozbeh Pournader, PAG:

U+1051C □ ELBASAN LETTER SHE ≠ LATIN CAPITAL LETTER C

RP: Looks like a typo in the codepoint. The correct codepoint is U+1051B □ ELBASAN LETTER SE.

U+118C4 □ WARANG CITI SMALL LETTER YA ≠ 007A z LATIN SMALL LETTER Z

RP: Looks like another typo. The correct codepoint is U+0079 y LATIN SMALL LETTER Y.

U+1F7D ώ GREEK SMALL LETTER OMEGA WITH OXIA ≠ U+1FF4 ῴ GREEK SMALL LETTER OMEGA WITH OXIA AND YPOGEGRAMMENI

RP: This doesn't make any sense, and will be ignored anyways, since both of these characters have canonical decompositions, so even if the source data includes them, as they get converted to NFD first, the implementation will never try to map one to the other. Should simply be removed.

U+14B7 ᒷ CANADIAN SYLLABICS WEST-CREE MWA ≠ <U+006C, U+00B7> l· LATIN SMALL LETTER L, MIDDLE DOT

RP: Another potential typo. The small L should be replaced by an uppercase L (U+004C)

U+011A Ě LATIN CAPITAL LETTER E WITH CARON ≠ U+0114 Ĕ LATIN CAPITAL LETTER E WITH BREVE

U+011B ě LATIN SMALL LETTER E WITH CARON ≠ U+0115 ĕ LATIN SMALL LETTER E WITH BREVE

U+0150 Ő LATIN CAPITAL LETTER O WITH DOUBLE ACUTE ≠ U+00D6 Ö LATIN CAPITAL LETTER O WITH DIAERESIS

RP: These three lines don't get used either, since none of the six characters above occur in NFD form. Should be removed.

0163 ; 01AB ; MA # (ţ → ƫ) LATIN SMALL LETTER T WITH CEDILLA → LATIN SMALL LETTER T WITH PALATAL HOOK
021B ; 01AB ; MA # (ț → ƫ) LATIN SMALL LETTER T WITH COMMA BELOW → LATIN SMALL LETTER T WITH PALATAL HOOK

RP: The two lines above don't get used either, since U+0163 and U+021B don't occur in NFD form. Instead, U+01AB ƫ LATIN SMALL LETTER T WITH PALATAL HOOK should be mapped to <t, combining palatized hook below> so that it becomes confusable with U+0163 and U+021B.

01CD ; 0102 ; MA # (Ǎ → Ă) LATIN CAPITAL LETTER A WITH CARON → LATIN CAPITAL LETTER A WITH BREVE
01CE ; 0103 ; MA # (ǎ → ă) LATIN SMALL LETTER A WITH CARON → LATIN SMALL LETTER A WITH BREVE
01CF ; 012C ; MA # (Ǐ → Ĭ) LATIN CAPITAL LETTER I WITH CARON → LATIN CAPITAL LETTER I WITH BREVE
01D0 ; 012D ; MA # (ǐ → ĭ) LATIN SMALL LETTER I WITH CARON → LATIN SMALL LETTER I WITH BREVE
01D1 ; 014E ; MA # (Ǒ → Ŏ) LATIN CAPITAL LETTER O WITH CARON → LATIN CAPITAL LETTER O WITH BREVE
01D2 ; 014F ; MA # (ǒ → ŏ) LATIN SMALL LETTER O WITH CARON → LATIN SMALL LETTER O WITH BREVE
01D3 ; 016C ; MA # (Ǔ → Ŭ) LATIN CAPITAL LETTER U WITH CARON → LATIN CAPITAL LETTER U WITH BREVE
01D4 ; 016D ; MA # (ǔ → ŭ) LATIN SMALL LETTER U WITH CARON → LATIN SMALL LETTER U WITH BREVE
01E6 ; 011E ; MA # (Ǧ → Ğ) LATIN CAPITAL LETTER G WITH CARON → LATIN CAPITAL LETTER G WITH BREVE
01E7 ; 011F ; MA # (ǧ → ğ) LATIN SMALL LETTER G WITH CARON → LATIN SMALL LETTER G WITH BREVE
01F5 ; 0123 ; MA # (ǵ → ģ) LATIN SMALL LETTER G WITH ACUTE → LATIN SMALL LETTER G WITH CEDILLA
0226 ; 00C5 ; MA # (Ȧ → Å) LATIN CAPITAL LETTER A WITH DOT ABOVE → LATIN CAPITAL LETTER A WITH RING ABOVE
0227 ; 00E5 ; MA # (ȧ → å) LATIN SMALL LETTER A WITH DOT ABOVE → LATIN SMALL LETTER A WITH RING ABOVE

RP: Don't get used since neither occur in NFD. They should all be removed.

