**L2/170-434R**

**Re:** **Emoji Regex & UTS #51 Definitions**
**From:** **Mark Davis**
**Date:** **2017-01-05** (originally 2017-12-22)

---

The Definitions that we have in UTS #51 are fairly complex. Yet in the end we still depend on the validity tests (for flags and tag sequences, for example) to filter the set of emoji sequences. Because of that dependency on validity tests, we could simplify the definitions to make a simpler EBNF and Regex for detecting possible emoji characters and sequences. This has advantages for parsing and scanning text for emoji. We may also want to simplify the UTS #51 definitions.

## Proposal

1. Add a section to the UTS #51 that describes the usage of the following EBNF and Regex expressions in scanning text to identify possible emoji.
2. Discuss in the UTC whether we would want to pursue simplifying our UTS #51 definitions to correspond more to the EBNF.

## Issue

The following EBNF can be used to scan for *possible* emoji, which can then be verified by performing validity tests as specified in UTS #51. For brevity here, it uses some proposed property aliases, such as EMD for Emoji_Modifier. It is much simpler than the expressions currently in Definitions: you can think of it as the convex hull of what is matched by those definitions. Of course, it also includes some degenerate cases as a by-product of that simplicity, but those are weeded out by validity tests in any event.

If #2 is done, then the check for valid skin tones would be characterized as a validity test rather than a syntactic feature.

| EBNF | Notes |
|---|---|
| `possible_emoji := possible_zwj_element (\x{200D} possible_zwj_element)+` | 200D = joiner |
| `possible_zwj_element :=`<br>`  \p{RI} \p{RI}`<br>`\| \p{Emoji} emoji_modification?` | RI = Regional_Indicator |
| `emoji_modification :=`<br>`  \p{EMD}`<br>`\| \x{FE0F}? \p{Mn}*`<br>`\| [\x{E0020}-\x{E007E}]+ \x{E007F}` | Mn= Nonspacing_Mark<br>FE0F = emoji VS<br>E00xx are tags<br>E007F is the TERM tag. |

From this a regex can be generated, as below. While it may seem complex, it is far simpler than what would result from the Definitions, which result in regex expressions which are many times more complicated, and yet still require verification with validity tests.

**Regex**

```
(\p{RI} \p{RI}
| \p{Emoji} (\p{EMD} | \x{FE0F}? \p{Mn}* | [\x{E0020}-\x{E007E}]+ \x{E007F})?)
   ( \x{200D}
(\p{RI} \p{RI}
| \p{Emoji} (\p{EMD} | \x{FE0F}? \p{Mn}* | [\x{E0020}-\x{E007E}]+ \x{E007F})?))+
```

## Notes

Note that this **EBNF** shares a characteristic with the UTS #51 Definitions: it is finer-grained than a grapheme cluster. That is, if you have “A<zwj><emoji>”, the Unicode 10 grapheme cluster rule GB11 will cause that to be one grapheme cluster, while a scanner based on the **EBNF** will find an emoji *within* that grapheme cluster. (Similarly, a scanner looking for ASCII letters will find an “A” within that same grapheme cluster.)

The proposed Unicode 11.0 GB11 rule is tighter, since it requires an emoji (or emoji-like) character before the ZWJ. So a sequence like “A<zwj><emoji>” will not count as a grapheme cluster under that new rule.

In any event, a subsequent more precise test for valid sequences would also “split” a grapheme cluster like “<RIa><RIb>”, by finding it to contain 2 emoji (if “<RIa><RIb>” doesn’t form a valid flag). It is a known feature of the stock grapheme cluster rules that they are not built to handle degenerate cases if that makes the rules too complicated.