UTC #166 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2021-jan-08

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), Asmus Freytag, Mark Davis, Christopher Chapman, Ken Whistler, Peter Constable, Andy Heninger

Public feedback

Feedback received via the Unicode reporting form, see L2/21-011 “Comments on Public Review Issues (September 23, 2020 - January 8, 2021)”.

F1: Unicode confusables data missing Sharp-S letter to Capital-B confusable

Recommended UTC actions

1. AI for Mark Davis: For Unicode 14 confusables.txt, add ß ~ β and ß ~ B if feasible.

Feedback (verbatim)

Date/Time: Wed Sep 23 12:23:38 CDT 2020
Name: Wes
Report Type: Error Report
Opt Subject: Unicode confusables data missing Sharp-S letter to Capital-B confusable

It seems the confusables at ftp://ftp.unicode.org/Public/security/latest/confusables.txt
failed to include the German sharp-S or Eszett letter (https://en.wikipedia.org/wiki/%C3%9F)
as a possible confusable with the latin capital "B".
This is a fairly obvious confusable, and the wikipedia article even mentions

"Not to be confused with the Latin letter B." at the top of the article.

Would it be possible to add this to the official Unicode confusables data mapping ?

Background information / discussion

Asmus: Should be considered in context with capital sharp S (which would be even more of a confusable). However, not sure whether the latter is PVALID. Even if not, should be noted.

Mark: I think this is omitted because the NFKC_CF form of ß is "ss". But would take more investigation. That investigation would simplest if the equivalence were added, then retracted if the tests break.

So: Recommend an action to add ß ~ β and ß ~ B if feasible.

F2: Emoji properties missing from UTS #18

Mark Davis sent this directly to Markus Scherer

Recommended UTC actions

1. AI for Mark Davis and the editorial committee: For Unicode 14, modify the table in UTS #18 section 2.7 Full Properties to add
    a. ***RGI_Emoji_Flag_Sequence**** and
    b. ***Emoji_Keycap_Sequence****

Feedback (verbatim)

Email from Mark Davis on 2020-oct-22 “Missing properties in TR18”

I noticed that we are missing some emoji properties in https://unicode.org/reports/tr18/#Full_Properties

For completeness, we should have those listed in:
1. https://unicode-org.github.io/unicode-reports/tr51/tr51.html#Emoji_Properties
2. Plus those in ED-20, ED-21, ED-22, ED-23, ED-24, ED-25, and ED-27

We are missing two of them:

ED-21. emoji keycap sequence set
ED-23. RGI emoji flag sequence set.

Background information / discussion

These were overlooked in the previous version, but are the missing partitions of RGI Emoji.

F3: Javanese script: limited use → recommended?

Recommended UTC actions

1. Respond with this information:
   UTC's understanding of the ICANN process (as explained in "Maximal Starting Repertoire - MSR-4: Overview and Rationale") is that a script with Identifier_Type=Recommended does not automatically mean that it is approved for the Root Zone (nor does a script being Limited_Use mean that it cannot be included). Therefore, the requesters are encouraged to work with ICANN and provide the necessary evidence for "wide-spread everyday common use". The issue of changing the Identifier_Type in Unicode from Limited_Use to Recommended is a separate one, but would also require similar evidence. The UTC is in the process of formalizing the criteria.

2. AI for Asmus Freytag: Begin work on a set of criteria to be met by scripts to be considered for Identifier_Type=Recommended. Determine what information needs to accompany any proposal to reclassify.
   a. Note: On jan08, Asmus submitted L2/21-030 "Requirements and Process for Changing Script Status for Identifier Use".

Feedback (verbatim) — dec14

From: PANDI ID Registry
Sent: Monday, December 14, 2020, 12:22:02 AM PST
Subject: Re: PANDI Inquiries

We would like to know how can we increase the status of the Javanese script from limited use to recommended?
should we send more evidence that the script are still actively being used by the community?
because we needed it As soon as possible for our IDN process to ICANN.

