ISO/IEC JTC 1/SC 22 liaison report to UTC #175

To: UTC, ICU-TC
From: Robin Leroy
Date: 2023-06-13

At its 173rd meeting in November 2023, by consensus 173-C31, the UTC recommended to the officers *that they establish a Category A liaison between The Unicode Consortium and ISO/IEC JTC 1/SC 22 (Programming languages, their environments and system software interfaces), with Robin Leroy as the liaison officer*.

The liaison application was sent in December, balloted in January, and registered in March.

Most of the active working groups published a revision of their language standards recently. In light of the UTC's recent efforts, and being aware of of long-standing issues with normative references to the Unicode Standard vs. ISO/IEC 10646 in these standards, the liaison officer looked at the drafts with an eye towards identifier definitions and normative references to these two standards.

WG 4 COBOL

ISO/IEC 1989:2023 (COBOL 2023) has been published.

This standard has no normative reference to the Unicode Standard; but they need it, being a case-insensitive language with Unicode identifiers. Instead they list ranges of characters and case pairs, noting the correspondence to Unicode properties. A similar practice had been followed by C11 through C20 and C++11 through C++20; but see below under WG 14 and WG 21.

The standard has an unversioned normative reference to ISO/IEC 10646.

The language uses an identifier definition that can be expressed as

Start (*Interior** *End*)?

where *Start*, *Interior* and End are, as described in the notes, in the versioned UnicodeSet notation used by the invariants tests:

1. *Start* = [\p{U13.0.0:XID_Start} 0-9]
2. *End* = [\p{U13.0.0:XID_Continue} - [_]]
3. *Interior* = [*End* [\-_ \p{Name=KATAKANA MIDDLE DOT}]]

The language uses equivalent case-insensitive identifiers, but the pairs it lists to define that appear to be based on the simple case *mapping* rather than *folding*, except that Cherokee and the circled capitals are both unmapped. A note states that the mapping of U+03C2 ς to σ (which would be the correct case *folding*) was removed in this version of the standard because it was believed to be erroneous, since ς is lowercase. Normalization is not mentioned in the standard.

> **Note:** The addition of KATAKANA MIDDLE DOT to *Continue* appears related to the Unicode 4.1 stability violation noted by UTC 174-C3; a note in the standard discusses it in the context of backward compatibility with ISO/IEC TR 10176:2003.

The liaison officer has not interacted with this WG so far.

WG 5 Fortran

ISO/IEC 1539-1:2018 Cor 2 was published 2023-03-09.
ISO/IEC DIS 1539-1 (Ed 5) was approved 2023-01-31.

The standard has no no normative reference to Unicode. It has a normative unversioned reference to ISO/IEC 10646.

The language does not support non-ASCII identifiers.

The liaison officer has not interacted with this WG so far.

WG 9 Ada

ISO/IEC 8652:2023 (Ada 2022) has been published.

Ada (since Ada 2005) technically has no normative references to the Unicode Standard, but it needs it (being case-insensitive, like COBOL).

Instead, it uses the case mappings and foldings defined by "the documents referenced by the note in section 1 of ISO/IEC 10646:2003" (in more recent editions of the Ada standard, this is "documents referenced in Clause 2" of more recent editions of ISO/IEC 10646, this being implicitly a versioned reference to the Unicode Standard). The standard has a normative versioned reference to ISO/IEC 10646. In light of more recent work by WG 14 and WG 21, it may be possible to dispense with this trick in the future.

The language uses an identifier definition based on the Unicode 3.0 one, which was current when non-Latin-1 identifier support was added in Ada 2005; this (along with similar situations in the C# and VB programming languages) is one of motivations for the discussion of migration from Unicode 3.0 added to the latest draft of DUTS #55.

Normalization is swept under the carpet by making non-NFC programs implementation-defined. Ada 2022 forbids characters that are NFKC_Quick_Check=No in identifiers in order to reduce the implementation-definedness; but there is room for improvement when it comes to the handling of normalization.

As technical work in this WG mainly takes place in the WG and in its Ada Rapporteur Group prior to the Committee Draft stage, the liaison officer has been added as an observer of the Ada Rapporteur Group in order to advise on Unicode-related issues going forward.

Preliminary discussions in User-Community-Input issues are ongoing to provide a unified interface for strings regardless of their underlying representation (Ada has fixed-size strings, bounded strings, and unbounded strings, for Latin-1 Character, BMP Wide_Character, and UTF-32 Wide_Wide_Character), and to provide a strongly-typed solution UTF-8 strings (transcoding between standard encoding forms is supported since Ada 2012, but the type UTF8_String is a renaming of the Latin-1 String).

WG 14 C

A committee draft for C23, CD 9899.2, went out for comments until 2023-05-31.

The liaison officer was contacted by the chair of SC 22 and convenor of WG 14 requesting permission of the Consortium to have normative references to annexes of the Unicode Standard, and whether the Consortium was willing to keep the WG up to date if these are updated, pursuant to the ISO/IEC Directives, Part 2, subclause 10.2, item b. This was answered in the affirmative, and noting the existence of the Proposed Updates. The standard has an unversioned normative reference to ISO/IEC 10646.

The identifier definition is aligned with that of C++; see under WG 21.

WG 17 Prolog

The liaison officer has not interacted with this WG so far.

WG 21 C++

The committee draft for C++23, CD 14882, went out for comments by 2022-11-02.

The Draft International Standard for the previous version, C++20, had a versioned normative reference to a UAX. While it was removed from the standard, this was a response to an *editorial* comment (**-002) from the ISO/CS, as it was not actually used normatively; see the editor's report N4867. There were no objections in principle to such a reference.

