Technical Reports

L2/08-248

Working Draft for Proposed Update

Unicode Technical Report #33

UNICODE CONFORMANCE MODEL

Authors	Ken Whistler and Asmus Freytag (asmus@unicode.org)
Date	2008-07-03
This Version	http://www.unicode.org/reports/tr33/tr33-4.html
Previous Version	http://www.unicode.org/reports/tr33/tr33-3.html
Latest Version	http://www.unicode.org/reports/tr33/
Revision	4

Summary

This report defines conformance terminology, specifies different areas and levels of conformance, and describes what it means to make a claim of conformance or "support" of the standard. This conformance model presented here is not a framework for conformance verification testing.

Status

This document is a proposed update of a previously approved Unicode Technical Report. This document may be updated, replaced, or superseded by other documents at any time. Publication does not imply endorsement by the Unicode Consortium. This is not a stable document; it is inappropriate to cite this document as other than a work in progress.

> ***A Unicode Technical Report (UTR)*** *contains informative material. Conformance to the Unicode Standard does not imply conformance to any UTR. Other specifications, however, are free to make normative references to a UTR.*

Please submit corrigenda and other comments with the online reporting form [Feedback]. Related information that is useful in understanding this document is found in the References. For the latest version of the Unicode Standard see [Unicode]. For a list of current Unicode Technical Reports see [Reports]. For more information about versions of the Unicode Standard, see [Versions].

Contents

1 Overview

The Unicode Standard [Unicode] is a very large and complex standard. Because of this complexity, and because of the nature and role of the standard, it is often rather difficult to determine, in any particular case, just exactly what conformance to the Unicode Standard means.

The Unicode Standard forms the foundation ~~which supports~~ supporting a large variety of operations on textual data, from data interchange protocols to complex tasks like sorting, rendering or content analysis. All of these processes expose implementations to the complexities of human languages and writing systems.

Earlier character sets were ~~either~~ small, or had a clearly limited field of application, (such as by geographical area), or both. By contrast, the Unicode

Standard aims to be universal. A universal character encoding standard cannot rely on implicit agreements about the nature and behavior of the characters it encodes; it must provide explicit constraints on their identity and intended use. This affects not only how characters are interchanged, but the specification of many common text processes as well. At the same time, the standard must allow implementations the necessary flexibility to address the expectations of its users, while providing enough constraints to guarantee predictable interchange of data and consistency between implementations.

This Conformance Model explains the issue of conformance relating to the Unicode Standard so that users better understand the contexts in which products are making claims for support of the Unicode Standard, and implementers better understand how to meet the formal conformance requirements while satisfying the expectations of their users. It does not alter, augment or override the actual Unicode conformance requirements found in the text of the Unicode Standard. Rather it attempts to provide a conceptual framework to make it easier for users and implementers to identify and understand the specific conformance requirements contained in [Unicode].

This model defines conformance terminology, specifies different areas and levels of conformance, and describes what it means to make a claim of conformance or "support" of the standard. This model is not a framework for conformance verification testing, although it could be used to develop such a framework, should that prove desirable. At this time no such framework has been developed by the Unicode Consortium, nor have any conformance verification tests been required or sanctioned.

Many of the concepts presented here are equally applicable to other standards developed by the Unicode Consortium, such as the Unicode Collation Algorithm [UCA], and the specifications for Unicode support in regular expressions [RegEx].

2 Terminology

This section ~~gives a basic introduction to the~~ introduces basic terminology that will be discussed in more detail in sections below.

A great number and variety of standards exist. Standards may be regulatory, mandatory or voluntary. For the purposes of this document, a standard is defined to be a formally developed specification. The organization developing that standard is then called a Standards Develop~~ment~~ing Organization or SDO.

2.1 Conformance

In the context of formal standards, *conformance requirements* refer to a set of rules or criteria that allow a relevant entity such as an element of information interchange, a device, an application, or a piece of hardware, ~~can~~ to be evaluated as either meeting or not meeting the specification in the standard.

