L2/23-017

Recommendations of the source code working group for UTC #174

To: PAG, UTC
From: Robin Leroy, SCWG
Date: 2023-01-13

The SCWG recommends that the UTC take the following dispositions:

1. The UTC notes the amendments to the proposed update for Unicode Standard Annex #31, *Unicode Identifiers and Syntax*, mentioned in sections A and B of document L2/23-017, for Unicode Version 15.1.

2. The UTC notes the amendments to Proposed Draft Unicode Technical Standard #55, *Unicode Source Code Handling*, mentioned in section C of document L2/23-017.

3. **[Proposed] Consensus:** Advance Proposed Draft Unicode Technical Standard #55, *Unicode Source Code Handling*, to Draft Unicode Technical Standard #55, *Unicode Source Code Handling*.

In re feedback from Reini Urban on "TR31" [Wed Jan 19 03:46:02 CST 2022], see 171-A121:

4. **[Proposed] Action Item for** Rick McGowan: Thank the correspondent for their feedback of [Wed Jan 19 03:46:02 CST 2022], and relay the SCWG response in section D of L2/23-017.

5. The UTC notes document L2/22-267, *Comments on L2/22-230 "Mathematical notation profile for default identifiers"*, but takes no further action.

> **Note:** Following discussion at UTC #173 and in the SCWG, adjustments to wording have been made, and it has been clarified that the use of these characters in identifiers is discouraged in general use; this follows from the recommendations in PDUTS #55. No further action is required.

Contents:

Notation

In sections A, B, and C of this document, technical changes with respect to the Proposed Update for UAX #31, Version 15.1, draft 2 or Proposed Draft Unicode Technical Standard #55, Version 1, draft 1 (which correspond to the text presented to UTC #173) are shown in cyan. In the case of UAX #31, changes shown in yellow are changes with respect to the text in Version 15.0 that have already been presented to UTC #173. In the actual proposed update and proposed draft, the changes are shown only in yellow, with respect to Unicode Version 15.0 in the case of UAX #31, and with respect to draft 1 in the case of PDUTS #55.

A. Unversioned references to the UCD

The following proposed technical changes have been included in the Proposed Update (draft 6) to UAX #31. See the Rationale section below.

Section 1.4, Conformance

> ***UAX31-C1**. An implementation claiming conformance to this specification shall identify the version of this specification.*
>
> > **Note:** An implementation can make use of the definitions from a given version of this specification while backing them with property assignments from an unversioned reference to the Unicode Character Database. In this case, the implementation should specify a minimum version of Unicode for the properties.

Section 2, Default Identifiers

> ***UAX31-R1b. Stable Identifiers:** To meet this requirement, an implementation shall guarantee that identifiers are stable across versions of the Unicode Standard: that is, once a string qualifies as an identifier, it does so in all future versions of Unicode.*
>
> > **Note:** The UAX31-R1b requirement is relevant when an identifier definition is based on property assignments from an unversioned reference to the Unicode Standard, as property assignments may change in a future version of the standard. It is typically achieved by using grandfathered characters. See *Section 2.5, Backward Compatibility*. Where profiles are allowed, management of those profiles may also be required to guarantee backwards compatibility. Typically such management also uses grandfathered characters. Because of the stability policy [Stability], if an implementation meets either requirement UAX31-R1 or UAX31-R2 without declaring a profile, that implementation also meets requirement UAX31-R1b.
>
> > **Example:** Consider an identifier definition which uses UAX31-R1 default identifiers with a profile that adds digits (characters with General_Category=Nd) to the set Start, and uses an unversioned reference to the Unicode Character Database, with a minimum version of 5.2.0.
> >
> > With property assignments from Unicode Version 5.2.0, both ▫ (U+19DA) and A▫ (U+0041, U+19DA) are valid identifiers under this definition: U+19DA has General_Category=Nd.

In Unicode Version 6.0.0, U+19DA has General_Category=No. The identifier A□ (U+0041, U+19DA) remains valid, because XID_Continue includes any characters that used to be XID_Continue. However, □ is not a valid identifier, because U+19DA is no longer in the set [:Nd:].

In order to meet requirement UAX31-R1b, the definition would need to be changed to add to the set Start all characters that have the property General_Category=Nd in any version of Unicode starting from Unicode 5.2.0 and up to the version used by the implementation.

Rationale

Readers have interpreted UAX31-R1b as implying a stability policy for language standards that claim conformance; for instance, as meaning that "once a string qualifies as an identifier in some version of C++, it does so in all future versions of C++", which is too restrictive (for instance, keywords can be added), and is neither the intent nor the letter of the requirement. Standards have unversioned references to Unicode (and we encourage that), but we did not mention how this should be done in the context of identifier definitions.

B. Exclusion of the inappropriate VS in the Emoji Profile.

The following proposed technical changes have been included in the Proposed Update (draft 6) to UAX #31. See the Rationale section below.

Section 7.2, Emoji Profile

The emoji profile for default identifiers consists of: ~~the~~

1. The addition of the RGI emoji set defined by ED-27 in Unicode Technical Standard #51, Unicode Emoji [UTS51] for a given version of Unicode to the sets *Start* and *Continue* in definition UAX31-D1.
2. The removal of the code point U+FE0E VARIATION SELECTOR-15 (the Text Presentation Selector) from the set *Continue*.

[...]