The C++23 Committee Draft has an unversioned normative reference to the Unicode Standard—this time actually used—; likewise there were no comments objecting to it. There was however a comment FR-010-133 objecting to the multiplicity of Unicode versions referenced, both directly and indirectly via the versioned and unversioned references to ISO/IEC 10646.

The accepted resolution to that comment was to remove the reference to ISO/IEC 10646, citing only the Unicode Standard; the project editor interprets the chaos of multiple version as a case of *the absence of appropriate ISO or IEC documents*, per the ISO/IEC Directives, Part 2, subclause 10.2.

While the UTC is already aware of the issues surrounding the C & C++ identifier definitions from earlier documents (*inter alia*, L2/22-102, L2/22-230) a refresher may be welcome.

The identifier definition used in C++11, C++14, C++17, and C++20 was based on code point ranges listed in those standards; the répertoire defined by these ranges appears to correspond to that of UAX31-R2 *immutable identifiers*, with the additional exclusion of those characters that were GC=Zs at the time of C++11, see UnicodeSet comparison. Notably, the Swift programming language also went with this identifier definition.

The C++11 through C++20 syntax can be expressed as *Start Continue*** with

1. *Start* = *Continue* - [\p{Block = Combining_Diacritical_Marks}
   \p{Block = Combining_Diacritical_Marks_Supplement}
   \p{Block = Combining_Diacritical_Marks_For_Symbols}
   \p{Block = Combining_Half_Marks}]

2. *Continue* = [\P{Pattern_Syntax}
   & \P{Pattern_White_Space}
   & \P{General_Category = Surrogate}
   & \P{Noncharacter_Code_Point}
   & \P{General_Category = Private_Use}
   & \P{General_Category = Control}
   & \P{U6.0.0:General_Category = Space_Separator}]

In C++23, because of the structural issues associated with immutable identifiers (in particular when it comes to normalization), the identifier definition was changed to one closely based on UAX31-R1 *default identifiers* (this was even done retroactively as a defect report going back to C++11). This is a backward incompatible change, and as the previous definition had been used for more than a decade, implementers quickly encountered incompatibilities in real code. Some of the material in DUTS #55 and PU UAX #31 is informed by these difficulties; while C++23 is closed for technical changes, we expect that changes will be made in the C++26 timeline based on these documents.

WG 21/SG 16 Unicode

WG 21 has a study group (SG 16, meeting twice a month) for Unicode-related matters.

The liaison officer took part in SG 16 meetings and discussions on the sg16 mailing list. Minutes may be found at https://github.com/sg16-unicode/sg16-meetings/.

2023-02-09: As part of ongoing work to address the aforementioned French NB comment on CD 14882, Corentin Jabot asked for a term to specifically refer to the three UTF encoding forms. Following discussion with Jens Maurer on the mailing list, it was found that such a term did not exist; the Standard frequently abuses *Unicode encoding form* for that purpose.

The liaison officer relayed this issue to the PAG, which is considering the issue. The liaison officer has kept SG 16 informed of the progress of the proposal through the PAG.

2023-02-22, 2023-03-08: P2773R0: Considerations for Unicode algorithms. The study group is planning to add support for Unicode algorithms to the standard library. The priorities are notably in a reasonable order, with normalization first (after transcoding), before case transformations etc. Many languages fail to get that right.

The liaison officer clarified the nature of tailoring, being distinct from language dependence, and pointed out that things that can depend on language don't always do so in practice (CLDR grapheme clusters are not language-dependent, Turkic case folding needs to be used with great care rather than whenever something is Turkish).

2023-03-22, 2023-04-12: P2728R0. The study group considered a proposal to add transcoding between the standard encoding forms to its standard library.

Of relevance to ICU-TC: Compatibility with wchar_t-as-UTF-16 (rather than the modern char16_t) and char-as-UTF-8 (rather than char8_t) were discussed. The group is prioritizing APIs based on range adapters, so the question ends up being whether one needs to insert an explicit "char means UTF-8" or "wchar_t means UTF-16" adapter. The situation would be different for an eager algorithm taking a pair of pointers to charN_t, as one would then require a copy not to fall afoul of the strict aliasing rule.

WG 23 Programming Language Vulnerabilities

ISO/IEC DIS 24772-1 *Programming languages — Avoiding vulnerabilities in programming languages — Part 1: Language independent catalogue of vulnerabilities* was ballotted from January through March.

The liaison officer, having been involved in work spurred by the so-called "Trojan Source exploit", looked at this document in search of standard terminology that could be used in the discussions in Unicode Technical Report #36 and Standards #39 and #55.

There are only three mentions of Unicode in the DIS:

- In a discussion of cross-site scripting:

  > Attackers frequently use a variety of methods to encode the malicious portion of the tag, such as using Unicode [...]

- For *Resource names* and *Unrestricted file upload*, under *Avoiding the vulnerability or mitigating its effects*:

  > Avoid all Unicode characters [...] in filenames and the extensions.

One supposes that a lack of text helps with most programming language vulnerabilities indeed.

The liaison officer has not interacted with this WG so far.

WG 24 Linux

A new revision of POSIX, CD 9945, is out for comments by 2023-05-30.

The liaison officer did not look very much into this document. It has no normative references to Unicode. It has a normative reference to ISO/IEC 10646-1:2000, which is not very fresh; however the draft has a review note stating that *This list will be updated in a later draft*.

The liaison officer has not interacted with this WG so far.

Next steps

SC 22 will hold a virtual plenary in September; liaison reports are due 2023-08-04.

The liaison officer intends to work with the ARG to get rid of the "documents referenced" trick. Once Unicode 15.1 is published, work can begin on addressing the issues encountered by the incompatible changes made by C++23 and C23, and on migrating Ada to an identifier definition more modern than that given by Unicode 3.0, based on the updated guidance in UAX #31 and UTS #55.