UTC #175 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2023-apr-20

Participants

The following people have contributed to this document:

Markus Scherer (chair), Josh Hadley (vice chair), Asmus Freytag, Elango Cheran, Ken Whistler, Manish Goregaokar, Mark Davis, Ned Holbrook, Peter Constable, Rick McGowan, Robin Leroy, Steven Loomis

1. Core spec

This section intentionally left blank.

2. UCD

2.1 Add Simple_Case_Folding mappings for three existing characters

L2/23-062 from Markus Scherer

Recommended UTC actions

1. Consensus: Add Simple_Case_Folding mappings for U+1FD3, U+1FE3, and U+FB05, see L2/23-062; for Unicode 15.1.
2. Action Item for Markus Scherer, PAG: In CaseFolding.txt, add Simple_Case_Folding mappings for U+1FD3, U+1FE3, and U+FB05, see L2/23-062; for Unicode 15.1.

Summary

These three characters have (full) Case_Folding mappings by which they each match a related character, and they map to the same sequences via NFKC, but they do not have Simple_Case_Folding mappings. Therefore, surprisingly, these do not match their related characters under simple folding.

2.2 Annotated version of UAX #14

Recommended UTC actions

1. Action Item for Robin Leroy, EDC: Publish the annotated version of the UAX #14, Unicode Line Breaking Algorithm, as a new UTN (Unicode Technical Note).

Feedback

From Robin Leroy:

We keep running into a certain kind of question whereby we wonder why some part of the standard is the way it is.

Sometimes such questions come directly from outside. Sometimes we have similar questions that arise because some obscure rule causes unforeseen issues; we then need to do archæology to figure out why these rules are the way they are in order to understand how we can change them without resurrecting forgotten bugs.

This kind of problem is not specific to Unicode. Asmus has mentioned that IDNA has references to source proposals in its data structures. The Ada Rapporteur Group's answer to that problem is the Annotated Ada Reference Manual, which is to produce a *separate* document, which consists of the standard, plus:

- differences from the preceding version (much like our yellowed reports);
- references to proposals, discussions, and ARG or WG9 dispositions behind the change;
- additional annotations pointing out reasons for rules, ramifications of rules, etc., that go in greater depth than is relevant to the intended audience of the standard, but that are useful to the standardizers.

Having noticed in PRI-446 and L2/22-243 the abundance of « why is the sky blue? » questions on this particular document, I have produced an annotated version of UAX14 in that vein. Temporary demo: eggrobin.github.io/unicode-annotations/alba.html.

This has facilitated the work on UAX #14 this cycle, such as

- identifying and fixing outdated wording (which used pre-5.0 terminology) in 173-A6, and CP1252 holdovers from Unicode 3.0.0 in 173-A13.
- understanding long-standing issues with quotation marks in L2/23-063,
- quickly iterating on amendments to the proposed rules in L2/22-080 to deal with the dotted circle issue, with a good understanding of the interactions between rules.

Making this available would facilitate the work of contributors, and help users answer their own questions about the origins of rules and wording. A Unicode Technical Note seems like an appropriate medium for such a standing document; compare UTN45.

2.3 value "none" can mean the absence of a value

Recommended UTC actions

1. No action. Ken Whistler has already modified UAX #44 to say that <none> indicates that no value is defined for a code point. No longer referring to the “empty string” and not referring to “absence of a string”. Instead now points to 4.2.11 Empty Fields for contrast.

Feedback

From Markus Scherer:

UAX 44 https://www.unicode.org/reports/tr44/#Missing_Conventions documents

> The special tag values which may occur in the default_prop_val field in an @missing line are interpreted as follows:
> <none> = the empty string

However, in some (many?) cases it's really not mapping a code point to the empty string, but documenting that for that code point there is no meaningful value -- and “none” says that nicely.

For example, Bidi_Paired_Bracket (in BidiBrackets.txt) uses <none> to indicate that for any character not listed in the file there is no other character that is its paired bracket.

Whether a programmatic API returns an empty string or a null value or a None value or a special dummy string... is beside the point for the UCD.

I suggest that we amend the UAX 44 “Interpretation” to “the empty string, or the absence of a value”.

2.4 UAX #31 "grandfathered character"

PRI #462 “Proposed Update UAX #31 Unicode Identifiers and Syntax”

Recommended UTC actions

1. Action Item for Markus Scherer, Asmus Freytag, PAG: Modify UAX #31 removing the use of "grandfathered" or replacing it with other language as appropriate without creating novel terms, for Unicode 15.1. See L2/23-079 item 2.4.

