UTC #172 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2022-jul-20

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

Sources of feedback

We review general feedback received via the Unicode reporting form, see L2/22-123 “Comments on Public Review Issues (April 11 - July 11, 2022)”. We also review feedback on public review issues and documents in the UTC document register.

UCD

UCD1: Todhri vowel encoding model

L2/22-074 “Todhri encoding options” from Roozbeh Pournader
L2/20-188R2 “Proposal for encoding the Todhri script ...” from Michael Everson

Recommended UTC actions

1. Note: Regarding L2/22-074 “Todhri encoding options”, the properties & algorithms group recommends using character sequences (option 3 in that doc) rather than encoding new composite characters, unless there is a really strong argument for why precomposed forms are essential. In particular, we do not see such a strong argument for the Todhri vowels discussed in L2/22-074.
2. Withdraw the composite vowel code points U+105C9 TODHRI LETTER EI and U+105E4 TODHRI LETTER U that were approved in consensus 171-C17.
3. Action item for Ken Whistler, EDC: Remove U+105C9 TODHRI LETTER EI and U+105E4 TODHRI LETTER U from the pipeline.

Summary

Todhri is a new script (not in Unicode 15). Its “e” and “u” vowels look like other vowels with a dot above. 22-074 and sections 2.2.1-2.2.3 of 20-188R2 present three encoding options:

1. Atomic encoding of “e” and “u”, no decompositions.
2. Encoding composite characters with canonical decompositions with other vowels + U+0307 dot above.
3. Requiring the use of vowel+dot-above sequences.

Michael favors atomic encoding for its simplicity.
Roozbeh finds atomic encoding problematic, assuming that “Some content creators will use U+0307 in the script anyway, causing multiple representation issues”

The proposed encoding uses several U+03xx combining marks for stress and other variations.

Background information / discussion

https://www.unicode.org/L2/L2022/22061.htm#171-C17

> **[171-C17] Consensus:** Accept 52 Todhri characters in a new Todhri block (U+105C0..U+105FF) for encoding in a future version of the standard, but amending the properties on page 6 of L2/20-188R2 with the following two decompositions and changing the general category property for all the characters from “Ll” to “Lo”:
>
> 105C9;TODHRI LETTER EI;Lo;0;L;105D2 0307;;;;N;;;;;
> 105E4;TODHRI LETTER U;Lo;0;L; 105DA 0307;;;;N;;;;;
>
> (Reference section 2 of L2/22-068 and L2/20-188R2)

SAH pointed PAG to L2/22-074 on May 18, after the UTC #171 meeting. Todhri is not in Unicode 15.

Ken: SAH consensus is turning sharply against “do not use” tables. (This character looks like that sequence but don’t use one or the other because not canonically equivalent…)

Peter: Given the limited use (not in modern use, not to be recommended for use in identifiers), I wouldn't be too concerned about that mixed representations. However, given that 03xx combining marks are already required regardless, I don't think there's a strong case for atomic w/o decomposition. Option 2 (compromise) or 3 (cleaner).

Mark: Option 3.

Asmus: Would need a really strong argument for why a precomposed form is essential, and I don’t see that. Communicate to SAH that in general we would frown upon composites without strong reasons. Peter: SAH may still want some composites for Indic scripts.

UCD2: Soft_Dotted for new characters in Cyrillic Extended-D

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Give the Soft_Dotted property to U+1E04C and U+1E04D and U+1E068, for consistency with other modifier letters, and for consistency with the regular Cyrillic letters; for Unicode 15. See L2/22-124 item UCD2.

Feedback (verbatim)

Date/Time: Tue May 31 13:46:24 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 453

The following new characters in the Cyrillic Extended-D block should be given
the Soft_Dotted property for consistency with their base forms:

U+1E04C MODIFIER LETTER CYRILLIC SMALL BYELORUSSIAN-UKRAINIAN I
U+1E04D MODIFIER LETTER CYRILLIC SMALL JE
U+1E068 CYRILLIC SUBSCRIPT SMALL LETTER BYELORUSSIAN-UKRAINIAN I

Background information / discussion

Delta code chart: https://www.unicode.org/charts/PDF/Unicode-15.0/U150-1E030.pdf

Soft_Dotted is for characters which lose the dot if another “above” combining mark is added. Is this the case for these characters?

These characters are already Soft_Dotted:
U+0456 CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I
U+0458 CYRILLIC SMALL LETTER JE

The list of Soft_Dotted characters that have decompositions (Unicode 14) does include some gc=Lm.

We recommend against adding further modifier letters, because this could be an endless stream. Need to start recommending markup. Also, the i/j modifier letters could have been shared between Latin & Cyrillic scripts, since they look the same and since we have scx.

UCD3: Other_Lowercase for new U+1E06D MODIFIER LETTER CYRILLIC SMALL STRAIGHT U WITH STROKE

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Give the Other_Lowercase property (and thus also Cased) to modifier letter U+1E06D, for Unicode 15. See L2/22-124 item UCD3.

Feedback (verbatim)

Date/Time: Tue May 31 13:53:07 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 453

In the Cyrillic Extended-D block, new character U+1E06D MODIFIER LETTER CYRILLIC SMALL STRAIGHT U WITH STROKE was accidentally excluded from the Other_Lowercase property that all other modifier letters in the block have been assigned.

Background information / discussion

Multi-property view of the current data:
https://raw.githubusercontent.com/unicode-org/icu/main/icu4c/source/data/unidata/ppucd.txt

```
block;1E030..1E08F;age=15.0;Alpha;blk=Cyrillic_Ext_D;Cased;CI;CWKCF;Dia;dt=Sup;gc=Lm;Gr_Base;IDC;IDS;lb=AL;Lower;NFKC_QC=N;NFKD_QC=N;SB=LO;sc=Cyrl;WB=LE;XIDC;XIDS
# 1E030..1E08F Cyrillic Extended-D
# Superscript modifier letters
cp;1E030;dm=0430;na=MODIFIER LETTER CYRILLIC SMALL A;NFKC_CF=0430
cp;1E031;dm=0431;na=MODIFIER LETTER CYRILLIC SMALL BE;NFKC_CF=0431
…
# Superscript modifier letters
cp;1E06B;dm=04AB;na=MODIFIER LETTER CYRILLIC SMALL ES WITH DESCENDER;NFKC_CF=04AB
cp;1E06C;dm=A651;na=MODIFIER LETTER CYRILLIC SMALL YERU WITH BACK YER;NFKC_CF=A651
cp;1E06D;-Cased;dm=04B1;-Lower;na=MODIFIER LETTER CYRILLIC SMALL STRAIGHT U WITH STROKE;NFKC_CF=04B1;SB=LE
unassigned;1E06E..1E08E
```

UCD4: Other_Lowercase for five existing modifier letters

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Give the Other_Lowercase property (and thus also Cased) to the modifier letters U+10FC and U+A7F2..A7F4 and U+AB69, for Unicode 15. See L2/22-124 item UCD4.

Feedback (verbatim)

Date/Time: Tue May 31 20:39:22 CDT 2022
Name: David Corbett
Report Type: Error Report
Opt Subject: PropList.txt

The following characters are missing Other_Lowercase. All other modifier letters that decompose to cased letters have Other_Lowercase.

• U+10FC MODIFIER LETTER GEORGIAN NAR
• U+A7F2 MODIFIER LETTER CAPITAL C
• U+A7F3 MODIFIER LETTER CAPITAL F
• U+A7F4 MODIFIER LETTER CAPITAL Q
• U+AB69 MODIFIER LETTER SMALL TURNED W

Background information / discussion

Modifier letters with decompositions, grouped by Lowercase (Unicode 14)

These characters are already lowercase:
U+10DC GEORGIAN LETTER NAR
U+028D LATIN SMALL LETTER TURNED W

UCD5: Four Nag Mundari signs should have lb=CM

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. No further action needed.
2. FYI: Ken Whistler has already made this change during the Unicode 15 beta period: Change the Nag Mundari signs 1E4EC..1E4EF to lb=CM.

Feedback (verbatim)

Date/Time: Wed Jun 1 19:09:27 CDT 2022
Name: David Corbett
Report Type: Public Review Issue

Opt Subject: 453 [PAG]

LineBreak-15.0.0d6.txt has this line:

1E4EC..1E4EF;AL # Mn [4] NAG MUNDARI SIGN MUHOR..NAG MUNDARI SIGN SUTUH

Those characters should have lb=CM because they are combining marks.