0419 ; 040D ; MA # (Й → Ѝ) CYRILLIC CAPITAL LETTER SHORT I → CYRILLIC CAPITAL LETTER I WITH GRAVE
048A ; 040D 0326 ; MA # (Ҋ → Ѝ̦) CYRILLIC CAPITAL LETTER SHORT I WITH TAIL → CYRILLIC CAPITAL LETTER I WITH GRAVE, COMBINING COMMA BELOW
045D ; 0439 ; MA # (ѝ → й) CYRILLIC SMALL LETTER I WITH GRAVE → CYRILLIC SMALL LETTER SHORT I

RP: These are all equating a grave accent with a breve accent which doesn't make sense. On top of that, U+0419 and U+045D don't occur in NFD form. The first and last line should be removed and the middle line should be replaced by a version using U+0306 COMBINING BREVE.

1E9A ; 1EA3 ; MA # (ẚ → ả) LATIN SMALL LETTER A WITH RIGHT HALF RING → LATIN SMALL LETTER A WITH HOOK ABOVE

RP: The right side has a canonical decomposition, which makes it suboptimal. Also, the left side has a compatibility decomposition which is a closer resemblance to the character: aʾ. This should be replaced by a mapping from U+1E9A to its compatibility decomposition: <U+0061 a, U+02BE ʾ>.

2365 ; 0629 ; MA #* (⍥ → ة) APL FUNCTIONAL SYMBOL CIRCLE DIAERESIS → ARABIC LETTER TEH MARBUTA
00F6 ; 0629 ; MA # (ö → ة) LATIN SMALL LETTER O WITH DIAERESIS → ARABIC LETTER TEH MARBUTA

RP: The APL symbol is closer to an uppercase Ö than a lowercase ö, and the second mapping will not get used since U+00F6 doesn't happen in NFD form. Instead, U+0629should be mapped to <o, combining diaeresis> and U+2365 should be mapped to <O, combining diaeresis>.