Feedback (verbatim)

Date/Time: Fri Jan 6 16:44:32 CST 2023
Name: Markus Scherer
Report Type: Error Report
Opt Subject: UAX #31 "grandfathered character"

I just stumbled on the term "grandfathered character" in UAX #31.
We should replace that; see

- https://en.wikipedia.org/wiki/Grandfather_clause#Origin
- https://medium.com/@nriley/words-matter-why-we-should-put-an-end-to-grandfathering-8b19efe08b6a
- https://developers.google.com/style/inclusive-documentation
- https://developers.google.com/style/word-list#grandfathered

There are five occurrences in `UAX #31`. Suggestions:

2.5 Backward Compatibility

Unicode General_Category values are kept as stable as possible, but they can change across versions of the Unicode Standard. The bulk of the characters having a given value are determined by other properties, and the coverage expands in the future according to the assignment of those properties. In addition, the Other_ID_Start property provides a small list of characters that qualified as ID_Start characters in some previous version of Unicode solely on the basis of their General_Category properties, but that no longer qualify in the current version. These are called grandfathered characters.
-->

Just remove the last sentence.

2 Default Identifiers

Note: The UAX31-R1b requirement is typically achieved by using grandfathered characters. See Section 2.5, Backward Compatibility. Where profiles are allowed, management of those profiles may also be required to guarantee backwards compatibility. Typically such management also uses grandfathered characters.

-->

Note: The UAX31-R1b requirement is typically achieved by using a small list of characters that qualified as identifier characters in some previous version of Unicode. See Section 2.5, Backward Compatibility. Where profiles are allowed, management of those profiles may also be required to guarantee backwards compatibility. Typically such management also uses a list of characters that qualified previously.

5.2 Case and Stability

Casing stability is also an issue for bicameral writing systems. The assignment of General_Category property values, such as gc=Lu, is not guaranteed to be stable, nor is the assignment of characters to the broader properties such as Uppercase. So these property values cannot be used by themselves, without incorporating a grandfathering mechanism, such as is done for Unicode identifiers in Section 2.5 Backward Compatibility. That is, the implementation would maintain its own list of special inclusions and exclusions that require updating for each new version of Unicode.

-->

Casing stability is also an issue for bicameral writing systems. The assignment of General_Category property values, such as gc=Lu, is not guaranteed to be stable, nor is the assignment of characters to the broader properties such as Uppercase. So these property values cannot be used by themselves, without incorporating a compatibility-preserving [stability-enforcing?] mechanism, such as is done for Unicode identifiers in Section 2.5 Backward Compatibility. That is, the implementation would maintain its own list of special inclusions and exclusions that require updating for each new version of Unicode.

6 Hashtag Identifiers

The grandfathering techniques mentioned in Section 2.5 Backward Compatibility may be used where stability between successive versions is required.

-->

The compatibility-preserving [stability-enforcing?] techniques mentioned in Section 2.5 Backward Compatibility may be used where stability between successive versions is required.

Background information / discussion

Asmus: Most places where we mention Section 2.5 we can delete grandfathered as "techniques" are already for backwards compatibility. We don't need to add a new qualification for that cross reference.

Don't use "compatibility-preserving" in Case and Stability as that looks like a defined term, but use "that preserve backward compatibility".

2.5 Case convention of the name of NFKC_SCF

Recommended UTC actions

1. Consensus: Change the name of the informative property created by Consensus 174-C2 to NFKC_Simple_Casefold, with a low line between the words Simple and Casefold. Its alias NFKC_SCF and its definition are unchanged. For Unicode Version 15.1.
2. Action Item for Ken Whistler, PAG: Update the name of the property NFKC_SCF in the Property Table of Unicode Standard Annex #44, Unicode Character Database, for Unicode Version 15.1. See document L2/23-079 item 2.5.

Feedback

From Robin Leroy:

Spotted while working on the following AI:

> [174-C2] Consensus: Create a new informative, derived property NFKC_SimpleCasefold (NFKC_SCF), derived as its non-Simple counterparts except for the use of the Simple_Case_Folding instead of the Case_Folding, for Unicode Version 15.1.
> [174-A8] Action Item for Mark Davis, PAG: Add NFKC_SimpleCasefold to DerivedNormalizationProps.txt and PropertyAliases.txt, for Unicode Version 15.1. See document L2/23-008 item 1.2.

The name NFKC_SimpleCasefold is odd, in that it has a mix of CamelCase and Low_Lines. None of the other properties in Table 7 of UAX44 use CamelCase. Unihan uses CamelCase, but no low lines; this is not Unihan anyway.

