Coded Hashes of Arbitrary Images

(or: the last frontier of Emoji encoding)

Steven R. Loomis
srl@icu-project.org
(individual contribution)

Keith Winstein
keithw@cs.stanford.edu
Stanford University

2016-05-02

1 Introduction

> Emoji are pictographs (pictorial symbols) that are typically presented in a colorful cartoon form and used inline in text. […] In Unicode 8.0, there is a total of 1,282 emoji, which are represented using 1,051 code points.[1]

Recently, there has been considerable interest in adding newly created pictorial symbols, not found in any existing character set, to the Unicode Standard as emoji.[2] Advocacy groups and others request these code points because Unicode plain text remains the dominant interoperable interchange format for messaging. In practice, before a new emoji can be used, a code point must be assigned and be recognized by the sender and receiver. However, a longer-term goal for Unicode is that implementations should support "embedded graphics, in addition to the emoji characters".[3]

In this proposal, we describe a mechanism to uniquely identify arbitrary images within a plain-text Unicode character sequence. This will allow implementors to create their own emoji without needing to request and wait for the assignment of a code point. The basic idea is to encode a globally-unique *secure hash* of the emoji in a Unicode character sequence. Once the receiver knows the image's hash, it may already have the corresponding image, or may have a choice of several mechanisms to retrieve it.

Our technique will gracefully degrade on legacy Unicode implementations, but our proposal is limited to allowing Unicode to *uniquely identify* an arbitrary image. We propose to leave to implementors the details of how to retrieve the actual image.

[1] Mark Davis and Peter Edberg. *Unicode Technical Report 51: Unicode Emoji*. 2015. URL: `http://www.unicode.org/reports/tr51/`.

[2] *E.g.,* Taco Emoji campaign, Beard Emoji campaign, Dumpling Emoji campaign. There are currently 79 candidate emoji that have been assigned tentative code points.

[3] Davis and Edberg, *Unicode Technical Report 51: Unicode Emoji*, op. cit., Section 8, "Longer Term Solutions".

2 Proposal

We propose that implementations be able to uniquely identify an arbitrary image within a Unicode character sequence. The means will be to encode, in a series of coded characters, a secure hash of a canonical representation of the image.

3 Non-Goals

UTR # 51 outlines some possible use-case scenarios as well as challenges with embedded graphics.[4] It is not the goal of this document to address all aspects of embedded graphics. This document will focus on those aspects related to character encoding only, and leave to domain experts and implementers to determine standardized approaches to topics such as privacy and security, actual data transfer of the image content, reliability and availability, and the like.

4 Overview

This document proposes:

1. The encoding of a new base character for image transfer, `U+FFF8 EMBEDDED IMAGE BASE`
2. The allocation of the entire plane `0C` for the purpose of image hashes

(TODO TODO)

To generate a hash, the image content (a standardized size, 128x128 png), plus the metadata (content-type, alternates, etc) is SHA-256 hashed.

The actual encoding is:

`U+FFF8 + U+0Cxxxx + U+0Cxxxx + U+0Cxxxx ...`

where each `U+0C` code point, from `U+0C0000 – U+0C7FFF` contains 15 bits of the SHA-256 hash.

The U+0C code points will have a combining character general category (which?).

The more `U+0C` present (up to 20 - 300 bits) the longer and more specific the hash is.

(TODO TODO)

[4] Ibid., Section 8, "Longer Term Solutions".

References

Davis, Mark and Peter Edberg. *Unicode Technical Report 51: Unicode Emoji*. 2015. URL: http://www.unicode.org/reports/tr51/.

Colophon

Typeset by LaTeX. Made with 100% recycled bits. All opinions belong to the authors and do not reflect the opinions of their associated employers.

Thank you to Keith Winstein for the discussion which finally kicked off this document.