UTC #171 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2022-apr-18

Properties & algorithms

We are a group of Unicode contributors who take an interest in properties and algorithms.
We look at relevant feedback reports and documents that Unicode receives, do some research, and submit UTC documents with recommendations as input to UTC meetings. Since 2021q4 we are responsible for developing and maintaining UCD/UCA/idna/security data (not Unihan).

Participants

The following people have contributed to this document:

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

Public feedback

Feedback received via the Unicode reporting form, see L2/22-063 “Comments on Public Review Issues (Jan 18, 2022 - April 11, 2022)”.

F1: TR31 Security Bugs (UCD Versions 1-14)

Recommended UTC actions

1. Action Item for Mark Davis, Source Code Working Group: Review feedback from Reini Urban on “TR31” [Wed Jan 19 03:46:02 CST 2022] and provide a recommendation. See L2/22-064 item F1.

Feedback (verbatim)

Date/Time: Wed Jan 19 03:46:02 CST 2022
Name: Reini Urban
Report Type: Error Report
Opt Subject: tr31-latest

TR31 Security Bugs (UCD Versions 1-14)
==

1. U+FF00..U+FFEF not as ID

Most of the U+FF00..U+FFEF Full and Halfwidth letters have incorrectly
`ID_Start` resp. `ID_Continue` properties. XID ditto.

They should not, because they are confusable with the normal
characters in the base planes. E.g. LATIN A-Z are indistuingishable
from A..Z, LATIN a-z from ａ..ｚ, likewise for the Katakana ｦ..ｯ and
ｱ..ﾝ, and the Hangul ..ﾾ, ￂ..ￇ, ￊ..ￏ, ￒ..ￗ ￚ..ￜ halfwidth letters.

This is esp. for TR39 a security risk. TR39 provides Identifier Type
properties to exclude insecure identifiers, but I cannot find any
other type property to set these U+FF21..U+FFDC IDs to, than
\`Not_XID\`. Thus the \`ID_Start\`/\`ID_Continue\` property should be
deleted for all of them. If they are not identifiable, they should not
be marked as such.
Since XID's are guaranteed stable and nobody cares yet about TR39, I
would accept a new TR39 Identifier Type property Confusable, or just
set the Not_XID property there for these.
But really, defects, esp. security defects should be fixed.

2. Medial letters in \`ID_Start\`, not \`ID_Continue\`
---

DerivedCoreProperties lists all of the Arabic and Thai MEDIAL letters,
which are part of identifiers in \`ID_Start\`, not in \`ID Continue\`. Only
the Combining marks are in \`ID_Continue\`. Thus all unicode-aware
parsers accept all MEDIAL letters incorrectly in the start
position. They should only be allowed in the \`ID_Continue\` position,
and parsers should disallow them in the end positions for identifiers.

All the other medial letters (Myanmar, Canadian Aboriginal, Ahom,
Dives Akuru) are not part of Recommended Scripts, so they do not
affect TR39 security. But since almost nobody but Java, cperl and Rust
honor TR39 it's still affecting most parsers.

Other medial exceptions are noted in TR31 at 2.4 Specific Character
Adjustments, but the tables DerivedCoreProperties and TR39 Identifier
tables and thus all user parsers are wrong.
<https://www.unicode.org/reports/tr31/#Specific_Character_Adjustments>

Background information / discussion

The interactions between #31 and #39 are complicated, and more extensive work is needed to clarify them.
https://www.unicode.org/reports/tr31/#Stability
https://www.unicode.org/policies/stability_policy.html#Property_Value

Example: “Once a character is ID_Start, it must continue to be so in all future versions.”

https://www.unicode.org/reports/tr39/#Identifier_Status_and_Type

Not_XID: Characters that do not qualify as default Unicode identifiers; that is, they do not have the Unicode property *XID_Continue=True*.

New Identifier_Type=Confusable ?? — could have Confusable_With_ASCII?
Ken: Adds to confusion; we have decent data here.

Asmus: Confusable_With_ASCII and Confusable_With_Punctuation somewhere would be useful.
Ken: That belongs in UTS #39.

Arabic MEDIAL presentation forms are Identifier_Type=Not_NFKC:
https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5B%3Asc%3DArab%3A%5D%26%5B%3Ana%3D%2FMEDIAL%2F%3A%5D&g=Identifier_Type&i=
… and thus Identifier_Status=Restricted.

