L2/21-135

Reserved Emoji (RE), Fleshed-Out

From: Mark Davis

Date: 2021-July-13

This document is for the following action.

> **Action 167A094 Flesh out approaches RE and RID ofdocument L2/21-071 and bring back for discussion at the next UTC meeting.** See Future Unicode Emoji Options [L2/21-078]

The primary goal of Reserved Emoji (RE) is to allow implementations that run on previous versions of Unicode / Emoji to support display emoji from (some) later versions: that is, some degree of future-proofing. The main problem is that even if the implementation is upgraded with a font that supports the new emoji, the processing of emoji for line break and other segmentation purposes can fail. What the user can see, for example, is half of an emoji sequence at the end of one line, and the rest at the start.

As it turns out, we had already defined the property *Extended_Pictographic* for the purpose of new emoji, and these are already used for future-proofing in cases such as UAX #29 . We can utilize this definition to implement RE more broadly. The key policy change is that we guarantee to only allocate new emoji characters from the set of *Extended_Pictographic* from several versions ago. Then those versions would be ‘future proofed’ for display, with some small changes. Key to this is making sure that segmentation algorithms (including line break) work with *Extended_Pictographic.* As it turns out, only one change for segmentation needs to be changed. We could also document this approach in UTS #51 for other kinds of algorithms.

Define a *potential emoji* (PE) as any Extended_Pictographic code point that is unassigned.

There are 1,525 such code points in U14.0. We thus have plenty of room for the next few versions, so we need no change in U14.0 in Extended_Pictographic. To make sure that all new emoji come from Extended_Pictographic, we need the following policy recorded in the UTC minutes:

Proposed Emoji allocation policy

New Emoji characters are to be allocated from unassigned code points that are in \p{Extended_Pictographic} for the previous 2 versions of Unicode.

> For example, a new Emoji character in U15.0 must have had the property Extended_Pictographic=Yes in U13.0 and U14.0.

We also need to make sure that new emoji sequences are forward-compatible in *emoji sequences*. That means that any emoji sequence will not be broken if an emoji character is replaced by an unassigned Extended_Pictographic character. That can be accomplished with the following change:

Proposed LineBreak enhancement

Change

***LB30b** Do not break between an **emoji base** and an **emoji modifier**.*

EB × EM

To

***LB30b'** Do not break between an **emoji base** (or potential emoji) and an **emoji modifier**.*

EB × EM

[\p{Extended_Pictographic}&\p{Cn}] × EM

Proposed UTS #51 enhancement

Add a new section as follows (wording subject to editorial improvements). Note: this section can come in a later version than the Proposed Emoji allocation policy and Proposed LineBreak enhancement.

1.4.10 Future Proofing

Sometimes an implementation will receive an emoji from a newer version of software. Even if the implementation has a font that supports the emoji, processing of the emoji can fail because the implementation doesn't have newer Unicode properties. Unicode segmentation algorithms handle that situation because they recognize that any new emoji (for at least the next few versions) are all allocated from characters in \p{Extended_Pictographic}.

If an implementation is using the emoji definitions and data files in this specification directly (rather than through the segmentation algorithms), then a few adjustments need to be made to allow for future-proofing.

Define a *potential emoji* (PE) as any Extended_Pictographic code point that is unassigned.

- That is, PE :=[\p{Extended_Pictographic}&\p{Cn}]

The key is to treat all PE characters as if they were emoji for purposes of validation. This also rests on the fact that *emoji components* [ED-5] are very unlikely to change over versions. The following extended definitions are then applied in three key cases.

Future-Proofing Definitions

Definiendum	**Definiens**	**Future-Proofed Definiens**
`emoji_character`	`\p{Emoji}`	`[\p{Emoji}`**`PE`**`]`
`default_emoji_presentation_character`	`\p{Emoji_Presentation}`	`[\p{Emoji_Presentation}`**`PE`**`]`
`emoji_modifier_base`	`\p{Emoji_Modifier_Base}`	`[\p{Emoji_Modifier_Base}`**`PE`**`]`

This process is similar to the way Unicode treats certain other unassigned code points as having specific default properties. For example, certain unassigned code points recommended to be treated as if they were Arabic letters for the purpose of the Bidirectional Algorithm.

Background

RID

NOTE: As for RID, I think that the RE approach satisfied the major need without as complicated a mechanism, so I don't think it necessary to flesh that out or bring it back to the UTC.

Emoji Modifier Sequences

An emoji modifier sequence is:

emoji_modifier_sequence := emoji_modifier_base emoji_modifier

So for forward compatibility of emoji modifier sequences in segmentation, we would just need to prevent a break between a potential emoji and an emoji modifier. The following are used in the specifications of different kinds of segmentation that are in question. These include all and only the emoji_modifier characters:

Line_Break=E_Modifier (EM)

Word_Break=Extend

Grapheme_Cluster_Break=Extend

Word_Break and Grapheme_Cluster_Break are already covered, since we have the following, which doesn't depend on the nature of the preceding code point.

Do not break before extending characters or ZWJ.

GB9 × (Extend | ZWJ)

and

Ignore Format and Extend characters, except after sot, CR, LF, and Newline. (See Section 6.2, Replacing Ignore Rules.) This also has the effect of: Any × (Format | Extend | ZWJ)

WB4 X (Extend | Format | ZWJ)* → X

So the only change needed for modifier sequences would be for LineBreak. That can be done with the change listed in Proposed LineBreak enhancement.

Emoji ZWJ Sequences

These are defined by ***ED-15a*** and ***ED-16*** in UTS #15

```
emoji_zwj_element :=
  emoji_character
| emoji_presentation_sequence
| emoji_modifier_sequence


emoji_zwj_sequence :=
  emoji_zwj_element ( ZWJ emoji_zwj_element )+
```

Again, Word_Break and Grapheme_Cluster_Break are handled already by the following rules.

Do not break within emoji modifier sequences or emoji zwj sequences.

GB11 \p{Extended_Pictographic} Extend* ZWJ × \p{Extended_Pictographic}

Do not break within emoji zwj sequences.

WB3c ZWJ × \p{Extended_Pictographic}

Ignore Format and Extend characters, except after sot, CR, LF, and Newline. (See Section 6.2, Replacing Ignore Rules.) This also has the effect of: Any × (Format | Extend | ZWJ)

WB4 X (Extend | Format | ZWJ)* → X

For ZWJ sequences, LineBreak also works with no change:

LB8a** Do not break after a **zero width joiner.

ZWJ ×

A **ZWJ** will prevent breaks between most pairs of characters. This behavior is used to prevent breaks within emoji zwj sequences.

***LB9** Do not break a combining character sequence; treat it as if it has the line breaking class of the base character in all of the following rules. Treat **ZWJ** as if it were **CM**.*

Treat X (CM | ZWJ)* as if it were X.