UTC #167 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2021-apr-26

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.

This group started with the UCD file and production tool maintainers, and with Markus Scherer as the chair. Several UTC participants have requested and received invitations to join. We discuss via email, shared documents, and sometimes video meetings.

Participants

The following people have contributed to this document:

Markus Scherer (chair), Mark Davis, Asmus Freytag, Christopher Chapman, Ken Whistler, Peter Constable

Public feedback

Feedback received via the Unicode reporting form, see L2/21-068 “Comments on Public Review Issues (January 8, 2021 - April 22, 2021)”.

F1: Contradictory requirements for U+2044 and default ignorable code points

Recommended UTC actions

1. AI for Markus Scherer: Provide small edits for chapters 5 & 23 to clarify that default ignorable code points can have effects on display beside their intended function. Include an example or two, such as fraction slash sequences, and/or sequences with U+0600 ARABIC NUMBER SIGN followed by one or more Arabic digits. For Unicode 14. Reference: L2/21-069 item F1.

Feedback (verbatim)

Date/Time: Mon **Feb 1** 10:58:28 CST 2021
Name: David Corbett
Report Type: Error Report
Opt Subject: Contradictory requirements for U+2044 and default ignorable code points

Chapter 6 says that the **fraction slash** creates fractions only in the environment `\p{Nd}+\u2044\p{Nd}+`. However, chapter 5 says that default

ignorable code points should sometimes be ignored for display, with the example that “U+200B ZERO WIDTH SPACE affects word segmentation, but has no visible display”, and chapter 23 says that outside of a defined variation sequence, “use of a variation selector character does not change the visual appearance of the preceding base character from what it would have had in the absence of the variation selector.” How should these contradictory requirements be resolved? For example, should <digit, variation selector, slash, digit> and <digit, ZWSP, slash, digit> be displayed as fractions or not?

Date/Time: Sat **Apr 24** 13:03:04 CDT 2021
Name: David Corbett
Report Type: Other Question, Problem, or Feedback
Opt Subject: Response to L2/21-069

> David does not include a use case for combinations of fractions with default
> ignorable code points in his submission.

The use case is the slashed zero in fractions. L2/21-069’s recommendation in F1 implies that <zero, VS1, fraction slash, one> should be rendered as <full-sized slashed zero, slash, full-sized one>, but that <one, fraction slash, zero, VS1> may be rendered <numerator one, fraction slash, denominator slashed zero>.

Background information / discussion

https://www.unicode.org/versions/Unicode13.0.0/ch05.pdf

- p. 246 “Format characters also typically have no visible display of their own, but may impact the display of neighboring graphic characters.”
- p. 249 “... However, it is not unusual for format characters and variation selectors to have a visible effect on other characters in their vicinity. … Finally, there are some format characters whose function is not intended to affect display. U+200B zero width space affects word segmentation, but has no visible display.”

https://www.unicode.org/versions/Unicode13.0.0/ch06.pdf p. 271
https://www.unicode.org/versions/Unicode13.0.0/ch23.pdf p. 899

Markus:

- Looks like chapters 5 & 23 do not foresee the use of variation selectors or format controls in fraction sequences.
- The Standard says “have no visible display of their own” but does not promise that default ignorable characters have no effect at all on display or other processing.
    - The Standard gives specific examples of intended effects on display (e.g., for ZWJ, bidi controls, and SHY).
    - We *could* add a more explicit statement about possible effects on display that are not related to the character’s intended function, and in particular clarify the “not intended to affect display” and “not change the visual appearance” statements. The fraction slash display could be an example; a processing example could be collation contraction matching.

- David does not include a use case for combinations of fractions with default ignorable code points in his submission.
    - 2021-apr-26: See David's second submission in response to an earlier version of this document.

Asmus: Reaffirm that the fraction slash spec is a specific code point sequence, and additional characters should not be inserted. Markus: Fraction slash spec is already very precise.

Asmus: Format characters are *intended* to have *some* effect.

Peter: Another example would be U+0600 ARABIC NUMBER SIGN, which is discussed in ch. 9.2 (pg 372) and described as being "followed by a sequence of one or more Arabic digits... The sequence terminates with the occurrence of any non-digit character."

