L2/20-240

UTC #165 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2020-sep-30

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, with Markus Scherer as the chair. Several UTC participants have requested and received invitations to join. So far, discussion has been via email and shared documents, and one video meeting.

Participants

The following people have contributed to this document:

Markus Scherer (chair), Mark Davis, Christopher Chapman, Roozbeh Pournader, Asmus Freytag

Public feedback

Feedback received via the Unicode reporting form, see L2/20-239 "Comments on Public Review Issues (July 20, 2020 - September 23, 2020)".

F1: UAX14 quotation marks vs ID

This was item F7 in L2/20-175 for UTC #164. From the meeting minutes: "The properties and algorithms group will discuss this item further."

Feedback (verbatim)

Date/Time: Mon Jun 22 16:19:50 CDT 2020
Name: fantasai
Report Type: Error Report
Opt Subject: UAX14 quotation marks vs ID

The UAX14 rules concerning QU are too strict, and don't work for Chinese by
default, because they rely on spaces to be a reasonable default. This can
probably be solved by allowing breaks between ID + Pi and between Pf + ID.
See https://github.com/w3c/clreq/issues/245 for more info.

Recommended UTC actions

1. We recommend not to make any changes in UAX #14.
2. AI for Mark: Forward L2/20-240 item F1 to CLDR for discussion: Handling of quotation marks in line breaking needs language-specific tailoring.

Background information / discussion

[discussion before UTC #164]

Mark: I recommend putting out a PRI for this (and maybe other issues) to get feedback from a broader community.

Andy: I don't know much about Chinese, so I'll take the description of the problem on faith.

Splitting the character class QU into Pf and Pi would carry a cost in data size.

The change would be to LB-19, currently
    × QU
    QU ×
Something like
    [^ID] × Pi
    Pf × [^ID]
Which raises the question of whether any other character categories beyond ID would want to have this behavior.

And what should be done with quotes that are neither Pf or Pi, like ASCII quotes?

In any event, I don't think anything should go into UAX-14 without first being implemented and released in ICU, to see what unanticipated complications turn up. Maybe it could be put out as a CLDR tailoring.

Koji Ishii: Using ID doesn't solve the case presented in https://github.com/unicode-org/icu/pull/223#issuecomment-435642278 as Andy pointed out above.

ASCII quotes are fine, they're not full-width in Chinese/Japanese and that authors will put spaces before or after appropriately. The problem of U+201C/201D/etc. is that they're full-width in C/J fonts, and they have embedded spacing in their glyphs, similar to full-width parenthesis (see http://unicode.org/reports/tr50/#table_2 for example glyphs of full-width parenthesis,) so we can't expect space characters to exist.

Given Andy's comment on the data size, maybe similar approach as LB30 can solve?

[discussion after UTC #164]

Chris Chapman: As Fuqiao Xue noted in this comment in the original issue, this problem is already at least partially solved by the provision in UAX 14 for tailoring quotation marks to OP and CL based on language, and this appears to be implemented (at least for U+201C and U+201D in Chinese) in Chrome, Firefox, and Safari.

I'm guessing that's thanks to this change in ICU. I think for the default handling of punctuation around ideographs to work in the absence of language-based tailoring, there'd have to be a rule like:
"Treat any quotation mark adjacent to an ideograph as OP if its general category is Pi, or CL if its general category is Pf."

Asmus: Re suggested [^ID] × Pi — Would need to know which punctuation is opening/closing. Especially in Swedish, the same character is used on both sides (see https://en.wikipedia.org/wiki/Quotation_mark). Changes like this should be considered only if it can be shown exhaustively that they would avoid the need for tailoring for characters that are used in their native context, without requiring an unexpected need for tailoring all other languages.

F2: feedback on UAX#31: Identifier Characters

Feedback (verbatim)

Date/Time: Thu Jul 30 16:55:11 CDT 2020
Name: Peter Constable
Report Type: Error Report
Opt Subject: feedback on UAX#31

This feedback pertains to revision 33 of UAX#31:
http://www.unicode.org/reports/tr31/tr31-33.html

In section 1, the paragraph after Figure 1 says,

"The set consisting of the union of ID_Start and ID_Nonstart
characters is known as Identifier Characters ..."

Then in section 1.1, the second bulleted item in the list of stability guarantees says,

"The Identifier characters are always a superset of the ID_Start characters."

Given the definition of "Identifier Characters" given in
section 1, this statement is tautological—necessarily true,
by definition—so not useful to state as a stability guarantee.
Was "proper superset" meant?

Recommended UTC actions

We recommend not to make any changes.