FB2B ; FB2A ; MA # (שׂ → שׁ) HEBREW LETTER SHIN WITH SIN DOT → HEBREW LETTER SHIN WITH SHIN DOT
FB2D ; FB2C ; MA # (שּׂ → שּׁ) HEBREW LETTER SHIN WITH DAGESH AND SIN DOT → HEBREW LETTER SHIN WITH DAGESH AND SHIN DOT
FB30 ; FB2E ; MA # (אּ → אַ) HEBREW LETTER ALEF WITH MAPIQ → HEBREW LETTER ALEF WITH PATAH
FB39 ; FB1D ; MA # (יּ → יִ) HEBREW LETTER YOD WITH DAGESH → HEBREW LETTER YOD WITH HIRIQ
FB49 ; FB2A ; MA # (שּ → שׁ) HEBREW LETTER SHIN WITH DAGESH → HEBREW LETTER SHIN WITH SHIN DOT
114BC ; 09CB ; MA # (𑒼 → ো) TIRHUTA VOWEL SIGN O → BENGALI VOWEL SIGN O
114BE ; 09CC ; MA # (𑒾 → ৌ) TIRHUTA VOWEL SIGN AU → BENGALI VOWEL SIGN AU
00C7 ; 0043 0326 ; MA # (Ç → C̦) LATIN CAPITAL LETTER C WITH CEDILLA → LATIN CAPITAL LETTER C, COMBINING COMMA BELOW
00E7 ; 0063 0326 ; MA # (ç → c̦) LATIN SMALL LETTER C WITH CEDILLA → LATIN SMALL LETTER C, COMBINING COMMA BELOW
0146 ; 0272 ; MA # (ņ → ɲ) LATIN SMALL LETTER N WITH CEDILLA → LATIN SMALL LETTER N WITH LEFT HOOK
01A0 ; 004F 0027 ; MA # (Ơ → O') LATIN CAPITAL LETTER O WITH HORN → LATIN CAPITAL LETTER O, APOSTROPHE
01A1 ; 006F 0027 ; MA # (ơ → o') LATIN SMALL LETTER O WITH HORN → LATIN SMALL LETTER O, APOSTROPHE
01FE ; 004F 0338 0301 ; MA # (Ǿ → Ó̸) LATIN CAPITAL LETTER O WITH STROKE AND ACUTE → LATIN CAPITAL LETTER O, COMBINING LONG SOLIDUS OVERLAY, COMBINING ACUTE ACCENT
021A ; 0162 ; MA # (Ț → Ţ) LATIN CAPITAL LETTER T WITH COMMA BELOW → LATIN CAPITAL LETTER T WITH CEDILLA
0340 ; 0300 ; MA # (̀ → ̀) COMBINING GRAVE TONE MARK → COMBINING GRAVE ACCENT
0341 ; 0301 ; MA # (́ → ́) COMBINING ACUTE TONE MARK → COMBINING ACUTE ACCENT
0343 ; 0313 ; MA # (̓ → ̓) COMBINING GREEK KORONIS → COMBINING COMMA ABOVE
0623 ; 006C 0674 ; MA # (أ → lٴ) ARABIC LETTER ALEF WITH HAMZA ABOVE → LATIN SMALL LETTER L, ARABIC LETTER HIGH HAMZA
0624 ; 0648 0674 ; MA # (ؤ → وٴ) ARABIC LETTER WAW WITH HAMZA ABOVE → ARABIC LETTER WAW, ARABIC LETTER HIGH HAMZA
0625 ; 006C 0655 ; MA # (إ → lٕ) ARABIC LETTER ALEF WITH HAMZA BELOW → LATIN SMALL LETTER L, ARABIC HAMZA BELOW
0626 ; 0649 0674 ; MA # (ئ → ىٴ) ARABIC LETTER YEH WITH HAMZA ABOVE → ARABIC LETTER ALEF MAKSURA, ARABIC LETTER HIGH HAMZA
06C2 ; 06C0 ; MA # (ۂ → ۀ) ARABIC LETTER HEH GOAL WITH HAMZA ABOVE → ARABIC LETTER HEH WITH YEH ABOVE
0BCA ; 0BC6 0B88 ; MA # (ொ → ெஈ) TAMIL VOWEL SIGN O → TAMIL VOWEL SIGN E, TAMIL LETTER II
0BCB ; 0BC7 0B88 ; MA # (ோ → ேஈ) TAMIL VOWEL SIGN OO → TAMIL VOWEL SIGN EE, TAMIL LETTER II
0BCC ; 0BC6 0BB3 ; MA # (ௌ → ெள) TAMIL VOWEL SIGN AU → TAMIL VOWEL SIGN E, TAMIL LETTER LLA

RP: These won't get used since the left side doesn't occur in NFD form. They should be removed.

FBA5 ; 06C0 ; MA # (ﮥ → ۀ) ARABIC LETTER HEH WITH YEH ABOVE FINAL FORM → ARABIC LETTER HEH WITH YEH ABOVE
FBA4 ; 06C0 ; MA # (ﮤ → ۀ) ARABIC LETTER HEH WITH YEH ABOVE ISOLATED FORM → ARABIC LETTER HEH WITH YEH ABOVE

RP: The right side is not NFD, so it's better replaced by its canonical decomposition

1E43 ; AB51 ; MA # (ṃ → ꭑ) LATIN SMALL LETTER M WITH DOT BELOW → LATIN SMALL LETTER TURNED UI

RP: This won't get used since the left side doesn't happen in NFD. Instead, we should reverse it and map U+AB51 to <m, combining dot below>.

1FF6 ; 13EF ; MA # (ῶ → Ꮿ) GREEK SMALL LETTER OMEGA WITH PERISPOMENI → CHEROKEE LETTER YA

RP: The left side doesn't occur in NFD, so the rule won't get used. The characters are not that similar either. It should be removed.

A4ED ; 1660 ; MA # (ꓭ → ᙠ) LISU LETTER GHA → CANADIAN SYLLABICS CARRIER TSA

RP: The characters are not that similar. Instead, U+A4ED should be made confusable with U+10921 𐤡 LYDIAN LETTER B and U+15FA ᗺ CANADIAN SYLLABICS CARRIER KHA.