F2: Fix ambiguous statement in #51 Unicode Emoji

Recommended UTC actions

1. AI for Mark Davis and the editorial committee, for UTS #51 version 14:
   Change the phrase:
   This set (i.e. Basic_Emoji) excludes all instances of an emoji component, which are not intended for independent, direct input"…
   to
   This set (i.e. Basic_Emoji) excludes all those instances of an emoji component that are not intended for independent, direct input"…
   and add explanatory text based on the Background information below, including an example
2. AI for Mark Davis: Review differences between RGI_Emoji and emoji-test.txt. Reference: L2/21-069 item F2.

Feedback (verbatim)

A Googler sent the following via email:

1. https://github.com/node-unicode/node-unicode-data/blob/b41fc8e7048376d4c673abeb62a0d039637dd3df/data/13.0.0-emoji-sequences.txt#L282 includes U+1F9B0 in Basic_Emoji, which in turn means \u{1F9B0} is an RGI_Emoji string.
2. However, https://unicode.org/reports/tr51/#def_basic_emoji_set says "This set (i.e. Basic_Emoji) excludes all instances of an emoji component, which are not intended for independent, direct input"…
3. …and https://github.com/node-unicode/node-unicode-data/blob/3d0934638c5028feb54073f418a19f7a61bdb83c/data/13.0.0-emoji.txt#L757 explicitly lists U+1F9B0 as an Emoji_Component.

It sounds like the intention is for standalone emoji components to NOT be RGI_Emoji strings, but the latest emoji-sequences.txt includes U+1F9B0 nonetheless.

For context, this is causing a bug for us on the XXX project. It would help us decide how to proceed if you could answer this question:

Is U+1F9B0 RGI_Emoji or not?
In other words, is there a bug in https://unicode.org/reports/tr51/, or in emoji-sequences.txt? Or did I misunderstand what's going on?

Background information / discussion

Mark replied (and Jennifer agreed):
There are really two kinds of components:

1. The intent is for the skin tones and hair styles to be *displayed* even in isolation, but they should not (typically) be on the keyboard palette. These are included in Basic_Emoji
2. The others should never have an emoji presentation in isolation, but do occur as part of emoji sequences. These are not included.

So here is the breakdown:

EMOJI_COMPONENT && BASIC_EMOJI: [- 🦰-🦳]
https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5B%F0%9F%8F%BB-%F0%9F%8F%BF%F0%9F%A6%B0-%F0%9F%A6%B3%5D
[9 code points: skin tone modifiers + hair components]

EMOJI_COMPONENT -- BASIC_EMOJI: [#*0-9 🇦-🇿-]
https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5B%23*0-9%E2%80%8D%E2%83%A3%EF%B8%8F%F0%9F%87%A6-%F0%9F%87%BF%F3%A0%80%A0-%F3%A0%81%BF%5D&g=&i=
[137 code points: keypad components, regional indicators, tag characters, ZWJ, VS16]

Documents

D1: Clarify guidance for use of a BOM as a UTF-8 encoding signature

L2/21-038 from Tom Honermann

Recommended UTC actions

1. AI for Markus Scherer and the properties & algorithms group: Provide small edits for relevant sections of the core specfication to update guidance for the UTF-8 BOM, given that UTF-8 has become the predominant text encoding and that therefore more tools can assume that text is in UTF-8. In terms of the standard, this effectively constitutes an external protocol. Also, we might add that more tools check for well-formed UTF-8 instead of, or in addition to, the signature byte sequence. For Unicode 14. Reference: L2/21-069 item D1.

Summary

“The Unicode standard is clear that a BOM may be used as an encoding signature for UTF-8 encoded data, but its guidance regarding when a BOM is or is not recommended for such use is not consistently interpreted.

This paper seeks to clarify the guidance offered by the Unicode standard for use of a BOM as an encoding signature and proposes several possible resolutions ranging from removal of existing

guidance to expanding guidance tailored to protocol designers, software developers, and text authors."

Background information / discussion

Markus:

- In my view, the text in the standard continues to be good and correct, and necessarily vague.
- We could update it slightly, given that UTF-8 has become the predominant text encoding and that therefore more tools can assume that text is in UTF-8. In terms of the standard, this effectively constitutes an external protocol. Also, we might add that more tools check for well-formed UTF-8 instead of, or in addition to, the signature byte sequence.
- I disagree with adding a statement like "In situations where text is known to be encoded as UTF-8, a BOM consumes storage space unnecessarily." — because where text is known to be in UTF-8, there is external information, and the signature is dis-recommended anyway.