Background information / discussion

Mark responded: “No, superset was meant. It is a restatement, just to clarify the stability condition.”

F3: feedback on UAX#31: first mention of “XID properties”

Feedback (verbatim)

Date/Time: Thu Jul 30 17:23:29 CDT 2020
Name: Peter Constable
Report Type: Error Report
Opt Subject: feedback on UAX#31

This feedback pertains to revision 33 of UAX#31

In section 2, in the 4th paragraph, the last sentence says,

"The second column provides a general description of the coverage for the associated class, the derivational relationship between the ID properties and the XID properties, and an associated set notation for the class."

The concepts "ID property" and "XID property" are in this way introduced. If there were mention of only "ID property", that would be fine: in the context, it would be sufficiently clear that there will be character properties pertaining to IDs that are used for Default Identifier Syntax. However, with a second concept thrown in, "XID property", this becomes confusing. (Huh? What's an "XID property" and what does it have to do with identifier syntax?) It would help to introduce the pair of terms with some explanation of what "XID" is all about.

Recommended UTC actions

We recommend not to make any changes.

Background information / discussion

Same as for “ID properties”, in an overview paragraph immediately before the table that provides links to definitions. Seems fine.

F4: feedback on UAX#31, 2.3.1 Limitations / joiners

Feedback (verbatim)

Date/Time: Thu Jul 30 18:17:04 CDT 2020
Name: Peter Constable
Report Type: Error Report
Opt Subject: feedback on UAX#31, 2.3.1 Limitations

This feedback pertains to revision 33 of UAX#31:
http://www.unicode.org/reports/tr31/tr31-33.html

Section 2.3.1 discusses potential tightening of restrictions in regard to
A1, A2 or B (use of ZWNJ or ZWJ within IDs in certain contexts). The last
paragraph says the following:

"Comparison. Typically the identifiers with and without these
characters should compare as equivalent, to prevent security issues."

Examples given in the preceding descriptions of A1, A2 and B included cases
in which strings with or without the joiner were both linguistically valid;
e.g., Farsi words for "names" and "a letter". But a constraint on comparison
is, in effect, preventing a distinction from being made: strings with and
without the joiner are to be treated as the same ID.

That seems to amount to saying that the joiners should only be kept when
displaying IDs as typed by a user. In that case, it seems like this
paragraph in 2.3.1 should suggest that.

In addition, it seems like it would make sense for 2.3.1 to mention layout
and format control characters, when permitted in IDs, as a potential basis
for distinguishing between display format and comparison format.

Recommended UTC actions

1. Authorize a proposed update of UAX #31 for Unicode 14.
2. AI for Rick and the ed committee: Start & post a proposed update of UAX #31 for Unicode 14.
3. AI for Mark and the ed committee: In UAX #31 clarify when & why ZWJ/ZWNJ should be ignored vs. when not. See L2/20-240 item F4. For Unicode 14.
4. AI for Asmus and Michel: Provide a document proposing an option in UAX #31 to prohibit ZWJ/ZWNJ altogether, for identifier security.

Background information / discussion

Example from Mark: English-language identifier with ZWJ, want to ignore. Don’t want to ignore it when it’s important for the semantics of text, as in certain circumstances in Persian or Sinhala.

Data point from Asmus: At the root level for URLs, ZWJ/ZWNJ is prohibited for security.

F5: feedback on UAX#31, set notation

Feedback (verbatim)

Date/Time: Thu Jul 30 18:43:45 CDT 2020
Name: Peter Constable

Report Type: Error Report
Opt Subject: feedback on UAX#31, set notation

This feedback pertains to revision 33 of UAX#31:

In section 2, Table 2 includes descriptions of property values in terms of "set notation". This is introduced in the immediately-preceding paragraph:

"The second column provides ... an associated set notation for the class."

An example, the notation used for describing ID_Start:

"[\p{L}\p{Nl}\p{Other_ID_Start}-\p{Pattern_Syntax}-\p{Pattern_White_Space}]"

No explanation is provided for this notation. It might make sense to someone already familiar with Unicode and the notation from other contexts. For someone coming from, say, a mathematics background but without Unicode experience, this does not like any familar set notation. (Math convention is to use brace brackets to denote a set; that's also used in, e.g., Python.) There are many classes of readers that would get to this point in the doc and wonder where the notation is explained.

The doc continues with other use of the notation, without explanation. E.g., section 2.3, under A.1:

"This corresponds to the following regular expression (in Perl-style syntax): /$LJ $T* ZWNJ $T* $RJ/ where:

$T = \p{Joining_Type=Transparent}
$RJ = [\p{Joining_Type=Dual_Joining}\p{Joining_Type=Right_Joining}]
$LJ = [\p{Joining_Type=Dual_Joining}\p{Joining_Type=Left_Joining}]"

The first hint—if the reader recognizes it as such, is a mention in section 2.4, after Table 3b, of "UnicodeSet syntax".