Make it clear in #31 that for security reasons people need to look at the mechanisms in #39; specifically, Identifier_Type & Identifier_Status.
Improve 31 pointing more to 39 "for reducing the attack surface".
In 39, consider additional Identifier_Type values like Confusable_With_ASCII (at least letters & digits) and Confusable_With_Punctuation (e.g., letter looks like apostrophe).
FYI: Current draft proposed update for UAX #31: https://www.unicode.org/reports/tr31/tr31-36d1.html

F2: Typos in UAX #24 & UAX #31

Recommended UTC actions

1. No action: These typos have been corrected.

Feedback (verbatim)

Date/Time: Thu Jan 27 15:49:01 CST 2022
Name: Ivan Panchenko
Report Type: Error Report
Opt Subject: UAX #24 and UAX #31

UAX #24 contains the mistakes "GREEK LETTER SMALL LETTER OMICRON" (instead of "GREEK SMALL LETTER OMICRON") and "in provided in" (instead of "is provided in"). A period is missing after "can be classified by script".

UAX #31 contains "an definition" (instead of "a definition") and possibly some misplaced spaces (search for " ," and " .").

F3: Inconsistent "bottom-shaded" character names

Recommended UTC actions

1. No action.

Feedback (verbatim)

Date/Time: Sat Feb 19 16:32:04 CST 2022
Name: Karl Williamson
Report Type: Error Report
Opt Subject: UCD

U+1F8A0: LEFTWARDS BOTTOM-SHADED WHITE ARROW
U+1F8A1: RIGHTWARDS BOTTOM SHADED WHITE ARROW

While not technically an error, the names of these symmetric characters are asymmetric. One has a HYPHEN-MINUs between BOTTOM and SHADED and the other has a SPACE. It would be helpful to add a Name Alias to one or the other

Background information / discussion

In loose character name matching, the medial hyphen is ignored:
https://www.unicode.org/reports/tr44/#Matching_Names
Therefore, *on input*, this inconsistency should not matter.

Adjacent characters also have inconsistent spellings with “TOP SHADED” and “LEFT-SHADED”.

F4: Case Charts

Recommended UTC actions

1. Action Item for Mark Davis and the PAG: Check the case chart generator for how U+0130 is processed, and fix if necessary; see L2/22-064 item F4 for Unicode 15.0 if possible.

Feedback (verbatim)

Date/Time: Thu Feb 24 03:26:53 CST 2022
Name: Martin J. Dürst
Report Type: Website Problem
Opt Subject: Case Charts

In the case charts at https://www.unicode.org/charts/case/,
together with Yusuke Endoh, a fellow Ruby committer, I discovered a
problem: It lists the lowercased version of U+0130 (İ) as U+0069 (i).
This is a simple case mapping, the full case mapping is U+0069 U+0307
at https://www.unicode.org/Public/UCD/latest/ucd/SpecialCasing.txt,
line 69. The case charts don't say anything about simple vs. full case
mappings, they should say something (it's unclear for me at the moment
exactly what they should say, because it's unclear to me exactly what they do).

Background information / discussion

https://www.unicode.org/charts/case/help.html
Rows like for U+00DF “ß” show the full case mapping.

U+0130 has multiple entries in SpecialCasing.txt. One is unconditional, and is what Martin and Yusuke expected to see. The others are language-specific. Maybe the chart generator ignores characters with conditional mappings, and accidentally discards the unconditional mapping for U+0130.

F5: Use of CANCEL TAG in emoji flags

Recommended UTC actions

1. Action Item for EdComm: Revise the core spec, section 23.9, based on the proposed replacement text in section F5 of L2/22-064, for Unicode 15.0.

Proposed replacement text for the entire section in https://www.unicode.org/versions/Unicode14.0.0/ch23.pdf:

Tag Characters: U+E0000–U+E007F

This block encodes a set of 97 special-use tag characters to enable the construction of tags using~~spelling out of ASCII-based string tags using~~ characters that can be strictly separated from ordinary text content characters in Unicode but that correspond to ASCII-based strings. ~~These tag characters can be embedded by protocols into plain text.~~ They can be identified and/or ignored by implementations with trivial algorithms because there is no overloading of usage for these tag characters—they can express only tag values and never textual content itself. ~~In addition to these 95 characters, two other characters are encoded: O~~One of these 97 characters is the deprecated language tag identification character~~ and one cancel tag character~~.

The current conformant use of the other 96 tag characters is specified in *UTS #51 Unicode Emoji* [UTS51]. See *ED-14a. emoji tag sequence (ETS)* and *Annex C: Valid Emoji Tag Sequences*. UTS #51 does not use nested tag sequences and is thus not stateful.