D2: Stability policy: Property of characters must not become property of strings

L2/21-091 from Mathias Bynens, Markus Scherer, Mark Davis

Recommended UTC actions

1. Recommend to the officers the proposal in L2/21-069 item D2 for a new encoding stability policy regarding the stability of the domains of properties.

Summary

Add a new Unicode Character Encoding Stability Policy (https://www.unicode.org/policies/stability_policy.html).

Summary: A property of characters must not become a property of strings, or vice versa. Applies to normative and informative properties.

Background information / discussion

Important for stability of processing (not breaking assumptions made by implementers).

Important for stability of validation of regular expressions, and other standards using Unicode properties, via property metadata.

Public Review Issues

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

PRI #417: Proposed Update UAX #29, Unicode Text Segmentation

https://www.unicode.org/review/pri417/

PRI417a: Korean word boundary example in UAX#29

Recommended UTC actions

1. AI for Rick McGowan to respond to Steven Luscher along the lines of:
    a. According to our analysis, solely under the UAX #29 rules “나는” should not be separated.
    b. Good word segmentation of CJK and certain other languages requires additional mechanisms, often involving dictionaries. With such an implementation, the separation into “나, 는” may be appropriate.
    c. The point of the example is illustrating the separation between the Latin “Chicago” and the Korean suffix “에”.
    d. Reference: L2/21-069 item P417a.

Feedback (verbatim)

Date/Time: Thu Jan 21 18:02:00 CST 2021
Name: Steven Luscher
Report Type: Error Report
Opt Subject: Korean word boundary example in UAX#29

Hi folks,

In the UAX #29 document (https://www.unicode.org/reports/tr29/#Word_Boundaries) it is written:

> According to Korean standards, the grammatical suffixes, such as “에” meaning “in”,
are considered separate words. Thus the above sentence would be broken into the following five words:
>
> 나, 는, Chicago, 에, and 산다.

A Korean speaking colleague of mine tells me that he, in fact, considers ‘나는’ to be one word.
In Mac OS, placing the cursor to the left of ‘나는’ and pressing Command-RightArrow moves you rightward past both graphemes.

Could the spec be wrong where it claims that 나 and 는 are two words?

Thank you,
Steven…

PRI417b: Unicode Standard Annex #29 - 3 Grapheme Cluster Boundaries - SpacingMark

Recommended UTC actions

1. AI for Markus Scherer and Deborah Anderson: Consider whether to remove the assignment of GCB=SpacingMark for U+11720..11721 AHOM VOWEL SIGN A & AA, in light of feedback and discussions over time. Reference: L2/21-069 item P417b.

Feedback (verbatim)

Date/Time: Fri Jan 29 18:14:29 CST 2021
Contact: johnsoneal@gmail.com
Name: Neal Johnson
Report Type: Error Report
Opt Subject: Unicode Standard Annex #29 - 3 Grapheme Cluster Boundaries - SpacingMark

Unicode Standard Annex #29 - 3 Grapheme Cluster Boundaries - SpacingMark (https://www.unicode.org/reports/tr29/#SpacingMark) states that U+11720 and U+11721 should be specifically excluded. However "GraphemeBreakProperty.txt" list both as included and as such {{UCharacter.getIntPropertyValue(0x11721, UProperty.GRAPHEME_CLUSTER_BREAK) }} return 10 "SPACING_MARK".

I am not sure if this an issue in the "GraphemeBreakProperty.txt" data file or an issue in Annex #29.

(submitted by Markus on behalf of Neal who mis-reported this as https://unicode-org.atlassian.net/browse/ICU-21438)

Background information / discussion

https://www.unicode.org/reports/tr29/#SpacingMark

SpacingMark	Grapheme_Cluster_Break ≠ Extend, *and* General_Category = Spacing_Mark, *or* *any of the following (which have* General_Category = Other_Letter*):* U+0E33 (◌ำ) THAI CHARACTER SARA AM U+0EB3 (◌ຳ) LAO VOWEL SIGN AM *Exceptions: The following (which have* General_Category = Spacing_Mark *and would otherwise be included) are specifically excluded:* U+102B (◌ါ) MYANMAR VOWEL SIGN TALL AA ... U+AA7B (◌ꩻ) MYANMAR SIGN PAO KAREN TONE U+AA7D (◌ꩽ) MYANMAR SIGN TAI LAING TONE-5 U+11720 (◌𑜠) AHOM VOWEL SIGN A U+11721 (◌𑜡) AHOM VOWEL SIGN AA