In general, a formal standard will have a conformance clause or clauses, which will be stated in terms of conditionals, such as "X conforms to Y specification of this standard if Z", or modals, often in uppercase, such as "An X that conforms to Y specification of this standard SHALL Z". The modal verbs that standards language

commonly associates with such statements are often carefully defined to avoid any ambiguities in interpretation. In common practice, they involve specialized usage of "SHALL" and "MUST" for requirements, but also "MAY" for permitted deviations and "SHOULD" for non-binding recommendations. (The Unicode Standard ~~itself~~ does not use the convention of uppercasing these terms).

If a standard is complex, the conformance clause or clauses ~~themselves~~ may also be complex. Occasionally, a conformance clause may simply be stated along the lines of "X conforms to this standard if it follows the specification in section W" where section W may consist of hundreds of pages and constitute most of the rest of the standard.

An implementation is said to be conformant if it meets all applicable conformance requirements.

2.2 Compliance

The term *compliance* is often used synonymously with the term conformance and will be used that way in this model.

2.3 Normative and Informative

Formal standards often distinguish between *normative* and *informative* content. This distinction may be highly conventionalized, or even subject to rules specified in other standards, such as for ISO standards, or the distinction may be less formally maintained.

Normative content of a standard is content which is required for all of the conformance requirements to be meaningful. Typically a standard will have normative definitions for terms used in the rest of the specification, normative references to other standards or sources whose content is referred to indirectly, and normative clauses, specifications, or sections, which ~~actually~~ define the content of the standard to which the conformance clauses apply.

Informative content of a standard is ~~all~~ material which has been added for clarification, but which, in the judgment of the standard's maintainers, could ~~in principle~~ be omitted without materially affecting the specification to which the conformance clauses refer. If a standard is changed over time, the status of some particular content could change from informative to normative, or vice versa, depending on whether it became required for conformance or was no longer required for conformance.

2.4 Conformance Verification and Certification

In the context of the Unicode Conformance Model, *conformance verification* means an external (third party) determination that an entity meets one or more requirements of the conformance clauses of the standard under some specified set of circumstances. In other words, while conformance clauses are merely a logical statement of requirements, conformance verification implies the existence of conformance verification tests that have been applied to entities in order to make such determinations.

- A conformance claim can simply be stated. It is an assertion that entity X meets a requirement of the standard.
- A verification of a conformance claim, on the other hand, is the result of the specific application of a test designed to determine the validity of a conformance claim. Such tests are called conformance verification tests.
- A certification of a conformance claim is the verification of a conformance claim by a certification authority.

The Unicode Consortium does not endorse a particular methodology for conformance verification.

2.5 Conformance Testing

A standard may include tests or "benchmarks" as part of the text of the standard, or as external documents associated with the standard. While there is some overlap in general usage of the terms "conformance test" and "conformance verification tests", a systematic distinction is drawn between the two in the Unicode Conformance Model.

2.5.1 Conformance Tests

A *conformance test* for the Unicode Standard is a list of data certified by the Unicode Technical Committee [UTC] to be "correct" with regard to some particular requirement for conformance to the standard. In some instances, as for example, the implementation of the Unicode Bidirectional Algorithm, producing a definitive list of correct results is difficult or impossible, and in such cases, a conformance test may consist of an implemented algorithm certified by the UTC to produce correct results for any pertinent input data. Conformance tests for the Unicode Standard are essentially benchmarks that someone can use to determine if their algorithm, or API, ~~etc.~~, claiming to conform to some requirement of the standard, does in fact match the data that the UTC asserts define such conformance.

2.5.2 Conformance Verification Tests

A *conformance verification test* for the Unicode Standard is a test, usually designed and implemented by a third party not associated with the Unicode Consortium or the UTC, intended to test a product which claims conformance to one or more aspects of the Unicode Standard, for actual conformance to the standard. Thus a conformance verification test is a *test of a product*. Such a test, may, of course, make use of one or more of the Unicode conformance tests to determine the results of its conformance verification.