The emoji profile includes characters that are in Pattern_Syntax; it is therefore associated with a profile for UAX31-R3b, which consists of replacing each emoji character of a certain subset of [:Pattern_Syntax:] by its text presentation sequence (ED-8a):

1. Remove the characters in the set [[:Pattern_Syntax:]&[:Emoji_Presentation:]] from the set of characters with syntactic use.
2. For all C in [[:Pattern_Syntax:]&[:Emoji_Presentation:]], add the sequence consisting of C followed by U+FE0E VARIATION SELECTOR-15 (the Text Presentation Selector) to the set of characters with syntactic use.

In addition, in order to avoid lexical ambiguities between identifiers and operators, the emoji profile includes a profile for UAX31-R3c, which consists of the removal of the character U+FE0F VARIATION SELECTOR-16 (the Emoji Presentation Selector) from the set *Continue*.

Rationale

Without these removals, it is ambiguous whether ☕ and 😊 are operators or identifiers when meeting requirements UAX31-R1 and UAX31-R3c with the emoji profile:

1. ☕ = 🟤 (Added to Start) + VS-15 (XID_Continue) matches the identifier syntax, and is directly added to the syntax set;
2. 😊 = ☺ (in syntax) + VS-16 (Mn) matches the operator syntax, and is directly added to the identifier *Start* set.

C. Recommended handling of ZWJ & ZWNJ in computer language identifiers.

The following proposed technical changes have been included in draft 5 of the Proposed Draft UTS #55. See the Rationale section below.

Section 5.1.3, General Security Profile

> As described in Section 6.1, Confusables Data Collection, in Unicode Technical Standard #39, Unicode Security Mechanisms [UTS39], the entirety of Unicode is not in scope for thorough confusables data collection. In order to ensure that confusable detection is effective, implementations should provide a mechanism to warn about identifiers that are not in the General Security Profile for identifiers, as defined in Section 3.1, General Security Profile for Identifiers, in [UTS39].
>
> For this purpose, implementations should use a modification of the General Security Profile which allows the characters U+200C ZERO WIDTH NON-JOINER and U+200D ZERO WIDTH JOINER. This is because these characters are necessary in the orthographies of some major languages. The potential issues arising from the use of these default ignorable characters are a small subset of the broader problem of confusability, which must be dealt with using the mechanisms described in Section 5.1.1, Confusable Detection.
>
> The contextual restrictions for these characters described in *Section 3.1.1, Joining Controls*, in *Unicode Technical Standard #39, Unicode Security Mechanisms* [UTS39], should be applied, as they can mitigate usability issues.
>
> > **Example:** Consider بـه ر و ز, the Persian word for "update" (which could be romanized be-ruz, where the hyphen roughly corresponds to the linguistic separation indicated by the ZWNJ in Persian). This word contains a zero width non-joiner; removing it yields بـهر و ز, which displays differently, and is a different word (the name Behrooz). An implementation should not warn about the use of the identifier بـه ر و ز.
> >
> > However, a programmer who uses Persian identifiers, having a key for that character, could accidentally type a zero width non-joiner in a place where it has no effect, as in the following:
> >
> > ```
> > if Version > Current_Version then  -- ZWNJ between V and e.
> >
> >   بـه روز (Version); -- ZWNJ between و and ز.
> > ```
> >
> > Unless confusable detection is performed prior to successful compilation, the programmer would be presented with baffling compilation errors, as the code would fail to compile even though it looks correct. A diagnostic based on the General Security Profile, which requires

only lexical analysis, and can therefore be performed early, would warn about those spurious zero width non-joiners.

Rationale

The security issues posed by these characters are addressed by confusable detection as amended by the proposed update to UTS #39 (they are removed from the skeleton). However, the contextual checks would help with usability issues, so we should not recommend removing them.

Note: An intermediate published draft had the following, which recommended removing the contextual checks:

For this purpose, implementations should use a modification of the General Security Profile which allows the characters U+200C ZERO WIDTH NON-JOINER and U+200D ZERO WIDTH JOINER, and does not apply the contextual restrictions for these characters described in Section 3.1.1, Joining Controls, in [UTS39]. This is because these characters are necessary in the orthographies of some major languages. The potential issues arising from the use of these default ignorable characters are a small subset of the broader problem of confusability, which must be dealt with using the mechanisms described in Section 5.1.1, Confusable Detection.

Review note: The preceding paragraph was added following discussion in the SCWG after UTC #173; it has not yet been reviewed by the UTC, but is included for public review.

D. SCWG response to feedback [Wed Jan 19 03:46:02 CST 2022]

The definitions in UAX #31 do not provide for identifiers secure against spoofing issues, let alone linguistic well-formedness with respect to the use of Myanmar medial consonants. Where spoofing concerns are relevant, the mechanisms described in UTS #39 should be used.

A clarification of that separation of concerns is given in the Proposed Updates for UAX #31 and UTS #39, PRIs #462 and #463; guidance on the appropriate use of the security mechanisms in programming environments is given in Proposed Draft Unicode Technical Standard #55, PRI #466.

The fullwidth characters and Arabic presentation forms mentioned in the feedback are all excluded by the General Security Profile defined in UTS #39.

The Myanmar script *is* part of the recommended scripts, and its medial letters are allowed by the General Security Profile; however, its medial letters are *not* part of XID_Start, being combining marks (some spacing, some nonspacing). For more on Myanmar and its medial consonants, see *The Unicode Standard*, Version 15.0.0, Section 16.3.

There is no such thing as a MEDIAL Thai letter.