https://www.unicode.org/Public/UCD/latest/ucd/auxiliary/GraphemeBreakProperty.txt

```
116AE..116AF  ; SpacingMark # Mc   [2] TAKRI VOWEL SIGN I..TAKRI VOWEL SIGN II
116B6         ; SpacingMark # Mc       TAKRI SIGN VIRAMA
11720..11721  ; SpacingMark # Mc   [2] AHOM VOWEL SIGN A..AHOM VOWEL SIGN AA
11726         ; SpacingMark # Mc       AHOM VOWEL SIGN E
1182C..1182E  ; SpacingMark # Mc   [3] DOGRA VOWEL SIGN AA..DOGRA VOWEL SIGN II
11838         ; SpacingMark # Mc       DOGRA SIGN VISARGA
```

The Ahom vowel signs A & AA were added to the exceptions list for SpacingMark in this draft: tr29-26.html

with this review note:

Review Note:
The additional exceptions proposed for Ahom [L2/12-309R] were inferred by extrapolating the former list of exclusions to the new repertoire in Unicode 8.0, following the rationale given in [L2/11-114] which was applied in [tr29-18.html] for Unicode 6.1. Careful review is advised. If the proposed exceptions are accepted, then U+11720..U+11721 will also be excluded from the list of SpacingMark characters in GraphemeBreakProperty.txt for Unicode 8.0.

References:
[L2/12-309R] Martin Hosken, Stephen Morey, *Revised Proposal to add the Ahom Script in the SMP of the UCS*
[L2/11-114] Martin Hosken, *Proposal to change grapheme extending properties of various characters*
[tr29-18.html] Mark Davis, *Proposed Update UAX #29 for Unicode 6.1*

Excluding these characters from SpacingMark means that their WB values change to Other.

Recommended UTC action: AI: Remove the assignment of GCB=SpacingMark for U+11720..11721 AHOM VOWEL SIGN A & AA, letting them default to GCB=Other. For Unicode 14. Reference: L2/21-069 item P417b.

Peter: On investigation I believe it is correct to change the data to exclude 11720..11721 from SPACING_MARK.

PRI #427: Proposed Update UTS #18, Unicode Regular Expressions

https://www.unicode.org/review/pri427/

PRI427a: Misuse of subscripts

Recommended UTC actions

1. AI for Mark Davis: Revise the text of the proposed update for UTS #18 to use `𝖢<sub>𝕊</sub>` and `𝖢<sub>ℙ</sub>` instead of subscript letter forms. Reference: L2/21-069 item P427a.

Feedback (verbatim)

Date/Time: Sun Mar 21 16:48:51 CDT 2021
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: PRI #427: Misuse of subscripts

The proposed update introduces the notations C_s and C_p for the complements

of 𝕊 and ℙ, using subscript plain lowercase letters in place of subscript double-struck capital letters. This use of U+209B and U+209A goes against Unicode's general principle of subscripts, as described in section 22.4, that "style or markup in rich text" is preferred when possible, except in phonetic alphabets. Because UTS #18 is written in HTML, it should use `ℂ<sub>𝕊</sub>` and `ℂ<sub>ℙ</sub>`.

PRI427b: UTS #18 Unicode Regular Expressions

Recommended UTC actions

1. Update the proposed draft as per L2/21-094 "Update proposed draft UTS 18"
2. Keep the PRI open for now.

PRI #428: Unicode 14.0.0 Alpha Review

https://www.unicode.org/review/pri428/

PRI428a: Diacritical Marks Extended: Rotate code points U+1AC9..1ACE

Recommended UTC actions

1. Consensus: For Unicode 14: Rotate six code points U+1AC9..1ACE as noted in L2/21-069 item PRI428a.
2. AI for Ken Whistler and Michel Suignard, for Unicode 14:
   Rotate six code points as follows (shown in new order):
   a. Used in extended IPA
   b. 1ACD → 1AC9 COMBINING DOUBLE PLUS SIGN ABOVE
   c. 1ACE → 1ACA COMBINING DOUBLE PLUS SIGN BELOW
   d. Used in Middle English Ormulum
   e. 1AC9 → 1ACB COMBINING TRIPLE ACUTE ACCENT
   f. 1ACA → 1ACC COMBINING LATIN SMALL LETTER INSULAR G
   g. 1ACB → 1ACD COMBINING LATIN SMALL LETTER INSULAR R
   h. 1ACC → 1ACE COMBINING LATIN SMALL LETTER INSULAR T
3. AI for Ken Whistler: Update the pipeline with rotated code points U+1AC9..1ACE as noted in L2/21-069 item PRI428a.

