0.1 General [uaxid.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.

0.2 R1 Default identifiers [uaxid.def]

0.2.1 General [uaxid.def.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 5.11, where *identifier* is formed from *identifier-start* or *identifier* followed by *identifier-continue*.

0.2.2 R1a Restricted format characters [uaxid.def.rfmt]

1 The clause R1a has been removed from UAX #31.

> The characters that were added when meeting this requirement are now part of the default; the contextual checks required by this requirement remain as part of the General Security Profile in Unicode Technical Standard #39, Unicode Security Mechanisms.

0.2.3 R1b Stable identifiers [uaxid.def.stable]

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.

0.3 R4 Equivalent normalized identifiers [uaxid.eqn]

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 5.11.

0.4 Requirements of UAX #31 for which no claims are made [uaxid.nonobservance]

1 UAX #31 version 15.1 has conformance requirements which either do not apply to C++ or which this document makes no claim about.

(1.1) — R2. Immutable Identifiers

(1.2) — R3. Pattern_White_Space and Pattern_Syntax Characters

(1.3) — R5. Equivalent Case-Insensitive Identifiers

(1.4) — R6. Filtered Normalized Identifiers

(1.5) — R7. Filtered Case-Insensitive Identifiers

(1.6) — R8. Hashtag Identifiers

2 This document also makes no claim about additional conformance points in any versions of UAX #31 in versions after 15.1.