Background information / discussion

Script proposal: https://www.unicode.org/L2/L2021/21031r-mundari.pdf

Multi-property view of the current data:
https://raw.githubusercontent.com/unicode-org/icu/main/icu4c/source/data/unidata/ppucd.txt

```
block;1E4D0..1E4FF;age=15.0;Alpha;blk=Nag_Mundari;gc=Lo;Gr_Base;IDC;IDS;lb=AL;SB=LE;sc=Nagm;WB=LE;XIDC;XIDS
# 1E4D0..1E4FF Nag Mundari
# Letters
cp;1E4D0;na=NAG MUNDARI LETTER O
cp;1E4D1;na=NAG MUNDARI LETTER OP
…
# Various signs
cp;1E4EB;CI;gc=Lm;na=NAG MUNDARI SIGN OJOD
cp;1E4EC;-Alpha;bc=NSM;ccc=232;CI;gc=Mn;GCB=EX;-Gr_Base;Gr_Ext;-IDS;jt=T;na=NAG MUNDARI SIGN MUHOR;SB=EX;WB=Extend;-XIDS
cp;1E4ED;-Alpha;bc=NSM;ccc=232;CI;gc=Mn;GCB=EX;-Gr_Base;Gr_Ext;-IDS;jt=T;na=NAG MUNDARI SIGN TOYOR;SB=EX;WB=Extend;-XIDS
cp;1E4EE;-Alpha;bc=NSM;ccc=220;CI;gc=Mn;GCB=EX;-Gr_Base;Gr_Ext;-IDS;jt=T;na=NAG MUNDARI SIGN IKIR;SB=EX;WB=Extend;-XIDS
cp;1E4EF;-Alpha;bc=NSM;ccc=230;CI;gc=Mn;GCB=EX;-Gr_Base;Gr_Ext;-IDS;jt=T;na=NAG MUNDARI SIGN SUTUH;SB=EX;WB=Extend;-XIDS
```

Unicode 14 has 142 gc=Mn that are not lb=CM including a number of SEA “signs”. However, almost all of these have lb=SA (Complex_Context, need more than rule-based line breaking) while Nag Mundari letters have lb=AL.

UCD6: Line_Break for double diacritics

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Change the double diacritics U+1DCD and U+1DFC to lb=Glue, for Unicode 15. See L2/22-124 item UCD6.

Feedback (verbatim)

Date/Time: Wed Jun 1 19:14:47 CDT 2022
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: 453 [PAG]

Most double diacritics ([[:ccc=233:][:ccc=234:]]) have Line_Break=Glue. The two exceptions are U+1DCD COMBINING DOUBLE CIRCUMFLEX ABOVE and U+1DFC COMBINING DOUBLE INVERTED BREVE BELOW. They were probably overlooked because they were encoded later. They should also have Line_Break=Glue.

Background information / discussion

PVA.txt:
ccc; 233; DB ; Double_Below
ccc; 234; DA ; Double_Above

[[:ccc=233:][:ccc=234:]] grouped by Line_Break and Age

UCD7: Identifier_Status of U+A7AE and U+026A

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Mark Davis, PAG: Change the Identifier_Type of U+A7AE to Technical (and thus its Identifier_Status to Restricted), for Unicode 15. See L2/22-124 item UCD7.

Feedback (verbatim)

Date/Time: Sat Jun 4 21:11:32 CDT 2022
Name: David Corbett
Report Type: Other Document Submission
Opt Subject: Identifier_Status of U+A7AE and U+026A

It is weird that U+A7AE LATIN CAPITAL LETTER SMALL CAPITAL I has Identifier_Status=Allowed but its lowercase form U+026A LATIN LETTER SMALL CAPITAL I has Identifier_Type=Technical. That means the uppercase letter is

recommended for use in identifiers and the lowercase isn't. They should be treated consistently.

Background information / discussion

Asmus: This kind of inconsistency can happen if the review is done in the context of IDN where capital letters are excluded. I don't know whether this is the case here, but unless there's direct evidence that a capital form is inherently different my preference would be to always match the status of the capital to the lowercase.

Agree should be consistent and uppercase should follow lowercase.

UCD8: Identifier_Status of Mazahua letters

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Mark Davis, PAG: Change the Identifier_Type of U+A7B8 and U+A7B9 to Limited_Use (and thus their Identifier_Status to Restricted), for Unicode 15. See L2/22-124 item UCD8.
2. Action item for Mark Davis, EDC: In UTS# 39 Table 1 change the description of Limited_Use to not be limited to whole scripts, for Unicode 15. See L2/22-124 item UCD8.

Feedback (verbatim)

Date/Time: Sat Jun 4 21:23:44 CDT 2022
Name: David Corbett
Report Type: Other Document Submission
Opt Subject: Identifier_Status of Mazahua letters

U+A7B8 LATIN CAPITAL LETTER U WITH STROKE and U+A7B9 LATIN SMALL LETTER U WITH STROKE have Identifier_Status=Allowed. U+023A, U+0246, U+0247, and U+2C65 are also used in Mazahua but have Identifier_Status=Restricted. If the first two, which are used only in that language, are allowed, then the other four should also be allowed.

Background information / discussion

Asmus: Use in Mazahua would constitute what we elsewhere call "limited_use". There is a **bug** in UTS#39 that seemingly limits the "Limited_Use" Identifier type to whole scripts. This is unnecessarily limiting for scripts like Latin that cover many unrelated orthographies. The MSR includes A7B9 and 0247 (to allow the Latin Generation Panel to review) and they concluded that these were not used commonly enough to include in the DNS Root Zone. Mazahua has between 75-150K speakers with 35% literacy.
Asmus: Not aware of other languages using these 6 characters. See Latin RZ-LGR.

UCD9: Bidi_Class for Kaktovik numerals

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Change the Kaktovik numerals to bc=L (like Mayan numerals), for Unicode 15. See L2/22-124 item UCD9.

Feedback (verbatim)

Date/Time: Tue Jun 7 02:42:41 CDT 2022
Name: Mikhail Merkuryev
Report Type: Error Report
Opt Subject: Kaktovik numerals

Are Kaktovik numerals really Other Neutral? If they a written both left-to-right and right-to-left, maybe. But it seems to me they should be Left-to-right, like for example Mayan numerals.

I'm the author of Unicodia, a simple encyclopedia of Unicode characters. I've pulled your beta bases yesterday, and noticed this peculiarity.

Thank you.

Background information / discussion

https://www.unicode.org/L2/L2021/21058r-kaktovik-numerals.pdf proposes bc=L for these. It says it is a positional system and all the examples are LTR.

bc=L would be more consistent with Mayan numerals and counting rods.

UCD10: Emoji keycap bases and RI should be ExtPict

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Discussed; no action

Feedback (verbatim)

Date/Time: Fri Jun 24 09:00:17 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 453

Now that UTS #51 allows emoji keycap and tag sequences to be valid components in ZWJ sequences, the keycap bases (U+0023, U+002A,

U+0030–U+0039) and regional indicator symbols (U+1F1E6–U+1F1FF) should be given the Extended_Pictographic property so that the line break and text segmentation algorithms can deal with them properly.

Background information / discussion

Line_Break values:

- #, * are lb=AL
- Digits are lb=NU
- RI are lb=RI

In the line breaking rules, these are mostly considered before considering ExtPict.

UCD11: Change certain symbols from lb=ID to AL

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Consider Line_Break values for Creative Commons symbols; for Unicode 16. See L2/22-124 item UCD11.
2. Action item for Mark Davis, PAG: Check Line_Break values of symbols (excluding Creative Commons) for inconsistencies with other similar characters; emoji should have lb=ID; for Unicode 16. See L2/22-124 item UCD11.