Feedback (verbatim)

Date/Time: Mon Feb 15 19:56:28 CST 2021
Name: **Eduardo Marín Silva**
Report Type: Public Review Issue
Opt Subject: Suggestions on the alpha code chart of Diacritical Marks Extended

1. Whenever a header says "Used in..." It should read instead "Marks for..."

2. The header above 1AC1 should say (after the current header) "... Do not use pairs of these marks as replacement for 1ABB or 1ABD"

3. The two marks "combining double plus above and below" should be moved up, to be next to the single "plus sign above" and the Ormulum marks shifted down two spots.

4. The bullet note above the "number sign above" currently reads "used extensively in J.P. Harrington’s transcriptional notation" I suggest for it to read "Used by J.P. Harrington to indicate heavy or contrastive stress"

5. The "combining triple acute accent" should have a mutual cross reference to the "combining double acute accent"

Date/Time: Mon Apr 12 16:47:19 CDT 2021
Name: **Michael Everson**
Report Type: Public Review Issue
Opt Subject: Combining Diacritical Marks Extended

Move COMBINING DOUBLE PLUS SIGN ABOVE and COMBINING DOUBLE PLUS SIGN BELOW to immediately after COMBINING PLUS SIGN ABOVE.

PRI428b: U+1CF42 and U+1CF43 have nonconformant names

Recommended UTC actions

1. Consensus: For Unicode 14: Insert required hyphens into the names of U+1CF42 & U+1CF43.
2. AI for Ken Whistler and Michel Suignard, for Unicode 14: Insert required hyphens into the names of U+1CF42 & U+1CF43. Reference: L2/21-069 item P428b.
3. AI for Ken Whistler: Update the pipeline with fixed names of U+1CF42 & U+1CF43, see L2/21-069 item P428b.

Feedback (verbatim)

Date/Time: Sun Feb 14 09:29:15 CST 2021
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: PRI #428: U+1CF42 and U+1CF43 have nonconformant names

The names of proposed characters U+1CF42 (ZNAMENNY PRIZNAK MODIFIER LEVEL 2) and U+1CF43 (ZNAMENNY PRIZNAK MODIFIER LEVEL 3) currently do not conform to section 4.8 of the Unicode Standard. A hyphen-minus needs to be inserted before the final digit in both names because a digit must not immediately follow a space.

PRI428c: General category of U+1DF0A

Recommended UTC actions

1. AI for Ken Whistler, for Unicode 14: Change U+1DF0A LATIN LETTER RETROFLEX CLICK WITH RETROFLEX HOOK from gc=Ll to Lo.

Feedback (verbatim)

Date/Time: Sun Feb 14 09:42:18 CST 2021
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: PRI #428: General category of U+1DF0A

Proposed character U+1DF0A LATIN LETTER RETROFLEX CLICK WITH RETROFLEX HOOK currently has general category Ll (Lowercase_Letter). A more appropriate value would be Lo (Other_Letter) which is shared by most other click letters, including its hook-less counterpart U+01C3 LATIN LETTER RETROFLEX CLICK.

PRI428d: Names of U+1FAF1 and U+1FAF2

Recommended UTC actions

1. Request the ESC to consider both feedback items in L2/21-069 item PRI428d.

Feedback (verbatim)

Date/Time: Sun Feb 14 09:58:07 CST 2021
Name: **Charlotte Buff**
Report Type: Public Review Issue
Opt Subject: PRI #428: Names of U+1FAF1 and U+1FAF2

The names of proposed characters U+1FAF1 RIGHTWARD BACKHAND and U+1FAF2 LEFTWARD HAND could potentially be changed to RIGHTWARDS BACKHAND and LEFTWARDS HAND respectively. The words “rightward” and “leftward” do not occur in any other Unicode character names; instead the spellings “rightwards” and “leftwards” are used every single time.