Deprecated Use for Language Tagging

The original intended use for the tags was for language tagging of plain text. That was only provided to help avoid a far worse external proposal to use malformed UTF-8 for language tagging. It became clear that language tagging using these characters was complicated and unnecessary, and the characters were deprecated in Unicode 5.1. In Unicode 8.0, most characters were de-deprecated, and repurposed for use with emoji.

To see the description of the original usage, see the corresponding chapter in Unicode 5.0.

~~The language tag identification character identifies a tag string as a language tag; the language tag itself makes use of BCP 47 language tag strings spelled out using the tag characters from this block. This character and the associated mechanism for language tagging are…~~
[Ed Note: Delete everything to the end of this section]

Feedback (verbatim)

Date/Time: Sun Jun 27 12:38:08 CDT 2021
Name: Alexei Chimendez
Report Type: Error Report
Opt Subject: Use of CANCEL TAG in emoji flags

UTS #51 allows for the interchange of various flags through "emoji tag sequences", specified as: an emoji character or sequence, followed by one or more component characters from the block Tags, and terminated with the character CANCEL TAG.

In the Unicode Standard, sec. 23.9 reads:

> There are two uses of cancel tag. To cancel a tag value of a particular type, prefix the cancel tag character with the tag identification character of the appropriate type. [...] To cancel any tag values of any type that may be in effect, use cancel tag without a prefixed tag identification character.

Continuing, it specifies:

> Inserting a bare cancel tag in places where only the language tag needs to be canceled could lead to unanticipated side effects if this text were to be inserted in the future into a text that supports more than one tag type.

However, the use of CANCEL TAG in flags is, in effect, a "bare cancel tag", because it is not preceded by a tag identification character (it is only preceded by tag component characters). The presence of an emoji flag in a text may thus inadvertently cause the canceling of all applicable tags.

While the Standard currently only specifies one kind of tag (the language tag, which is "strongly discouraged"), the use of CANCEL TAG in emoji flags may cause issues if other kinds of tags are introduced in the future, or for applications or protocols that make use of "private use" tags to signal in-band information.

The simplest solution is to change the wording in sec. 23.9 to read:

> To cancel any tag values of any type that may be in effect, use cancel tag without a prefixed tag identification character or other tag character.

With this change, the CANCEL TAG character in the sequence

> U+1F3F4 U+E0066 U+E006F U+E006F U+E007F

has no effect and is ignored, while in the sequence

> U+1F3F4 U+66 U+6F U+6F U+E007F

the CANCEL TAG character will cancel all tags. This change prevents the inadvertent canceling behavior of emoji tag sequences as described above.

Background information / discussion

The Editorial Committee has considered this feedback from last year and decided that it has technical implications, and has a bearing on the actual interpretation of a CANCEL TAG in the formal syntax of tag sequences, in cases where more than one kind of tag might interact. See the Editorial Committee report L2/21-127, item F1. The UTC then asked the properties & algorithms group via action item [168-A30] to consider this feedback.

The Unicode Standard (latest version), section 23.9, "Tag Characters"

[Ned]: While it is true that an emoji tag sequence will cancel any current language tag, considering that the language tag type is deprecated I don't see how it makes any difference in the absence of any new tag types (which I would be surprised to see from UTC). If we care to address this at all, I would suggest formalizing emoji tag sequences, perhaps by saying that an emoji tag base followed by one or more tag characters is considered an emoji tag identification *sequence* and let the CANCEL TAG apply to that. Agree with Mark that there's little value in defining how CANCEL TAG behaves with multiple tag types when there is only one such type and it is deprecated, and I would rather nuke the section than formally squeeze emoji into it.

[Mark] Given that the use of the tag sequences for representing flags is deprecated, and the standard doesn't actually mention the *real* use of the tag sequences (for emoji), I think this section needs to be mostly removed. I'd recommend:

- Remove everything from "**Deprecated Use for Language Tagging**" down
- Substitute something like the following:

> Tag sequences are used in defining certain Emoji constructs. For details see UTS #51 Unicode Emoji. [add ref]
>
> They were originally designed for use in language tags, to forestall use of non-conformant UTF-8 sequences, but that usage has been deprecated.

[Markus] I like Alexei's suggestion that the cancel tag be ignored (and not cancel anything) if it is preceded by a tag character that is not a tag identification character; that is, if it is preceded by any U+E0020..U+E007E. Small change, easy test.
The only incompatibility is that if you had text with a language tag immediately followed by the cancel tag, then the language tag was terminated but now no longer will be.
We could avoid this almost as easily by making the test slightly more complicated: Ignore the cancel tag if it is immediately after a sequence of one or more tag characters that does not include a tag identification character.