Should the property be called NFKC_Simple_Casefold?

3. New Scripts & Characters

3.1 New characters with no significant issues

PAG members reviewed the following proposals, provided feedback to SAH, and the feedback has been addressed.
No further recommended actions from our side.

- L2/22-268 Revised Proposal to Encode Alternate BA for the Bengali Language
- L2/23-019 Revised proposal to encode Sidetic in Unicode
- L2/23-024 Proposal to encode Tolong Siki in Unicode
- L2/23-065 Proposal to encode a blank character for Khitan Small Script

3.2 Forint sign

L2/23/060 Proposal to Encode a Hungarian Forint Symbol in the Unicode Standard

Discussion between PAG and SAH about this proposal is ongoing. We ask that the UTC not make any binding decision yet in favor of the proposal.

4. Bidi

4.1 Dependency of the bidi algorithm on normalization

Recommended UTC actions

1. Consensus: Unicode will not add further characters with both (a) canonical decompositions, and (b) Bidi_Paired_Bracket_Type ≠ None. Unicode will also not add further characters where the canonical decompositions *contain* characters whose Bidi_Paired_Bracket_Type ≠ None. See L2/23-079 item 4.1.
2. Consensus: The UTC establishes the above Consensus 175-C?? as a precedent according to UTC procedures 10.5.2.
3. Action Item for Manish Goregaokar, PAG: Change UAX #9 to point out that an implementation of the UBA need not perform normalization / canonical equivalence in general, and explicitly list the pairs of paired bracket characters relevant for canonical equivalence and merely motivate this list via normalization; note that this list is immutable unless Unicode overturns precedent 175-C??; for Unicode 15.1. See L2/23-079 item 4.1.

Feedback

From Manish Goregaokar:

In UAX #9:

> BD16. A bracket pair is a pair of characters consisting of an opening paired bracket and a closing paired bracket such that the Bidi_Paired_Bracket property value of the former or its canonical equivalent equals the latter or its canonical equivalent and which are algorithmically identified at specific text positions within an isolating run sequence.

The "canonical equivalence" bit exists so that text containing "mismatched" brackets that "match" under normalization will have the same bidi behavior before and after normalization.

However, there is only a single pair of characters for which this matters: U+2329 LEFT-POINTING ANGLE BRACKET and U+232A RIGHT-POINTING ANGLE BRACKET (which map to U+3008 and U+3009)

In previous discussion we determined that *current* Unicode stability policy does not prevent further such singleton normalizations from being introduced.

However, it is worth considering if we can change this, basically, have a policy against new characters being introduced that normalize to other Bidi_Paired_Bracket characters (in other words, introducing new types that are `NFC_Inert=No` + `Bidi_Paired_Bracket_Type = {Open, Close}`.

If we still want to maintain the ability to add such characters in the future, the algorithm *could* be changed to rely on BPB values that are pre-normalized (i.e. U+2329 maps to U+3009 and U+232A maps to U+3008) where for BD16 it checks both the value and its corresponding paired value (this will be tricky to get right, but ultimately doable). This makes the property no longer solely derived from Bidi_M, though. I don't like this particular path.

Background information / discussion

The problematic cases are "bracket" punctuation characters with canonical singleton decompositions. It is very unlikely that we will add any more such characters.

This edge case does not seem to warrant a stability policy.

Ken Whistler mentioned these characters on the public mailing list last year https://corp.unicode.org/pipermail/unicode/2022-March/010074.html, writing

> And it is vanishingly unlikely that the UTC is ever going to
> add more such paired brackets with canonical decomposition mappings.

and noting that

> The BidiReference code just does a hard-coded additional test (and
> explains why). For this particular edge case, that works just as well,
> is just as robust (see above assertion that UTC isn't going to add more
> exceptions to be dealt with), […]

5. Text Segmentation

5.1 Improve the handling of class QU in the line breaking algorithm

L2/23-063 “Line breaking around quotation marks” from Robin Leroy

Recommended UTC actions

1. Consensus: Replace rule LB 15 by LB 15a and LB 15b in UAX #14, as described in L2/23-063 *Line breaking around quotation marks*. For Unicode Version 15.1.
2. Action Item for Robin Leroy, PAG: Make the changes to the Proposed Update for UAX #14 described in L2/23-063. For Unicode Version 15.1.

