

Subject: Customized Emoji  
From: M. Davis, Emoji SC  
Date: 2015-07-25  
Working Draft: <https://goo.gl/ohqZjx>

The Emoji Subcommittee (ESC) was tasked with brainstorming ideas for interchanging customized emoji without requiring each one to be encoded. Here is the result of some of the discussions we've had.

*Disclaimer: This document represents a preliminary discussion which is subject to UTC review. Any and all details are therefore subject to change. Any characters that are contained in this proposal, along with their attributes, including, but not limited to, their names, code points, and representative glyphs, are purely hypothetical at this point.*

- [1. Status Quo](#)
- [2. Joiners](#)
- [3. Variation Sequences](#)
- [4. IVS](#)
- [5. Emoji Modifiers](#)
- [6. Standardized Tagged Emoji Sequence \(STES\)](#)
- [7. Tagged URL Emoji Sequence \(TUES\)](#)
  - [7.1 Security and other implementation considerations](#)
- [8. Tagged Encapsulated Graphics](#)

Here are some options we discussed.

## 1. Status Quo

Stay out of the business of providing more “generative” capability; people should use markup or other protocols for stickers.

## 2. Joiners

The use of joiners requires no action by the Unicode Consortium. This mechanism has limitations, since it is simply a sequence of existing emoji, and has no structure for placement. It also only works to combine emoji that are already defined. It should only be used where the sequence would make sense if simply listed serially. An example might be a devil with sunglasses.

To promote interoperability, we could provide a separate list of ones supported by one or more major vendors, and update it periodically (more often than Unicode releases).

In discussion at the 2015-feb UTC, there was some discussion of a "brackets" approach rather than joiners, with a fallback display using visible bracketing. Possibly even just using ASCII [brackets]. This would have issues, since one can search on a page for "[" and get a match inside an emoji (unless the browser takes special precautions).

## 3. Variation Sequences

We can add specific variation sequences for given emoji. This is a slower process (but faster than new characters), but limited to only 16 (or 14 for ones with text/emoji variants).

## 4. IVS

We could (this would require a change) allow the registration of emoji glyph sets, extending the usage of the Ideographic Variation Selectors to also affect emoji. This would allow for many more free-form variants, and decouple them from versions of the Unicode Standard.

However, even with 240 of them, they could quite quickly become exhausted unless we limited it in some way (like one VS per emoji per full member).

Options:

1. new block of Emoji variants?
2. maybe not make registration as free-form as IVS?

## 5. Emoji Modifiers

Where the fallback appearance needs to be distinctive (such as the case for skin tone), it may make sense to add new emoji modifiers, such as GINGER (to give the previous emoji red hair).

## 6. Standardized Tagged Emoji Sequence (STES)

Define a syntax for a sequence of tags following an existing emoji. Provide a “StandardizedTaggedEmoji.txt” which is a bit like StandardizedVariants.txt. People can propose additions to that file to the emoji subcommittee. It will assess them, and pass on recommendations to the UTC.

The base emoji should always be chosen to be the best fallback emoji possible, since it will appear if the system does not handle the particular tagged emoji sequence. If the system can handle tags, but doesn't have a corresponding image, then the base emoji should be shown with a dotted-box around it, to indicate that the appearance is a fallback glyph.

This would not be a registry: we would still examine any proposal to see that it made sense, and require a sample image (SVG?) for the proposal, and it would have to be a reasonable variation of the base emoji character. So one might have a FRANKENSTEIN image as an STES with a base of ☹ but not with 😬.

## 7. Tagged URL Emoji Sequence (TUES)

This would be a very flexible approach, providing a way to do “stickers” within plain text. We could (this would require a change) allow tagged emoji sequences of the form:

```
<tues> := <emoji><body>
<body> := char-tag{1,15}
<char-tag> := U+E002E TAG FULL STOP
               | U+E002F TAG SOLIDUS
               | U+E0030 TAG DIGIT ZERO .. U+E0039 TAG DIGIT NINE
               | U+E0061 TAG LATIN SMALL LETTER A .. U+E007A TAG LATIN SMALL LETTER Z
```

Let **{body}** be obtained by subtracting 0x#E0000 from each codepoint in **<body>**. The interpretation of **<tues>** is as the following HTML:

```
<img alt='<emoji>' src='https://{body}' style='height:1em; width:1em; vertical-align:text-top'>
```

Example:

The following text has one normal emoji followed by one customized emoji:

hi ☹️ 🤪 joe

It is encoded as the following:

```
hi &#x1F600&#x1F479;{goo.gl/p70vTG} joe
```

And should display as if it were HTML encoded as:

```
hi &#x1F600<img alt='&#x1F479;' src='https://{goo.gl/p70vTG'
style='height:1em; width:1em; vertical-align:text-top'> joe
```

If the body is not supported, it would appear as just the base emoji with an outline (the gray box below should be a

dotted outline):

hi  joe

The base emoji should always be chosen to be the best fallback emoji possible, since it will appear if the system does not handle tagged emoji sequences, or is not able to fetch the corresponding image. If the system can handle tags, but can't fetch an image using the body, then the base emoji should be shown with a dotted-box around it, to indicate that the appearance is a fallback glyph.

This also means that we would continue to add emoji bases, so that there would be suitable ones for the TUESs to fall back to. But the pressure would be vastly reduced.

We restrict the length of the body to keep from burdening plain text too much. Maintainers of images can always find a way to have short <body> sequences, either with general purpose [URL shorteners](#), or with their own (mystuff.com/i/A8nkQ1i).

The disadvantage of this and #8 below is that they would be too flexible; much more complicated for implementers and thus more prone to error.

### 7.1 Security and other implementation considerations

The same security considerations govern as when the corresponding HTML is used. In addition, while a system would probably cache commonly used images, the cache must not be unbounded. If the latency is too great, or the user is offline without a cached image, the system should show the fallback glyph instead.

As far as "stability" of images go, this is neither more nor less stable than the use of images in HTML, or stickers. In Gmail, the images underlying the emoji defined within Gmail remain stable; but that may or may not be the case for other providers.

For that matter, the images shown for emoji characters are not "stable", in that they depend on the font used to display them. Successive versions of Android or iOS, for example, have updated those images.

## 8. Tagged Encapsulated Graphics

In this approach, we would fully encapsulate any graphic using TAG characters. It corresponds to encoding as [Data URLs](#). The upside is that it is completely flexible; the downside is that these can be huge, and would load down plain text significantly.

Options: lower res than PNGs, maybe max size of 24x24? limited colors, maybe 16? One of the color glyph formats from OpenType?

```