Feedback (verbatim)

Date/Time: Fri Jun 24 09:56:01 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 453

The following characters in the Enclosed Alphanumeric Supplement, Alchemical Symbols, Geometric Shapes Extended, and Supplemental Arrows-C blocks currently are Line_Break=Ideographic (ID):

Intellectual property rights symbols:

U+1F10D..U+1F10F CIRCLED ZERO WITH SLASH..CIRCLED DOLLAR SIGN WITH OVERLAID BACKSLASH
U+1F16D..U+1F16F CIRCLED CC..CIRCLED HUMAN FIGURE
U+1F1AD MASK WORK SYMBOL

Astronomical and astrological symbols:

U+1F774..U+1F776 LOT OF FORTUNE..LUNAR ECLIPSE
U+1F77B..U+1F77F HAUMEA..ORCUS

Go stone markers (compare ⊙, ⚇, ⚈, ⚉):

U+1F7D5..U+1F7D8 CIRCLED TRIANGLE..NEGATIVE CIRCLED SQUARE

Star symbol (compare ☆, ✶, ❋ etc.):

U+1F7D9 NINE POINTED WHITE STAR

Arithmetic symbol dingbat (compare ➕, ➖, ✖, ➗):

U+1F7F0 HEAVY EQUALS SIGN

Arrows for legacy computing:

U+1F8B0..U+1F8B1 ARROW POINTING UPWARDS THEN NORTH WEST..ARROW POINTING RIGHTWARDS THEN CURVING SOUTH WEST

A more appropriate line break value for them would be Alphabetic (AL) as a matter of consistency, because all comparable characters are categorised as Alphabetic (or Ambiguous in a few cases) as well. The Creative Commons symbols in particular would benefit from this change because several of them are often used in sequence.

In fact, it would be a good idea to likewise set the default line break value for unassigned code points in these four blocks to Alphabetic since the encoding of Ideographic characters in these ranges seems to be the exception rather than the norm.

Background information / discussion

Symbols grouped by Line_Break

Most emoji have lb=ID. Others: [[:emoji:]-[:lb=ID:]-[:lb=e_base:]-[:lb=ri:]-[:lb=E_Modifier:]]

UCD12: Questionable ExtPict for non-emoji symbols

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Mark Davis, PAG: Check Extended_Pictographic values of non-emoji characters for inconsistencies with other similar characters; consider removing ExtPict from non-emoji characters; for Unicode 16. See L2/22-124 item UCD12.

Feedback (verbatim)

Date/Time: Fri Jun 24 10:24:49 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 453

There are some irregularities in how the Extended_Pictographic property has been assigned to non-emoji characters, which probably stem from default values that were never overridden. The following characters are

Extended_Pictographic=True even though none of the other non-emoji characters within the same blocks share that property:

U+1F10D..U+1F10F CIRCLED ZERO WITH SLASH..CIRCLED DOLLAR SIGN WITH OVERLAID BACKSLASH
U+1F12F COPYLEFT SYMBOL
U+1F16C..U+1F16F RAISED MR SIGN..CIRCLED HUMAN FIGURE
U+1F1AD MASK WORK SYMBOL
U+1F260..U+1F265 ROUNDED SYMBOL FOR FU..ROUNDED SYMBOL FOR CAI
U+1F774..U+1F776 LOT OF FORTUNE..LUNAR ECLIPSE
U+1F77B..U+1F77F HAUMEA..ORCUS
U+1F7D5..U+1F7D9 CIRCLED TRIANGLE..NINE POINTED WHITE STAR
U+1F8B0..U+1F8B1 ARROW POINTING UPWARDS THEN NORTH WEST..ARROW POINTING RIGHTWARDS THEN CURVING SOUTH WEST

While there is no real harm to these being Extended_Pictographic, there is no purpose to it either because none of these characters are ever going to be emojified and the Extended_Pictographic property has no use outside of emoji ZWJ sequences.

Background information / discussion

Mark: I would have no real objection to making assigned characters that are not emoji also not be Extended_Pictographic, if (a) we all agree that they can't be emojified (and I think we are there), and (b) we think it is worth the effort (as Buff points out, they don't really hurt anything either).

Line_Break and General_Category values for emoji that are not lb=ID

UCD13: Pahawh Hmong math symbols should be gc=Sm not So

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, EDC: In the core spec, add text to the Pahawh Hmong block description explaining the difference between “used in some mathematical sense” and “part of the repertoire used in international mathematical notation”. See L2/22-124 item UCD13.

Feedback (verbatim)

Date/Time: Fri Jun 24 15:15:21 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 453

The Pahawh Hmong script, which was encoded in version 7.0, includes four arithmetic symbols at U+16B3C..U+16B3F that serve as a plus, minus, multiplication, and division sign respectively. Despite being math symbols,

they currently belong to the general category Other_Symbol (So) rather than Math_Symbol (Sm). I propose changing them to Math_Symbol.

The original proposal (L2/12-013: Everson, “Final proposal to encode the Pahawh Hmong script in the UCS”) did in fact give them the general category value Sm, but this was changed to So at some unknown point before release. I cannot find any traces of this decision in publicly available documents, so the change might very well have been the result of a clerical error.

Background information / discussion

https://www.unicode.org/L2/L2012/12013-n4175-pahawh-hmong.pdf
16B3C;PAHAWH HMONG SIGN XYEEM NTXIV;Sm;0;ES;;;;;N;;;;;
16B3D;PAHAWH HMONG SIGN XYEEM RHO;Sm;0;ES;;;;;N;;;;;
16B3E;PAHAWH HMONG SIGN XYEEM TOV;Sm;0;ES;;;;;N;;;;;
16B3F;PAHAWH HMONG SIGN XYEEM FAIB;Sm;0;ES;;;;;N;;;;;

Some characters functioning as math symbols in a specific script does not automatically mean that they should be listed as math symbols in an international context.

[:Sm:]-[:sc=Common:] by script

UCD14: QUADRUPLE PRIME should be lb=PO not AL

From https://www.unicode.org/review/pri453/ Unicode 15.0.0 Beta

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Check the Line_Break values of the various PRIME characters for consistency; for Unicode 16. See L2/22-124 item UCD14.

Feedback (verbatim)

Date/Time: Mon Jun 27 08:44:43 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 453

Currently, U+2057 QUADRUPLE PRIME has Line_Break=Alphabetic (AL) while U+2032 PRIME, U+2033 DOUBLE PRIME, and U+2034 TRIPLE PRIME have Line_Break=Postfix_Numeric (PO). I propose changing U+2057 to Postfix_Numeric for consistency.

UCD15: Remove redundant @missing for several properties

From Asmus Freytag via email

Recommended UTC actions

1. Action item for Markus Scherer, PAG: In PropertyValueAliases.txt remove the @missing lines for Equivalent_Unified_Ideograph, bmg and NFKC_CF and scx, and for any other properties which have redundant @missing lines; for Unicode 15.

Feedback (paraphrased)

For the Equivalent_Unified_Ideograph (EqUIdeo) property, there are two @missing lines providing the default value for all code points not mentioned in the data file:

```
PropertyValueAliases.txt:
# @missing: 0000..10FFFF; Equivalent_Unified_Ideograph; <none>
EquivalentUnifiedIdeograph.txt:
# @missing: 0000..10FFFF; <none>
```

They specify the same default value, but having two such specifications risks their getting out of sync.

We should either remove the one in PVA.txt, or else document that in case of conflicts one of them “wins” (probably the one in the file that carries the data for the property).

Additional properties that are affected are <bmg> and <NFKC_CF> and <scx>.

Background information / discussion

This is the case in both Unicode 14 and in 15 beta, and probably earlier versions.

UCD16: Core spec on Default Properties

From Asmus Freytag via email

Recommended UTC actions

1. Action item for Asmus Freytag, EDC: Make changes to the core spec chapter 3, D26 “Default Property”, according to the feedback (additions & changes in blue) in L2/22-124 item UCD16, for Unicode 15.

Feedback

… discussion of default values. Markus Scherer gives a preference for the core spec referring to UAX #44 (section 4.2.9 Default Values) rather than duplicating material.

Asmus responds:

I see the role of UAX#44 in describing how default properties are represented in the UCD, not actually defining what a default property is.

As such, I agree that we definitely should not duplicate information such as the details of the @missing conventions, nor the thorough accounting for which properties have special defaults, etc.

However, the text passage in the Core Spec serve as part of a general introduction to the topic, and in particular as a further explanation of the concept of "Default Property Value" (which is defined in D.26).
As it stands, that overview is incomplete and a bit at odds with the UCD. The goal would be to fix it, while keeping it general enough that we don't need to revise it (unless we come up with some novel types of default properties in the future).

Asmus:

The text of the **Core spec** seems to not anticipate some types of default values. And is out of step with our efforts to explicitly document defaults (This is based on the 14.0.0 core spec, where these (informative) statements following ***D26 Default Property*** could use some tweaks). For example:

"• A default property value is typically defined implicitly, to avoid having to repeat

long lists of unassigned code points."