Thank you so much.
waiting for your reply

Best regards,
Alicia Nabilla
Business Development
PANDI .id Registry

Feedback (verbatim) — jan08

Date/Time: Fri Jan 8 03:52:33 CST 2021
Name: Alicia
Report Type: Error Report
Opt Subject: Javanese Script on table 7, should be on table 5

Dear UNICODE,

we are PANDI, a new association member on UNICODE, we registered in the first place to endeavour Indonesian Scripts to be able to be use on digital platforms. starting with Javanese script and having it appear on the table 7 confused us as it is being used widely even in the digital platform,
these are some of the websites evidences:

ꦢꦼꦤꦿꦏꦫꦲꦤꦶꦱ꧀ꦭꦸ.id
ꦩꦼ.id
ꦗꦮ.id
ꦥꦸꦗꦁꦤꦸꦠ꧀ꦤ.id
ꦩꦶꦤꦲꦫꦗꦮ.id
ꦗꦏꦸꦥ꧀ꦮ.id
ꦏꦿꦶꦲꦤ꧀ꦢꦁ.id
ꦩꦗ.id
ꦼꦤ꧀ꦢꦸꦮꦶꦪꦸꦝꦶ.id
ꦏꦶꦫꦏꦶꦲ꧀ꦤꦼꦤ꧀.id
ꦤꦥꦠꦼꦏ.id
ꦩꦫꦥꦁ.id
ꦫꦏꦁꦫꦠꦶ ꦏꦿꦸ.id
ꦫꦮꦶꦲꦸꦤ꧀ꦠꦠ꧀ꦤ.id
ꦥꦲꦸꦏꦶꦤ꧀ꦢꦿ.id
ꦗꦮꦏꦏꦶꦱꦿ.id
ꦩꦶꦏꦗꦕꦁ.id
ꦪꦲꦶꦪꦲꦶꦠꦤ.id
ꦩꦶꦕꦩꦶꦕꦶꦗꦮ.id
ꦫꦮꦶꦫꦗꦶꦤꦸꦗ.id
ꦢꦼꦪꦏꦩꦸꦪꦪꦸꦭ.id
ꦤꦪꦠ.id
ꦩꦸꦏꦠꦼꦏ.id
ꦪꦸꦮꦶꦮꦶꦤ꧀.id
ꦥꦕꦸꦕꦩꦶꦪꦕꦲꦤ꧀.id
ꦲꦶꦪꦸꦠꦼꦏ.id
ꦗꦮꦫꦲꦶꦫꦠ.id
ꦥꦿꦲꦩꦶꦲ.id
ꦥꦤꦸꦗꦏꦸꦥ꧀ꦮ.id
ꦲꦤꦩꦏꦠ.id
ꦢꦼꦏꦏꦸꦠꦏꦶ.id
ꦲꦠꦼꦏꦫꦥꦤꦶ.id
ꦲꦏꦪꦲꦲꦸ.id
ꦲꦪꦸꦢꦼꦠꦼꦏ.id
ꦩꦩꦏꦠꦤ꧀ꦏꦶꦕꦤ꧀.id
ꦪꦤꦶ.id
ꦲꦤꦶꦥꦿꦪꦏ.id
ꦕꦿꦸꦩꦏꦶꦢꦼꦠꦼꦏꦗꦮ.id
ꦠꦸꦠꦕꦸꦤꦪꦤ꧀.id
ꦲꦸꦕꦶꦩꦸꦩ꧀.id
ꦫꦥꦁꦏꦿꦶꦪꦲꦸꦩ꧀.id
ꦢꦼꦤꦶꦩꦸꦏ.id

ꦥꦤꦶꦁꦤꦶꦗꦸꦮ.id
ꦗꦮꦤ.id
ꦥꦗꦿꦤ.id

Other than that, pointing to https://www.unicode.org/reports/tr31/ where Javanese is listed as 'Limited Used scripts' (table 7) when it should be on the table 5 (Recommended scripts) based on the Iso 10646 evidences. if there are also more information about what can we input to give more evidence please do mention on your answer.