Date/Time: Fri Apr 16 17:37:02 CDT 2021
Name: **Michael Everson**
Report Type: Public Review Issue
Opt Subject: Symbols and Pictographs Extended-A

1FAF1 RIGHTWARD BACKHAND and 1FAF2 LEFTWARD HAND are misnamed. "Backhand" refers to a kind of tennis swing; it does not refer to the back of a hand. Handedness is something the UCS should have dealt with long ago.

RIGHT-POINTING BACK OF HAND is what the first one is, and LEFT-POINTING FRONT OF HAND is what the other one is. All of the existing hands should be looked at with regard to this.

Note that the THUMBS UP and THUMBS DOWN hands are not completely encoded. Users should be able to select whether they wish to show hands with thumbs up or down based on how the would see it if they were holding their hands out in front of them. When I look at my right hand thumbs up I see the palm. When I look at ny right hand thumbs down, I see the back.

This is Alpha, so if there is a wish to make some of these hands make sense, now is the time to complete the set logically. I would help between now and beta if asked.

PRI428e: Names of dezh and tesh digraphs with hooks

Recommended UTC actions

1. Consensus: For Unicode 14: Insert the word “digraph” into the names of four dezh and tesh digraphs with hooks: U+1DF12, 1DF17, 1DF19, 1DF1C.
2. AI for Ken Whistler and Michel Suignard, for Unicode 14: Insert the word “digraph” into the names of four dezh and tesh digraphs with hooks: U+1DF12, 1DF17, 1DF19, 1DF1C. Reference: L2/21-069 item P428e.
3. AI for Ken Whistler: Update the pipeline with modified names of four dezh and tesh digraphs with hooks, see L2/21-069 item P428e.

Feedback (verbatim)

Date/Time: Sun Feb 14 10:25:15 CST 2021
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: PRI #428: Names of dezh and tesh digraphs with hooks

The names of the following proposed characters should be adjusted to include the word “digraph” for consistency with their respective hook-less counterparts (U+02A4 LATIN SMALL LETTER DEZH DIGRAPH and U+02A7 LATIN SMALL LETTER TESH DIGRAPH):

U+1DF12: LATIN SMALL LETTER DEZH WITH PALATAL HOOK → LATIN SMALL LETTER DEZH DIGRAPH WITH PALATAL HOOK
U+1DF17: LATIN SMALL LETTER TESH WITH PALATAL HOOK → LATIN SMALL LETTER TESH DIGRAPH WITH PALATAL HOOK
U+1DF19: LATIN SMALL LETTER DEZH WITH RETROFLEX HOOK → LATIN SMALL LETTER DEZH DIGRAPH WITH RETROFLEX HOOK
U+1DF1C: LATIN SMALL LETTER TESH WITH RETROFLEX HOOK → LATIN SMALL LETTER TESH DIGRAPH WITH RETROFLEX HOOK

PRI428f: General category of Znamenny priznak modifiers

Recommended UTC actions

1. AI for Ken Whistler, for Unicode 14: Change the Znamenny priznak modifiers U+1CF42..U+1CF46 from gc=Cf to Mn.

Feedback (verbatim)

Date/Time: Sun Feb 14 10:57:51 CST 2021
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: PRI #428: General category of Znamenny priznak modifiers

The Znamenny priznak modifiers (U+1CF42..U+1CF46) were given the general category Cf (Format). A more appropriate value would be Mn (Nonspacing_Mark) because they apply directly to the preceding character, comparable to variation selectors for instance. Other properties like bidi class and grapheme cluster break would need to be adjusted accordingly as well.

Background information / discussion

Looks ok, like in Miao, better GCB behavior.

PRI428g: Script Extensions for Arabic Punct used for N'ko and Adlam

Recommended UTC actions

1. AI for Mark Davis, for Unicode 14 Script_Extensions:
    a. Add Nkoo for U+060C ARABIC COMMA & U+061B ARABIC SEMICOLON.
    b. Add Nkoo & Adlm for U+061F ARABIC QUESTION MARK.
    c. Reference: L2/21-069 item PRI428g.