The word "implicit" seems at odds with explicitly defined @missing statements. While an empty string is a good example at an "implicit" definition, there are just too many defaults that are explicitly chosen and don't just naturally arise from the absence of a specified value.

Suggestion 1:

Reword the above to:

"• A default property value is typically omitted when listing property values to avoid having to repeat long lists of unassigned code points. The default value may instead be specified by explicit directives or in the description of the property."

(As highlighted. Retaining the weasel words like "may" and "typically" to cover edge cases).

Suggestion 2:

Add a sentence at the end to make clear that we no longer rely on the reader to "imply" the value, but that we explicitly state the "implied" default:

"• In the case of some properties with arbitrary string values, the default property

value is an implied null value. For example, the fact that there is no Unicode

character name for unassigned code points is equivalent to saying that the

default property value for the Name property for an unassigned code point is a

null string. This may also be indicated by an explicit directive."

(Suggesting that we avoid using the expression "@missing" here, but perhaps there are other ways to qualify the term "directive" a bit better).

Suggestion 3:

Finally, we should add a mention of two special classes of default property values (new text):

"• For properties that map from code points to string values, the default is typically the identity mapping as opposed to a constant value over a range of code points."

"• In select cases, the default property for a code point may be the value of another property for that code point (including its default property values). For example, the default for *Script_Extensions* for a code point is the value of the *Script* property for that code point."

(Using "select" to indicate that this is not common, but wanting to allow the language to survive if we add another instance of this kind of "indirection" to another property).

UCD17: @missing lines do not work for binary properties

From Markus Scherer in discussion with the properties & algorithms group

Recommended UTC actions

1. Action item for Ned Holbrook, Markus Scherer, PAG: In emoji-data.txt, remove the @missing lines; for Unicode 15.
2. Action item for Ken Whistler, EDC: In UAX #44 for Unicode 15,
   (a) revert the changes to the paragraph that used to say that an @missing line is never provided for a binary property (so that it continues to say that for Unicode 15), and
   (b) change the example for multiple @missing lines from using Extended_Pictographic to using one of bc/ea/lb.
3. Action item for Ken Whistler, EDC: In UAX #44 for Unicode 15,
   change "@missing lines are also supplied for many properties in the file PropertyValueAliases.txt. In this case, because there are many @missing lines in that single data file, each @missing line ~~contains an additional second field specifying the property name for which it defines a default value~~."
   to "@missing lines are also supplied for many properties in the file PropertyValueAliases.txt. In this case, because there are many @missing lines in that single data file, each @missing line **in that file uses the syntactic pattern code_point_range; property_name; default_prop_val**".
   See L2/22-124 item UCD17.
4. Action item for Markus Scherer, PAG: Propose a UCD file syntax for explicit "No" values for binary properties, and use it for multiple @missing lines for Extended_Pictographic in emoji-data.txt; for Unicode 16. See L2/22-124 item UCD17.
5. Action item for Markus Scherer, PAG: For Unicode 16, change PropertyAliases.txt to list Bidi_Mirroring_Glyph and Equivalent_Unified_Ideograph under "String Properties", and to list Bidi_Paired_Bracket under "Enumerated Properties".
6. Action item for Ken Whistler, PAG: For Unicode 16, in UAX #44 table 9 (Property Table), change the types of Bidi_Mirroring_Glyph and Equivalent_Unified_Ideograph to "S" (String-valued).

Summary

The file ucd/emoji/emoji-data.txt has @missing and data lines like this:

```
# All omitted code points have Emoji=No
# @missing: 0000..10FFFF  ; Emoji ; No
```

```
0023        ; Emoji              # E0.0   [1] (#)      hash sign
002A        ; Emoji              # E0.0   [1] (*)      asterisk
```

Compare with www.unicode.org/reports/tr44/#Missing_Conventions & #Complex_Default_Values

1. These @missing lines should not be there at all. UAX #44 says “An @missing line is never provided for a binary property …”
    a. Note: The proposed update of UAX #44 modifies this sentence, allowing @missing lines with value “Yes”, giving Extended_Pictographic as an example.
2. These @missing lines use a different format compared with the regular data lines. For these properties, as for binary properties elsewhere, the data lines only show the property name; the value “Yes” is implied. However, the @missing lines have an extra field with the value “No”. UAX #44 says that @missing lines should have the same syntax as data lines in the same file.
3. These @missing lines could easily be mis-parsed. If a parser treated these lines like data lines, and ignored the additional fields, then it would set this property for all Unicode code points.

In addition,

4. We have started using multiple @missing lines per property (e.g., Bidi_Class), for certain ranges with default values. These only work when the syntax allows data lines to override the values for specific code points as needed. However, for binary properties, there is currently no UCD syntax for the “No” value. As a result, we cannot currently use multiple @missing lines for Extended_Pictographic, which is one of the properties with complex default values.
5. (Unrelated, but Asmus Freytag observed this issue.) PropertyAliases.txt lists Bidi_Mirroring_Glyph, Bidi_Paired_Bracket, and Equivalent_Unified_Ideograph under “Miscellaneous Properties”. They should be listed under more specific categories. UAX #44 shows Bidi_Paired_Bracket_Type as Enumerated.

Bidi

Bidi1: Glyph mirroring: ExtraMirroring.txt

L2/22-026R from Kent Karlsson
Revised document. The original proposal suggested a new data file named NonBidiMirroring.txt. The revision proposes ExtraMirroring.txt.

Recommended UTC actions

1. Action item for Mark Davis, Asmus Freytag, PAG: Continue the discussion of L2/22-026R (ExtraMirroring.txt) with Kent Karlsson and Karl Williamson.

Summary

Proposed new data file ExtraMirroring.txt

… make a data file similar to *BidiMirroring.txt*, but for symbols that have the *Bidi_Mirrored=No* property value and have a mirror character, call it *ExtraMirroring.tx*t. Mostly for arrows and arrow-like symbols, but also other symbols. Note that various arrows are commonly used in math expressions. And in editing math expressions

or other text, one may (for whatever reason, like error fixing, swapping the arguments, “no, you should go right, not left” symbolised with an arrow in the text, …) want to mirror also symbols that have the *Bidi_Mirrored=No* property value.

Both *BidiMirroring.txt* and *ExtraMirroring.txt* (proposed here) can be used by typeface foundries, or even typeface editing tools, be used to make consistently looking mirror glyphs for mirror character pairs.

Related feedback (verbatim)

Date/Time: Sun Apr 17 12:41:19 CDT 2022
Name: Karl Williamson
Report Type: Other Document Submission
Opt Subject: NonBidiMirroring.txt

https://www.unicode.org/L2/L2022/22026-non-bidi-mirroring.pdf is a proposal from Kent Karlsson for creation of this UCD file

I saw that a proposed response to it was that it was "speculative".

I can tell you that Perl 5 already has had to workaround the absence of such information in the UCD, and the presence of this would be helpful going forward.

The issue for us is delimiters surrounding string-like constructs. These constructs include literal text, and regular expression patterns, among others. Perl has long allowed one to use any of 4 pairs of delimiters for these, like
qr(this is a pattern)

The 4 sets are () <> {} []. These stem from before Unicode came along, and now Unicode has added hundreds of potential such delimiters. We've had longstanding requests to use this, and the next release of Perl will add many of them. It would have been better to have used this proposed file if it had existed, and I did go looking for something suitable, to no avail. It would be better in the future to use this file, as it gets updated to correspond with new Unicode versions.

Background information / discussion

(We briefly looked at L2/22-026 for UTC #171, wrote the comment below about unclear use cases, and assigned an AI for KenW+EDC to review annotations. Karl Williamson wrote his feedback in response to our recommendation.)

(PAG comment from UTC #171) NonBidiMirroring.txt: Unclear use cases. Seems to be proposed for speculative uses.

Markus: The Perl usage sounds like it’s about special delimiter pairs, not about mirroring, and those pairs would likely be a subset of the pairs in the proposed data file. That is, the Perl implementation would need its

own data, although the Perl implementers could pick the subset from a Unicode property. Question: Why couldn't Perl implementers pick a subset using existing properties? For example: [:Ps:][:Pe:][:Pi:][:Pf:]

Ken: Another data file to look at for paired delimiters is BidiBrackets.txt.

Asmus: We have been aware of when we encoded pairs of characters. Some of these are listed in BidiBrackets.txt. For going further, we would need a good explanation for what makes a notional pair, and good justification for why we need a machine-readable data file for it.

