UTC #170 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2022-jan-19

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), Mark Davis, Asmus Freytag, Ken Whistler, Ned Holbrook, Rick McGowan, Christopher Chapman, Peter Constable, Elango Cheran

Public feedback

Feedback received via the Unicode reporting form, see L2/22-018 “Comments on Public Review Issues (Sept 25, 2021 - Jan 18, 2022)”.

F1: Persian digits are not considered Arabic Numbers

Recommended UTC actions

1. No changes in Unicode. These properties are very much intentional.
2. We have responded to the submitter.

Feedback (verbatim)

(Note: Two feedback items from the same person on the same topic.)

Date/Time: Mon Oct 25 15:20:07 CDT 2021
Name: Gary Wade
Report Type: Other Document Submission

Originally submitted against CLDR at https://unicode-org.atlassian.net/browse/CLDR-15118

If there is a better avenue, please provide a direct link as I saw no other
appropriate place to do so.

Persian digits (U+06F0-U+06F9) are not considered Arabic Numbers in UnicodeData.txt

Values for U+06F0 to U+06F9 are considered to be European Numbers rather than Arabic Numbers, and so based on a bidi property lookup, these are not considered to be "RTL-weak" for lack of a better phrase like values U+0660 to U+0669, and so some algorithms will always consider them to be "LTR-weak".

06F0;EXTENDED ARABIC-INDIC DIGIT ZERO;Nd;0;EN;;0;0;0;N;EASTERN ARABIC-INDIC DIGIT ZERO;;;;
06F1;EXTENDED ARABIC-INDIC DIGIT ONE;Nd;0;EN;;1;1;1;N;EASTERN ARABIC-INDIC DIGIT ONE;;;;
06F2;EXTENDED ARABIC-INDIC DIGIT TWO;Nd;0;EN;;2;2;2;N;EASTERN ARABIC-INDIC DIGIT TWO;;;;
06F3;EXTENDED ARABIC-INDIC DIGIT THREE;Nd;0;EN;;3;3;3;N;EASTERN ARABIC-INDIC DIGIT THREE;;;;
06F4;EXTENDED ARABIC-INDIC DIGIT FOUR;Nd;0;EN;;4;4;4;N;EASTERN ARABIC-INDIC DIGIT FOUR;;;;
06F5;EXTENDED ARABIC-INDIC DIGIT FIVE;Nd;0;EN;;5;5;5;N;EASTERN ARABIC-INDIC DIGIT FIVE;;;;
06F6;EXTENDED ARABIC-INDIC DIGIT SIX;Nd;0;EN;;6;6;6;N;EASTERN ARABIC-INDIC DIGIT SIX;;;;
06F7;EXTENDED ARABIC-INDIC DIGIT SEVEN;Nd;0;EN;;7;7;7;N;EASTERN ARABIC-INDIC DIGIT SEVEN;;;;
06F8;EXTENDED ARABIC-INDIC DIGIT EIGHT;Nd;0;EN;;8;8;8;N;EASTERN ARABIC-INDIC DIGIT EIGHT;;;;
06F9;EXTENDED ARABIC-INDIC DIGIT NINE;Nd;0;EN;;9;9;9;N;EASTERN ARABIC-INDIC DIGIT NINE;;;;

It was noted that these digits are not considered Arabic digits, but since their names literally have the word "Arabic" in them, this seems incorrect; consider also by that same logic the HANIFI ROHINGYA DIGIT and RUMI digits which are considered in this class.

Date/Time: Tue Oct 26 14:05:04 CDT 2021
Name: Gary L. Wade
Report Type: Error Report
Opt Subject: UnicodeData.txt

Values for U+06F0 to U+06F9 are considered to be European Numbers(EN) rather than Arabic Numbers (AN) for the bidi class, and so based on a bidi property lookup, these are not considered to be "RTL-weak" for lack of a better phrase like values U+0660 to U+0669, and so some algorithms will always consider them to be "LTR-weak". Since these digits are used in Persian, which is an RTL language, these should also have the bidi class of AN just like the HANIFI ROHINGYA DIGIT and RUMI digits.

