Update Annex E onto Unicode 16

Steve Downey <sdowney@gmail.com>
<sdowney2@bloomberg.net>

Document #:	P3717R0
Date:	2025-05-19
Project:	Programming Language C++
Audience:	SG16, EWG, CWG

Abstract

Update the non-normative Annex E to reflect Unicode 16. Remove the reference to the UAX31-R1a Restricted Format Characters requirement removed from UAX31-41.

Contents

1 Introduction

The current version of Unicode©Standard Annex #31 – Unicode Identifiers and Syntax clarifies and improves some of the rules for identifiers and white space syntax. Parallel work is investigating using the new Mathematical Compatibility Notation Profile to allow certain characters such as ∂ and ∇, to better support translation of math and physics equations into C++ code. There are also opportunities to clean up the description of white space in parsing C++.

This paper proposes no changes in either area. It simply updates the current Annex E to match the forms used in Unicode Annex 31 and removes a conformance point that Unicode has removed, UAX31-R1a Restricted Format Characters.

2 Proposal

Strike the reference to UAX31-R1a. Update all reference names to the UAX31 form in the current UAX31-41.

3 Wording

Update Annex E as follows.

[uaxid.general] General

1 This Annex describes the choices made in application of UAX #31 (“Unicode Identifier and Pattern Syntax”) to C++ in terms of the requirements from UAX #31 and how they do or do not apply to this document. In terms of UAX #31, this document conforms by meeting the requirements R1 “Default Identifiers” and R4 “Equivalent Normalized Identifiers” from UAX #31. The other requirements from UAX #31, also listed below, are either alternatives not taken or do not apply to this document.

[uaxid.def] R1 Default identifiers

[uaxid.def.general] General

1 UAX #31 specifies a default syntax for identifiers based on properties from the Unicode Character Database, UAX #44. The general syntax is

```
<Identifier> := <Start> <Continue>* (<Medial> <Continue>+)*
```

where `<Start>` has the XID_Start property, `<Continue>` has the XID_Continue property, and `<Medial>` is a list of characters permitted between continue characters. For C++ we add the character U+005F LOW LINE, or _, to the set of permitted `<Start>` characters, the `<Medial>` set is empty, and the `<Continue>` characters are unmodified. In the grammar used in UAX #31, this is

```
<Identifier> := <Start> <Continue>*
<Start> := XID_Start + @\textrm{\ucode{005f}}@
<Continue> := <Start> + XID_Continue
```

2 This is described in the C++ grammar in **??**, where *identifier* is formed from *identifier-start* or *identifier* followed by *identifier-continue*.

[uaxid.def.rfmt] UAX31-R1a Restricted format characters

1 If an implementation of UAX #31 wishes to allow format characters such as U+200D ZERO WIDTH JOINER or U+200C ZERO WIDTH NON-JOINER it must define a profile allowing them, or describe precisely which combinations are permitted.

2 C++ does not allow format characters in identifiers, so this does not apply.

[uaxid.def.stable] R1b Stable identifiers

1 An implementation of UAX #31 may choose to guarantee that identifiers are stable across versions of the Unicode Standard. Once a string qualifies as an identifier it does so in all future versions.

2 C++ does not make this guarantee, except to the extent that UAX #31 guarantees the stability of the XID_Start and XID_Continue properties.

[uaxid.immutable] UAX31-R2 Immutable identifiers

1 An implementation may choose to guarantee that the set of identifiers will never change by fixing the set of code points allowed in identifiers forever.

2 C++ does not choose to make this guarantee. As scripts are added to Unicode, additional characters in those scripts may become available for use in identifiers.

[uaxid.pattern] UAX31-R3 Pattern_White_Space and Pattern_Syntax characters

1 UAX #31 describes how formal languages such as computer languages should describe and implement their use of whitespace and syntactically significant characters during the processes of lexing and parsing.

2 This document does not claim conformance with this requirement from UAX #31.

[uaxid.eqn] UAX31-R4 Equivalent normalized identifiers

1 UAX #31 requires that implementations describe how identifiers are compared and considered equivalent.

2 This document requires that identifiers be in Normalization Form C and therefore identifiers that compare the same under NFC are equivalent. This is described in **??**.

[UAX31-R5 Equivalent case-insensitive identifiers] [uaxid.eqci]

1 This document considers case to be significant in identifier comparison, and does not do any case folding. This requirement from UAX #31 does not apply to this document.

[UAX31-R6 Filtered normalized identifiers] [uaxid.filter]

1 If any characters are excluded from normalization, UAX #31 requires a precise specification of those exclusions.

2 This document does not make any such exclusions.

[UAX31-R7 Filtered case-insensitive identifiers] [uaxid.filterci]

1 C++ identifiers are case sensitive, and therefore this requirement from UAX #31 does not apply.

[UAX31-R8 Hashtag identifiers] [uaxid.hashtag]

1 There are no hashtags in C++, so this requirement from UAX #31 does not apply.

4 Impact on the standard

No normative change. Modifies non-normative text describing the conformance points with the Unicode Identifier standard.

Document history

Initial version.