Ken: We don't have data about arrows that could be notional pairs. We do not need them for bidi processing. Arrows have more complex rotational symmetry, not just left & right. If we were to provide data for arrows, it should probably be more comprehensive than what the document suggests.

Bidi2: Motivation for certain design decisions

From https://www.unicode.org/review/pri449/ Proposed Update UAX #9, Unicode Bidirectional Algorithm

Recommended UTC actions

1. Action item for Asmus Freytag, Ken Whistler, EDC: In UAX #9, add explanation/motivation for certain overall design decisions; see the feedback as modified in discussion in L2/22-124 item Bidi2; for a future version of Unicode.

Feedback (verbatim)

Date/Time: Fri Jul 1 12:52:18 CDT 2022
Name: Asmus/
Report Type: Public Review Issue
Opt Subject: 449

An issue has been raised on the public mailing list by A. Prilop that points to the fact that the description of the Bidi Algorithm apparently fails to state the motivation for certain design decisions. For example, here's the entire paragraph summarizing the algorithm from the Intro:

"Each character has an implicit bidirectional type. The bidirectional types left-to-right and right-to-left are called strong types, and characters of those types are called strong directional characters. The bidirectional types associated with numbers are called weak types, and characters of those types are called weak directional characters. With the exception of the directional formatting characters, the remaining bidirectional types and characters are called neutral. The algorithm uses the implicit bidirectional types of the characters in a text to arrive at a reasonable display ordering for text."

Some of the goals for "reasonable" display ordering include:

(1) getting the correct ordering of words when separated by punctuation

(2) getting the correct ordering of groups of digits
(3) getting the correct placement of numerical punctuation

Each of those depends not only on adjacent characters but sometimes more distant context, or on the overall paragraph direction. Different writing systems differ, for example in the handling of different sets digits. This is reflected by specific bidirectional types for Arabic letters or different types of digits.

Reverse engineering this motivation is not something that the reader of the spec should be required to do. A simple suggestion would be to include something like the preceding text immediately after the quoted passage. Alternatively, a paragraph summarizing the intent could be added in each section, or both.

Background information / discussion

Change proposed text to:

Some of the goals for "reasonable" display ordering include:

(1) getting the correct ordering of words when separated by punctuation
(2) getting the correct ordering of groups of digits separated by punctuation
(3) getting the correct placement of numerical punctuation like currency symbols

Each of those depends not only on adjacent characters but sometimes more distant context, or on the overall paragraph direction. Different writing systems differ, for example in the handling of different sets of digits. This is reflected in the algorithm by specific bidirectional types for Arabic letters or different types of digits.

Normalization

Norm1: Misleading intro to examples

Recommended UTC actions

1. No further action.
2. FYI: Ken Whistler has already made this change in the UAX #15 proposed update: Replace the sentence “For consistency, all of these examples use Latin characters, although similar examples are found in other scripts.” with “Examples like these can be found in many scripts.”

Feedback (verbatim)

Date/Time: Fri Apr 22 12:02:13 CDT 2022
Name: Tim Pederick
Report Type: Error Report
Opt Subject: tr15-51.html

UAX #15, §1.2 Normalization Forms, says of figures 3 to 6 that "
[f]or consistency, all of these examples use Latin characters". This is not
true of figure 3, in which the second example uses only the Greek
characters U+2126 and U+03A9. (And to be pedantic, figure 5 has an example
with only the Common characters U+0032, U+2075, and U+0035.)

I don't propose replacing the examples with ones that do use Latin
characters, but rather changing the note itself, or even removing it. I'm
not really sure what is meant by "for consistency"; is it
really "inconsistent" to use non-Latin examples? Is the intent of the note
to head off complaints of Latin-script parochialism?

Text Segmentation

Seg1: PU-UAX29 excludes some Kawi characters from GCB=SpacingMark

From https://www.unicode.org/review/pri441/ Proposed Update UAX #29, Unicode Text Segmentation

Recommended UTC actions

1. No action. This topic has been discussed and agreed in UTC #171, resulting in action items which have been done in time for Unicode 15 beta.

Feedback (verbatim)

Date/Time: Tue Mar 29 00:43:30 CDT 2022
Name: Norbert Lindenberg
Report Type: Public Review Issue
Opt Subject: 441

The proposed update for UAX 29 excludes the following Kawi characters
from having the Grapheme_Cluster_Break property value SpacingMark:

U+11F03 (◌) KAWI SIGN VISARGA
U+11F34 (◌) KAWI VOWEL SIGN AA
U+11F35 (◌) KAWI VOWEL SIGN ALTERNATE AA
U+11F41 (◌) KAWI SIGN KILLER

Being excluded from having SpacingMark means that they receive the
Grapheme_Cluster_Break property value Other. In consequence, these

characters do not combine with other characters into extended grapheme clusters; they always form their own separate grapheme clusters.

I don't see any reason in the proposal for Kawi, L2/20-284R, or anywhere else why that should be the case. The purpose of grapheme clusters isn't well defined, but one case where the Unicode Standard recommends using them is in emergency line breaking (see UAX 14, section 3, Introduction). If a line break is introduced before a combining mark of a complex script, fonts or rendering systems commonly insert a dotted circle as a base for that mark, which is undesirable.

The corresponding spacing combining marks in the three most closely related scripts, Javanese, Balinese, and Sundanese, all have the Grapheme_Cluster_Break property value SpacingMark or (in one case, 1B35) Extend. I suggest that Kawi is handled the same way.

Background information / discussion

Proposed update: https://www.unicode.org/reports/tr29/tr29-40.html#SpacingMark

This topic has been discussed and agreed in UTC #171, resulting in action items [171-A68] and [171-A69] which have been done in time for Unicode 15 beta.

Seg2: Kawi line break: Western style for now

From https://www.unicode.org/review/pri442/ Unicode 15.0.0 Alpha Review

Recommended UTC actions

1. FYI: The PAG reviewed and agrees with Western-style line breaking behavior of Kawi in Unicode 15, based on PRI #442 feedback from Norbert Lindenberg on 2022-apr-08, and using Line_Break properties values as suggested there.
2. No further action items: These changes have been made in time for Unicode 15 beta.

Feedback (verbatim)

Date/Time: Fri Apr 8 17:49:03 CDT 2022
Name: Norbert Lindenberg
Report Type: Public Review Issue
Opt Subject: 442

The data for Kawi in the LineBreak.txt draft for Unicode 15 uses the South East Asian style of context analysis for line breaking. This style implies that a complex context-dependent analysis is required for Kawi. That is not actually the case, as the proposal L2/20-284R documents line breaking at orthographic syllable boundaries. That style of line breaking however isn't actually supported in the Unicode line breaking algorithm in Unicode 15 yet.

For now, Kawi syllables should use the Western style to align with the script's descendants Javanese, Balinese, and Sundanese. For punctuation, I suggest using the values proposed in L2/22-080.

I propose the following changes:

11F00..11F01;SA # Mn [2] KAWI SIGN CANDRABINDU..KAWI SIGN ANUSVARA
→ change to CM
11F02;SA # Lo KAWI SIGN REPHA
→ change to AL
11F03;SA # Mc KAWI SIGN VISARGA
→ change to CM
11F04..11F10;SA # Lo [13] KAWI LETTER A..KAWI LETTER O
→ change to AL
11F12..11F33;SA # Lo [34] KAWI LETTER KA..KAWI LETTER JNYA
→ change to AL
11F34..11F35;SA # Mc [2] KAWI VOWEL SIGN AA..KAWI VOWEL SIGN ALTERNATE AA
→ change to CM
11F36..11F3A;SA # Mn [5] KAWI VOWEL SIGN I..KAWI VOWEL SIGN VOCALIC R
→ change to CM
11F3E..11F3F;SA # Mc [2] KAWI VOWEL SIGN E..KAWI VOWEL SIGN AI
→ change to CM
11F40;SA # Mn KAWI VOWEL SIGN EU
→ change to CM
11F41;SA # Mc KAWI SIGN KILLER
→ change to CM
11F42;SA # Mn KAWI CONJOINER
→ change to CM
11F43..11F4F;SA # Po [13] KAWI DANDA..KAWI PUNCTUATION CLOSING SPIRAL
→ change to BA for 11F43..11F44
→ change to ID for 11F45..11F4F
11F50..11F59;NU # Nd [10] KAWI DIGIT ZERO..KAWI DIGIT NINE
→ keep

Background information / discussion

These assignments look reasonable, and have been made in time for Unicode 15 beta.