6.9 Wrong confusables data for dž digraphs [#459]

Recommended UTC actions

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

Confusables source

From Roozbeh Pournader, PAG, based on feedback from David Corbett:

Confusables data for dž digraphs is currently as follows:

01C4 ; 0044 017D ; MA # (Ǆ → DŽ) LATIN CAPITAL LETTER DZ WITH CARON → LATIN CAPITAL LETTER D, LATIN CAPITAL LETTER Z WITH CARON #

01C5 ; 0044 017E ; MA # (ǅ → Dž) LATIN CAPITAL LETTER D WITH SMALL LETTER Z WITH CARON → LATIN CAPITAL LETTER D, LATIN SMALL LETTER Z WITH CARON #

01C6 ; 0064 017E ; MA # (ǆ → dž) LATIN SMALL LETTER DZ WITH CARON → LATIN SMALL LETTER D, LATIN SMALL LETTER Z WITH CARON #

But confusables data is supposed to be in NFD form. So instead of the mapping to Ž and ž, they should map to <Z/z, combining caron>.

Data

Remove:
01C4 ; 0044 017D
01C5 ; 0044 017E
01C6 ; 0064 017E

Add:
01C4 ; 0044 005A 030C
01C5 ; 0044 007A 030C
01C6 ; 0064 007A 030C

7. Other

7.1 Improve readability of Unicode Set spec [#440]

Recommended UTC actions

1. **Action Item** for Robin Leroy, PAG: In Proposed Draft Unicode Technical Standard #61, Unicode Set Notation, change the definitions of “string-literal” and “bracketed-element” to be space-sensitive, and remove the syntactic category “optional-white-space”. See L2/25-228 item 7.1.

Feedback (verbatim)

PRI-523 for PD-UTS #61 Unicode Set Notation

Date/Time: Tue July 01 18:53:21 PDT 2025
ReportID: ID20250701185321
Name: Mark Davis
Report Type: Public Review Issue
Opt Subject: Improve readability of Unicode Set spec

I think that https://www.unicode.org/reports/tr61/ would be more readable if it used shorter names for optional-white-space and white-space,. In message format, for example, o and s are used, such as:

literal-expression = "{" o literal [s function] *(s attribute) o "}"

I find that easier to read than if it were written with the long names:

literal-expression = "{" optional-white-space literal [white-space function] *(white-space attribute) optional-white-space "}"

where the whitespace obscures the meat of the syntax. The main place that would benefit would be:

bracketed-element ⩴ { optional-white-space string-element optional-white-space }

string-literal ⩴

{ optional-white-space }

{ optional-white-space string-elements optional-white-space }

string-element ⩴

bracketed-literal-element escaped-element named-element

string-elements ⩴

string-element optional-white-space string-element

string-elements optional-white-space string-element

which could become

bracketed-element ⩴ { o string-element o }

string-literal ⩴ { o } { o string-elements o }

string-element ⩴ bracketed-literal-element escaped-element named-element

string-elements ⩴ string-element o string-element string-elements o string-element

Question: the last line would also be simpler and easier to understand as the following.

string-elements ⩴ string-element (o string-element)* // or ABNF equivalent

I recall though that the spec is using the less-readable format to be more easily parsable by tooling.

"ows" and "ws" would also be more readable than optional-white-space and white-space, though not quite as pithy.

Background information / discussion

As noted in the feedback, the main place that would benefit would be the definition of the lexical elements *string-literal* and *bracketed-element*; in fact these are the *only*rules in the grammar that use *optional-white-space*.

This should appear strange at first sight, since UnicodeSet syntax is full of optional white space: (`[ ^ a - z ]` is the same as `[^a-z]`). The reason why *optional-white-space* does not appear elsewhere in the grammar is that the grammar separates lexical analysis (lexing) from syntactic analysis (parsing); once the lexical elements (tokens) are defined, the optional white space between them is described by the sentence “One or more white-space character is allowed between any two adjacent lexical elements; this is not indicated explicitly in the grammar for UnicodeSet”.

However, the original implementation in ICU did not have lexing separate from parsing (nor did the grammar in UTS #35). See the “Dragon Book” [ASU85, pp. 84 sq.] for reasons for distinguishing these phases; “Simpler design is perhaps the most important consideration. […] If we are designing a new language, separating the lexical and syntactic conventions can lead to a cleaner overal language”.

The lack of separate lexing meant that when strings were added to UnicodeSet, the character-by-character iterator that skipped over white space was used to parse those, and thus string literals ignore white space: `{this is a string without spaces}`represents the character sequence `thisisastringwithoutspaces`. The presence of *optional-white-space* within the grammar for a lexical element is effectively a scar of this *post hoc* separation of lexing from parsing.

Discussions with Mark Davis and Markus Scherer reveal that this behaviour was unintentional, and the properties and algorithms working group likewise found that it is highly misleading for a string literal to discard spaces. The properties and algorithms group therefore recommends that the *optional-white-space* simply be removed from the *bracketed-element* and *string-literal* productions, and that white space be added to *string-element*, rendering the issue of the readability of the syntactic category *optional-white-space* moot.