Mark: No known implementation of tag sequences. Our users are better off if we remove the section from the core spec, replace with "only used in UTS #51…". Could point to the last version that had the text for the deprecated structure.

Asmus: Weary of saying "no known implementation" when we can't be sure. — Mark: Maybe, but vanishingly small.

Asmus: Maybe ok. But then say that the cancel tag will never be used for anything but emoji.

F6: Vai line breaking vs. L2/22-080

Recommended UTC actions

1. Action Item for SAH: Review feedback on Vai line breaking [Mon Mar 28 18:33:50 CDT 2022] and provide a recommendation.

Feedback (verbatim)

Date/Time: Mon Mar 28 18:33:50 CDT 2022
Name: David Corbett
Report Type: Other Document Submission
Opt Subject: Vai line breaking

This is feedback on L2/22-080. Another script with line breaks between orthographic
syllables is Vai. The description in chapter 19 indicates that most Vai letters
should have lb=ID, and U+A60B and U+A60C should have lb=BA. The “h-” characters might be ID or BA.

Background information / discussion

L2/22-080 “Line breaking at orthographic syllable boundaries” by Norbert Lindenberg, et al

Documents

D1: Glyph mirroring: NonBidiMirroring.txt

L2/22-026 from Kent Karlsson

Recommended UTC actions

1. Action Item for Ken Whistler, EdComm: Review proposed revisions to annotations mentioned in L2/22-026.

Summary

Proposed new data file NonBidiMirroring.txt

… make a data file similar to BidiMirroring.txt, but for symbols that have the *bidiMirrored=No* property value and have a mirror character. Mostly for arrows and arrow-like symbols, but also other symbols. Note that various arrows are commonly used in math expressions. And in editing math expressions or other text, one may (for whatever reason, error fixing, swapping the arguments, “no, you should go right, not left” symbolised with an arrow in the text, …) want to mirror also symbols that have the *bidiMirrored=No* property value.

Both *BidiMirroring.txt* and *NonBidiMirroring.txt* can be used by typeface foundries, or even typeface editing tools, be used to make consistently looking mirror glyphs for mirror character pairs.

Also: Proposed removal of an annotation for 223D REVERSED TILDE.

Also: Proposed removal of an annotation (and maybe something else) for 2205 EMPTY SET.

Background information / discussion

NonBidiMirroring.txt: Unclear use cases. Seems to be proposed for speculative uses.

D2: Bidi in programming languages and markup languages

L2/22-028 from Kent Karlsson

Recommended UTC actions

1. No action: We have notified the Source Code Working Group and requested review/discussion of this doc. The WG has done so and provided recommendations. See section D7 in this document.

Summary

We will here focus on how bidi should be handled in IDE editors and to some extent compilers and interpreters that allow bidi characters (or, more precisely, strong RTL characters). Not every programmer uses an IDE, or an editor that “knows” which programming language is being edited. That case is not covered here.

In https://www.unicode.org/L2/L2022/22007-avoiding-spoof.pdf (Avoiding Source Code Spoofing), there are some ideas for how to handle bidi for programming languages, and avoid some possible spoofing, or readability, issues. Here I’ll give my take on some of the issues raised, and my recommended handling of bidi in programming language source code including script languages, but also extended to (data) markup languages (like HTML, XML, JSON) now often also edited via IDEs, particularly for XML and JSON files that are used in systems and edited “at the same time” as some program source related to such XML/JSON files and have some of the same issues as programming language sources

D3: Proposal for amendments to UAX#9 and UAX#31

L2/22-072R from Robin Leroy, Mark Davis, Source code ad hoc working group

Recommended UTC actions

1. Accept the amendments to UAX #9 and UAX #31 proposed in L2/22-072R
2. Action Item for Ken Whistler, Robin Leroy, EdComm: incorporate changes to Proposed Update UAX #9 based on the proposal in document L2/22-072R, for Unicode 15.0
3. Action Item for Mark Davis, Robin Leroy, EdComm: incorporate changes to Proposed Update UAX #31 based on the proposal in document L2/22-072R, for Unicode 15.0.

Summary

The Source Code Working Group recommends amendments to non-normative text in Unicode Standard Annexes #9 and #31.