"In UnicodeSet syntax, the characters in these tables are:

Table 3: [\$_]
Table 3a: ['\-.\:·‿״׳-’·゠・]
Table 3b: [\u200D ʹ]"

This appears to be the same notation, but referred to in a different way: "UnicodeSet syntax" (versus "set notation" earlier—same notation? Or different?).

This appears to be using the "UnicodeSet notation" specified in section 5.3.3 of UTS#35

http://unicode.org/reports/tr35/#Unicode_Sets

If that is what is intended, then:

- UAX #31 should give an introduction to the notation and reference to the specification for it at or before the first usage of the notation.
- UAX #31 should use consistent terminology for how it refers to the notation; if an informal expression is preferred, then that should be introduced when the notation is first introduced.

(E.g., "At several points in this document, character classes will be described using UnicodeSet notation (hereafter, "set notation"). This notation is defined in [UnicodeSets].")

Recommended UTC actions

1. AI for Mark and the ed committee: In UAX #31 more clearly and consistently refer to CLDR for UnicodeSet syntax, according to L2/20-240 item F5. For Unicode 14.

Background information / discussion

We should point to the CLDR spec for the UnicodeSet notation, as a concrete syntax (rather than UTS #18 which provides many options for syntax).

F6: IdentifierType of Balinese musical symbols

Feedback (verbatim)

Date/Time: Tue Aug 4 17:50:07 CDT 2020
Name: Manish Goregaokar
Report Type: Error Report
Opt Subject: IdentifierType of Balinese musical symbols

In IdentifierType.txt:

1B6B..1B73 ; Limited_Use # 5.0 [9] BALINESE MUSICAL SYMBOL COMBINING TEGEH..BALINESE MUSICAL SYMBOL COMBINING GONG

These should probably be "Limited_Use Technical", not just Limited_Use

Recommended UTC actions

1. AI for Mark: In security/.../IdentifierType.txt, for U+1B6B..U+1B73 add Identifier_Type=Technical as proposed in L2/20-240 item F6, unless other UTC action items about Identifier_Type classifications contradict this. For Unicode 14.

F7: UTS #46 should validate ACE label edge cases

Feedback (verbatim)

Date/Time: Fri Aug 14 16:04:06 CDT 2020
Name: Markus W Scherer
Report Type: Error Report
Opt Subject: UTS #46 should validate ACE label edge cases

The IDNA2008 ToUnicode operation validates ACE labels ("xn--" plus Punycode)
by decoding them, then re-encoding via ToASCII, and verifying that the
round-trip output is the same as the input (case-insensitive).

The UTS #46 ToUnicode operation and its Processing step uses a cheaper
Convert/Validate step which wants to be equivalent.

However, it misses two edge cases which pass Convert/Validate step but which
IDNA2008 catches with its round-trip verification:

1. "xn--" decodes to an empty string
2. "xn--ASCII-" decodes to just "ASCII"

I propose that we modify
https://www.unicode.org/reports/tr46/#ProcessingStepPunycode (section 4
Processing > step 4 "Convert/Validate" > If the label starts with
“xn--”) so that it catches these cases.

Note that it is possible to check for these cases before/without
Punycode-decoding the label, except that, for equivalent error handling,
"xn---" should be skipped, letting Punycode decode fail instead. (In
IDNA2008 ToUnicode, a Punycode decode error preempts the round-trip
verification, and a quirk in the decoding procedure lets the "last
delimiter" slip into the main decoding loop if that delimiter immediately
follows the ACE prefix. The loop fails because the hyphen is not a valid
Punycode digit.)

Recommended UTC actions

1. AI for Markus Scherer and the ed committee: Update UTS #46 to validate ACE label edge cases, see L2/20-240 item F7. For Unicode 14.

Background information / discussion

Agreed: UTS #46 should treat these cases as invalid, in a way that is equivalent to how they are treated in IDNA2008.

F8: Missing Indic shaping properties for Common script Vedic characters

Feedback (verbatim)

Date/Time: Sun Aug 16 18:43:48 CDT 2020
Name: Norbert Lindenberg
Report Type: Error Report
Opt Subject: Missing Indic shaping properties for Common script Vedic characters

The Vedic signs 1CE9..1CEC and 1CEE..1CF1 are missing Indic syllabic category
definitions in the Unicode 13.0 data. At least some of these characters are
attested in L2/07-343, figures 8H–8J, as carrying marks, so the default category
Other is incorrect for them. For others, the default category Other might be
correct, but if that’s the case, I think it would be preferable to explicitly
provide the value.