To see the difference between how these digits are laid out unexpectedly, Apple's TextEdit app on the Mac running under US English can be used to enter these with the appropriate Arabic and Persian keyboards on separate lines with a space between each digit:

1. Launch TextEdit on macOS under US English locale
2. Choose Arabic keyboard
3. Type each digit with a space between each (1, space, 2, space, etc.); notice the RTL direction to lay out the text
4. Press the return key to enter a new line

5. Choose Persian keyboard
6. Type each digit with a space between each; notice the LTR direction is used to lay out the text

This software and much more expect to use the properties in UnicodeData.txt
for the bidi algorithm, and adding an override in each app to make Persian
digits RTL goes against its purpose.

06F0;EXTENDED ARABIC-INDIC DIGIT ZERO;Nd;0;EN;;0;0;0;N;EASTERN ARABIC-INDIC DIGIT ZERO;;;;
06F1;EXTENDED ARABIC-INDIC DIGIT ONE;Nd;0;EN;;1;1;1;N;EASTERN ARABIC-INDIC DIGIT ONE;;;;
06F2;EXTENDED ARABIC-INDIC DIGIT TWO;Nd;0;EN;;2;2;2;N;EASTERN ARABIC-INDIC DIGIT TWO;;;;
06F3;EXTENDED ARABIC-INDIC DIGIT THREE;Nd;0;EN;;3;3;3;N;EASTERN ARABIC-INDIC DIGIT THREE;;;;
06F4;EXTENDED ARABIC-INDIC DIGIT FOUR;Nd;0;EN;;4;4;4;N;EASTERN ARABIC-INDIC DIGIT FOUR;;;;
06F5;EXTENDED ARABIC-INDIC DIGIT FIVE;Nd;0;EN;;5;5;5;N;EASTERN ARABIC-INDIC DIGIT FIVE;;;;
06F6;EXTENDED ARABIC-INDIC DIGIT SIX;Nd;0;EN;;6;6;6;N;EASTERN ARABIC-INDIC DIGIT SIX;;;;
06F7;EXTENDED ARABIC-INDIC DIGIT SEVEN;Nd;0;EN;;7;7;7;N;EASTERN ARABIC-INDIC DIGIT SEVEN;;;;
06F8;EXTENDED ARABIC-INDIC DIGIT EIGHT;Nd;0;EN;;8;8;8;N;EASTERN ARABIC-INDIC DIGIT EIGHT;;;;
06F9;EXTENDED ARABIC-INDIC DIGIT NINE;Nd;0;EN;;9;9;9;N;EASTERN ARABIC-INDIC DIGIT NINE;;;;

Background information / discussion

bc=AN digits are not “RTL-weak”. Both Arabic and Persian digits are LTR, but they behave differently in relation to grouping and decimal separators and currency symbols. This is very much intended in the UBA and in these property values, and has been this way for a very long time.

F2: Mistake about U+0953 and U+0954

Recommended UTC actions

1. Remove the explicit Indic_Positional_Category values for U+0953 DEVANAGARI GRAVE ACCENT and U+0954 DEVANAGARI ACUTE ACCENT, letting them default to NA (not applicable), for Unicode 15.
2. Action item for Markus Scherer: Remove the explicit Indic_Positional_Category values for U+0953 & U+0954.

Feedback (verbatim)

Date/Time: Sat Sep 18 15:50:43 CDT 2021
Name: David Corbett
Report Type: Error Report
Opt Subject: Mistake about U+0953 and U+0954

Chapter 12 says “Because U+0953 and U+0954 are not intended to be used
with the Devanagari script, they have no explicit property values for
Indic_Positional_Category and Indic_Syllabic_Category”, but that is

not true. They both still have the explicit Indic_Positional_Category value of Top.