Background information / discussion

Ken:

> Note that the proposed update draft I have for UAX #9 is already responsive to this proposal:
> https://www.unicode.org/reports/tr9/tr9-45d2.html
> with the suggested text change (more or less) for HL4 and a much more extended example worked out.
>
> I'm hoping that Mark can do the same to get a proposed update draft up for UAX #31, with language responsive to the request there, so it can be viewed and evaluated in place. — DONE

D4: Line breaking at orthographic syllable boundaries

L2/22-080 from Norbert Lindenberg, et al

Recommended UTC actions

1. Action Item for Christopher Chapman, PAG: reassess after further refinements

Summary

This document proposes:

- To define the term "orthographic syllable" in section 6.1 of The Unicode Standard
- To introduce a new style of context analysis in line breaking for certain Brahmic scripts, which breaks lines at the boundaries of orthographic syllables
- To update the descriptions of line breaking in The Unicode Standard for several scripts
- To update the Line_Break property to use this new style of line breaking for these scripts
- To update the Line_Break property value of the character U+25CC DOTTED CIRCLE to enable its use as a placeholder for subjoined consonants

Background information / discussion

Mark: This document looks well formed and well thought out. My only concern is whether we are lined up to update ICU and CLDR in the Unicode 15.0 timeframe, which are already pretty booked. If we can find resources for that, great. Otherwise, it might be better to target U16.0.

D5: Proposal for new identifier type values

L2/22-081 from Asmus Freytag & Michel Suignard

Recommended UTC actions

1. Action Item for Mark Davis, Source Code Working Group: Add the characters listed under "Confusable with Punctuation" in document L2/22-081 to Restricted as a part of their re-work of Identifier_Status and Identifier_Type
2. Action Item for Mark Davis, Source Code Working Group: Review the characters listed under "Confusable with ASCII" in document L2/22-081 and provide a recommendation.

Summary

From the doc's Overview section:

This proposal requests the addition of two (non-exclusive) values for Identifier_Type in UTS#39 to cover some scenarios that are being distinguished in existing identifier implementations and might be useful more generally.

It is probably useful to recognize that providing these Identifier_Type values for excluded scripts (and code points) is a pointless exercise. One could go further, and consider that rolling this out across limited_use scripts is likewise something that we could (perhaps should) formally defer at this point; therefore, the explicit examples in this proposal focuses on recommended scripts.

→
- Confusable with Punctuation
- Confusable with ASCII

Background information / discussion

Mark: I think this is a pretty reasonable approach, but there are two issues standing in the way of incorporating it into Unicode 15.0.
1. It does not clearly propose exactly what the new property values are named (long and short names), exactly what the textual changes in UTS #39 would be, and exactly which characters should have the new property values. Until we see what those characters are, we can't assess them in time for the upcoming UTC.
2. The problem I foresee is that the group might end up with a result for Unicode 16.0 that would cause us to need to retract this proposal. That is, we are likely to have some broader changes coming out of the working groups efforts to the Identifier_Types. Ideally this should be subsumed in the work of the Source Code Working Group.

D6: Proposal for an option in UAX #31 to prohibit ZWJ/ZWNJ for identifier security

L2/22-082R from Asmus Freytag & Michel Suignard (with additional input from Mark Davis)

Recommended UTC actions

1. Action Item for Mark Davis, Source Code Working Group: Review the recommendations.

Summary

In response to action item 165-A44:

This document proposes a profile option to be added in UAX #31 and or UTS#49 to prohibit ZWJ/ZWNJ altogether, for better identifier security. This behavior would match the DNS Root Zone, where ZWJ and ZWNJ are strictly prohibited. Similarly strict restrictions are likely appropriate for at least Second Level Domain.

Background information / discussion

Mark: The premise of the document is incorrect; ZWJ/NJ are not permitted in the default identifiers. However, by the fact that two experts (Asmus and Michel) stumbled over this, it became clear in discussions in the Source Code Working Group that the text in both UAX #31 and UTS#39 is not sufficiently clear. Document D8 is the result of that discussion, and addresses the issues raised in D6, as much as it can be for Unicode 15.0.

D7: Status report of the source code ad hoc working group for UTC #171

L2/22-088 from Robin Leroy, Mark Davis, Source code ad hoc working group

Related topics:

- D3 (L2/22-072R)
- D8 (L2/22-087)

Recommended UTC actions

1. Review & discuss the report and its recommendations in UTC #171.

Summary

The source code ad hoc working group was created by consensus 170-C2 of the UTC [...]

Progress report; review & recommendations on recent L2 documents.