Seg3: Improvement to Word_Boundary_Rules

From https://www.unicode.org/review/pri441/ Proposed Update UAX #29, Unicode Text Segmentation

Recommended UTC actions

1. Action item for Chris Chapman, EDC: Change UAX #29 as suggested in L2/22-124 item Seg3, for Unicode 15.

Feedback (verbatim)

Date/Time: Fri Jun 17 09:24:45 CDT 2022
Contact: richard.gibson@gmail.com
Name: Richard Gibson
Report Type: Error Report
Opt Subject: Unicode® Standard Annex #29 UNICODE TEXT SEGMENTATION

TC39 technical group 2 would like to push for an improvement
in #Word_Boundary_Rules that provides an example above WB6 similar to the
one above WB8.

Proposed change from the tc39/ecma402 GitHub repository issue 656
issuecomment-1158026888 :

-Do not break letters across certain punctuation.
+Do not break letters across certain punctuation, such as within “e.g” or “example.com”.

Background information / discussion

The link referenced in the feedback.

Seg4: U+23B6 is not lb=QU

From https://www.unicode.org/review/pri446/ Proposed Update UAX #14, Unicode Line Breaking Algorithm

Recommended UTC actions

1. Action item for Chris Chapman, PAG: In UAX #14, remove the note about the special behavior of U+23B6 and its lb=QU value, for Unicode 15.

Feedback (verbatim)

Date/Time: Fri Jun 3 10:22:13 CDT 2022
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: 446

UAX #14 says that U+23B6 BOTTOM SQUARE BRACKET OVER TOP SQUARE BRACKET is a

member of class QU, but that has not been true for many years.

In Unicode 5.0, the properties of the three vertical brackets U+23B4..U+23B6 were changed to consistently have lb=AL.

Background information / discussion

https://www.unicode.org/reports/tr14/tr14-48.html#QU

> U+23B6 BOTTOM SQUARE BRACKET OVER TOP SQUARE BRACKET is subtly different from the others in this class, in that it is *both* an opening and a closing punctuation character at the same time. However, its use is limited to certain vertical text modes in terminal emulation. Instead of creating a one-of-a-kind class for this rarely used character, assigning it to the **QU** class approximates the intended behavior.

LineBreak.txt: 23B4..23DB;AL # So [40] TOP SQUARE BRACKET..FUSE
(Details on that range)

Seg5: What is the rationale for LB21b? Delete or expand?

From https://www.unicode.org/review/pri446/ Proposed Update UAX #14, Unicode Line Breaking Algorithm

Recommended UTC actions

1. Action for Rick McGowan: Respond to David Corbett about L2/22-124 item Seg5, pointing out that LB21b was introduced based on rationale in L2/13-211.

Feedback (verbatim)

Date/Time: Fri Jun 3 09:10:34 CDT 2022
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: 446

The SY class is motivated by the commonness of URLs. Hebrew letters can appear in URLs. What is the rationale for LB21b? Why is Hebrew special among all scripts that can appear in URLs? Documenting the reason would help implementers decide how to tailor the algorithm.

Maybe the reasoning is that, although Hebrew can appear in URLs, most URLs are still ASCII, so a slash in Hebrew is probably not a URL slash and so isn't a break opportunity. However, if so, that reasoning applies to all non-ASCII characters; the only reason Hebrew is treated specially is that it happens to have its own line break class for an unrelated reason, not because Hebrew is actually different from other scripts. If this is the reason, there are two ways to make the algorithm more consistent. The first is to delete LB21b. The second is to expand LB21b to all non-ASCII alphabetic/symbol characters.

Background information / discussion

https://www.unicode.org/reports/tr14/tr14-48.html
LB21b *Don't break between Solidus and Hebrew letters*.
SY × HL

This was added in Unicode 8.0 per Consensus 137-C9. The rationale is L2/13-211.

Seg6: lb=Close_Parenthesis for more brackets

From https://www.unicode.org/review/pri446/ Proposed Update UAX #14, Unicode Line Breaking Algorithm

Recommended UTC actions

1. Action item for Ken Whistler, PAG: Change the Line_Break values for U+2E55..U+2E5C to match those for the ASCII square brackets; for Unicode 15. See L2/22-124 item Seg6.

Feedback (verbatim)

Date/Time: Fri Jun 3 19:49:05 CDT 2022
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: 446

L2/21-042 gives examples of U+2E55..U+2E5C within words, just like how U+0029 is used in "(s)he". It is central to these characters' purpose to appear within words, so it is likely that their line breaking works the same as for U+0029. The closing characters U+2E56, U+2E58, U+2E5A, and U+2E5C should therefore have Line_Break=Close_Parenthesis.

Background information / discussion

https://www.unicode.org/L2/L2021/21042-phonetic%20punct.pdf
e.g. U+2E56 RIGHT SQUARE BRACKET WITH STROKE

Line_Break values of [()\u2E55-\u2E5C]

Asmus: Because the proposal shows these were added as a part of a notation that also includes [ ], it is reasonable for the line break property for the square brackets [ ] and these characters to align (technically not with U+0029, but since ']' does have the same LB class as ')', that is just a footnote.)

Seg7: Tighten UAX #29 conformance

L2/22-159 from Mark Davis & Markus Scherer

Recommended UTC actions

1. Action item for Mark Davis, EDC: Flesh out the conformance section of UAX #29 as in document L2/22-159 for Unicode 15.0.
2. Action item for Rick McGowan: Update the existing proposed update of UAX #29 to incorporate the improved conformance section, for Unicode 15.

Summary

The text and the conformance section of UAX #29 give wide latitude for implementations to tailor text segmentation. Implementations frequently differ, and some standards organizations (e.g., TC39 for ECMAScript) only use basic testing to allow for such differences. This makes it hard for programs to achieve consistent behavior between OSes and browsers. (Example bug discussion.) Some TC39 members have asked for our help in addressing the situation.

The biggest issue is that unlike many other UAXes and UTSes, #29 does not provide clear conformance clauses with the specification of what it means to declare a profile. We already have a profile of Grapheme Cluster Boundaries in CLDR, and anticipate adding other profiles for Word Boundaries. In fact, the UTC has asked for changes to be “baked” in CLDR before bringing into #29 or #14.

We propose to tighten UAX #29 conformance language to be more like in other Unicode specs, for example UAX #31 (identifiers). That is, allowing clear conformance to the default behavior, and requiring that a “profile” be specified where the behavior differs. This clarifies the relation between UAX #29 and profiles of it (whether in CLDR or elsewhere).

Seg8: Anatolian hieroglyphic line breaks

From https://www.unicode.org/review/pri446/ Proposed Update UAX #14, Unicode Line Breaking Algorithm

Recommended UTC actions

1. Action item for Rick McGowan: Respond to David Corbett about L2/22-124 item Seg8, asking whether the current lb classifications cause problems, if so, which, how serious, and rationale for making a change.

Feedback (verbatim)

Date/Time: Mon Jul 11 20:35:10 CDT 2022
Name: David Corbett
Report Type: Other Document Submission
Opt Subject: Anatolian hieroglyphic line breaks

The standard says that “Spaces are used in modern renditions of [Anatolian] hieroglyphic text”; accordingly, most Anatolian hieroglyphs

have Line_Break=Alphabetic, such that there are no line break opportunities within words. The only exceptions are U+145CE and U+145CF. If U+145CF appears within a word, there is a line break opportunity after it. Is that really true? It seems more likely that modern renditions of Anatolian hieroglyphic text break on spaces, not within words. U+145CE and U+145CF should therefore get Line_Break=Alphabetic.

Background information / discussion

U+145CE **lb=OP** ANATOLIAN HIEROGLYPH A410 **BEGIN LOGOGRAM MARK**
U+145CF **lb=CL** ANATOLIAN HIEROGLYPH A410A **END LOGOGRAM MARK**

IDNA

IDNA1: Should UseSTD3ASCIIRules apply to Validity Criterion 6?

Recommended UTC actions

1. Action for Markus Scherer, Mark Davis, PAG: Investigate the report in L2/22-124 item IDNA1. If necessary, propose a clarification of UTS #46, and/or update the code that generates the IdnaTestV2.txt file.