Summary

This document is a proposal for changes to Unicode Standard Annex #14, Unicode Line Breaking Algorithm, in order to improve its handling of « this kind » of quotation marks, and fix strange edge cases in the existing handling of quotation marks

5.2 UAX #29: WB4 should be expanded and clarified

PRI #469 Proposed Update UAX #29, Unicode Text Segmentation

Recommended UTC actions

1. Consensus: Change the Word_Break property of U+0600–U+0605, U+06DD, U+0890, U+0891, U+08E2, U+110BD, and U+110CD from Format to Numeric, and the Word_Break property of U+070F from Format to ALetter. See L2/23-079 item 5.2.
2. Action Item for Josh Hadley, PAG: Update Table 3 of Unicode Standard Annex #29, Unicode Text Segmentation, to exclude GCB=Prepend from Word_Break=Format, include U+0600–U+0605, U+06DD, U+0890, U+0891, U+08E2, U+110BD, and U+110CD in Word_Break=Numeric, and include U+070F in Word_Break=ALetter. For Unicode Version 15.1.
3. Action Item for Robin Leroy, PAG: Update WordBreakProperty.txt according to L2/23-079 item 5.2. For Unicode Version 15.1.

Feedback (verbatim)

Date/Time: Fri Jan 6 18:26:42 CST 2023
Name: Marshall Stoner
Report Type: Error Report
Opt Subject: www.unicode.org/reports/tr29/

The Rule WB4 should be expanded and clarified. As is, the algorithm may break an Arabic numeric heading such as U+061C U+0600 U+0664 U+0666 in the wrong place. The word break rules should lead to "U+061C ÷ U+0600 × U+0664", *not* "U+061C x U+0600 ÷ U+0664". According to the same document, the sequence "U+0600 U+0664" is a grapheme cluster that should not be broken. I think there should be a rule in addition to WB4 that clarifies the break should come *after* most 'Format', 'Extend', or 'ZWJ', code points, but 'Format' should exclude any format characters that are subtending marks. Format characters that are subtending marks should be placed in a new category and there should then be two rules..

```
Unset
WB4a:    Any × ( Extend | Format | ZWJ )
WB4b:    Prepend × Any
```

Therefore, if there is a sequence `[some letter] ( Extend | Format | ZWJ )* Prepend* [ another letter ]`, the break should always occur after the `"( Extend | Format | ZWJ)*"` string but *before* the `"Prepend*"` string. Prepend should be characters excluded from Format.

Background information / discussion

http://www.unicode.org/reports/tr29/#WB4

> Ignore Format and Extend characters, except after sot, CR, LF, and Newline. (See Section 6.2, Replacing Ignore Rules.) This also has the effect of: `Any × (Format | Extend | ZWJ)`
> WB4 `X (Extend | Format | ZWJ)* → X`

There is a GCB=Prepend value, but there is no WB=Prepend. The GCB=Prepend characters have either WB=Format or WB=ALetter.

See Unicode 15.0, Figure 9-7. Arabic Signs Spanning Numbers.

Characters mentioned in the feedback and in discussion:

cp	ARABIC...	char	chart heading	gc	GCB	WB
0600	NUMBER SIGN	ـــ؀	Subtending marks	Cf	Prepend	Format
0601	SIGN SANAH	ـــ؁	Subtending marks	Cf	Prepend	Format
0602	FOOTNOTE MARKER	ـــ؂	Subtending marks	Cf	Prepend	Format
0603	SIGN SAFHA	ـــ؃	Subtending marks	Cf	Prepend	Format
0604	SIGN SAMVAT	ـــ؄	Subtending marks	Cf	Prepend	Format
0605	NUMBER MARK ABOVE	ـ؅	Supertending mark	Cf	Prepend	Format
061C	LETTER MARK		Format character	Cf	Control	Format
0664	DIGIT FOUR	٤	digits	Nd	Other	Numeric
0666	DIGIT SIX	٦	digits	Nd	Other	Numeric
06DD	END OF AYAH	◌۝	Quranic annotation sign	Cf	Prepend	Format

U+061C U+0600 U+0664 U+0666 = ALM, NUMBER SIGN, DIGIT FOUR, DIGIT SIX

5.3 Proposal to Update Properties for Two Khmer Characters

L2/23-018 from Steven Loomis

Recommended UTC actions

1. Consensus: Add U+17D4 KHMER SIGN KHAN & U+17D5 KHMER SIGN BARIYOOSAN to Sentence_Terminal and Sentence_Break=STerm, for Unicode 15.1.
2. Action Item for Josh Hadley, PAG: Add U+17D4 KHMER SIGN KHAN & U+17D5 KHMER SIGN BARIYOOSAN to Sentence_Terminal and Sentence_Break=STerm, for Unicode 15.1.

Summary

Add U+17D4 ។ KHMER SIGN KHAN & U+17D5 ៕ KHMER SIGN BARIYOOSAN to Sentence_Terminal and Sentence_Break=STerm.