D8: Profile Changes in UAX #31 / UTS #39

L2/22-087

Recommended UTC actions

1. Accept the proposals in L2/22-087 for changes to UAX #31 and UTS #39, for Unicode 15.0.
2. Action Item for Mark Davis, Robin Leroy, EdComm: incorporate changes to UAX #31 based on the proposal in section I of document L2/22-087, for Unicode 15.0.
3. Action Item for Mark Davis, Robin Leroy, EdComm: incorporate changes to UTS #39 based on the proposal in section III of document L2/22-087, for Unicode 15.0.
4. Action Item for Mark Davis: In the property files IdentifierStatus.txt and IdentifierType.txt, remove the Joiner_Control characters ZWJ and ZWNJ from Identifier_Type=Inclusion and Identifier_Status=Allowed. For Unicode 15.0.

Summary

This addresses the issues raised in discussions of L2/22-082, insofar as they can be addressed before the Source Code Working Group produces a more comprehensive proposal.

While the source code ad hoc working group has not finished their work, there is agreement that additional changes could be made to UAX #31 and UTS #39 to avoid some problems with identifier profiles, and especially those that allow for ZWJ/ZWNJ.

Background information / discussion

Mark: Asmus to propose an additional paragraph for 2.3 Layout and Format Control Characters; Mark and Robin to incorporate after review.

Public Review Issues

https://www.unicode.org/review/

PRI #438: Proposed Update UAX #44, Unicode Character Database

https://www.unicode.org/review/pri438/

PRI438a: Character name loose matching vs. character name namespace

Recommended UTC actions

1. No further action.

Feedback (verbatim)

Date/Time: Tue Dec 7 15:49:18 CST 2021
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: 438

UAX44-LM2 defines a medial hyphen in terms of “the normative Unicode character name, as published in the Unicode names list”. That means that hyphens in name aliases, named character sequences, and code point labels, which are mentioned later in that section, are not medial hyphens. For example, the alias “LOCKING-SHIFT ONE” does not contain a medial hyphen, and the string “locking shift one” does not match it. This was probably not the intent.

Background information / discussion

This has been fixed and clarified based on this and other recent feedback.
See the updated definition of the “Unicode namespace for character names” (UAX34-D3) and the clarified description of loose character name matching (UAX44-LM2).

PRI #440: Proposed Update UTS #10, Unicode Collation Algorithm

https://www.unicode.org/review/pri440/

PRI440a: UTS #10 typos

Recommended UTC actions

1. No action: Ken Whistler has already fixed these in the current draft.

Feedback (verbatim)

Date/Time: Sat Jan 15 06:48:13 CST 2022
Name: Ivan Panchenko
Report Type: Error Report
Opt Subject: UTS #10 [PRI #440]

UTS #10 contains the following minor mistakes: “An low” (instead of “A low”), “An setting” (instead of “A setting”), “of of” (instead of “of”), “UTS #10, shall” (instead of “UTS #10 shall”), “weight Level 3” (instead of “weight. Level 3”), “sorting, should” (instead of “sorting should”), “Note:The” (instead of “Note: The”), “out-of range” (instead of “out-of-range”), “level level” (instead of “level”), “Case differences (uppercase versus lowercase), are” (instead of “Case differences (uppercase versus lowercase) are”).

Background information / discussion

Ken has already fixed these in https://www.unicode.org/reports/tr10/tr10-46.html

PRI #441: Proposed Update UAX #29, Unicode Text Segmentation

https://www.unicode.org/review/pri441/

PRI441a: PU-UAX29 excludes some Kawi characters from GCB=SpacingMark

Recommended UTC actions

1. No action recommended at this time — PAG requires additional time for review.

Feedback (verbatim)

Date/Time: Tue Mar 29 00:43:30 CDT 2022
Name: Norbert Lindenberg
Report Type: Public Review Issue
Opt Subject: 441

The proposed update for UAX 29 excludes the following Kawi characters from having the Grapheme_Cluster_Break property value SpacingMark:

U+11F03 (◌) KAWI SIGN VISARGA
U+11F34 (◌) KAWI VOWEL SIGN AA
U+11F35 (◌) KAWI VOWEL SIGN ALTERNATE AA
U+11F41 (◌) KAWI SIGN KILLER

Being excluded from having SpacingMark means that they receive the Grapheme_Cluster_Break property value Other. In consequence, these characters do not combine with other characters into extended grapheme clusters; they always form their own separate grapheme clusters.