Feedback (verbatim)

Date/Time: Thu May 5 19:38:08 CDT 2022
Name: Karl Wagner
Report Type: Error Report
Opt Subject: UTS #46: UNICODE IDNA COMPATIBILITY PROCESSING
UTS #46

Version: 14.0.0
Date: 2021-08-24
Revision: 27
URL: https://www.unicode.org/reports/tr46/

I only just started writing my own implementation of this recently, so apologies if I'm misunderstanding, but there are two locations where code-points are checked. Using the same format as the IdnaTestV2.txt file for describing those locations, they would be P1 and V6 ("Processing" step 1, and "Validation" step 6).

- P1 is applied to the entire domain, as given. So it may see (decoded) Unicode text, or Punycode. It takes the value of UseSTD3ASCIIRules in to account, so a domain like "≠☪≯/.com" triggers the

error at P1 only if UseSTD3ASCIIRules=true, because it contains a code-point which STD3ASCIIRules disallows. "xn--jbf911clb.com" will never trigger the error at this location, regardless of UseSTD3ASCIIRules, because it is just ASCII and hasn't been decoded yet.

- V6 is applied to the result of Punycode-decoding a domain label, so it will only see decoded Unicode text. As written, it would appear **not** to take UseSTD3ASCIIRules in to consideration, meaning that both (original inputs) "≠ઉ≯.com" and "xn--jbf911clb.com" would trigger errors at this location, regardless of UseSTD3ASCIIRules.

Here is the text of Section 4.1, Validity Criteria
(https://www.unicode.org/reports/tr46/#Validity_Criteria), Step 6:

> Each code point in the label must only have certain status values according to Section 5, IDNA Mapping Table:
> - For Transitional Processing, each value must be valid.
> - For Nontransitional Processing, each value must be either valid or deviation.

It is not clear whether these status values are supposed to take the value of UseSTD3ASCIIRules in to account. As described above, if this step does not consider UseSTD3ASCIIRules, "≠ઉ≯.com" and "xn--jbf911clb.com" will always be invalid domains. This leads me to believe that it **should** respect UseSTD3ASCIIRules, otherwise the parameter would be meaningless; it does not matter that P1 considers UseSTD3ASCIIRules, because it will be caught by V6 later anyway.

I'll have to apologise again because I am not very familiar with the codebases I am about to cite, but from what I can glean this is actually causing confusion in practice:

- Unicode-org implementation of IDNA not appear to consider UseSTD3ASCIIRules here:

https://github.com/unicode-org/unicodetools/blob/main/unicodetools/src/main/java/org/unicode/idna/Uts46.java#L610-L625

- This appears to be confirmed by the IdnaTestV2 file. For example, Version 14.0.0 (Date: 2021-08-17, 19:34:01 GMT) lines 571 and 573:

[571] xn--jbf911clb.xn----p9j493ivi4l; ≠ઉ≯.숯-ᥴ; [V6]; xn--jbf911clb.xn----p9j493ivi4l; ; ; # ≠ઉ≯.숯-ᥴ
[573] xn--jbf911clb.xn----6zg521d196p; ≠ઉ≯.숯-ᥴ; [V6]; xn--jbf911clb.xn----6zg521d196p; ; ; # ≠ઉ≯.숯-ᥴ

"V6" is not an optional validation step tied to any parameter; it does not appear to be something implementations can decide whether or not it applies to them. It always applies, and these domains should always be considered invalid IIUC, according to the tests.

- JSDOM implementation does consider UseSTD3ASCIIRules, considers these to be valid domains:
  https://github.com/jsdom/tr46/blob/e937be8d9c04b7938707fc3701e50118b7c023a5/index.js#L100

- Browsers effectively do in URLs. Safari 15 and JSOM both consider "http://≠☉>/.com.xn--jbf911clb" to be a perfectly fine URL:

https://jsdom.github.io/whatwg-url/#url=aHR0cDovL+KJoOGimeKJry5jb20ueG4tLWpiZjkxMWNsYg==&base=YWJvdXQ6Ymxhbms=

So I think it is worth adding an explicit mention of UseSTD3ASCIIRules and whether or not it applies to the mapping table lookup from step V6.

Thanks,

Karl

Background information / discussion

ICU does consider UseSTD3ASCIIRules after decoding and mapping.

In UTS #46, section 4.1.1 UseSTD3ASCIIRules is a subsection of 4.1 Validity Criteria, and it discusses how the validity criteria test changes depending on the flag, and on the implementation when the flag is false.

The test data file “only provides test cases for UseSTD3ASCIIRules=true”.

Markus: I *think* this is reasonably clear in the spec (I think the flag should be considered) but it could be clarified. I also want to read the unicodetools code (to which Karl points). (Possible that the Unicode Tools code never considers the flag because it only generates data for it being true.)

Collation

Coll1: Hoist Hebrew tailoring from CLDR into DUCET

Recommended UTC actions

1. Action item for Markus Scherer, PAG: Continue the discussion about the desired sort order of Geresh & Gershayim in CLDR and in the DUCET. See L2/22-124 item Coll1.

Feedback (verbatim)

Date/Time: Tue May 3 05:59:18 CDT 2022
Name: Henri Sivonen
Report Type: Error Report
Opt Subject: DUCET

https://github.com/unicode-org/cldr/blob/main/common/collation/he.xml has the following tailoring (apart from script reordering):

&[before 2]''<<′ # GERESH just before APOSTROPHE (secondary difference)
&[before 2]'\"'<<″ # GERSHAYIM just before QUOTATION MARK (secondary difference)

The other Hebrew-script language in CLDR, Yiddish, has this same tailoring (and further tailorings).

https://github.com/unicode-org/cldr/blob/main/common/collation/yi.xml

It seems generally unfortunate, both from the user perspective and from the binary size perspective of shipping an implementation, when a language requires a tailoring even though its tailoring doesn't collide with the needs of other languages in CLDR. By hoisting this tailoring into DUCET, Hebrew could use the root collation with script reordering, like, for example, Greek and Georgian. The handling of й/Й in the Cyrillic script in DUCET looks like precedent of hoisting collation complexity shared by merely the majority (not even all) of languages for a script into DUCET. In this case, the tailoring applies to both languages for the script.

(I'm filing this about DUCET as opposed to filing this about CLDR root, because CLDR root seeks to minimize differences from DUCET.)

Background information / discussion

These two characters have been tailored in CLDR since 2013, based on interchangeable use: CLDR-5576
Note that they sort secondary-before the similar-looking ASCII characters (primary equal).

Other similar-looking characters sort primary-after the ASCII characters by default, and are not usually (if ever) tailored. For example, the default sort order yields O'Connor < O'Neill < O’Connor

https://www.unicode.org/Public/UCA/14.0.0/allkeys.txt
https://www.unicode.org/charts/collation/chart_Punctuation.html

Looks like these Hebrew characters have multiple distinct functions:

- https://en.wikipedia.org/wiki/Geresh
- https://en.wikipedia.org/wiki/Gershayim

It seems desirable to have the default sort order collate the various single/double look-alike quote punctuation characters consistently, and together, and avoid language-specific tailorings for them. Most of them sort primary-after the ASCII characters, so Geresh and Gershayim could be moved to those groups. However, since the characters in these groups are often typed interchangeably, it might be useful to make all of these look-alikes consistently sort primary-equal, that is, secondary-after the ASCII characters.

All of these distinctions are mostly ignored when using alternate=shifted collation. Except: Primary-different punctuation characters become quaternary-different under alternate=shifted. Primary-equal punctuation characters become completely ignorable under alternate=shifted.

There are also apostrophe look-alikes that are letters, such as U+02BB (Hawaiian ‘Okina) and U+02BD (which sort among Latin letters). As such, they sort primary-different from the ASCII and other apostrophes, and very far away, and are not affected by alternate=shifted.

Coll2: Hoist Armenian tailoring from CLDR into DUCET

Recommended UTC actions

1. No UTC action.
2. FYI: The new ticket CLDR-15840 proposes to change the CLDR hy tailoring, rather than modifying the default sort order. The default sort order of the Armenian script will remain the same as the sort order for Western Armenian (hyw, which currently has no collation tailoring), and remain consistent with immutable Unicode mappings of the ech-yiwn ligature.

Feedback (verbatim)

Date/Time: Tue May 3 06:00:27 CDT 2022
Name: Henri Sivonen
Report Type: Error Report
Opt Subject: DUCET

https://github.com/unicode-org/cldr/blob/main/common/collation/hy.xml
has the following tailoring (apart from script reordering):

&p<և<<<ԵՒ

There are no other Armenian-script languages in CLDR.