Feedback (verbatim)

Date/Time: Mon Feb 15 15:51:33 CST 2021
Name: Neil S Patel
Report Type: Public Review Issue
Opt Subject: Script Extensions for Arabic Punct used for N'ko and Adlam

Hello,

Recently, I have been working with a couple of W3C groups to look into script itemization issues. We have noticed that with both Adlam and N'ko when Arabic punctuation, typically used with both scripts, appears in a string of text it triggers unexpected fall backs. This occurs even when the tested font includes the appropriate Arabic punctuation. After some

discussion it was suggested that the script extensions could be responsible.

Reference: https://github.com/w3c/afrlreq/issues/18

Currently the script extensions for Arabic punctuation is listed as follows. There are no references to African scripts.

```
# ================================================
# Script_Extensions=Arab Rohg Syrc Thaa Yezi

060C          ; Arab Rohg Syrc Thaa Yezi # Po       ARABIC COMMA
061B          ; Arab Rohg Syrc Thaa Yezi # Po       ARABIC SEMICOLON
061F          ; Arab Rohg Syrc Thaa Yezi # Po       ARABIC QUESTION MARK

# Total code points: 3
# ================================================
```

I would like to propose the following update to include Adlam and N'ko.

```
# ================================================
# Script_Extensions=Arab Nko Rohg Syrc Thaa Yezi

060C          ; Arab Nko Rohg Syrc Thaa Yezi # Po       ARABIC COMMA
061B          ; Arab Nko Rohg Syrc Thaa Yezi # Po       ARABIC SEMICOLON

# Total code points: 2
# ================================================

# ================================================
# Script_Extensions=Adlm Arab Nko Rohg Syrc Thaa Yezi

061F          ; Adlm Arab Nko Rohg Syrc Thaa Yezi # Po       ARABIC QUESTION MARK

# Total code points: 1
# ================================================
```

Thanks.

PRI428h: Inconsistent identifier types for Komi letters

Recommended UTC actions

1. AI for Mark Davis, for Unicode 14: Change Komi letters U+0500..U+050F to Identifier_Type=Obsolete.
2. AI for Ken Whistler and Michel Suignard, for Unicode 14: Add a NamesList annotation to mark Komi letters U+0500..U+050F and U+052A..U+052D as for an obsolete alphabet.

Feedback (verbatim)

Date/Time: Fri Mar 19 19:46:34 CDT 2021
Name: David Corbett
Report Type: Error Report
Opt Subject: Inconsistent identifer types for Komi letters

The obsolete Komi letters U+052A..U+052D have Identifier_Type=Obsolete but
the other obsolete Komi letters U+0500..U+050F have
Identifier_Type=Recommended.

Background information / discussion

Obsolete alphabet from 1920s: https://en.wikipedia.org/wiki/Komi_language

PRI428i: U+08C8 ArabicShaping name

Recommended UTC actions

1. AI for Ken Whistler, for Unicode 14: Change ArabicShaping.txt for U+08C8 to use “KEHEH WITH EXTENDED HAMZA ABOVE”. Reference: L2/21-069 item PRI428i.

Feedback (verbatim)

Date/Time: Wed Mar 24 16:19:46 CDT 2021
Name: Lorna Evans
Report Type: Error Report
Opt Subject: U+08C8 ArabicShaping name

While I did laugh at this name in ArabicShaping, I think we could
come up with a better name:

08C8; KEHEH WITH DOOHICKEY ABOVE; D; GAF

It seems that the Arabic Shaping name was never discussed as far as
I can tell from script-adhoc notes, nor from UTC minutes.

L2/19-077 originally requested the character to be ARABIC LETTER
KEHEH WITH HAMZA ABOVE which indicates to me there is some association
with a hamza.

This was later changed to ARABIC LETTER GRAF in L2/19-252

I would suggest something like this:

08C8; KEHEH WITH EXTENDED HAMZA ABOVE; D; GAF

Lorna

PRI428j: Chinese numerals are not classified as numerals

Recommended UTC actions

1. AI for Rick McGowan: Respond to “r00ster” along the lines of:
    a. The ideographs used for numerals are not exclusively numeric.
    b. The numeric types of all relevant characters are listed in https://www.unicode.org/Public/UCD/latest/ucd/extracted/DerivedNumericType.txt
    c. Reference: L2/21-069 item PRI428j.

Feedback (verbatim)

Date/Time: Sat Apr 10 11:49:32 CDT 2021
Name: r00ster
Report Type: Error Report
Opt Subject: Chinese numerals are not classified as numerals

Hello Unicode,

I noticed that you classify Chinese numerals as Lo (other letters) which does not seem very correct to me because I believe Chinese numerals should be classified as numerals and not as other letters.