Best Regards,
Alicia Nabilla
PANDI .id-Registry
Icon Business Park, LT1-LT2 Cisauk, BSD, Tangerang, Indonesia.

Background information / discussion

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

Example for prior discussion: unicore thread 20200629+ “UTS 39: Inclusion of Identifier_Type=Limited_Use amongst Identifier_Status=Restricted”

Asmus:

The ICANN process for Root Zone Label Generation Rules is currently limited to scripts from the Recommended set in UAX #31. However, that relationship is not 1:1, for example “Bopomofo” was excluded as not being appropriate for the Root Zone. (As explained in “Maximal Starting Repertoire - MSR-4: Overview and Rationale”). Additionally, that same document identifies all Excluded scripts as “permanently excluded from the Root Zone”, at least as long as they retain that status in Unicode. For the Limited Use scripts, the panel appointed by ICANN to make this determination declares a “neutral” attitude.
It is clear from reading the MSR document that the number of supported scripts has grown: The set in MSR-1 was a much smaller subset of the Recommended scripts. The process therefore appears to allow additions, however, beyond the detailed discussion given in the MSR itself on what makes a code point eligible to be part of a Root Zone label, there is no specific process outlined for evaluating limited-use scripts. Presumably, the same evaluation criteria used to ratify each of the Recommended scripts would be applicable. Therefore, it is clearly not required for Unicode to change the status of Javanese, solely so that it could be considered for the Root Zone.
The question before UTC then reduces to whether to raise the Identifier_Type of Javanese to Recommended based on Unicode’s own criteria for a Recommended script. It may be that these criteria would need to be revisited.
A separate UTC document [L2/21-030 “Requirements and Process for Changing Script Status for Identifier Use”] outlines some of the information about a limited use script that should probably go into such an evaluation (and should be documented in the context of making any such decision).
The case for such a change made for Unicode as a global source for other standards would likely need to be even stronger than that for one specific area like ICANN. In particular, a dependent specification (like the Root Zone Label Generation Rules for ICANN) can always tailor the set by making specific exceptions for individual Limited_Use scripts on a case-by-case basis. And such exceptions may be different for each zone in the DNS.

A suggested response points out that Unicode is not the gate-keeper for the DNS.

F4: Statements of canonical & other equivalences

Recommended UTC actions

1. AI for Markus Scherer and the editorial committee: For Unicode 14, add & clarify terminology for testing for canonical and other equivalences in chapter 3 (such as sections 3.7, 3.11, and 3.13) and UAX #15, and use that terminology in appropriate places, as outlined in L2/21-012 item F4.

Feedback (verbatim)

Date/Time: Tue Dec 15 13:25:44 CST 2020
Name: Zach Lym
Report Type: Submission (FAQ, Tech Note, Case Study)
Opt Subject: Normalization Generics (NFx, NFKx, NFxy)

I have been tracking down the rationale behind the normalization choices in filesystems. One problem area is the misleading use of strict logician terminology. Take the definition of Unicode's caseless matching algorithm [D145]:

> A string X is a canonical caseless match for a string Y if and only if:
> NFD(toCasefold(NFD(X))) = NFD(toCasefold(NFD(Y)))

The W3C Canonical Case Fold Normalization algorithm is defined as being compatible with [D145], but uses NFC in the last step [w3c-charmod-norm], leading to an apparent contradiction. Even though Unicode explains that "case folding is closed under canonical normalization" it took me a long time to find that passage and convince myself that the W3C and Unicode matching algorithms are equivalent. I am not alone: Linux kernel hackers couldn't figure it out either [linux-norm]!

>> Is there any case where
>> NFC(x) == NFC(y) && NFD(x) != NFD(y) , or
>> NFC(x) != NFC(y) && NFD(x) == NFD(y)
>
>This is good question. And I think we should get definite answer for it prior inclusion of normalization into kernel.