Background information / discussion

Related: UTC #164 properties feedback & recommendations / paragraph modified for [164-A62]
This retained the already existing claim that “they have no Indic shaping properties”.
In fact, they have been Indic_Positional_Category=Top at least since Unicode 8.

Discussion: Given the description, these characters should not have explicit values for either of these properties.

F3: Bad word break of RI ZWJ RI RI

Recommended UTC actions

1. Action item for Mark Davis and Chris Chapman: Start a new document that subsumes all old AIs concerning consistency between different segmentation algorithms, with explicit examples of problem cases where available. Once the document is submitted to the registry, request that the old AIs be closed: 160-A73, 149-A50, 142-A64 and possibly more. See L2/22-019 item F3.

Feedback (verbatim)

Date/Time: Fri Oct 15 17:59:27 CDT 2021
Name: Yannick Duchêne
Report Type: Error Report
Opt Subject: UAX29

Referring to version 13, unless I’m wrong, the sample at line #1725 of WordBreakTest.txt, exposes a case of a grapheme being broken.

The test sequence is: ALetter RI ZWJ RI RI ALetter
As graphemes, I believe it is: (ALetter) (RI ZWJ) (RI RI) (ALetter)
But the sample says, as words, it is: (ALetter) (RI ZWJ RI) (RI) (ALetter)
The third grapheme, (RI RI), is broken in two parts, its first RI goes to one word and its second RI, to another word.

The comment is correcte about the rule applied, so may be this is an unintended effect of the rules for word boundaries or for grapheme boundaries in UAX #29. It may be not intended, since §6 says “The other default boundary specifications never break within grapheme clusters”.

Background information / discussion

unicodetools/data/ucd/13.0.0-Update/auxiliary/WordBreakTest.txt#L1725
÷ 0061 ÷ 1F1E6 × 200D × 1F1E7 ÷ 1F1E8 ÷ 0062 ÷ #

÷ [0.2] LATIN SMALL LETTER A (ALetter)
÷ [999.0] REGIONAL INDICATOR SYMBOL LETTER A (RI)
× [4.0] ZERO WIDTH JOINER (ZWJ_FE)
× [16.0] REGIONAL INDICATOR SYMBOL LETTER B (RI)
÷ [999.0] REGIONAL INDICATOR SYMBOL LETTER C (RI)
÷ [999.0] LATIN SMALL LETTER B (ALetter)
÷ [0.3]

Partially related to these AIs and possibly more:

160-A73	Mark Davis	Investigate the best way to resolve inconsistencies in text segmentation and linebreak algorithms, and report back to the UTC. See feedback in PRI #396 from Charlotte Buff [Sat Jul 6 16:57:48 CDT 2019].		In progress

149-A50	Peter Edberg	Review cases where LineBreak and WordBreak are not aligned, for the January 2017 (--> 2018) UTC meeting. (Retargeted for 13.0, then 14.0.)	L2/16-336

142-A64	Laurențiu Iancu	Prepare a proposal for recommendations to address remaining inconsistencies between word break and sentence break in UAX #29, for Unicode 9.0. (postponed to 10.0, then 11.0, then 13.0, then 14.0, then 15.0)

F4: U+0019 in ISO vs. NameAliases.txt vs. chart/NamesList.txt

Recommended UTC actions

1. For U+0019, add a Name alias of type abbreviation, for Unicode 15.
2. Action item for Ken Whistler: Add one line to NameAliases.txt: 0019;EM;abbreviation [done already]

Feedback (verbatim)

Date/Time: Thu Nov 25 10:22:52 CST 2021
Name: Giacomo Catenazzi
Report Type: Error Report
Opt Subject: NameAliases.txt

C0 chart (https://www.unicode.org/charts/PDF/U0000.pdf) uses the abbreviation EM for 0x19, but in NamesAlias.txt only EOM is listed as abbreviation. Because EM is used in various ISO (and ANSI, and ECMA, e.g. ECMA-48 and the C0 table is linked also in ECMA-6 [ISO 646]), I think NameAliases.txt should include also a third line:

0019;END OF MEDIUM;control
0019;EOM;abbreviation

0019;EM;abbreviation <- NEW LINE HERE

Note: the name 'EM' seems available in Unicode.