Background information / discussion

These two characters have Line_Break=Break_After.

5.4 Multiple notes of support for line-breaking at orthographic syllable boundaries

PRI #472 “Line breaking at orthographic syllable boundaries”

Recommended UTC actions

1. No action. The UTC recognizes and appreciates the review feedback in favor of the proposal.

Summary

Several one-line (non-substantive) notes of support were received for the proposal to introduce line breaking at orthographic syllable boundaries.

5.5 Grapheme clusters for Indic scripts

Recommended UTC actions

1. Consensus: Modify Indic grapheme clusters as described in L2/23-079 item 5.5, for Unicode 15.1.
2. Action Item for Mark Davis, PAG: In UAX #29, add the three macros Virama, LinkingConsonant, and ExtCccZwj, as well as the new rule 9.c, as described in L2/23-079 item 5.5, for Unicode 15.1.

Feedback

From Mark Davis:

We held back on changes to the grapheme clusters to Indic scripts, and said they should first go into CLDR.

It has been 4 years since these were deployed in CLDR & ICU (which probably account for the majority of end-users affected by grapheme clusters) and there are no objections.

I propose that we make the corresponding additions to UTS #29, namely:

1. Adding 3 new macros to https://unicode.org/reports/tr29/#Grapheme_Cluster_Break_Property_Values
   a. `Virama=[\p{Gujr}\p{sc=Telu}\p{sc=Mlym}\p{sc=Orya}\p{sc=Beng}\p{sc=Deva}&\p{Indic_Syllabic_Category=Virama}]`
   b. `LinkingConsonant=[\p{Gujr}\p{sc=Telu}\p{sc=Mlym}\p{sc=Orya}\p{sc=Beng}\p{sc=Deva}&\p{Indic_Syllabic_Category=Consonant}]`
   c. `ExtCccZwj=[\p{gcb=Extend}-\p{ccc=0}] \p{gcb=ZWJ}]`

It is not necessary for the macros to have disjoint categories.
The list of scripts can be added to over time, as test files for them become available.

2. Adding a new rule:
   9.c `LinkingConsonant ExtCccZwj* Virama ExtCccZwj* × LinkingConsonant`

5.6 Proposed changes for line breaking on orthographic syllables

L2/23-072 from Robin Leroy

based on earlier feedback on

PRI #472 “Line breaking at orthographic syllable boundaries”

Recommended UTC actions

1. Consensus: Add line breaking classes AF, AK, AP, AS, VI, and VF, as well as a new line breaking rule LB 28b, and change Line_Break property values, as described in L2/23-072.
2. Action Item for Robin Leroy, PAG: Incorporate the changes to UAX #14 described in L2/23-072 into the Proposed Update. For Unicode Version 15.1.
3. Action Item for Robin Leroy, PAG: Incorporate the changes to UAX #29 described in L2/23-072 into the Proposed Update. For Unicode Version 15.1.
4. Action Item for Norbert Lindenberg, PAG: Provide an updated description for line breaking class BA, classifying the additions to that class from L2/23-072. For Unicode Version 15.1.
5. Action Item for Robin Leroy, PAG: Provide an updated description for line breaking class GL, conveying that characters such as hieroglyphic joiners and the Brahmi number joiner are included in this class. For Unicode Version 15.1.
6. Action Item for Robin Leroy, PAG: Update LineBreak.txt and PropertyValueAliases.txt as described in L2/23-072. For Unicode Version 15.1.

Feedback (verbatim)

Date/Time: Tue Mar 07 04:09:12 CST 2023
Name: Robin Leroy
Report Type: Public Review Issue
Opt Subject: 472

The proposal L2/22-080R2 affects a number of Brahmic scripts, for which it appears to be a clear improvement.
However, its effect is not quite restricted to these scripts, as it changes line breaking class of the Common character ◌ (U+25CC DOTTED CIRCLE).
This introduces line break opportunities, for instance, between a letter and a dotted circle, or between two dotted circles.

See, for instance, a◌̀ and e◌̂◌̣ on the demo:
https://www.unicode.org/review/pri472/background.html?text=a%E2%97%8C%CC%80%0Ae%E2%97%8C%CC%82%E2%97%8C%CC%A3.

Such usage of the dotted circle is attested to describe a sequence of combining marks; see the comments in http://www.unicode.org/Public/UCD/latest/ucd/NormalizationTest.txt.
While the use cases to which this change would be disruptive may be niche, any usage of the dotted circle is niche, including the one motivating the change.