I was originally going to propose additions to D145 textual description, cross-references to the implementation section, and adding discussion of W3C charmod-norm. However, I don't think this would help as the text is already quite dense and most people will just ignore everything outside the example anyway [minimalist-manual].

I would instead like to propose normalization form generics for use in pseudo code definitions:

```
NFx = NFD|NFC //NFx != NFy
NFKx = NFKD|NFKC
NFxy = NFD|NFC|NFKD|NFKC
```

Freestanding `X`/`Y` variables should be probably be replaced to disambiguate them from the `NFx` nomenclature. `s1`/`s2` would work but `foo`/`bar` is less dense:

```
NFx(caseFold(NFD(foo))) = NFx(caseFold(NFD(bar)))
```

`NFx` does not currently appear within the Unicode standard itself, but is used in the normalization technical note [UAX15]. However, **UAX15 defines `NFx` twice**, first as NFD|NFC|NFKD|NFKC and later on as NFD|NFC. I think the proposed convention gets the most mileage out of the nomenclature and is how I have seen `NFx` used in the real world [linus].

Thank you!
-Zach Lym

[w3c-charmod-norm]: https://w3c.github.io/charmod-norm/#CanonicalFoldNormalizationStep
[linux-norm]: https://lwn.net/ml/linux-fsdevel/20190206084752.nwjkeiixjks34vao@pali/
[minimalist-manual]: https://dl.acm.org/doi/10.1207/s15327051hci0302_2
[UAX15]: https://unicode.org/reports/tr15/
[linus]: https://lore.kernel.org/linux-fsdevel/CAHk-=wiFtZL5rK3T-HQPm0oG4vekDJEKS47P8BbzHSXt_6SHuA@mail.gmail.com/

Background information / discussion

Markus:

- Both NFD and NFC are transformations that preserve canonical equivalence, but they yield different normal forms.
    - We might want to say exactly this in chapter 3 and in UAX #15.
- The most explicit expression of this so far is probably in https://www.unicode.org/reports/tr15/#Design_Goals “Goal 1: Uniqueness”.
    - Turn the “if strings are equivalents, then…” into “if and only if” constructs.
- D145 is overly specific in its use of normalization functions for achieving less specific goals.
    - The outer NFD normalization functions establish canonical equivalence. Instead, we could use a binary function like isCanonicallyEquivalent(left side, right side) and then say that that can be tested by normalizing each side using the same function that produces a canonically equivalent normal form, such as NFD or NFC, or by other means as long as they yield the same answer.
        - **Mark**: I think this actually obscures it. Why have multiple options for that?
        - Chapter 3: We have D70 Canonical equivalent: Two character sequences are said to be canonical equivalents if their full canonical **decompositions** are identical.
        - That means toNFD(X) = toNFD(Y), so we might as well just say that.
        - Notes:

        - 1. I searched for "canonically equivalent" and failed to find all instances, since sometimes we use the form "canonical-equivalent"(s). Would be clearer if we used exactly 1 term for one concept.
        - 2. We also use sometimes "full canonical decomposition" (14 times) and sometimes "canonical decomposition" (40 times).
        - In our definitions we say that the full one requires applying decomposition mappings recursively (until no change). So these are very different. That means that there is important difference between them, and we should almost always be using "full canonical decomposition" or maybe "decomposition mapping" (the first level found in the data files.
    - **Markus**: The problem is that someone like the reporter who has not read and comprehensively understood everything about normalization is easily confused. They think that they must test for toNFD(X) = toNFD(Y) using exactly those functions, and need convincing that toNFC(X) = toNFC(Y) or possibly other means achieve the same purpose.
    - Maybe we need to beef up D67 & D70 and define isXyzEquivalent(x, y) functions?
    - And somewhere we should add a point that there are different ways to test these conditions.
    - Editorially, the challenge is that some of these definitions come before we define the normalization forms, so either we get circular definitions or we define functions and implementation tips much later than the concepts they test.
    - **Mark**: We have plenty of circularity in the definitions anyway, so that is not a blocker.
    - Not a bad idea to have a definition of "canonical equivalent to" early on. Could define as IFF toNFD(X) = toNFD(Y), but then also include in a note that this is also equivalent to toNFC(X) = toNFC(Y).
    - And the same mutatis mutandis for compatibility equivalence.
  - **The inner NFD function cannot always be replaced with NFC because that applies discontiguous composition and defeats the purpose**
    - We might want to point this out.
    - The following text already says that it can be optimized by only dealing with U+0345 and composites that include that.
    - We could also point to https://www.unicode.org/notes/tn5/#FCD.
- The same applies to D146 (with compatibility equivalence) and D147 (equivalence under NFKC_Casefold).
- In my opinion, using isCanonicallyEquivalent(left side, right side) would be much clearer than NFx and NFxy etc. and help better explain what are relevant and necessary steps.