On the question of using the Kleene star, the grammar is indeed using the less concise format partly to be compatible with more tooling (including the parser generator that the author had been using to check the grammar for ambiguities), but also partly because the author finds that more readable (cf. the C++ standard, which also defines a named nonterminal symbol for sequences; it does have subscript *opt* to indicate an optional symbol), and partly because naming everything makes it much easier to refer to constructs in the prose that describes the semantics.

[ASU85] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1985.
ISBN: 0-201-10088-6.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.
ISBN: 0-201-10194-7.

7.2 Unicode 17 UAX #42 Modification section is missing the renames [#467]

Recommended UTC actions

1. **Action Item** for John Wilcock, PAG: In the proposed update of UAX #42 for Unicode 18, amend the revision 38 Modifications to add the omitted items. For Unicode 18.0. See L2/25-228 item 7.2.

Feedback (verbatim)

Date/Time: Mon Aug 27 17:06:31 PT 2025
ReportID: ID20250827170631
Name: Daniel Bünzli
Report Type: General Feedback
Subject: UAX #42 Modification section is missing the renames

Hello,

I noticed that in the description of revision 37 of UAX #42. The following lines are missing:

- Renamed attribute `kRSTUnicode` to `kTGT_RSUnicode`
- Renamed attribute `kSrc_NushuDuben` to `kNSHU_DubenSrc`
- Renamed attribute `kReading` to `kNSHU_Reading`

Thanks for your work on keeping the ucdxml alive.

Best,

Daniel

7.3 PRI-509: DUTS58 suggestions for Summary [#471]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The D-UTS #58 Summary has been updated in revision 1 draft 5, based on this feedback.
2. In addition, D-UTS #58 revision 1 draft 5 includes many changes based on review and feedback by other PAG members.

Feedback (verbatim)

PRI-509

Date/Time: Mon Sep 15 14:07:47 PST 2025
ReportID: ID20250915140747
Name: Peter Constable
Report Type: Public Review Issue
Opt Subject: PRI 509: suggestions for Summary

Editorial suggest for the intro summary of DUTS 58:

URLs processed in communication protocols are parsed in conformance with particular protocol specifications. When URLs appear in
text content, however, the character sequences are not always intended to be read in exactly the same way they would be parsed in
a protocol. Some characters that are often used as sentence-level punctuation in text can also be valid characters within a URL.
Software that applies the protocol rules when parsing URLs in text content often produce the wrong results.

When a URL is inserted into text, percent encoding can be used to avoid the above ambiguity, though often this is not done. Also,
when a URL that includes non-ASCII characters is inserted into text, implementations often over-use percent encoding for those
characters, resulting in a URL that is illegible for a human reader. Thus, percent encoding is often both underused and overused
leading to less beneficial results.

This document specifies...

7.4 PRI-509: DUTS58 paired punctuation within path, etc. [#472]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The relevant D-UTS #58 text has been modified in revision 1 draft 5.

Feedback (verbatim)

PRI-509

Date/Time: Mon Sep 15 17:27:18 PST 2025
ReportID: ID20250915172718
Name: Peter Constable
Report Type: Public Review Issue
Opt Subject: PRI 509: paired punctuation within path, etc.

In the Link_Termination Property section, the description for Close mentions "subparts" within a path, query or fragment:

"If the character is paired with a previous character in the same Part (path, query, fragment) and in the same subpart
(that is, not across interior '/' in a path, or across '&' or '=' in a query, it is treated as Include."

(It might also have mentioned "/" and "?" within query and fragment but doesn't.)

Two issues:

First, is seems to make sense that counterpart open/close punctuation is unlikely to be used with an intent of being paired
across segment boundaries within a path. So, not searching for a pair across a "/" boundary within a path seems to make sense.
However, it seems less obvious that the same can be said for query or fragment elements of a URL. For example, it seems
conceivable that a "(" ... ")" pair might surround a sequence of key/value pairs that comprise a logical grouping. E.g.,

...(k1=a&k2=b)...

I have no idea if this kind of pairing is done in practice, so perhaps this isn't more than a remote, hypothetical possibility.
But I do know it is permissible in RFC 3986.

But -- the second point -- the algorithm, as written, does not include anything to recognize such "subpart" segments or to
incorporate awareness of such subparts into the logic.

