# Unicode Conformance Model House Cleaning

Asmus Freytag, based on input by Mark Davis and Ken Whistler

July 15, 2023

This document captures my take on an ongoing discussion with the intent of making it visible to a larger audience from which we will hopefully gain some collaborators and reviewers, and with the aim to ask UTC to assign specific actions to resolve the various issues and prepare suitable draft documents.

***The Unicode Conformance Model***, as current specified in UTR#33 (revision 5) is dated from about 15 years ago. In the meantime, we have learned a thing or two, and also implemented new ways of specifying conformance in the context of algorithm (see the recent work of UAX#31 and UTS#39, both in proposed updates). At the same time, the Conformance model is generic enough, so as to not need a complete rewrite. With input from the PAG the report's authors have prepared a draft for a proposed update to bring it in line with current practice.

One key insight of this work is to highlight that conformance claims have to be testable (in some way) to be meaningful. This is not the same as saying that Unicode must provide the testing, or the infrastructure (although we do supply test files in certain cases, or model implementations in others). It also doesn't mean that the tests must be mechanical or amenable to automation.

A separate effort is underway to create a ***Conformance FAQ*** around conformance related questions to address some of the issues and questions we've encountered recently. This work, like all FAQ related work is proceeding in the "FAQ of the month club" that is headed by Ben Yang as part of the Editorial Committee, and we are getting some good discussion there.

Finally, having taken a fresh look at all of these, the ***conformance clauses in TUS*** (and in some UTS and UAXs not recently reviewed) may need a careful going over. The following are suggestive examples that describe the start, and not the endpoint, of the discussion how to improve, where possible, the statement of conformance requirements. They intend to be suggestive of the kind of discussion we need to have to either confirm existing language or make improvements.

Examples:

> *C4 A process shall interpret a coded character sequence according to the character semantics established by this standard, if that process does interpret that coded character sequence.*

It is impossible to determine whether or not one conforms to this, because "character semantics established by this standard" is not well-defined.

This implicitly refers to D3, which refers to "identity, normative properties, and behavior". Those are specified in D2 and D1. Arguable the phrasing should be simply "normative behavior". Many of the

behaviors in D1 are only relevant if the implementation is performing a given function (like normalization).

By the same argument “Normative” only has meaning in reference to a specific operation. For example, “The character combination properties and the canonical ordering behavior cannot be overridden by higher-level protocols.” The real meaning of that is that if you perform normalization on Unicode text, the results must be as specified in XXX; the canonical ordering behavior and “character combination properties” are a means to that end.

However, a counter argument is that the Unicode Standard is not a collection of specifications for all possible operations on text. One part of the standardization effort is to ensure that sender and receiver agree on the identity of the text elements that they interchange. This is true, even if there’s no specific operation defined by Unicode that tests a particular aspect of this common understanding. In many ways, this reflects the fact when we assert U+0041 is for LATIN CAPITAL LETTER A, that there is something that distinguishes it from U+0042 LATIN CAPITAL LETTER B. Unicode has no “consonant” or “vowel” property for the Latin script, by which one could distinguish “A” from “B” and in both cases, lowercasing applies the same offset.

Nevertheless, a spell checker that codes any instance of “B” using U+0041 would not be Unicode conformant, even if it correctly case folds this letter by adding the correct offset.

This argument then rests on character identity and behavior that is independent of operational specifications found in the Unicode Standard or its annexes (or any UTSs), but instead rests in the traditional understanding and shared convention among users of the script as to the identity of the abstract characters “Latin Capital Letter A” and “B”.

Conformance to the Unicode Standard should imply the promise to “interpret” characters based on their conventional identity in all cases where there isn’t a specific operational specification that narrows down further what it means to treat a character according to its intent in a specific algorithm or text operation.

The latter would be strictly testable, even in an automated way, while the former may require considerable judgment to evaluate — not least because there’s some latitude grounded in the history of each script, on the permissible “interpretations” for the purpose of general operations on text. An example of such latitude is the acceptable range of glyphs for a given character, which may be very wide in any situation where there is enough context to allow identification of the character despite a very fanciful representation.

> *C16 Normative references to the Unicode Standard itself, to property aliases, to property value aliases, or to Unicode algorithms shall follow the formats specified in Section 3.1, Versions of the Unicode Standard.*

Some discussion is happening around the need for unversioned references. On the face of it, an unversioned conformance claim is not testable. If it is understood that some product or specification is compatible with the "latest" version of the Unicode Standard, that can only be guaranteed for anything that is released in strict synchronicity to the releases of the Unicode Standard. Even then, just as it is for the Unicode Standard, it is of interest to know that version a given copy of that product or specification has been updated to.

One argument centers on complex implementations where documentation may be distributed and where making sure everything is up to date may be burdensome. But that argument is about the physical representation of the conformance claim itself, not the logical content of the claim. For example, a runtime library could have an API that returns the active supported version. In that case, the documentation may not need to contain that information, but the full conformance claim remains de-facto versioned.