Documents

D1: Proposal on material issues in UAX #14 relevant to French

L2/20-243 from Marcel Schneider

Recommended UTC actions

1. We recommend not to make any changes.

Summary

Chris:
The author raises concerns about the handling of spaces in the "6.2 Tailorable Line Breaking Rules", especially in the context of French usage. The author proposes that "Rules LB13 through LB18 should be aligned on current best practice by raising LB18 before LB13, and by deleting "SP*" in LB14–LB17." So, effectively, break after spaces regardless of adjacent punctuation.

The author also asserts that "the leading industry does not seem ready to implement rules LB13 through LB17". I checked ICU's text segmentation behavior, and it appears to implement the rules LB13 through LB17 correctly, and ICU is widely used in the industry, so I question this assertion.

Background information / discussion

Chris:
These rules have been in place since UAX #14 revision 19, which is dated 2006-08-22. I am concerned that changing such long-standing rules would have negative consequences, such as causing unexpected document reflow, for over 14 years worth of documents.

Also, these rules are tailorable as noted in the section title, and "can be tailored by a conformant implementations" as noted in the section introduction, so the author's concerns can be addressed by implementors tailoring their implementations to meet their target audience's needs.

Mark: I strongly second "These rules have been in place since UAX #14 revision 19, which is dated 2006-08-22. I am concerned that changing such long-standing rules would have negative consequences, such as causing unexpected document reflow, for over 14 years worth of documents." Changing the LB rules needs to be done only with extremely compelling evidence.

Andy: I also agree with the recommendation to not make the suggested changes. They are to sequences of spaces and punctuation that commonly occur; they would definitely be noticed.

D2: Variation Sequences for Combining Marks

Multiple documents:

- L2/16-162 Cleanup of constraints on variation sequences from Mark Davis
    - Discussed at UTC #147 2016-May with two AIs that are no longer open:
        - 147-A136 explanatory text: done in Unicode 9, but not with the exact proposed wording
        - 147-A137 superseded by 152-A5a “Draft a new section for Chapter 3 on variation selectors and variation sequences”
- L2/20-244 Variation sequences for combining marks from Norbert Lindenberg
    - Revised: L2/20-244r — This adds a further restriction that the first character must not itself be a variation selector.
- L2/20-247 Restrictions on base characters of variation sequences (L2/20-244) from Charlotte Buff
    - Reporting form feedback on this doc from David Corbett on Wed Sep 30 19:49:25 CDT 2020: A third option is to adopt a new rule that a composed code point (e.g. U+09CB BENGALI VOWEL SIGN O) may be the base of a variation sequence if and only if its decomposed trailing code point (e.g. U+09BE BENGALI VOWEL SIGN AA) is also the base of a variation sequence, and those two variation sequences are harmonized to represent essentially the same variation.

- L2/20-250 Recommendations to UTC #165 October 2020 on Script Proposals, item 22 (last page)

Recommended UTC actions

1. Approve in principle adding a formal definition of a variation sequence (etc.) including the five constraints for an initial character of a variation sequence in L2/21-012 item D2 “In summary, the initial character…”; for Unicode 14.
2. AI for Markus Scherer and the editorial committee: For Unicode 14, propose changes to the specification of variation sequences in TUS chapter 23.4 based on L2/21-012 item D2. Include examples of characters and sequences that are excluded.