I don't see any reason in the proposal for Kawi, L2/20-284R, or anywhere else why that should be the case. The purpose of grapheme clusters isn't well defined, but one case where the Unicode Standard recommends using them is in emergency line breaking (see UAX 14, section 3, Introduction). If a line break is introduced before a combining mark of a complex script, fonts or rendering systems commonly insert a dotted circle as a base for that mark, which is undesirable.

The corresponding spacing combining marks in the three most closely related scripts, Javanese, Balinese, and Sundanese, all have the Grapheme_Cluster_Break property value SpacingMark or (in one case, 1B35) Extend. I suggest that Kawi is handled the same way.

Background information / discussion

Proposed update: https://www.unicode.org/reports/tr29/tr29-40.html#SpacingMark

It has been long-standing practice to explicitly exclude post-base spacing combining marks (gc=Mc, InPC=Right) of most SE Asian scripts (lb=SA) from GCB=SpacingMark.

This particular change was made in github.com/unicode-org/unicodetools/pull/181 in order to satisfy a consistency check for the UCD — look for “postbase” in the file changes.

PRI #442: Unicode 15.0.0 Alpha Review

https://www.unicode.org/review/pri442/

PRI442a: KHANDA = ADI SHAKTI

Recommended UTC actions

1. Action item for Ken Whistler, EdComm: Add cross-references in NamesList.txt for KHANDA and ADI SHAKTI, for Unicode 15; see L2/22-064 item PRI442a.

Feedback (verbatim)

Date/Time: Sat Feb 12 14:53:23 CST 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 442

Proposed character U+1FAAF KHANDA is in every aspect identical to existing character U+262C ADI SHAKTI – and deliberately so. I am aware of the reason behind this duplicate encoding, as the ESC no longer wishes to emojify already assigned codepoints.

However, if the concept of character identity still means anything, there should at least be established a formal, immutable relationship between these two identical characters. I propose defining a compatibility decomposition mapping from KHANDA to ADI SHAKTI.

1FAAF;KHANDA;So;0;ON;<compat> 262C;;;;N;;;;;

This will not interfere with U+1FAAF's main use as an emoji – twenty-two other emoji characters already have decomposition mappings without issue, and one even has a case mapping on top of that – but would allow systems to formally recognise that KHANDA is in fact nothing more than a stylistic variant of ADI SHAKTI and subsequently treat them as equivalent for certain purposes such as loose-match searching. If one were to search a document for all instances of "the Khanda symbol", it would make little sense for half of all such instances to be randomly omitted from the results.

While the preliminary code chart glyph for KHANDA looks distinct from ADI SHAKTI (the former being drawn as an outline and the latter as solid), this difference is not actually part of either character's identity but simply a whim of the glyph designer. A font supporting both ADI SHAKTI and KHANDA (and these *are* going to exist) has no reason to make these characters look distinct from one another because they are after all the exact same symbol and only encoded separately due to some technicality involving the new guidelines for emoji submissions.

Therefore, as there is no discernable difference between these two characters, they should be compatibility equivalents of each other.

Background information / discussion

While there are emoji characters (as well as some dingbats etc.) with compatibility decompositions, this is not complete and consistent, and we believe that there is little value in these decompositions.

Mark:

1. There are emoji with compat decompositions, so it is acceptable.

2. However, there are many other characters that lack compat decomposition (including heavy equals) also.

There are no emoji characters that decompose canonically. 22 have decomposition mappings
https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5B%3Aemoji%3A%5D-%5B%3Adt%3Dnone%3A%5D&g=dt&i=

One character is not NFC_Inert
https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5B%3Aemoji%3A%5D-%5B%3Anfc_inert%3A%5D&g=dt&i=

PRI442b: Cf or Mn: EGYPTIAN HIEROGLYPH MIRROR HORIZONTALLY

Recommended UTC actions

1. Comment: PAG reviewed the feedback and the recommendations of SAH. Given the understanding that U+13440 EGYPTIAN HIEROGLYPH MIRROR HORIZONTALLY affects the preceding character, we concur with SAH that this character should be treated as a combining character, and with property changes as proposed by SAH in section 16 of L2/22-068.

Feedback (verbatim)

Date/Time: Mon Feb 28 11:36:43 CST 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 442

Proposed character U+13440 EGYPTIAN HIEROGLYPH MIRROR HORIZONTALLY currently
has general category Cf (Format). A more appropriate value would be Mn
(Nonspacing_Mark).

U+13440 applies only to the hieroglyph immediately preceding it, with
possibly a variation selector for rotation intervening. As such, it behaves
like a combining mark. Its line break property value would also need to
change from GL (Glue) to CM (Combining Mark) accordingly, and its
bidirectional class from L (Left_to_Right) to NSM (Nonspacing_Mark).

