Proposal for an option in UAX #31 to prohibit ZWJ/ZWNJ for identifier security

In response to action item 165-A44

Asmus Freytag and Michel Suignard (with additional input from Mark Davis)

Date: April 12, 2022

This document is in response to an action item to allow a profile option to be added in UAX #31 and or UTS#49 to prohibit ZWJ/ZWNJ altogether, for better identifier security. This behavior would match the DNS Root Zone, where ZWJ and ZWNJ are strictly prohibited. Similarly strict restrictions are likely appropriate for at least Second Level Domain.

Review of existing language in UAX#31 and UTS#39 points to the fact that the existing specification is sufficient for the purpose of defining identifier syntax that restricts ZWJ/ZWNJ. However, a case can be made that the mechanism could be clarified and the rationale for employing better explained.

Rationale for restricting ZWJ / ZWNJ

ZWJ and ZWNJ are invisible format characters that request a change for a more or less connected typographical rendition of adjacent characters. In many contexts their use is entirely optional and may not have any visible effect on an identifier. In some other cases, a script-specific selection of rendering behavior may be intended (but even for those scripts, not all contexts would likely result in a visible change to the identifier). In a few cases, such as in Sinhala, a word may appear malformed or misspelled without one of these characters.

There is a widely held view that the few cases where ZWJ or ZWNJ have true orthographic status (as opposed to more stylistic selection among different appearance) do not outweigh the security risks associated with invisible characters, for which the presence may not always be detectable, or not reliably detectable.

In this context it's useful to note that the purpose of identifiers is always to have a "useful mnemonic" and not to allow faithful representation of every word or phrase for some language(s).

As already alluded to above, while these are allowed (with strict context rules) in IDNA2008, they are categorically excluded from the DNS Root Zone[1], as well as from Reference LGRs for the Second Level, except for specifically enumerated sequences in certain scripts[2] where they have strong orthographic function, and where their presence would result in a reliably distinct appearance of the identifier.

---

[1] See https://www.icann.org/resources/pages/root-zone-lgr-2015-06-21-en and look for the "Procedure to Develop and Maintain Label Generation Rules for the Root Zone with Respect to IDNA Labels".

[2] See the Second Level Reference LGR for the Sinhala Script, https://www.icann.org/sites/default/files/packages/lgr/lgr-second-level-sinhala-script-22apr21-en.html

Suggested changes

Add a section that lays out, in human-readable terms, the intention behind “Default identifiers” and that describes clearly the kinds of characters that are included or excluded with focus on achieving an understanding . Relate this definition to other identifier definitions in wide use. How close does it come to IDNA 2008, how close does it come to the DNS Root Zone Label Generation Rules (the latter are a subset of IDNA 2009 that excluded CONTEXTJ code points as well as letters not in everyday common use)?[3]

Improve the explanation on how to create identifier syntaxes that use context rules and enumerated subsequences. Reflect (relatively) recent advances in the space of IDNs as implemented in specifications such as RFC 7940.

For example, below R1, add a note that
• R1 does not include ZWJ or ZWNJ, and
• Implementations that support ZWJ or ZWNJ should provide a profile of R1a

Improve property lookup. When reading a statement about properties/property values it should be possible to access the actual listing in the UCD by following the shortest number of links. This process should not assume the reader “knows” where to find a property.

Take suitable actions to further restrict ZWJ/ZWNJ so their inclusion into identifier profiles requires explicit intent.

Background

It appears that UAX31-R1. *Default Identifiers* already doesn't include ZWJ/ZWNJ, so the core definition of identifiers doesn't include them. However, short of sifting through several levels of property files, none of which are directly accessible by hyperlinks, it is not clear that the definition as given bypasses ZWJ/ZWNJ.

However, there is already a profile option in "UAX #31 to prohibit ZWJ/ZWNJ".

Namely the main conformance clause, UAX31-C2, provides a set of R clauses that one can choose from (and an implementation MUST choose a subset since some of them conflict). R1, R1a, and R1b are listed separately. Thus an implementation can simply claim conformance to R1 and R1b, and exclude R1a, to get what is being asked for in L2/22-082.

It appears that the following provides a tool to match the desirable forms of restrictions for ZWJ/ZWNJ when they are not entirely prohibited:

[3] The actual language should be based on further discussion in the property or algorithms group, possibly with input from the ad-hoc on source code.

> UAX31-R1a. *Restricted Format Characters:* To meet this requirement, an implementation shall define a profile for UAX31-R1 which allows format characters as described in Section 2.3, Layout and Format Control Characters.

An implementation may further restrict the context for ZWJ or ZWNJ, such as by limiting the scripts allowed or limiting the occurrence of ZWJ or ZWNJ to specific character combinations, if a clear specification for such a further restriction is supplied.

However, in the discussion of section 2.3, the three conditions A, B and C are referenced, none of which match what would be natural for IDNs governed by Label Generation Rules (LGR)[4]. In an LGR, instead of generic restrictions, specific context rules would be defined (or, as in the only case where ZWJ and ZWNJ are currently supported in the Second Level Reference LGRs) the actual conjuncts where their presence is essential are enumerated explicitly.

This treatment is hinted at in the subsubsection "2.3.1 Limitations". The discussion there focuses on parsing issues. In many situations, the detailed context rules only apply on first encounter, definition, or "registration" of an identifier. Subsequent "look-up" only needs to match against the registered value. Such matches would fail if format characters are present where they are not present in the "registered" version of the identifier.

For the separate, if related purpose, of segmenting text, a looser definition of identifier can be used; generally, being able to identify potential, if possibly invalid identifiers for verification against a definition or a registered value is more useful that detecting contextually invalid identifiers during segmentation.

[4] See RFC 7940, "Representing Label Generation Rules in XML"