Summary

The Unicode Standard, 23.4 Variation Selectors (pp. 898ff)
Mark, Norbert, and Charlotte point out problems with the requirements for the initial character:

- “Variation Sequence. A variation sequence always consists of a base character or a spacing mark (gc = Mc) followed by a single variation selector character.”
- “The initial character in a variation sequence is never a nonspacing combining mark (gc = Mn) or a canonical decomposable character. These restrictions on the initial character of a variation sequence are necessary to prevent problems in the interpretation of such sequences in normalized text.”

Background information / discussion

Markus:

- I agree with Norbert’s doc 244 that “the restriction against nonspacing combining marks is both too loose and too restrictive to meet its goal” as he explains. We should restrict against characters with ccc≠0 regardless of their General_Category. (That is what was proposed in L2/16-162, but not yet done.)
- I agree with Charlotte’s doc 247 pointing out problems with characters that have NFC_QC=Maybe.
- **In summary, the initial character of a variation sequence should be**
    - Graphic Character (TUS chapter 3 D50: gc=L, M, N, P, S, Zs)
    - Not a Variation_Selector
    - ccc=0 (does not reorder)
    - NFD_QC=Yes (does not decompose)
    - NFC_QC=Yes (does not get consumed in composition)
- Note: L2/16-162 had a similar list:
    - Variation Sequence. A variation sequence ~~always consists of~~ is a sequence of two characters, where the final character is a variation selector character. There are some additional constraints on the initial character:
    - 1. It must have a zero canonical combining class
    - 2. It must not be a variation selector
    - 3. It must not be a canonical decomposable character
    - 4. It must not have a General Category value of Other (Cc | Cf | Cs | Co | Cn)
    - These constraints are required because it is important that variation sequences remain stable under normalization, and that the effects of variation selector can always be characterized as unambiguously applying to a single character. Versions of the Unicode Standard prior to version 9.0 had a more limited statement of constraints on variation sequences.

- It is theoretically possible that a new pair of characters A and B get encoded with a two-way canonical decomposition A=BC with an existing character C that used to fulfill the requirements for a variation sequence base character but consequently changes from NFC_QC=Yes to Maybe. We should add a test to the Unicode Tools that all variation sequence base characters continue to fulfill the requirements.

Asmus does not want to treat “this simply as a mathematical exercise in canonical equivalency. We must not only document the restriction, but also their practical effect, even if the latter doesn't change anything about the fact that the restrictions are required and necessary.”
For example, neither the current nor the proposed rules allow for a variation sequence for different shapes of the acute accent according to French vs. Polish typographic conventions. We need to make sure this is spelled out in terms of the actual typographic variation people may be familiar with, not merely in terms of normalization equivalence for the encoding. (and also not simply for hypotheticals).

For examples of characters where the old and proposed rules apply differently, see the documents from Norbert and Charlotte.

These lists address a different and essential aspect, because they document a formal effect of the proposed change. They don’t address the issue of known typographical variants that cannot ever (could never) be addressed with the variation sequence method. Both need to be documented.
——
Markus: Manual check on first characters in StandardizedVariants.txt vs. proposed requirements — the following set comes up empty (as desired). This is the set of each initial character, except CJK, and removing all characters that fulfill the requirements.
[[0\u2205\u2229\u222A\u2268\u2269\u2272\u2273\u228A\u228B\u2293\u2294\u2295\u2297\u229C\u22DA\u22DB\u2A3C\u2A3D\u2A9D\u2A9E\u2AAC\u2AAD\u2ACB\u2ACC\uFF10\u3001\u3002\uFF01\uFF0C\uFF0E\uFF1A\uFF1B\uFF1F\u1000\u1002\u1004\u1010\u1011\u1015\u1019\u101A\u101C\u101D\u1022\u1031\u1075\u1078\u107A\u1080\uAA60-\uAA66\uAA6B\uAA6C\uAA6F\uAA7A\uA856\uA85C\uA85E\uA85F\uA860\uA868\x{10AC5}\x{10AC6}\x{10AD6}\x{10AD7}\x{10AE1}\u1820-\u1826\u1828\u182A\u182C\u182D\u1830\u1832\u1833\u1835\u1836\u1838\u1844-\u1849\u184D\u184E\u185D\u185E\u1860\u1863\u1868\u1869\u186F\u1873\u1874\u1876\u1880\u1881\u1887\u1888\u188A]-[[:ccc=0:]&[:NFD_QC=Yes:]&[:NFC_QC=Yes:]]]

