L2/15-052

Subject: **Proposed changes for UAX31 for Hashtags**
From: **Mark Davis**
Date: **2015-01-29**

Re: 138-A42 For 8.0, look at the broader question of incorporating more natural language words into looser identifiers in contexts like hashtags. See consensus 138-C16 and UAX #31.

Proposal: make the following changes to http://unicode.org/reports/tr31/ to add support for hashtags.

1.4 Conformance
add:

UAX31-C3. *An implementation claiming conformance to this specification for parsing Hashtags shall do so in accordance with* ***Section 6.1*** **Parsing Hashtags**

UAX31-C4. *An implementation claiming conformance to this specification for determining when two Hashtags are identical shall do so in accordance with* ***Section*** **6.2 HashTag Identity**

2.4 Specific Character Adjustments

add definition:

An InclusionCharacter is defined to be any character included in *Table 3. Candidate Characters for Inclusion in Identifiers.*

add to Table 3:

The following characters, to prevent surprises when words are pasted after hashtags:

U+00AD ( ) SOFT HYPHEN
U+2011 (‑) NON-BREAKING HYPHEN
U+200B ( ) ZERO WIDTH SPACE
U+2060 ( ) WORD JOINER
U+180E ( ) MONGOLIAN VOWEL SEPARATOR

Issues:

1. *Alternatively, we could have a separate table 3a with the new characters, and modify the definition.*
2. *Should we add compatibility variants of items in the table? Such as U+FF0D (－) FULLWIDTH HYPHEN-MINUS. There is text in that section that says they can be added, but if we want them, we should have a definitive list to reference.*

(new) Section 6 Hashtags

add:
Hashtags have grown to be quite popular in social media, but there is little consistency across platforms. As noted in Section 2.4, compared to other identifiers, hashtags are expected to more fully encompass words or phrases used in natural languages. The following are recommended for parsing hashtags.

6.1 Parsing Hashtags

A hashtag consists of a hash trigger (such as #), followed by a Unicode XID_Start character, then followed by zero or more of XID_Continue characters. The latter can have interspersed single InclusionCharacter (see Section 2.4).

HashTrigger = [#﹟＃] XID_Start (InclusionCharacter? XID_Continue+)*

6.2 HashTag Identity

Hashtags should be compared loosely, that is, insensitive to case, compatibility variants, and format characters. Thus two hashtags should compare as equivalent if they are the same under Compatibility Normalization and Case Folding:

X ~ Y
iff
NFKC_CF(X) = NFKC_CF(Y)

The simplest way to deal with this is to perform NFKC_CF on the hashtag immediately after it is parsed. However, the input form of the hashtag should be retained in any original messages, so that the original display is not distorted by the lack of Join_Controls.

6.3 Security

Based on usage, it does not appear that confusability of Hashtag strings is a significant issue. If that comes to present a problem, then the techniques described in UTR #36 and UTS #39 can be applied, such as in restricting mixed scripts.

6.3 Stability

With each new version of Unicode, new characters become allowed in Hashtags. However, sometimes the properties of Unicode characters may also change, thus disallowing characters that were allowed previously.

The XID_Start and XID_Continue are kept backwards compatibility: see Section 2.5 Backward Compatibility. However, there is no guarantee about the InclusionCharacters; while unlikely, it is possible that characters might be removed from that set.

There are two main strategies for dealing with this.

1. If it is not important for the implementation to maintain absolute backwards compatibility for hashtags, then this possibility can be ignored.
2. If it is important, then the implementation can maintain its own set of grandfathered InclusionCharacters, and simply union-in new characters from successive versions of Unicode.