PRI442c: Decomposition of MODIFIER LETTER CYRILLIC SMALL BARRED O

Recommended UTC actions

1. No action: This has been fixed for Unicode 15.0 beta.

Feedback (verbatim)

Date/Time: Mon Feb 28 11:52:19 CST 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 442

Proposed character U+1E04E MODIFIER LETTER CYRILLIC SMALL BARRED O currently decomposes to U+04D9 CYRILLIC SMALL LETTER SCHWA, but the correct mapping would be to U+04E9 CYRILLIC SMALL LETTER BARRED O.

Background information / discussion

The proposal proposed the correct decomposition mapping; the draft data differs from that. (Typo?)

Proposal: https://www.unicode.org/L2/L2021/21107-cyrillic-mod.pdf (p. 7) has the following decomp:
1E04E;MODIFIER LETTER CYRILLIC SMALL BARRED O;Lm;0;L;<super> 04E9;;;;N;;;;;

PRI442d: Confusable sequence due to new KHOJKI LETTER SHORT I

Recommended UTC actions

1. No action. See the analysis by Peter Constable in L2/22-083 “Khojki Confusable Sequences” and associated recommendations from SAH (section 22 of L2/22-068).

Feedback (verbatim)

Date/Time: Mon Mar 7 14:37:54 CST 2022
Name: Eduardo Marín Silva
Report Type: Public Review Issue
Opt Subject: 442

With the addition of 11240 KHOJKI LETTER SHORT I, there is now a new confusable sequence: 11202 is now confusable with 11240+1122C. I write this as a reminder to mention this in the relevant documentation.
I would also like to link my recently submitted document reviewing the alpha code charts: https://www.unicode.org/L2/L2022/22056-uni15-alpha-resp.pdf

PRI442e: gc of Ornate Parentheses

Recommended UTC actions

1. No action.

Feedback (verbatim)

Date/Time: Fri Mar 4 13:21:23 CST 2022
Name: Gregg Tavares
Report Type: Error Report
Opt Subject: PropList-15.0.0d2.txt

Is this correct?

```
FD3E        ; Pattern_Syntax # Pe      ORNATE LEFT PARENTHESIS
FD3F        ; Pattern_Syntax # Ps      ORNATE RIGHT PARENTHESIS
```

All the other brackets/parenthesis have Ps for left and Pe for right.

Background information / discussion

These General_Category values are correct. These characters are mostly used in RTL text but are not Bidi_Mirrored.
See "β4 ORNATE LEFT PARENTHESIS should be Ps ?" on pages 5-6 of L2/21-126
See www.unicode.org/versions/Unicode7.0.0/#Migration "Segmentation-related Changes"
See UTC consensus [138-C21]

PRI #446: Proposed Update UAX #14, Unicode Line Breaking Algorithm

https://www.unicode.org/review/pri446/

PRI446a: Descriptions of PO/PR don't match behavior

Recommended UTC actions

1. Action Item for Christopher Chapman, PAG: Review feedback from Charlotte Buff on PRI #446 [Tue Apr 5 07:14:53 CDT 2022] and provide recommendations.

Feedback (verbatim)

Date/Time: Tue Apr 5 07:14:53 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 446

In UAX #14, the descriptions of the line breaking classes Postfix_Numeric
(PO) and Prefix_Numeric (PR) don't match the actual behaviour of the line
breaking algorithm when it comes to the treatment of intervening spaces.

The description of PO states:

»Characters that usually follow a numerical expression may not be separated from preceding numeric characters or preceding closing characters, even if one or more space characters intervene. For example, there is no break opportunity in “(12.00) %”.«

And similarly, the description of PR states:

> »Characters that usually precede a numerical expression may not be separated from following numeric characters or following opening characters, even if a space character intervenes. For example, there is no break opportunity in “$ (100.00)”.«

However, the actual line breaking rules that govern these classes (LB23a, LB24, LB25, LB27) don’t actually contain a special provision for intervening spaces. As a result, the strings given as examples *do* in fact contain line breaking opportunities simply due to rule LB18 (Break after spaces) – before the percent sign in the former and before the opening parenthesis in the latter. This can be confirmed via the use of Unicode’s online utility tool for breaks and segmentation (https://util.unicode.org/UnicodeJsps/breaks.jsp).

Background information / discussion

[PAG discussion 12 Apr 2022]: UAX #14 editor(s) should review and recommend.