CJK unified ideographs and emoji variation sequence initial characters are known/assumed to fulfill the requirements as well.

D3: UTS #18 Editorial additions

L2/21-002 UTS #18 Editorial additions from Mark Davis
L2/21-003 UTS #18 Editorial additions:EBNF from Mark Davis

Recommended UTC actions

1. Action item for Mark Davis and the editorial committee: Make editorial changes and EBNF clarifications to UTS #18 as proposed in L2/21-002 and L2/21-003, in a new proposed update.

Summary

Proposed changes for 3 areas in **UTS #18 Unicode Regex**:
1. Adding examples and clarifying text for Annex D: Resolving Character Classes with Strings

2. Adding a new Annex F. Parsing Character Classes
3. Cleaning up some omissions, typos, and stylistic inconsistencies in the EBNF

Public Review Issues

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

P1: UAX#39 wording issues

See https://www.unicode.org/review/pri423/

Recommended UTC actions

1. AI for Mark Davis and the editorial committee: Add to the proposed update for UTS #39 text to address the PRI #423 feedback from Asmus on 2020-dec-31. Also add the term “widespread” to “everyday common use”.

Feedback (verbatim)

Date/Time: Thu Dec 31 17:50:56 CST 2020
Name: asmus
Report Type: Error Report
Opt Subject: UAX#39 wording issues

In Table 1 of UAX#39 there are obfuscatory and possibly incorrect uses of “explicit”. In the definition of “Limited_Use” the text

> and no explicit script from Table 5, Recommended Scripts.

Should be changed to

> and no script from Table 5, Recommended Scripts, other than “Common” or “Inherited”.

That way, the reader does not have to hunt for the definition of “explicit” script in UAX#24. Because “explicit” is not capitalized, it cannot be understood by the reader as a defined term, so anyone not familiar with UAX#24, will not even understand that it means anything different than “explicitly listed in Table 5”. Also, if it is desired to make it a defined term, it should be given an actual (numbered) definition in UAX#24, not simply a gloss.

Likewise for Excluded the language:

> and no explicit script from Table 7, Limited Use Scripts or Table 5, Recommended Scripts.

should be changed to

> and no script from Table 7, Limited Use Scripts or Table 5, Recommended Scripts, other than “Common” or “Inherited”.

In the definition of Recommended, the text

> containing an explicit script in Table 5, Recommended Scripts in [UAX31], except for those characters that are Restricted above.

appears to rule out the ASCII digits or indeed combining marks as part of identifiers (script=Common or inherited). Here the remedy would simply be to drop the word “explicit”.

While making the change, there is an inconsistency between the phrasing of the description of Limited_Use and Recommended.

Compare:

> Characters from scripts that are in limited use: with Script_Extensions values containing a script in Table 7, Limited Use Scripts in [UAX31], and no explicit script from Table 5, Recommended Scripts.

to

> Characters with Script_Extensions values containing an explicit script in Table 5, Recommended Scripts in [UAX31], except for those characters that are Restricted above.

It would make the intent clearer if the description echoed the rationale for making a script recommended. As follows:

> Characters from scripts that are in everyday common use: with Script_Extensions values containing a script in Table 5, Recommended Scripts in [UAX31], except for those characters that are Restricted above.