2.6 Support

In the context of the Unicode Conformance Model, the term *support* refers to a more generalized claim of intent to conform to one or another requirement of the standard. A claim of Unicode support may be difficult to verify, because its statement can be vague or lack detail. However, it indicates that the developer or user of an entity intends conformance. More specifically, support often refers to a claim of particular repertoire coverage. For example, an application may claim support for Unicode Greek. That should be interpreted as meaning that Unicode

Greek characters will be handled in conformance with the Unicode Standard, and that all other relevant aspects of processing of those characters with which that particular application is concerned, will be done in such a way as not to violate the conformance clauses of the Unicode Standard.

2.7 Versioning

The Unicode Standard is regularly versioned, as new characters are added. A formal system of versioning is in place, involving three levels of versions:

1. major versions
2. minor versions
3. update versions

All three levels have carefully controlled rules for the type of documentation required, handling of the associated data files, and allowable types of change between versions. For more information about Unicode versioning see [Versions]. Other standards developed by the Unicode Consortium may use a single level versioning scheme.

Conformance claims ~~clearly~~ must be specific to versions of the Unicode Standard, but the level of specificity needed for a claim may vary according to the nature of the particular conformance claim ~~being made~~. Some standards developed by the Unicode Consortium require separate conformance to a specific version (or later), of the Unicode Standard. This version is sometimes called the *base version*. In such cases, the version of the standard and the version of the Unicode Standard to which the conformance claim refers must be compatible.

2.8 Corrigenda and Errata

If a technical deficiency in the specifications of the Unicode Standard is identified, it may be corrected by a change in the next version, or, if sufficiently important, by a formal corrigendum. A corrigendum often applies to several earlier versions, but, unlike the practice for other SDOs, does not retroactively change them. Implementations can claim conformance to any of these versions with the given corrigendum applied. For more on corrigenda see [Versions].

Errata are used to describe other known defects in the text. Unlike corrigenda they cannot be referenced in a conformance claim. For more information on errata see [Errata].

2.9 Stability and Invariance

Each version of the Unicode Standard, once published, is absolutely stable and will never change. Implementations or specifications that refer to a specific version of the Unicode Standard can rely upon this stability. If future versions of these implementations or specifications upgrade to a future version of the Unicode Standard, then some changes may be necessary.

Some formal standards are developed once and then are essentially frozen and stable forever. For such standards, stability of content and the corresponding stability of conformance claims is not an issue.

For a standard aimed at the universal encoding of characters, such stability is not possible. The Unicode Standard ~~is necessarily evolving and expanding~~ necessarily evolves and expands over time, to extend its coverage to include all the writing systems of the world. Because of the interaction between the way characters are processed and the definition (or identity) of the characters that have been encoded, many aspects of character processing must be specified in conjunction with encoding the characters, ~~in order~~ to guarantee predictable interpretation and reliable interchange of text. As experience in implementing the Unicode Standard accumulates, further aspects of character processing are added to the formal content of the Unicode Standard as needed.

This fundamentally dynamic quality of the Unicode Standard complicates issues of conformance: the content to which conformance requirements pertain is continually expanding. This expansion is both an expansion in breadth by adding more characters and scripts, and an expansion in depth by adding more aspects of character processing.

Invariance refers to those aspects of the content of the Unicode Standard that have been formally defined as unchangeable, even as the standard continues its development. The guarantee of the stability of the formal Unicode character names is a fairly trivial example. While in principle such names *could* be changed, and were changed once between Version 1.0 and Version 1.1, the [UTC] has determined that such changes are too disruptive and have too little benefit to be tolerated. Accordingly, the stability of character names has been promoted to the status of an invariant in the standard.

A further discussion of invariance and invariants can be found in [PropertyModel]. Invariants guard against change for the sake of change, or technological drift, but they also prevent the correction of clerical errors, which is not a negligible issue in a standard as large and complex as the Unicode Standard. For a current list of invariants and a discussion of the tradeoffs, see the Unicode ~~Stability Policy for Character Encoding~~ Stability Policy~~and Character Properties~~ [Stability].