Thus, unless there is thought to elaborate the algorithm in some way to support these subparts, it doesn't make sense to mention
them at all.

Background information / discussion

While the RFCs which describe URL syntax allow many punctuation characters in query parts, in normal usage there is no consistent pattern of parentheses having precedence over '&' or '='.

The term “subpart” was not intended to have a specific meaning in the context of URLs. It was used like “substring” or “character sequence”. Regardless, this sentence has been rephrased and hopefully slightly clarified.

7.5 PRI-509: DUTS58 pairing across path/query/frag boundaries [#473]

Recommended UTC actions

1. **No Action**: PAG recommends no action. The formal statement of the Termination Algorithm is explicit about the processing of paired punctuation. Usage of the term “part” has been regularized.

Feedback (verbatim)

PRI-509

Date/Time: Mon Sep 15 18:15:59 PST 2025
ReportID: ID20250915181559
Name: Peter Constable
Report Type: Public Review Issue
Opt Subject: PRI 509: pairing across path/query/frag boundaries

Given potential ambiguity of closers like ")" at the end of a candidate URL sequences, it makes sense not to attempt to infer a pairing across the boundaries between the top-level elements of a URL: e.g., opener in a path paired with a closer in query.
The algorithm, as it is intended to be implemented, ensures this by having the steps within the Link-Detection Algorithm section applied separately, and in turn, to path, query and fragments elements.

However, there is potential for implementations to overlook that detail and to apply the steps in that section to a sequence
spanning path + query + fragment. The intent for the steps to be applied to those elements separately is not stated in the
Link-Detection Algorithm section, so if an implementer hasn't read and paid adequate attention to earlier sections, they might overlook that very important detail.

Partially related is that the terms "part" and "Part" are used inconsistently through the document. So, for instance, use of "part" in section 3 "Parts of a URL" includes references to protocol and port elements.

The first mention of this crucial Part-at-a-time logic is in passing, in the second paragraph of the Termination section:

"The key is to be able to determine, given a Part (such as query)..."

The next is buried in the description of Close:

"If the character is paired with a previous character *in the same Part (path query fragment)* ..."

Of course, it is the handling of potential open/close pairs for which the Part-at-a-time logic matters the most. But the doc
should call this out more clearly.

The clearest statements coming in the Termination Algorithm section:

"This algorithm then processes each final Part [path, query, fragment] of the URL in turn. ... A Link_Termination=Close character... that does **not** have a matching Open character *in the same Part* of the URL."

If we assume implementers read and pay attention to these sections, that *should* be adequate. Even so, to be most explicit, it makes sense to call out this detail directly within the Link-Detection Algorithm section. The following is a suggestion:

```
## Link-Detection Algorithm
```

The following steps are performed, logically, over path, query and fragment elements of a URL separately. The are applied first over the path, if present. If no termination is detected within the path, the steps are repeated for the query, if present, with variables reset. If no termination is detected within the query, the steps are repeated for the fragment, if present. Crucially, the openStack must be cleared on the transitions from path to query and to fragment.

In the following:

- Part refers to one of {path, query, fragment}.

...

====

As noted earlier, usage of "Part" and "part" is not consistent. E.g.,

- one of {path, query, fragment} — the intended meaning
- other main URL elements: e.g., protocol, host, port (top of section 3); also in section 4, "Process each Part up to the Path, Query,
  and Fragment in the normal fashion..."
- individual characters: e.g., "... when a trailing period should be counted as part of a link or not."
- positions within the URL: e.g., "... the last location in the current Part that is still safely considered part of the link."
- a span of URL characters: e.g., "... vs. URLs that contain a part that is enclosed in parens, etc."

Also inconsistent is casing when referring to path, query and fragment (some instances are capitalized), and in references to the three elements as a set---"... Part (path, query, fragment)" vs. "... Part [path, query, fragment]".

Background information / discussion

The Introduction cannot serve as a complete specification. Implementers are expected to read the formal, detailed algorithm specification.

The semi-formal Termination Algorithm *does* state clearly that the URL Parts are processed in sequence, with paired punctuation within each Part:

- “This algorithm then processes each final URL Part [path, query, fragment] of the URL in turn. ...”
- “... A Link_Termination=Close character, such as a] that does **not** have a matching Open character *in the same Part* of the URL. ...”

The formal specification via the Link-Detection Algorithm pseudocode is explicit about this.

The use of the term “part” has been regularized, either with an explict URL context or as the “local-part” of an email address. Editorial cleanup of casing “part” vs. “Part” is subject to later drafts.