It seems generally unfortunate, both from the user perspective and from the binary size perspective of shipping an implementation, when a language requires a tailoring even though its tailoring doesn't collide with the needs of other languages in CLDR. By hoisting this tailoring into DUCET, Armenian could use the root collation with script reording, like, for example, Greek and Georgian. The handling of й/Й in the Cyrillic script in DUCET looks like precedent of hoisting collation complexity shared by merely the majority (not even all) of languages for a script into DUCET. In this case, the tailoring applies to the only language for the script.

(I'm filing this about DUCET as opposed to filing this about CLDR root, because CLDR root seeks to minimize differences from DUCET.)

Background information / discussion

The tailoring is for ligature ech-yiwn (to sort after keh) and the titlecase (capital+small) sequence of ech+yiwn, but not for the corresponding lowercase and uppercase sequences.
CLDR has had this tailoring since 2009, from the first version of the Armenian tailoring.

Note that the ligature is seen as standing for different sequences in Western vs. Eastern Armenian; see L2/20-143 and L2/20-175 item F2. (Note that the UTC did not take action at the time, see below https://www.unicode.org/L2/L2020/20172.htm#164-A49). ICU did implement language-specific case mappings for hy; it follows Unicode for all other languages including hyw.

- Ligature ech-yiwn: և
- Compat. decomp.: եւ (ech+yiwn) [this is immutable]
- Case-fold: եւ (ech+yiwn) [this is immutable]
- Titlecase: Եւ (Ech+yiwn)
- Uppercase: ԵՒ (Ech+Yiwn)
- Titlecase / hy (ICU): Եվ (Ech+vew)
- Uppercase / hy (ICU):ԵՎ (Ech+Vew)

Sorting:

Line	Text	Char. names
1	դ	da
2	ե	ech
3	**և**	**ech-yiwn ligature**
4	եւ	ech+yiwn
5	Ե	Ech
6	Եւ	Ech+yiwn
7	ԵՒ	Ech+Yiwn
8	եվ	ech+vew
9	Եվ	Ech+vew
10	ԵՎ	Ech+Vew
11	զ	za

Default	hy tailoring	hy proposed below
<1 [1] դ	<1 [1] դ	<1 [1] դ
<1 [2] ե	<1 [2] ե	<1 [2] ե
<3 [5] Ե	<3 [5] Ե	<3 [5] Ե
<1 [8] եվ	<1 [8] եվ	<1 [8] եվ
<3 [9] Եվ	<3 [9] Եվ	**<3 [3] և**
<3 [10] ԵՎ	<3 [10] ԵՎ	<3 [9] Եվ
<1 [4] եւ	<1 [4] եւ	<3 [10] ԵՎ
<3 [3] և	<3 [7] ԵՒ	<1 [4] եւ
<3 [6] Եւ	<1 [11] զ	<3 [6] Եւ
<3 [7] ԵՒ	<1 [12] ւ	<3 [7] ԵՒ
<1 [11] զ	<3 [13] Ւ	<1 [11] զ

12	ւ	yiwn
13	Ւ	Yiwn
14	ք	keh
15	Ք	Keh
16	o	oh

<1 [12] ւ	<1 [14] ք	<1 [12] ւ
<3 [13] Ւ	<3 [15] Ք	<3 [13] Ւ
<1 [14] ք	**<1 [3] և**	<1 [14] ք
<3 [15] Ք	<3 [6] Եւ	<3 [15] Ք
<1 [16] o	<1 [16] o	<1 [16] o

<1 = primary difference <3 = tertiary difference [12] = input text line number

Markus: The existing hy tailoring looks defective. If the ech-yiwn ligature is to sort together with titlecase Ech+yiwn, as the tailoring does, then it should also sort together with lowercase ech+yiwn and uppercase Ech+Yiwn — but these two groups actually sort far apart.

Markus: I assume that the default sort order works well for hyw, since it is consistent with the default and hyw case mappings.

Discussion: The default sort order is aligned with NFKC, case folding, and hyw locale behavior.

Markus: I propose that we not change the default sort order, and instead change the CLDR hy tailoring to align with the ICU hy case mappings:

&ևվ<<<և

with the result as shown in the table (last column). → CLDR-15840

Regex

Regex1: Unclear namespace in UTS #18

Recommended UTC actions

1. Action item for Mark Davis, EDC: In UTS #18, change the reference to the “namespace for character names plus name aliases”, aligning with and pointing to UAX34-D3.

Feedback (verbatim)

Date/Time: Tue May 31 21:17:28 CDT 2022
Name: David Corbett
Report Type: Other Document Submission
Opt Subject: Unclear namespace in UTS #18

UTS #18 says “The namespace for the \p{name=...} syntax is the namespace for character names plus name aliases.” This could be misinterpreted to mean that that namespace excludes code point labels, even though code point labels are discussed earlier in that section. It would be clearer to say “The namespace for the \p{name=...} syntax is the Unicode namespace for character names”, using the term defined in UAX34-D3, which in its next version will mention code point labels.

Background information / discussion

See UAX34-D3 in the proposed update for Unicode 15.

Security

Sec1: Omissions in confusables.txt

Recommended UTC actions

1. Action item for Asmus Freytag, Mark Davis, PAG: Update confusables.txt according to L2/22-114; for a future version of Unicode.
2. Action item for Asmus Freytag, Mark Davis, PAG: Update the infrastructure (Unicode Tools & confusables data format) to support confusable & intentional data from L2/22-107 and L2/22-108; for a future version of Unicode. See L2/22-124 item Sec1.

Feedback (verbatim)

Date/Time: Tue Jun 14 17:53:16 CDT 2022
Name: A./
Report Type: Public Review Issue
Opt Subject: pri451

After reviewing UTS#39 we found that there are a number of potential omissions
in the confusables.txt data file.

The result of our findings are available as a separate document.

L2/22-114

Background information / discussion

Asmus: Update from linked document (forward to whomever updates this file). Related documents L2/22-107 Proposal to Add Data for Pairs of Confusable sequences and L2/22-108 Proposal to Add Data for Pairs of Identical sequences

Sec2: Addressing inconsistencies in UAX #31

L2/22-110 from Robin Leroy, Mark Davis, Source code ad hoc working group

Recommended UTC actions

1. Action item for Robin Leroy, Asmus Freytag, EDC: Apply the changes in L2/22-110R to UAX #31, for Unicode 15.
2. Action item for Robin Leroy: Notify the release manager of the completion of changes to UAX #31 for Unicode 15.

Summary

While working on UAX #31, the source code ad hoc working group noticed some inconsistencies in Unicode Standard Annex #31 Unicode Identifier and Pattern Syntax. These have been called out by review notes in revision 36, draft 5 of the annex for Unicode 15.0β. This document proposes changes to UAX #31 to address these inconsistencies.

Sec3: Status report of the source code working group for UTC #172

L2/22-161 from Robin Leroy, Source code ad hoc working group

Recommended UTC actions

1. Action item for Robin Leroy, EDC: Apply the changes to UAX #31 described in L2/22-161, section I.2, for Unicode 15.

Summary

WG report of progress towards goals of the group.

Includes an editorial proposal in section I.2. Numbering paragraph requirements in UAX #31.

Background information / discussion

The source code ad hoc working group was created by consensus 170-C2 of the UTC, on the recommendation of the Properties & Algorithms Group, as described in document L2/22-007R2, section “Proposed Plan”, with Mark Davis as the chair.

Emoji

Emoji1: RGI_Emoji_Qualification

L2/22-160 from Mark Davis

Recommended UTC actions

1. Action item for Mark Davis and the ESC: Produce proposed updates of UTS #51 & UTS #18 that contain the changes outlined in document L2/22-160, for future versions of these standards (incl. UTS #51 version 15.1 or 16). See L2/22-124 item Emoji1.
2. Action item for Rick McGowan: Post proposed updates of UTS #51 & UTS #18. See L2/22-124 item Emoji1.

Summary

Proposal 1: Add an additional property of strings, RGI_Emoji_Qualification, with property values defined by the [existing] corresponding status values in emoji-test.txt.

Proposal 2: Add additional sequences to emoji-test.txt with the new status value ‘over-qualified’, and add the corresponding property value Over_Qualified to RGI_Emoji_Qualification.