If I go to articles listed on the right of https://en.wikipedia.org/wiki/Numeral_system and try out a few characters listed on these articles, they mostly work (except for some rather outdated scripts such as Tangut numerals) and they are detected by Unicode as numeric, but for Chinese numerals, this is not the case. None of the numerals are detected as numeric. Especially for such a widely spoken language I would expect Unicode to correctly classify the numerals of that language. It is true that in Chinese there is an overwhelmingly large amount of (single) numeral characters, but I believe it is possible to maybe just classify at least the very basic 零/〇、一、二、三、四、五、六、七、八、九 (0-9) as numerals, and leave all other numerals beyond that classified as other letters.

Is it possible for you to reclassify them as numerals in a future version? See also: https://github.com/rust-lang/rust/issues/84056. Classifying Chinese numerals as numerals will of course mean support for other East Asian languages too, such as Japanese and Hokkien.

Thank you in advance.

PRI428k: kana letters proposed at 1B11F-1B122

Recommended UTC actions

1. Request the Script Ad Hoc group to consider the feedback from Mikoto Ohtsuki questioning whether the kana letters proposed at 1B11F-1B122 should be encoded. Reference: L2/21-069 item PRI428k.

Feedback (verbatim)

Date/Time: Sat Apr 10 18:49:00 CDT 2021
Name: Mikoto Ohtsuki
Report Type: Public Review Issue
Opt Subject: 1B11F-1B122 in Unicode 14.0 Alpha (PRI #428: Unicode 14.0 Alpha Review)

If kana letters proposed at 1B11F-1B122 became candidate for Unicode 14 based on L2/19-381, rationale seem insufficient. AFAIK, they are assumed to be just inventions primarily to fill up empty cells in syllabary chart called gojuonzu (50 sound chart). Usually they appear in some gojuonzu compiled in around late 19th century-early 20th century and lack examples in text actually used to spell words in accordance with proposed characteristics.

Existing of YI syllable separate from I syllable, and of WU syllable separate from U syllable has not been attested in history of Japanese phonology or orthography. Therefore it is not possible to happen that native Japanese words such as いもうと, まうす, ようべ in page 6 of L2/19-381, and やいば, ついたち, ちひさい in page 10 were written using kana intended for WU or YI syllable. Note that standard う (U) was used in corresponding hiragana forms of them in page 6 instead of kana intended for WU. Chart contradicts itself.

Pages 2 and 7 show 衣 (U+8863) as Kanji Derivation for 1B12D, now shifted to 1B121, KATAKANA LETTER ARCHAIC YE. However 衣 is origin of 1B000 KATAKANA LETTER ARCHAIC E. It is inconsistent evidently. Rather than thinking this kana was derivation from single kanji, thinking it was compound form of イ (I) and エ (E) would be more appropriate as mentioned in footnote. It would be KATAKANA LETTER LIGATURE IE.

It is strongly suspected that referenced books were written without scholarly knowledge. Including them with current characteristics in Unicode 14 is questionable. I'd like UTC to consider two matters.

First, please postpone inclusion of them to Unicode Standard till their characteristics are confirmed by expert input or examples actually in use with proposed characteristics are provided. If such input is unavailable, please consider another way like encoding them as itaigana (kana variant) for standard I and U kana letters.

Second, please reconsider their names. Using same ARCHAIC prefix to both kana dating from Heian era (8th-12th century) and kana invented by `there should be to fill up gojuonzu` attempt in early modern period gives odd feeling. Please don't call latter kana ARCHAIC.

PRI428L: Soft_Dotted property of U+1DF1A

Recommended UTC actions

1. AI for Ken Whistler, for Unicode 14: For U+1DF1A LATIN SMALL LETTER I WITH STROKE AND RETROFLEX HOOK set Soft_Dotted=True.
    a. Note: This has already been taken care of.

Feedback (verbatim)

Date/Time: Fri Apr 16 09:40:46 CDT 2021
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: PRI #428: Soft_Dotted property of U+1DF1A

Proposed character U+1DF1A LATIN SMALL LETTER I WITH STROKE AND RETROFLEX HOOK should have the Soft_Dotted property like other variants of the letter i.