Refinement of the behaviour of the dotted circle could be relegated to a dedicated proposal.
However, absent that change in line breaking class, the proposal would degrade the behaviour of sequences (dotted circle, virama) in the affected scripts.

Instead, replacing `AK | AS by AK | AL | AS` in the first three sub-rules of the proposed rule LB28b would preserve the usability of the dotted circle as a placeholder base for

1. pre-base consonants: `AP × (AK | AL | AS)` handles AP × ◌;
2. virama: `(AK | AL | AS) × (VF | VI)` handles ◌ × (VF | VI);
3. conjunct consonants: `(AK | AL | AS) VI × AK` handles ◌ VI × AK.

At the same time, this would not affect the behaviour of class AL except in degenerate cases (cross-script virama or pre-base consonant usage).

This would not support the usage of the dotted circle itself as a subjoined consonant to demonstrate the forms of combining marks or conjuncts thereon, as cited and demonstrated in Section “Enabling the use of dotted circle as a placeholder for subjoined consonants” of L2/22-080R2: there would be a break in AK × VI ÷ ◌ × CM.

Changing sub-rule 3 of LB28b to `(AK | AL | AS) VI × (AK | AL)` may have unwanted effects in nondegenerate mixed-script cases.
Since both the cited example and the one shown in the proposal itself only subjoin the dotted circle to another dotted circle, an additional sub-rule
5. AL VI × AL
would address the use cases shown while not disrupting nondegenerate cases.

Note: in the above, the term *degenerate* is used in the sense defined in UAX #29.

5.7 clear statement that each emoji is a grapheme cluster

Recommended UTC actions

1. Action Item for Josh Hadley, PAG: In Unicode Standard Annex #29, add a note to the Grapheme Cluster Boundary rules stating that each emoji sequence (UTS #51 ED-17) is a single grapheme cluster. For Unicode Version 15.1.
2. Action Item for Mark Davis, ESC: In Unicode Technical Standard #51, update the section on emoji ZWJ sequences to state that an emoji sequence is a single grapheme cluster, with a reference to UAX #29, and reword the section on emoji modifier sequences accordingly. For Unicode Version 15.1.
3. Action item for Manish Goregaokar, PAG: Provide a proposal for changes to Unicode Standard Annex #29 and Unicode Technical Standard #51 highlighting that, depending on fonts and rendering engines, some grapheme clusters can be rendered as multiple glyphs, which are perceived as separate units by the user; and some single code points can appear to be multiple grapheme clusters. For Unicode Version 16.0.

Feedback

From Markus Scherer:

Someone asked whether each emoji is a single grapheme cluster.
One could look at all possible, well-formed emoji sequences and compare them with the grapheme cluster properties and rules and conclude that this is the case.
However, the rules are complex enough to make this not obvious.

Please add clear, simple statements to each of UAX #29 and UTS #51 to the effect of "each emoji is a single grapheme cluster".

6. IDNA

6.1 UTS #46: declare the transition period to be over

Recommended UTC actions

1. Consensus: Change UTS #46 to say that the transitional processing and the deviation mappings are deprecated, and that implementations generally only use the nontransitional processing, for Unicode 15.1.
2. Action Item for Markus Scherer, PAG: Change UTS #46 to say that the transitional processing and the deviation mappings are deprecated, and that implementations generally only use the nontransitional processing, for Unicode 15.1.

Feedback (verbatim)

Date/Time: Mon Jan 23 04:59:25 CST 2023
Name: Anne van Kesteren
Report Type: Error Report
Opt Subject: UTS46

Chromium will ship Nontransitional Processing soon: https://chromestatus.com/feature/5105856067141632. That covers all browser engines. I suggest taking that opportunity to simplify this document and its test suite and declare the transition period for which this conditional existed to be over.

6.2 UTS #46: change U+2260 (≠), U+226E (≮), and U+226F (≯) from disallowed_STD3_valid to valid

Recommended UTC actions

1. Consensus: In IdnaMappingTable.txt, change U+2260 (≠), U+226E (≮), and U+226F (≯) from disallowed_STD3_valid to valid, for Unicode 15.1.
2. Action Item for Mark Davis, Markus Scherer, PAG: In IdnaMappingTable.txt, change U+2260 (≠), U+226E (≮), and U+226F (≯) from disallowed_STD3_valid to valid, for Unicode 15.1.
3. Action Item for Mark Davis, Markus Scherer, PAG: In UTS46 section 4.1.1 UseSTD3ASCIIRules, remove the special behavior of U+2260 (≠), U+226E (≮), and U+226F (≯); modify section 6 Mapping Table Derivation (especially Step 7) as necessary so that these characters are no longer disallowed; for Unicode 15.1.