BTW it seems EOM was previously used in first version of ASCII as abbr. of 0x03 instead of ETX (as end of message) (according Wikipedia and the scanned version). EM will just avoid confusion, and it is more used(you use it on C0 chart).

Background information / discussion

https://www.unicode.org/Public/UCD/latest/ucd/NameAliases.txt
0019;END OF MEDIUM;control
0019;EOM;abbreviation
https://www.unicode.org/Public/UCD/latest/ucd/NamesList.txt
0019 <control>
      = END OF MEDIUM
https://en.wikipedia.org/wiki/ISO/IEC_646

- Shows EM
- Unicode NameAliases include all of the other abbreviations listed here for other C0 control codes.

https://en.wikipedia.org/wiki/ASCII#Control_code_chart
https://en.wikipedia.org/wiki/End_of_message#Origin

F5: UAX44-LM2 medial-hyphen clarification

Recommended UTC actions

1. Action item for Ken Whistler and the editorial committee for Unicode 15:
    a. Clarify the definition of “medial hyphen” in UAX44-LM2 and UAX34-R3, for example changing “between two letters” to “between two alphanumeric characters” (and defining “alphanumeric”).
    b. Refer to core spec section 4.8 Name which defines the Name syntax (including which hyphen is meant here).
    c. Add examples of names with alphanum-alphanum combinations that are not letter-letter.
    d. Add a note about this to the Unicode 15 release notes/migration section.
    e. [This is already in the latest draft of UAX #44 but not yet in UAX #34: https://www.unicode.org/reports/tr44/tr44-29.html#UAX44-LM2]

Feedback (verbatim)

Date/Time: Wed Dec 1 10:57:02 CST 2021
Name: J. S. Choi
Report Type: Error Report
Opt Subject: UAX44-LM2 medial-hyphen clarification

The UAX44-LM2 rule defines “medial hyphen” as a “hyphen occurring immediately between two letters”; however, it does not clarify whether a medial hyphen also may be between a letter and a numeral. For example, if the answer is

yes, then “VARIATION SELECTOR 15” and “VARIATION_SELECTOR_15” would match “VARIATION SELECTOR-15”, and if the answer is no, then they would not match.

Background information / discussion

https://www.unicode.org/reports/tr44/#Matching_Names “5.9.2 Matching Character Names” → UAX44-LM2

The name matching is intended to ignore hyphens between *alphanumeric* characters.

See also https://www.unicode.org/reports/tr34/#UAX34-R3 Name uniqueness, and The Unicode Standard, section 4.8 Name.

Other examples of letter+hyphen+digit:

```
1087;MYANMAR SIGN SHAN TONE-2;Mc;0;L;;;;;N;;;;;
23BA;HORIZONTAL SCAN LINE-1;So;0;ON;;;;;N;;;;;
2679;RECYCLING SYMBOL FOR TYPE-7 PLASTICS;So;0;ON;;;;;N;;;;;
2680;DIE FACE-1;So;0;ON;;;;;N;;;;;
286A;BRAILLE PATTERN DOTS-2467;So;0;L;;;;;N;;;;;
1B060;HENTAIGANA LETTER TA-3;Lo;0;L;;;;;N;;;;;
1B173;NUSHU CHARACTER-1B173;Lo;0;L;;;;;N;;;;;
2F803;CJK COMPATIBILITY IDEOGRAPH-2F803;Lo;0;L;20122;;;;N;;;;;
```

F6: Name collision: LATIN SMALL LETTER N WITH LEFT HOOK

Recommended UTC actions

1. Change the names of 6 characters already approved for Unicode 15:
   1DF25 LATIN SMALL LETTER D WITH LEFT HOOK..1DF2A LATIN SMALL LETTER T WITH LEFT HOOK
   Change each “WITH LEFT HOOK” suffix to “WITH MID-HEIGHT LEFT HOOK”.
2. Action item for Ken Whistler and the properties & algorithms group: Update the data files. [done already]
3. Action item for Ken Whistler: Update the pipeline.
4. Action item for Ken Whistler: Coordinate with ISO 10646. [done already]