Another case cited is that of true invariants. For example, there's no difference in the specification of UTF-8 between Unicode 15.1 and Unicode 16.0. (However, the specification changed in minor ways in earlier versions). Technically, the correct conformance claim would be something like "UTF-8 for Unicode x.y and later."

Where stability policies guarantee invariance, it should be possible to say "UTF-8 for Unicode (unversioned)" because tracking which was the last early version that still contained tweaks becomes less and less interesting.

There is another kind of invariant. The status "isNormalized" is guaranteed to not change once it was true for a given string (and as long as the first evaluation was performed for a conformant system that did not detect an unassigned character in the string). This is a bit more tricky because of the conditional, but once you have a database of strings, if each string was normalized with a system updated so the whole string was assigned, then it's not necessary to maintain the information for each string which version it had been normalized for.

> *C17 Higher-level protocols shall not make normative references to provisional properties.*
>
> • *Higher-level protocols may make normative references to informative properties.*

There is in general a problem telling any other specification what they can or cannot reference in references that are normative *in the context of the other specification.* At best, the Unicode Standard can outline which elements are not *intended* for external references (and where possible, give the reasons, such as the lack of cross-version stability for provisional properties).

Both normative and informative properties can also be changed (subject to stability policies) but the UTC guarantees that an effort will be made to consider the effect of backwards compatibility up to and including not deleting or reusing the name of an informative property.

Further recommendation may be appropriate, e.g: *Unversioned normative references to provisional properties shall state that those properties may be changed or withdrawn in any future Unicode version*.

*C18 If a process purports to implement a Unicode algorithm, it shall conform to the specification of that algorithm in the standard, including any tailoring by a higher-level protocol as permitted by the specification.*

Where possible, we should use the concept of profile, so that C18 becomes “implements a profile of a Unicode algorithm”. e.g. in C19.

However, the concept of tailoring is not useless, as long as it is invoked more precisely as, for example, *“...including the specification of a specific permitted tailoring*”

*D15 Reserved code point: Any code point of the Unicode Standard that is reserved for future assignment. Also known as an unassigned code point.*

A big omission was to not specify even if redundantly, who is permitted to “assign” characters. So this should become “future assignment by the Unicode Consortium” and we might add a bullet about PUA being for other standards or implementers.

*D16 Higher-level protocol: Any agreement on the interpretation of Unicode characters that extends beyond the scope of this standard.*

- *Such an agreement need not be formally announced in data; it may be implicit in the context.*
- *The specification of some Unicode algorithms may limit the scope of what a conformant higher-level protocol may do.*

This obviously intersects with conformant profiles, but the general notion of a higher-level protocol is anything that builds on the Unicode Standard. It can therefore not be limited to conformant profiles in the general case, but the relation should be carefully teased apart. Particularly in contexts were we assert that HLPs are restricted if they want to remain conformant.

**Table 3-1. Named Unicode Algorithms**

While this includes some UTS conformance, it is not complete.

*D32 Catalog property: A property that is an enumerated property, typically unrelated to an algorithm, that may be extended in each successive version of the Unicode Standard.*

This looks like an alias for *Open enumeration* (an enumeration that is not Closed), but there may be a useful distinction between something that pops a new value every time a novel class of items is encountered vs. something that pops a new value for each new item (e.g. Block names). “Catalog property” seems more like the latter, and “Open enumeration” seems more like the former. Define both?

*D34 Overridable property: A normative property whose values may be overridden by conformant higher-level protocols.*

The argument is made that a Normative property is only for a given other conformance clause. The Bidi algorithm makes normative reference to X bidi properties. However, there's another conception of properties which may or may not be used in the specification of a fully worked out algorithm.

It should be possible to normatively assert (in the sense of "it would fundamentally alter the standard if it was changed") some quality (property) of a character related to the identity that is being encoded, even without it being consumed by a defined algorithm.

"Overridable", however, truly only makes sense when that is testable. Which it usually is in the context of a property normatively referenced by an algorithm.

However, when we state "it is this, and not that" character, and you don't want this intent you must chose the other one, that's a statement that in the global context of the standard should not be "informative".

These examples are not exhaustive but give an indication of the kind of work and its scope, and of the tension between:

- On the one hand, the original conception of Unicode conformance that largely centered on the challenge of being the first character encoding that made explicit the underlying conventional identification of the abstract characters being encoded as opposed to leaving it implicit (and hinted at by lists of shapes).
- On the other hand, the understanding of the collection of Unicode and its technical Standards as the seminal collection of text processing specifications, whether they represent required performance that is essential for interoperability or a minimal, default level of performance that can serve as a foundation for generalization to either higher quality treatments or localized customizations.

**Proposed UTC Action**

Discuss the issue, its scope and direction in general terms and, as appropriate, delegate to appropriate subgroups the tasks of preparing actionable draft documents for future review and publication.