Feedback (verbatim)

Date/Time: Mon Jan 23 05:13:16 CST 2023
Name: Anne van Kesteren
Report Type: Error Report
Opt Subject: UTS46

Please change U+2260 (≠), U+226E (≮), and U+226F (≯) from disallowed_STD3_valid to valid.

These code points are not decomposed so they can never conflict with =, <, and >. And they are not inherently more confusing than any of the other allowed code points, which include hieroglyphics and emoji. These code points also work as-is in all browser engines (while < and > are forbidden) and on balance preference ought to be given to retaining compatibility so end users are not prevented from visiting websites or seeing subresources that might use these code points in their domain for one reason or another.

For further background and discussion please see whatwg/url#733.

Thank you!

6.3 IdnaTestV2.txt issues mostly with status annotation

Recommended UTC actions

1. Action Item for Mark Davis, PAG: Review IdnaTestV2.txt and ensure that the documentation and output of status codes conforms to the specification; see (doc & item) for a report of mismatches; for Unicode 15.1.

Feedback (verbatim)

Date/Time: Mon Jan 23 06:35:46 CST 2023
Name: Anne van Kesteren
Report Type: Error Report
Opt Subject: IdnaTestV2.txt

I have worked on importing IdnaTestV2.txt into web-platform-tests, the test framework used by all web browsers. The goal was to meet the requirements of the domain to ASCII algorithm specified at https://url.spec.whatwg.org/#idna with beStrict initialized to false.

As such, I attempted to filter out ToASCII statuses for UseSTD3ASCIIRules, CheckHyphens, and VerifyDnsLength. Hoping that any statuses that are left would indicate a failure requirement.

You can find my work at web-platform-tests/wpt#38080.

I ran into the following issues. Most of them relate to status annotation. IPv4 address confusion was the one issue that did not relate to statuses.

- VerifyDnsLength is not P4, but rather A4_1 and A4_2.
- Tests that use trailing ASCII digit labels (or such a label followed by a dot) are not useful for browsers as that will trigger the IPv4 parser.

Which will then usually return failure as the input was not actually an IPv4 address string. This is a problem for a number of the A4_1 and A4_2 tests. And also a large number of tests later on, such as ToASCII ("xn--gl0as212a.8.") or ToASCII("1.27"). I wrote a filter to exclude them, but it would be better if they were adjusted slightly (e.g., made to contain one non-EN code point) so what they aim to test can also be tested in browsers. (Note that the IPv4 parser runs after domain to ASCII, but the web platform doesn't provide a way to invoke domain to ASCII on its own and probably never will.)

- The test for ToASCII("$") is marked P1 and V6, not U1. This also effects numerous tests with <, >, and =. If they continue to have multiple statuses that will also make it impossible to filter them in an automated fashion. (This also applies to non-ASCII UseSTD3ASCIIRules code points, but I filed a separate request to remove those.)
- NV8 is not used as a status.
- A3 and X3 do not appear to be used as a status. (These are catered for by P4 presumably.)
- CheckBidi is not V8. V8 does not appear to be used. You'd have to filter out all B1-6 statuses instead.

Background information / discussion

https://www.unicode.org/Public/idna/15.0.0/IdnaTestV2.txt

7. Regex

7.1 Proposed Update for UTS #18

Recommended UTC actions

1. Consensus: The UTC authorizes a proposed update of UTS #18 for the purpose of addressing action items 174-A9, 174-A22, 173-A16, 172-A87, 172-A104, 170-A17, 168-A13, and 166-A70, and 162-A23 if appropriate.

Feedback

From Robin Leroy:

There are a number of open action items targeting UTS18, namely: 174-A9, 174-A22, 173-A16, 173-A74, 172-A87, 172-A104, 170-A17, 168-A13, 166-A70, 162-A23.
Many of these are requests for specific edits that could be done when anyone finds some time; but there is no proposed update, which means a separate consolidated proposal would have to be submitted instead.
We should have a proposed update to help us get through that backlog.

It is unclear whether 162-A23 is still relevant; that one is fairly nonspecific so it should probably have its own proposal (or at least discussion in the background section of a PAG report). The others seem clear enough.

8. Security

8.1 UTS39: Confusables: Letter coptic vida 'B'

PRI #463 Proposed Update UTS #39, Unicode Security Mechanisms

Recommended UTC actions

1. Action Item for Mark Davis, PAG: Consider the feedback in L2/23-079 item 8.1 for a future update of the confusables data.