Feedback (verbatim)

Date/Time: Tue Nov 16 12:31:51 CST 2021
Name: Jack Varanelli
Report Type: Error Report
Opt Subject: Unicode request for legacy Malayalam

To whom it may concern:

I am a student with no real position in Unicode. However, I noticed that the names of U+0272 and the proposed character for U+1DF27 in this document [https://www.unicode.org/L2/L2021/21156-legacy-malayalam.pdf] have

the same name (LATIN SMALL LETTER N WITH LEFT HOOK). This has been added to the Unicode Pipeline, so I am left to assume its inclusion is planned.
Knowing this may cause confusion, I'd advise a name change to the proposed character, if possible.

Apologies if this was intentional and an oversight on my part.

Sincerely,
Jack Varanelli

Background information / discussion

Ken:
... there are *6* characters involved, with identical raised left hooks. The existing 0272 has a baseline left hook, implicitly based on 0321 COMBINING PALATALIZED HOOK BELOW. Most of the existing characters named "WITH LEFT HOOK" are based on the en shape, with a baseline left hook (019D, 0272, 0528, 0529, 1DAE), but then there is the u "WITH LEFT HOOK" which is a raised left hook similar to the 6 from Kirk. We could either fix just the one name or fix *all six* in parallel. I actually suggest the latter as the more expedient solution, because there are other contrasting "left hooks" in different positions. In particular, of Kirk's d, l, n, r, s, t, there are "WITH PALATAL HOOK" versions at 1D81, 1D85, 1D87, 1D89, 1D8A, 01AB.

The cleanest might be to just suggest renaming all 6 of the new characters [...], to have a clear contrast with the other two usages.

Documents

D1: Proposal to add a derived data file for the “IDNA_Property” to /public/idna

L2/21-227 from Asmus Freytag & Ken Whistler
(Link to IANA data: https://www.iana.org/assignments/idna-tables-11.0.0/idna-tables-11.0.0.txt)

Recommended UTC actions

1. Create a new data file for derived data in a new folder “idna2008derived” under https://www.unicode.org/Public/idna/ with data contents as described in L2/21-227 and, for the purpose of regular expressions, with a property name of IDNA2008_Category.
2. Action item for Asmus Freytag and Ken Whistler: Post data files with derived IDNA2008 data for Unicode versions 6.1..15; see L2/22-019 item D1.
3. Action item for Asmus Freytag and Ken Whistler: In UTS #46, document the new files with derived IDNA2008 data; see L2/22-019 item D1.
4. Action item for Markus Scherer and the Unicode Tools team: For generation of derived IDNA2008 data, either adopt a new tool from Asmus & Ken, or modify the tools that generate the existing idna files to also generate the new file for future versions.

Summary

Establish a new derived property (IDNA_property)and add it /public/idna updated each version, starting with Unicode 15.0
…
We propose to provide a purely derived data file that essentially matches these derived data, but provides timely updates anytime the Unicode Standard adds characters or tweaks General Category values that influence the derivation.
The proposed derived property provides only the classification of code points by IDNA2008_property value and does so in a way that can be easily compared to the files in the IANA repository.
…
We suggest that this derived file be published any time a new version of Unicode (and therefore UTS#46) is published and be made available for public review ahead of release. We believe this would facilitate speedier IETF review or speedier adoption of new Unicode characters for implementations supporting IDNA that had relied heretofore on the convenience of the IANA registry data.