Conformance claims need to be distinguished in terms of their relationship to invariants and non-invariants in the standard because of their different risk levels for stability.

3 Structure of Unicode Conformance

This section will serve as a guide to the particular way that the Unicode Standard expresses conformance requirements, both in terms of where they are located and how they are expressed. It also explores the peculiar aspects of conformance related to the synchronized status of the Unicode Standard and the independent but closely aligned International Standard ISO/IEC 10646, which has its own conformance clauses expressed using ISO conventions.

3.1 Definitions

Chapter 3, "Conformance" of [Unicode] contains formal definitions of terms referenced in the conformance clauses. While modifications of these definitions between versions of the Unicode Standard have been, and will continue to be necessary, every effort is made to keep the numbering of the definitions stable.

This makes it easier to maintain external specifications that cite a particular definition. (In Version 5.0 of the Unicode Standard, all definitions were renumbered, with a cross reference table linking the new numbers to the earlier numbering.)

3.2 Conformance Clauses

The conformance clauses in Section 3.2, "Conformance Requirements" of [Unicode] define the requirements for a conformant implementation. They are expressed in terms of the definitions, but also refer to additional specifications contained in Unicode Standard Annexes. While modifications of these clauses between versions of the Unicode Standard have been, and will continue to be necessary, every effort is made to keep the numbering of the clauses stable. This makes it easier to maintain external specifications that cite a particular clause. (In Version 5.0 of the Unicode Standard, all clauses were renumbered, with a cross reference table linking the new numbers to the earlier numbering.)

3.3 Unicode Standard Annexes

A Unicode Standard Annex (UAX) contains part of the standard, published online as a standalone document. The relation between conformance to the Unicode Standard and conformance to each of the Unicode Standard Annexes is spelled out in detail in Section 3.2, "Conformance Requirements" of [Unicode]. Some of the conformance clauses refer explicitly to specifications contained in UAXs, such as ~~the~~ UAX #9: *Unicode Bidirectional Algorithm* [Bidi] or UAX #15: *Unicode Normalization Forms* [Normalization]. Normative material in other UAXs is defined by any of the mechanisms described below.

Other standards developed by the Unicode Consortium have their own conformance models.

3.4 Identification of Normative Content

In the Unicode Standard, Chapter 3, *Conformance* contains a set of numbered conformance clauses, as well a set of numbered definitions that are referenced by these clauses. For algorithms, a numbered set of rules is defined as well. Unicode Standard Annexes containing normative material make use the same convention. The conformance clauses use the letter C with a number; definitions use the letter D with a number. Rules for algorithms are numbered using a unique prefix specific to the algorithm.

Bullets and other textual devices are used to separate explanatory and informative text from the normative text of the conformance clauses, definitions and rules. Such explanatory and informative text does not form part of the actual conformance clauses, definitions or rules. Noteworthy text is set off by the use of "Note:", but unlike the usage in other standards, where this device is required to identify informative material, there is no set scheme for labeling informative statements in the Unicode Standard. In a few cases, an entire section of text is identified as containing normative or informative material respectively, but this is not a universal scheme.

3.5 Properties

The Unicode Character database [UCD] includes an overview file (UCD.html) with a table specifying which properties and property files are normative. For more information on the concept of normative properties, see UTR #23: *The Unicode Character Property Model* [PropertyModel].

The contents of normative files and normative fields within data files are formally decided by the Unicode Technical Committee. In particular, any fields listing properties on which conformance requirements depend are normative.

In principle, the contents of informative files and informative fields within data files are similar to all other informative material in the Unicode Standard, which may be changed by its editors without formal review by the Unicode Technical Committee. However, for some properties considered informative the practice is nevertheless to get binding committee decision on the values. Conformance requirements do not depend on informative material.

3.6 Algorithms

Unicode algorithms are specified as a series of logical steps. