UTC #177 properties feedback & recommendations

Markus Scherer & Josh Hadley / Unicode properties & algorithms group, 2023-oct-25

Participants

The following people have contributed to this document:

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

1. Core spec

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

L2/21-038 from Tom Honermann (C++ SG16)

Recommended UTC actions

1. Action Item for Robin Leroy, EDC: In the core specification, amend the text of Section 2.6, sub “Byte Order”, and Section 2.13, sub “Unicode Signature”, as described in Option 1 of L2/21-038. In addition, in Section 3.10, reword the third bullet under D95, changing “neither required nor recommended” to “not required”. For Unicode 16. See L2/23-234 item 1.1.
2. Action Item for Robin Leroy, EDC: In the core specification, section 23.8 Specials, add guidance for the use of a BOM in UTF-8 similar to L2/21-038 option 2. For Unicode 16. See L2/23-234 item 1.1.

Summary

From the abstract of the document:

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.

PAG recommended changes

We agree with the core spec changes in option 1 of the document, plus adding guidance similar to option 2 to core spec section 23.8 Specials.

We might modify the first bullet on L2/21-038 page 6 recommending that consumers strip the BOM (as in other bullets) rather than diagnose it as an error.

Consider for UTF-8, where we already explain that byte order does not apply, whether to use the term “signature” (short for “Unicode signature byte sequence”) rather than “BOM”. (Core spec section 23.8 Specials already uses “signature” and “signature byte sequence”.)

1.2 Remove ambiguity from D14 Noncharacters for Unicode 16.0

L2/23-201 by Asmus Freytag (copied here in full)

Recommended UTC actions

1. Action Item for Asmus Freytag, EDC: Clarify D14 Noncharacters according to L2/23-201 Alternative A. For Unicode 16. See L2/23-234 item 1.2.

Summary

Problem

The word “internal” is ambiguous in definition D14 Noncharacters.
In reviewing an IETF internet draft I came across language that cited the “reserved for internal use” language from D14. However, without the context of text passages in §2.13 “Special Character“ or §23.7 “Noncharacters” the meaning of this reservation is not unambiguous to the reader of that internet draft.
In the context of a body of work defining protocols (like the IETF RFCs) “reserved for internal use” may refer to items that are reserved to be used solely for protocol internal purposes, whereas the sense in Unicode encompasses use that is internal to a system or application (or protocol) that implements the Unicode Standard. Instead of noncharacters being somehow for internal use by the Unicode Standard, they are internal in the sense of not being intended for interchange.
The existing language does not make that distinction, or, in other words, does not prevent a reader from applying their understanding of the word “internal”, leading to an ambiguity.

D14 Noncharacter: A code point that is permanently reserved for internal use. Noncharacters consist of the values U+nFFFE and U+nFFFF (where n is from 0 to 10 [base] 16) and the values U+FDD0..U+FDEF.
with one of the bullets just repeating the statement without any help in disambiguation
• These code points are permanently reserved as noncharacters

Other text in the Standard spells out the key features of noncharacters in different ways that make the intent much clearer and rule out alternative interpretation of the word “internal”. Those statements describe noncharacters as follows:

1. They are code points that can never be assigned to abstract characters
2. They are not intended for interchange
3. Applications may (freely) use them for application internal purposes
4. Some may be suitable as sentinel values
5. If encountered outside of an implementation they have no interpretable semantics other than their status as noncharacter code point.

Preferred Solution

The preferred fix would be to update the language of D14 as follows:

D14 Noncharacter: A code point that is permanently reserved and will never be assigned to an abstract character. Noncharacters consist of the values U+nFFFE and U+nFFFF (where n is from 0 to 10 16) and the values U+FDD0..U+FDEF. They are not intended for interchange, but may be used by an implementation for internal purposes.
• Possible use cases include application internal sentinel values
• For more information, see Section 23.7, Noncharacters.

Alternatives:

A. Same as preferred solution, but with the final sentence moved into a bullet
B. Same as existing, but with new bullets covering items (1) through (5) added and the existing duplicative bullet removed.

Glossary

Change the glossary item to match the suggested language for the “preferred solution” instead of repeating the current definition with its ambiguous use of the word “internal”.

Discussion

We do want most low-level tools and protocols to treat noncharacters the same as private use or as unassigned code points.

Example of CLDR using noncharacters: We have some additional contractions in the CLDR version of the default sort order data, and in some tailorings. These contractions map to collation elements with primary weights at the start of each script, and each CLDR reordering group (spaces, digits, ...), in order to support parametric script reordering and alphabetic-index processing. In order not to interfere with real text, and to make it easy to enumerate them, each of the contraction strings starts with one of two noncharacters. They can show up unescaped in some data files.

Including noncharacters in a transformation form does not make it ill-formed. May want to reject some for internal use. Similar to other types of input validations.

2. UCD

2.1 Future maintenance of UAX #42 (UCDXML)

UTC-176-A5 Action item for Markus Scherer, PAG: Evaluate alternatives for future maintenance of UAX #42 and provide a recommendation to UTC at meeting #177.

Recommended UTC actions

1. Consensus: Authorize a Public Review Issue announcing the stabilization of UAX #42 UCDXML with spec and data frozen at Unicode 15.1.
2. Action Item for Markus Scherer, Rick McGowan, PAG: Post a Public Review Issue announcing the stabilization of UAX #42 UCDXML with spec and data frozen at Unicode 15.1. For Unicode 16.0. See draft PRI text in L2/23-234 item 2.1.

Summary

During UTC-176, PAG noted that so far none of its members had committed to maintaining UAX #42 (UCDXML).
Ken Whistler noted that, if no new (capable, committed) maintainer can be found, the proper approach would be to "stabilize" UAX #42 (keeping old versions available and documented).
We could issue a PRI announcing this step and giving reviewers an opportunity to respond if they are able and committing to taking over maintenance of UAX #42. We could also write a blog post about it to add visibility.

We might point to ICU's "preparsed UCD" (https://unicode-org.github.io/icu/design/props/ppucd) as a possible alternative format for the UCD, although its intended scope is currently limited to use in ICU.

Draft PRI text

PRI title:
Stabilization of UAX #42, Unicode Character Database in XML (UCDXML)

Description of Issue:
As noted in L2/23-187 Release Management Group Report to UTC #176, the editors of UAX #42 are no longer available to continue maintaining the spec and data for future versions of Unicode.
In the absence of committed maintainers, the UTC is proposing to stabilize UAX #42 and freeze the UAX and its data at the Unicode 15.1 level.
Users of UCDXML are encouraged to either parse the UCD files directly or use libraries that provide API access to Unicode properties.
Please let us know of other standards or projects which refer to UAX #42 or use the UCDXML data.

(When we actually publish the PRI, we should elaborate on what it means to stabilize a UAX.)

2.2 Changes for new characters in 16.0, continued

From Robin Leroy, Ken Whistler, et al., PAG

Recommended UTC actions

1. Consensus: The UTC approves correcting the Indic_Syllabic_Category range `13B8..113BA ; Vowel_Dependent` (spanning 65539 code points) originally proposed in L2/22-031 and approved by UTC-170-C9, to `113B8..113BA ; Vowel_Dependent` (spanning three code points). For Unicode Version 16.0. See L2/23-234 Section 2.2.
2. Note: The first proposed line of UnicodeData.txt in L2/23-191 has one semicolon too many; U+A7DB should be the Simple_Lowercase_Mapping of U+A7DA, not its Simple_Titlecase_Mapping.
3. Consensus: The UTC approves the code point change for GARAY HYPHEN to U+10D6E, from the conflicting U+10D6D approved by UTC-171-C18. GARAY CONSONANT NASALIZATION MARK remains at U+10D6D. For Unicode Version 16.0. See L2/23-234 Section 2.2.
4. Consensus: The UTC approves the change in canonical combining class for the GARAY DIGITS ONE through NINE U+10D40..U+10D49, from equal to their numeric value as approved by UTC-171-C18 to 0 (Not_Reordered). For Unicode Version 16.0. See L2/23-234 Section 2.2.
5. Consensus: The UTC approves the change in Titlecase_Mapping for U+1C8A CYRILLIC SMALL LETTER TJE from itself as approved by UTC-172-C4 to U+1C89 CYRILLIC CAPITAL LETTER TJE. For Unicode Version 16.0. See L2/23-234 Section 2.2.

6. Consensus: The UTC approves the change in General_Category for U+106DF GARAY REDUPLICATION MARK from Other_Symbol (So) as approved by UTC-171-C18 to Modifier_Letter (Lm). For Unicode Version 16.0. See L2/23-234 Section 2.2.
7. Consensus: The UTC approves the change in General_Category for U+10D4E GARAY VOWEL LENGTH MARK from Other_Letter (Lo) as approved by UTC-171-C18 to Modifier_Letter (Lm). For Unicode Version 16.0. See L2/23-234 Section 2.2.
8. Note: The Script_Extensions values suggested in L2/21-157R should not be added to the UCD; see the comments about a similar proposal in Section 6 of L2/23-012.

Summary

The PAG spotted some issues while preparing the UCD changes for the Unicode Version 16.0 pipeline.

There is a bad InSC range in L2/22-031:

```
Unset
13B8..113BA ; Vowel_Dependent # Mc [3] TULU-TIGALARI VOWEL SIGN AA..TIGALARI VOWEL SIGN
VOCALIC II
```

This should be

```
Unset
113B8..113BA ; Vowel_Dependent # Mc [3] TULU-TIGALARI VOWEL SIGN AA..TIGALARI VOWEL SIGN
VOCALIC II
```

instead of setting an entire plane to InSC=Vowel_Dependent.

See unicode-org/unicodetools@d0fcc53.

There is one semicolon too many in the first proposed UnicodeData.txt line in L2/23-191; see unicode-org/unicodetools@044597d.

There is a duplicate code point in the proposed UnicodeData.txt lines in L2/22-048. See unicode-org/unicodetools@bbfffdc.

Further, the digits are given CCC=NV, which is a bad idea for the nonzero digits.

The CYRILLIC SMALL LETTER TJE approved by UTC-172-C4 has a Titlecase_Mapping to itself rather than to the CAPITAL LETTER TJE; see unicode-org/unicodetools#554. This was caught by comparison against Ken's UnicodeData draft.

Scripsit Ken Whistler:

The gc value for 106DF GARAY REDUPLICATION MARK in L2/22-048 is clearly wrong.

It is assigned gc=So there, but iteration and reduplication marks should typically be assigned gc=Lm and Extender=True.

The mark in question is discussed under Section 4.1 Punctuation, but it clearly is neither punctuation nor a symbol. It just *looks like* a symbol in the author's mind because it is shaped like an x.

In discussion of the Garay length mark, we concluded that it should be gc=Lm instead of gc=Lo.
See unicode-org/unicodetools#552 (comment)

2.3 Discrepancy between the numeric values for U+5146 and U+79ED

Recommended UTC actions

1. Action Item for Josh Hadley, PAG: Update the documentation for DerivedNumericValues.txt to indicate that the first-listed value of kAccountingNumeric, kOtherNumeric, or kPrimaryNumeric is what is used for the derived numeric value. For Unicode 16.0. See L2/23-234 item 2.3.
2. Action Item for Ken Whistler, PAG: Update UAX #44, Table 9, description of Numeric_Value and the description following Table 10 to indicate that the value comes from the first-listed of kAccountingNumeric, kOtherNumeric, or kPrimaryNumeric. For Unicode 16.0. See L2/23-234 item 2.3.
3. Action Item for Eric Muller, Laurentiu Iancu, PAG: update the UCDXML generator and data to list nv=Numeric_Value with at most one number. For Unicode 16.0. See L2/23-234 item 2.3.

Feedback (verbatim)

Date/Time: Tue Aug 22 05:57:09 CDT 2023
ReportID: ID20230822055709
Name: Andrew West
Report Type: Error Report
Opt Subject: DerivedNumericValues.txt

For Unicode 15.1, there is a discrepancy between the numeric values for U+5146 and U+79ED as given in DerivedNumericValues.txt (single value only) and Unihan and ucd.xml (two values each):

https://www.unicode.org/Public/draft/UCD/ucd/extracted/DerivedNumericValues.txt:

```
Unset
5146          ; 1000000.0 ; ; 1000000 # Lo       CJK UNIFIED IDEOGRAPH-5146
79ED          ; 1000000000.0 ; ; 1000000000 # Lo       CJK UNIFIED IDEOGRAPH-79ED
```

Unihan_NumericValues.txt:

```
Unset
U+5146 kPrimaryNumeric     1000000 1000000000000
```

```
U+79ED kPrimaryNumeric     1000000000 1000000000000
```

ucd.nounihan.flat.xml:

```
Unset
      <char cp="5146" age="1.1" na="CJK UNIFIED IDEOGRAPH-#" JSN="" gc="Lo" ccc="0" dt="none"
dm="#" nt="Nu" nv="1000000 1000000000000" .../>
      <char cp="79ED" age="1.1" na="CJK UNIFIED IDEOGRAPH-#" JSN="" gc="Lo" ccc="0" dt="none"
dm="#" nt="Nu" nv="1000000000 1000000000000" .../>
```

The derived numeric value should be based on kPrimaryNumeric:

```
Unset
# Derived Property:   Numeric_Value
#  Field 1:
#    The values are based on field 8 of UnicodeData.txt, plus the fields
#    kAccountingNumeric, kOtherNumeric, kPrimaryNumeric in the Unicode Han Database (Unihan).
#    The derivations for these values are as follows.
#      Numeric_Value = the value of kAccountingNumeric, kOtherNumeric, or kPrimaryNumeric, if
they exist; otherwise
#      Numeric_Value = the value of field 8, if it exists; otherwise
#      Numeric_Value = NaN
```

However, the format of the file only allows for a single value.

My personal opinion is that Numeric_Value should always be a single value, even in cases such as U+5146 and U+79ED where there are alternative interpretations of the numeric value, otherwise implementations which rely on UCD data to apply numeric value (e.g. for numeric sorting) will not know which of the space-separated list of numeric values to apply.

My preferred solution would be:

1. Allow multiple alternative numeric values in the Unihan database only (i.e. no change to kPrimaryNumeric for U+5146 and U+79ED);
2. Allow only a single numeric value for Numeric_Value in DerivedNumericValues.txt, selecting the most widely-used modern interpretation for U+5146 and U+79ED, and modifying accordingly the stated derivation for the value given in Field 1;
3. Derive the "nv" value in ucd.xml from Numeric_Value in DerivedNumericValues.txt.

2.4 Bug with "cursor" listings in Character Name Index

Recommended UTC actions

1. Action Item for Ken Whistler, PAG: correct "cursor" listings in ucd/Index.txt as detailed in feedback ID20230720191514. For Unicode 16.0. See L2/23-234 item 2.4.
2. FYI: Already fixed in Index.txt for Unicode 16.0. (add Index entry: down, fast cursor unicodetools#545)

Feedback (verbatim)

Date/Time: Thu Jul 20 19:15:14 CDT 2023
ReportID: ID20230720191514
Name: Leroy D. Geisse V.
Report Type: Website Problem
Opt Subject: Missing character name variant

I think that this is a minor issue. Regards.

By searching for "cursor" in the Character Name Index (https://www.unicode.org/charts/charindex.html), I found is not the variant "down, fast cursor".

cursor down, fast
cursor left, fast
cursor right, fast
cursor up, fast

fast cursor down
fast cursor left
fast cursor right
fast cursor up

left, fast cursor

right, fast cursor

up, fast cursor

2.5 Inconsistent use of semicolon reported in data files

From David Carlisle (via Asmus email to PAG)

Recommended UTC actions

1. Action Item for Ken Whistler, PAG: In UAX #44 section 4.2.1 Data Fields, remove the sentence “For legacy reasons, no spaces are allowed before or after the semicolon in LineBreak.txt and in EastAsianWidth.txt.” For Unicode 16.0. See L2/23-234 item 2.5.

Feedback

david Carlisle davidc@nag.co.uk is updating the date file for the math TR and reports:

I found the general guidance on ; separated file formats in TR44
https://www.unicode.org/reports/tr44/#Format_Conventions
(Incidentally changes to LineBreak.txt and in EastAsianWidth.txt in the recent 15.1.0 break that)

2.6 Alphabetic combining letters

From Robin Leroy & Ken Whistler, PAG

Recommended UTC actions

1. Consensus: Assign the property Alphabetic to the combining Latin letters U+0363..U+036F and U+1DD3..U+1DE6. For Unicode Version 16.0. See L2/23-234 item 2.6.
2. Action Item for Robin Leroy, PAG: In PropList.txt, assign the property Other_Alphabetic to the combining latin letters U+0363..U+036F and U+1DD3..U+1DE6, and update derived properties accordingly. For Unicode Version 16.0. See L2/23-234 item 2.6.

Summary

The medieval superscript letter diacritics from the Combining Diacritical Marks and Combining Diacritical Marks Supplement blocks do not have the alphabetic property, in contrast to most other combining letters (including those used in the *Ormulum* in the corresponding Extended block).

They should.

3. New Scripts & Characters

3.1 Review of Script Ad Hoc topics

PAG members reviewed the following proposals, provided feedback to SAH, and the feedback has been addressed.
No further recommended actions from our side.

- L2/23-181 Encoding proposal for an extended Egyptian Hieroglyphs repertoire
- SAH issue: Invariants between Indic positional and syllabic categories and general category
    - See the SAH report for recommended property value changes for U+1171E, U+0D41, U+0D42
    - PAG is implementing an invariants check
- L2/23-197 Three Latin Lambdas for version 16.0 Request

3.2 Create a new UAX for Tangut source data

Recommended UTC actions

1. Consensus: The UTC authorizes a new UAX for documentation of TangutSources.txt, for Unicode 17.0. See L2/23-234 item 3.2.
2. Action Item for Michel Suignard, SAH: Create a proposed draft UAX for documentation of TangutSources.txt, for Unicode 17.0. See L2/23-234 item 3.2.

Feedback

Asmus Freytag noted that there is a need for a new UAX for Tangut source data.

Whenever we add data files for source data, we either need a dedicated UAX for the source information for that script, like UAX #38 for Unihan, or if we have a number of similar scripts, we could coalesce. A Tangut UAX might be lightweight compared to Unihan.

4. Normalization

4.1 document optimization of mapping+normalization

From Markus Scherer, ICU

Recommended UTC actions

1. Action item for Robin Leroy, Markus Scherer, PAG: In the Implementation Notes section of UAX #15, add a discussion of the implementation of operations combining normalization and folding, such as toNFKC_Casefold and toNFKC_Simple_Casefold, using the same logic as a normalization form. For Unicode 16.0. See L2/23-234 item 4.1.

Feedback

Markus noted: In ICU we implement NFKC_CF normalization by combining the normalization (NFC) data with the NFKC_CF to create a custom normalization data file. We use this file via the regular Unicode normalization implementation for a single-step NFKC_CF operation, rather than the two-step, map-then-normalize operation described in the spec.

I tried to implement NFKC_SCF (using Simple_Case_Folding) analogously, but I needed to manually tweak the data in order to achieve the same results as the specified algorithm.

I could not simply add the NFKC_SCF mappings to the normalization data, because the normalization data builder applies mappings recursively, which in this combination yields bad mappings for several characters that violate constraints of the normalization algorithm.

After PAG discussion:

- We should document the optimization technique of creating custom normalization data files for things like combining a mapping table with normalization.
- It would be nice if we could document for the standard combinations what mapping tweaks are needed to make this work.

5. Text Segmentation

5.1 Line break: Revisit change of some digits lb=NU -> ID

From PAG discussion

Recommended UTC actions

1. Consensus Change the Line_Break property of the Balinese, Javanese, Cham, and Dives Akuru digits, namely U+1B50..U+1B59, U+A9D0..U+A9D9, U+AA50..U+AA59, and U+11950..U+11959, from Ideographic to Aksara_Start. For Unicode Version 16.0. See L2/23-234 item 5.1.
2. Action Item for Robin Leroy, PAG: Change the Line_Break property of the Balinese, Javanese, Cham, and Dives Akuru digits from Ideographic to Aksara_Start. For Unicode Version 16.0. See L2/23-234 item 5.1.
3. Action Item for Robin Leroy, PAG: In UAX #14, Update the description of line breaking class Aksara_Start to mention that all digits of scripts that use the brahmic style of line breaking are assigned this class. For Unicode Version 16.0. See L2/23-234 item 5.1.

Summary

From PAG discussion of the new line breaking of certain scripts at orthographic syllable boundaries. See L2/22-080R2, L2/23-072, UTC-175-C27.

PAG members wondered why L2/22-080 gave some digits lb=ID and others lb=AS (from lb=NU).
There was some discussion of the behaviour of lb=ID with numeric prefixes, e.g., $, and postfixes, e.g., %, which is not objectionable, but not relevant, since these characters are not used in those scripts.
Ultimately the reason is that Norbert had started with lb=ID for all digits, and switched to lb=AS where necessary.

Having lb=AS for all would make the property assignments simpler, with no effect on nondegenerate cases; it would allow us to check future assignments with an invariant test.

5.2 UAX #29 Section 6.3: “easily converted”

Robert Grimm (via email to Christopher Chapman & Josh Hadley)

Recommended UTC actions

1. Action Item for Josh Hadley, PAG: update Section 6.3 of UAX #29 with improved wording about converting grapheme cluster rules into regular expressions, for Unicode 16.0. See L2/23-234 item 5.2.

Feedback (verbatim)

I've spent the last month or so building a tool to better visualize fixed-width rendering of Unicode, https://github.com/apparebit/demicode. As part of the effort, I implemented the grapheme cluster break

algorithm from TR 29 and discovered a bug in the handling of CR NL. Yesterday, I drafted a longer explanation of that bug, but today I discovered your draft revisions for 15.1. So I'm sparing you the explanation :)

Alas, I do have a suggestion for improving the exposition of TR29. Section 6.3 starts with the assertion that "The rules for grapheme clusters can be easily converted into a regular expression." When I first started implementing the grapheme cluster break algorithm, I found that assertion rather frustrating because the conversion wasn't obvious or easy to me. I had also hoped for more guidance about how to best implement the algorithm and, frankly, was underwhelmed by 6.3 (The approach I ended up taking was to translate the string to be broken into grapheme clusters into a string of grapheme cluster break properties represented by a single letter each and then use a plain regular expression, in my case Python's re. While the initial overhead isn't great, it sure beats having to implement a regex engine.)

Now that I have an implementation that passes all tests in GraphemeBreakTest.txt (thank you for that file, it was super helpful!), I am starting to see how the translation into a regular expression works. But having discovered the bug in table 1c myself, I believe that the assertion in 6.3 is plain wrong. It can't be that easy, since there was a bug that made it into Unicode 15.0. Given all the review a draft goes through that seems like a strong argument against the assertion.

In short, I'd recommend striking the "easily converted" text and instead maybe add a sentence on the intuition behind the translation (nonbreaks are lined up as regular expression sequences) as well as about proven implementation strategies. I included mine in parentheses above but am not sure what the best or even a good one would be here.

5.3 Line_Break assignments of U+1F8B0 and U+1F8B1

From Ned Holbrook, PAG, while reviewing the UCD changes for the Smalltalk proposal.

Recommended UTC actions

1. Consensus: Change the Line_Break property of U+1F8B0 ARROW POINTING UPWARDS THEN NORTH WEST and U+1F8B1 ARROW POINTING RIGHTWARDS THEN CURVING SOUTH WEST, as well as that of all unassigned code points in the Supplemental Arrows C block, from Line_Break=Ideographic (ID) to Line_Break=Alphabetic (AL). For Unicode Version 16.0. See L2/23-234 item 5.3.
2. Action Item for Robin Leroy, PAG: In LineBreak.txt, change the Line_Break property of U+1F8B0 ARROW POINTING UPWARDS THEN NORTH WEST and U+1F8B1 ARROW POINTING RIGHTWARDS THEN CURVING SOUTH WEST, as well as that of all unassigned code points in the Supplemental Arrows C block, from Line_Break=Ideographic (ID) to Line_Break=Alphabetic (AL). For Unicode Version 16.0. See L2/23-234 item 5.3.
3. Action Item for Robin Leroy, PAG: In the description of Line_Break property value Ideographic in UAX #14, review the documentation of the ranges that default to lb=ID and correct it as needed. For Unicode Version 16.0. See L2/23-234 item 5.3.

Summary

 U+1F8B0 ARROW POINTING UPWARDS THEN NORTH WEST
and

 U+1F8B1 ARROW POINTING RIGHTWARDS THEN CURVING SOUTH WEST,
added in Unicode Version 13.0, are pretty much the only arrows that are LB=ID (and in particular the only ones in the supplemental arrows blocks).

Having consulted with Ken Whistler, this appears to be an oversight in 13.0, the defaults for those blocks being weird. We should rectify it.
We are also going to pick LB=AL for the new 16.0 supplemental arrow from the Smalltalk proposal, instead of LB=ID from my and Ken's initial draft; since the proposal did not specify Line_Break values, this is still at our discretion and need not be brought to the attention of the UTC.

Ken also noted:
UAX #14's discussion of ID is incomplete in its statement of which ranges default to lb=ID.

6. Collation

6.1 UTS #10 obsolete example v=w

General feedback

Recommended UTC actions

1. Action Item for Markus Scherer, PAG: In UTS #10 section 1.5 Other Applications of Collation, make the statement about sorting vs. searching more generic, replacing the v=w example. For Unicode 16.0. See L2/23-234 item 6.1.

Feedback (verbatim)

Date/Time: Tue Sep 26 05:26:46 CDT 2023
ReportID: ID20230926052646
Name: Henri Sivonen
Report Type: Error Report
Opt Subject: UTS #10

Hi,

https://www.unicode.org/reports/tr10/tr10-49.html#Other_Applications_of_Collation
has this sentence: “For example, if v and w are treated as identical
base letters in Swedish sorting, then they should also be treated the
same for searching.”

This example has become obsolete. See
https://unicode-org.atlassian.net/browse/CLDR-17050 and links backwards
from there to issues and CLDR changesets concerning both Swedish and
Finnish search collations.

(Perhaps it could be mentioned instead that ä and å are primary-distinct
from a in Swedish.)

Henri Sivonen

Background information / discussion

PAG suggest making the example more generic:
If two letters are treated as identical base letters for sorting, then those letters should also be treated as identical for searching.

6.2 DUCET contractions with 3+ equivalent sequences

Markus Scherer & Ken Whistler, PAG

Recommended UTC actions

1. Action Item for Ken Whistler, PAG: In the DUCET input file and its tool (unidata.txt & sifter), support at least three sequences that are canonically equivalent to a contraction, and add all of the necessary sequences for Kirat Rai. For Unicode 16.0. See L2/23-234 item 6.2.

Summary

The DUCET input data file includes (mostly) the canonical closure of its mappings. When we create a contraction, we need to explicitly list the canonically equivalent sequences. So far, the input data file and the internal tool support up to two such sequences. For Kirat Rai U+16D6A vowel sign AU we need at least one more.

7. Security

7.1 Should the default UTS #39 confusability be bidi-aware?

From private communication to Robin Leroy, PAG

Recommended UTC actions

1. Consensus: In UTS #39, change the definitions of skeleton and confusable to be equivalent to the current bidiSkeleton(-, LTR) and LTR-confusable, respectively. For Unicode 16.0. See L2/23-234 item 7.1.
2. Action Item for Robin Leroy, PAG: In UTS #39, change the definitions of skeleton and confusable to be equivalent to bidiSkeleton(-, LTR) and LTR-confusable, respectively. Use the term internalSkeleton for the intermediate operation used in the definition of bidSkeleton and skeleton. For Unicode 16.0. See L2/23-234 item 7.1.

Summary

In 15.1, UTS #39 has a concept of bidiSkeleton and LTR- and RTL-confusabilities, but it retains the old skeleton and (unqualified) confusability.

However, the old confusability is always wrong when bidirectional considerations are involved: It will have false negatives (A1ש vs. A1ש) and false positives (A1ש vs. Aשl).
In sufficiently-strongly-directional single-direction text, the new confusabilities are equivalent to the old one.

When implementing bidiSkeleton in ICU, the question therefore was brought up of whether the APIs associated with ordinary confusability should be made a deprecated aliases of those associated with LTR-confusability. Since these operations are defined in UTS #39, this seemed like a matter for UTC/PAG.

7.2 UAX #31 and UTS #55 should point to D145 and mention the need for NFD before toNFKC_Casefold

Robin Leroy, PAG

Recommended UTC actions

1. Action Item for Robin Leroy, PAG: In UAX #31, clarify that comparing identifiers after toNFKC_Casefold does not meet UAX #31-R4 with NFKC, and that toNFKC_Casefold∘NFD should be used instead. For Unicode Version 16.0. See L2/23-234 item 7.2.
2. Action Item for Robin Leroy, PAG: In UTS #55, correct the sixth paragraph of Section 3.1.1 to state that UAX #31-R4 and UAX #31-R5 is met by comparison after toNFKC_Casefold∘NFD. See L2/23-234 item 7.2.

Summary

⟨U+03A9, U+0345, U+0313⟩ ᾨ and U+1FA8 ᾨ are canonically equivalent, but

toNFKC_Casefold(⟨U+03A9, U+0345, U+0313⟩ ᾨ) = ωi̇

whereas

toNFKC_Casefold(U+1FA8 ᾨ) = ὠι.

In order to meet UAX #31-R4 with normalization form KC and UAX #31-R5 with full case folding, it is not enough to apply the operation toNFKC_Casefold before comparing the identifiers, contra the sixth paragraph of https://www.unicode.org/reports/tr55/#Normalization-Case. This should be mentioned in UAX #31 as it is an easy mistake, and UTS #55 should be corrected.

8. Emoji

8.1 Proposal to Define Variation Sequences for Emoji Mapped to Legacy Computing Symbols

L2/23-142

Recommended UTC actions

1. No Action: PAG recommends no action as it was determined to be moot.

Summary

From the document's background section:

The character repertoire proposed in L2/21-235r (Bettencourt et al., Proposal to add further characters from legacy computers and teletext to the UCS), AKA "Legacy Computing Supplement", unifies a number of different character sets. Symbols from these sets map to both newly proposed characters (planned to be released in Unicode 16.0) as well as existing code points, some of which are classified as emoji.

The document proposes adding variation sequences for explicit text vs. emoji style of the existing characters. These characters have the Emoji_Presentation property.

(A similar addition of variation sequences for Emoji_Presentation characters was approved in UTC-173-C29, see p. 5 of L2/22-229R.)

9. Math

9.1 Incorrect Math Classifications in UTR #25

General Feedback

Recommended UTC actions

1. Action Item for Asmus Freytag, Murray Sargent, and other authors of UTR #25: Revise UTR #25 data files and text taking into account feedback ID20230815121059 from David Carlisle. See L2/23-234 item 9.1.

Feedback (verbatim)

Date/Time: Tue Aug 15 12:10:59 CDT 2023
ReportID: ID20230815121059
Name: David Carlisle
Report Type: Error Report
Opt Subject: TR25 UNICODE SUPPORT FOR MATHEMATICS

https://unicode.org/reports/tr25/

Some of the Math Classifications in the MathClass-15 data file associated with
TR25 seem incorrect.
https://www.unicode.org/Public/math/revision-15/MathClassEx-15.txt

23B0;R; ⎰ ;lmoust;ISOAMSC;;UPPER LEFT OR LOWER RIGHT CURLY BRACKET SECTION

23B1;R;⎱;rmoust;ISOAMSC;;UPPER RIGHT OR LOWER LEFT CURLY BRACKET SECTION
27C5;R;⟅;;;;LEFT S-SHAPED BAG DELIMITER
27C6;R;⟆;;;;RIGHT S-SHAPED BAG DELIMITER

These are classified as R (infix relation, TeX \mathrel) when it would seem more appropriate to use O and C (\mathopen \mathclose) which are the assignments currently made by LaTeX.

I'm doing a systematic comparison with LaTeX Unicode-math, there are other differences as detailed in this github issue

wspr/unicode-math#619 (comment)

However in some of these cases we may choose to change the TeX settings or simply document the differences although for example as listed in that issue, TeX traditionally makes daggers U+2020 an dU+2021 binary operators (B) not relations (R) which would give them more space.

Background information / discussion

Add Robin Leroy and David Carlisle as additional authors of UTR #25.
Ask Asmus Freytag, Murray Sargent to add Robin Leroy and David Carlisle as additional authors of UTR #25.

10. Authorize proposed updates

Recommended UTC action

1. Consensus: Authorize proposed updates of UAX #14, UAX #15, UAX #29, UAX #31, UAX #42, UAX #44, UTS #10, and UTS #39, for Unicode 16.0.
2. Consensus: Authorize proposed updates of UTS #55 and UTR #25.