Background information / discussion

https://www.unicode.org/Public/idna/

Asmus: IETF talks about “IDNA derived property values”

D2: Exploratory document on a problematic casing pair used by some African orthographies

L2/21-229 from Eduardo Marín Silva

Recommended UTC actions

1. No action. We agree with the recommendation of the Script Ad Hoc group.

Summary

From SAH discussion: This document proposes different approaches to handle the casing pair Бƀ “that according to Wikipedia is used by some African orthographies [though]… [the author] couldn’t confirm the veracity of those claims.” The document discusses five possible encoding models: one that would involve changing casing relations (A1), a second option involving use of a pair of characters for the old Zhuang orthography (A2), a third option which uses a current case pair but involves a different glyph form for the uppercase (A3), a fourth option involving use of Cyrillic letters for languages that use Latin letters (B), and lastly encoding a new Latin case pair (C).

Background information / discussion

This document lacks specific information about the exact characters, orthography, and usage.

D3: Proposal to guarantee stability of spelling of property names, values, and aliases in UCD

L2/22-029 from Michael Ficarra

Recommended UTC actions

1. The UTC recommends to the executive officers a new stability policy for not changing the spelling of existing property aliases and property aliases of UCD properties going forward, excluding provisional and contributory properties, similar to www.unicode.org/policies/stability_policy.html#Alias_Stability; see L2/22-019 item D3.
2. Action item for Ken Whistler and the properties & algorithms group: Determine the earliest Unicode version since which the exact spelling of existing property (value) aliases has been unchanged; see L2/22-019 item D3.

Summary

Proposal to stabilize the exact spellings (including case and underscores) of property aliases and property value aliases in the Unicode Character Database, as shown in PropertyAliases.txt and PropertyValueAliases.txt.

Background information / discussion

https://www.unicode.org/reports/tr44/#Property_And_Value_Aliases

> Aliases are defined as ASCII-compatible identifiers, using only uppercase or lowercase A-Z, digits, and underscore "_". Case is not significant when comparing aliases, but the preferred form used in the data files for longer aliases is to titlecase them.
> …
> In PropertyAliases.txt, the first field typically specifies an abbreviated symbolic name for the property, and the second field specifies the long symbolic name for the property. These are the preferred aliases.

https://www.unicode.org/policies/stability_policy.html#Alias_Stability

It seems like long aliases are *intended* to be in the form that they are, and are *practically* stable as is. There is a precedent that supports this view: PVA.txt includes both aliases Arabic_Presentation_Forms_A and Arabic_Presentation_Forms-A. Under loose alias matching, we need only one of these. This implies that we already have stable spellings.

If aliases are to be stable under strict matching, and we wanted to change case or add/remove underscores, then we would need to add such a modified alias as a new, additional alias (farther right in the row of the aliases data file), rather than just change it.

Michael programmatically confirmed that there are no current spelling divergences for any property names, values, or aliases in ECMA-262.

See also
https://www.unicode.org/reports/tr44/#Matching_Names
https://www.unicode.org/reports/tr18/#property_syntax

Note that the proposal does not mention properties defined by other Unicode standards such as UTS #39.

D4: Avoiding Source Code Spoofing

L2/22-007 from Mark Davis, Robin Leroy, Peter Constable, Markus Scherer

Recommended UTC actions

1. Form a limited-duration, ad hoc working group as suggested in the document, with Mark Davis as the chair.
2. Have an open discussion of the goals & techniques discussed in the document, to provide feedback for the working group.

Summary

This is a proposal to form a working group / task force focusing on providing guidance for dealing with the so-called Trojan Source exploit. While we have long-standing documentation on dealing with bidirectional behavior and confusables, this would focus on source code. This task force would deliver its results to the Properties and Algorithms working group of the Unicode Technical Committee for review, and eventually to the UTC for approval.
We would look to involve people who are experts in the Unicode encoding, bidirectional algorithm, and security concerns/mechanisms, as well as experts in programming language standardization and tooling and security experts. The proposal also provides a (rough draft) sketch of what such types of guidance could look like.