Feedback (verbatim)

Date/Time: Sun Mar 12 07:37:17 CDT 2023
Name: Ray
Report Type: Error Report
Opt Subject: Confusables Sheet

Letter coptic vedi 'B' is not listed as confusable. It is in fact confusable with latin B and its lookalike characters. This leads to unicode text normalization errors where presense of these characters fail to reduce the unicoded string to its ASCII form.

8.2 Add bidi URL display order recommendations

Recommended UTC actions

1. Action Item for Robin Leroy, PAG: add an example of the application of protocol HL4 to URL display to the Proposed Update for Unicode Standard Annex #9, Unicode Bidirectional Algorithm. See L2/23-079 item 8.2. For Unicode Version 15.1.
2. Action Item for Mark Davis, PAG: In a future revision of Unicode Technical Report #36, add a mention of the applicability of the UTS #55 Basic Ordering to URLs and a reference to the example in UAX #9. See L2/23-079 item 8.2.

Feedback:

Regarding PRI #185 “Extension of UBA for improved display of URL/IRIs” (2011)

Robin Leroy writes:

As this had been brought up in another place, Mark mentioned it in the SCWG, thinking it was similar albeit off-topic. The SCWG noted that this it is on topic, and a special case of the UTS #55 Basic Ordering.

However, it seems unlikely that anyone thinking about URLs would look at *Unicode Source Code Handling*, so this should be referenced in a more appropriate place; either UAX #9 or UTR #36.

Mark proposed the text below for addition to UAX# 9.

HL4 Example 2 for URLs

When a URL is displayed simply using the BIDI algorithm, the following results (as per convention, uppercase represents RTL letters)

Environment	Display
LTR	http://ab.cd.com/mn/op http://ab.cd.**HG.FE.**com/**LK/JI/**mn/op http://**LK/JI/HG.FE**
RTL	http://ab.cd.com/mn/op mn/op/**LK/JI**/com**.HG.FE**.http://ab.cd **LK/JI/HG.FE**//:http

Note that the various fields of the URL can appear to the user in an jumbled order. Moreover, if any of the fields contain mixed bidi text (including digits), part of the contents of a field may flip around a delimiter, as in the following:

Memory Positions

Memory pos.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Character	/	o	א	ב	1	a	b	2	/	3	c	d	4	ו	ד	5	/

Display Positions

Display pos.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Memory pos.	16	15	14	13	4	5	6	7	8	9	10	11	12	3	2	1	0
Character	/	5	ד	ו	1	a	b	2	/	3	c	d	4	ב	א	o	/

The BIDI display process described in *Section 4.1* *Bidirectional Ordering* of [UTS55] can be applied to URLs to remedy this situation.

In applying the rules of that section, the atoms are the delimiters and the text between them (aka literals). Those delimiters include the characters that separate the scheme, host, path, query, and fragment, plus the delimiters within each of those parts. For example:

http	:	//	foo	.	com	/	dir1	/	dir2	?	hl	=	fr	&	rl	=	CA	#	fii

The atoms are then displayed in monotonic order (RTL or LTR), and each literal is displayed with a paragraph direction equal to that monotonic order. This results in the following orders:

Environment	**Display**
LTR	http://ab.cd.com/mn/op http://ab.cd.**FE.HG.**com/**JI/LK/**mn/op http://**FE.HG/JI/LK**
RTL	op/mn/com.cd.ab//:http op/mn**/LK/JI**/com**.HG.FE**.cd.ab//:http **LK/JI/HG.FE**//:http

Background information / discussion

Asmus Freytag requests that UAX #9 or maybe UTR #36 explicitly states that it is not in contention with RFC 5893.

8.3 SCWG update

Recommended UTC actions

1. Note: The UTC notes the amendments to Draft Unicode Technical Standard #55, Unicode Source Code Handling, mentioned in L2/23-079 item 8.3.

Summary

The SCWG does not plan to produce a separate report document this time around, as there have been changes to only one document (UTS #55). However, the UTC should take note of these changes.

The modifications since UTC #174 looked at it are:

- Advanced from Proposed Draft to Draft Unicode Technical Standard.
- Addressed comments made at UTC #174.
- Added a note highlighting the possibility of tailoring confusable data to the font in programming environments where the font is known.
- Clarified that atom order should be a user preference, not a heuristic.
- Clarified the implications of the basic ordering on the content of string literals, and added an implementation permission for an override.
- Clarified that the General Security Profile excludes letters that are confusable with ASCII punctuation or symbols.
- Added a section describing the compatibility considerations when migrating from Unicode 3.0 identifier definitions.