Recommended UTC actions

1. AI for Roozbeh: Re document L2/20-240 item F8, investigate what the right Indic shaping properties should be for certain Vedic characters. See also related AI 164-A63. For Unicode 14.

Background information / discussion

Norbert provided similar feedback for adjacent characters before, see L2/20-175 item F10 “Missing Indic shaping properties for Devanagari and Vedic characters”. That had resulted in **[164-A63] Action Item for** Roozbeh Pournader: Re document L2/20-175 item F10, Investigate what the right Indic shaping properties should be for certain Devanagari and Vedic characters.

Asmus: If there are modern/minority use characters, talk to Asmus & Michel about implications. Roozbeh would like to have documentation for relevant dependencies between properties and other specs. Asmus: Higher threshold to change properties of modern-use characters.

Example: UAX #31 (identifiers) “B. Allow ZWJ in the following context” is defined using the Indic_Syllabic_Category=Vowel_Dependent property value. Roozbeh points out that the Indic_Syllabic_Category is more driven by character identity, the positional category more by shaping behavior.

Documents

D1: Changing Indic Syllabic Category of Limbu Kehmphreng

L2/20-184 from Martin Hosken

Summary

This proposal, if accepted, will result in the Indic syllabic category of U+193A (᤺ SIGN KHEMPRENG) changing from Vowel_Dependent to Tone_Mark.

…

The proposal is to change the Indic syllabic category of U+193A (᤺ SIGN KHEMPRENG) from Vowel_Dependent to something that will result in a USE category of VM. There are a few options, but none adequately represent a vowel modifier (which is what KEMPRENG is). The most obvious to the author is Tone_Mark, but any category that results in a USE category of VM would be sufficient.

Recommended UTC actions

1. Forward document L2/20-240 item D1 to the script ad hoc for consideration.

D2: No U+FFFD Generation for Zero-Length Content between ISO-2022-JP Escape Sequences

L2/20-202 from Henri Sivonen

Summary

UTR #36: Unicode Security Considerations recommends treating consecutive state change sequences without text in between as an error, which may be handled by emitting U+FFFD. Chrome and Safari do this, using ICU. Internet Explorer doesn't. Firefox didn't used to but switched implementations. "After the change, the U+FFFD generation has been reported as a bug in the email context both when handling email subject and when handling email body." This seems to sometimes happen when encoded byte streams are concatenated, rather than concatenating text and then encoding it as a whole. For email subjects, a good workaround was found. For email body text the problem remains.

Proposal: Either of the following, preferring the first option.
1. "End state 1: Drop ISO-2022-JP State Transition-Related U+FFFD Generation Completely"
2. "End state 2: Drop U+FFFD Generation for Zero-Length Content in the ASCII State and Add U+FFFD Generation for Other Unnecessary Transitions"

Recommended UTC actions

We recommend not to make any changes.

Background information / discussion

http://www.unicode.org/reports/tr36/#Some_Output_For_All_Input — Section 3.6.2 "Some Output For All Input"

Section 3.6 “Secure Encoding Conversion”, including this recommendation, was added in UTR #36 version 9, dated 2010-08-04.

Origin:

UTC action item 117-A39 (Peter Edberg, Markus Scherer, Editorial Committee): Create text for a new section in UTR #36 on Unicode encoding conversion security issues. AI closed on 2009-11-13.

The minutes of UTC #117 (2008-nov-06) refer to L2/08-407 “Proposal to provide new UTC text discussing encoding conversion security issues”. On the agenda, this was item B.18 Proposal to provide new UTC text discussing encoding conversion security issues. The UTR #36 update was published for public review as PRI #154 which was closed on 2010-02-10.

The ICU converter behavior dates back to ICU-6175 “Security issue with empty segments in toUnicode converters for ISO-2022-JP, HZ, ...”, submitted by Peter Edberg (Apple) based on the requirements in RFC 1468. The priority was “critical”.

Peter implemented this for ICU 4.0 on 2008-mar-12 (in svn r23571, converted to GitHub commit 867af87…, see diffs; also unit test added in svn r23572 = git commit d47745…).

Markus: Given that this is a well-justified recommendation based on a major legacy encoding’s spec, has been documented in UTR #36 for 10 years, implemented in ICU for 12 years, and long used in a couple of major browsers, I recommend that we keep this recommendation. Note that adjacent, separately-encoded byte streams in an email body could be concatenated with an intervening space character.