D5: C0 and C1 stability for Unicode and 10646

L2/22-013 from Kent Karlsson

Recommended UTC actions

1. No action

Summary

Overview of control code implementations, especially ISO/IEC 6429 = ECMA-48 vs. Teletext.
“The C0/C1 need to be fixed, and fixed to what is in ISO/IEC 6429.”
“C0/C1 should be protected by conformity clauses, stability clauses, and a total rewrite of section 23.1 (Control Codes; see below for a suggested rewrite) of the Unicode standard, and a rewrite of at least section 13.4 (Identification of control function set) of ISO/IEC 10646 [...].”

Background information / discussion

https://www.unicode.org/versions/Unicode14.0.0/ch23.pdf (pp. 900-902)
https://www.unicode.org/versions/Unicode1.0.0/ch02.pdf (pp. 15-16)
From the beginning, the Unicode Standard has mostly left the semantics of control codes to higher-level protocols, basically providing a pass-through of control codes. This also allows implementers to use Unicode for implementing other protocols.

The situation is equivalent in ISO 10646: “This coded character set provides for use of control functions encoded according to ISO/IEC 6429 or similarly structured standards for control functions, and standards derived from these. A set or subset of such coded control functions may be used in conjunction with this coded character set.”

We reviewed the document and agree that there is no real value for Unicode to make changes as suggested.

Public Review Issues

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

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

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

PRI427a: UTS #18 matching of negative numeric values

Recommended UTC actions

1. Action item for Mark Davis: Clarify in UTS #18 that loose value matching applies to symbolic values of enumerated and catalog properties, and UAX #44 5.9.1 / UAX44-LM1 applies to matching numeric values; use the example of numeric value -0.5 of U+0F33; for details refer to UAX #44; see L2/22-019 item PRI427a.

Feedback (verbatim)

Date/Time: Thu Dec 16 22:29:51 CST 2021
Name: Karl Williamson
Report Type: Error Report
Opt Subject: UTS 18

U+0F33 TIBETAN DIGIT HALF ZERO has a numeric value of -0.5. (I believe the existence of this character in the wild is apocryphal however.) There is no rule against other code points becoming encoded with a negative value.

However, UTS 18 says the hyphen-minus sign is supposed to be ignored within \p{} constructs, leaving no way to legally specify negative values.

I suspect that UTS 18 should be clarified to indicate that the hyphen minus at the beginning of a number should not be ignored, even with loose matching. But then what to do about two in a row?

Background information / discussion

https://www.unicode.org/reports/tr18/#property_syntax
https://www.unicode.org/reports/tr18/#RL1.2

https://www.unicode.org/reports/tr44/#Matching_Numeric
https://www.unicode.org/reports/tr44/#Matching_Symbolic

PRI427b: Outdated JavaScript escape syntax

Recommended UTC actions

1. Action item for Mark Davis: Adjust the current examples of escape syntax in https://www.unicode.org/reports/tr18/#Hex_notation to account for improved capabilities in regex handling in Java, JavaScript, and ICU.

Feedback (verbatim)

Date/Time: Tue Jan 18 22:34:08 CST 2022
Name: Norbert Lindenberg
Report Type: Error Report
Opt Subject: UTS 18: Unicode Regular Expressions

In a table showing "current examples of escape syntax for Unicode code points", UTS 18: Unicode Regular Expressions shows "\uD83D\uDC7D" in the row that includes JavaScript. This example is no longer current: EcmaScript 2015 introduced the \u{xxxxxx} escape syntax for Unicode code points, so that "\u{1F47D}" now works.

PRI427c: Finish PRI #427

Recommended UTC actions

1. Approve UTS #18 for release, based on PRI #427.
2. Action item for Mark Davis, Markus Scherer, and the editorial committee: Finalize the text of UTS #18, based on PRI #427. Include changes for action items 168-A012, 168-A013, and 166-A070 [and for PRI427b] if feasible, but these are not blockers.
3. Action item for Rick McGowan: Post UTS #18 once the text is finalized, based on PRI #427.

Background information / discussion

All of the feedback has been accounted for. We were just waiting to see if there was anything resulting from the ECMAScript additions to regular expressions that should be reflected in the document, but we don’t anticipate anything pressing in the near future.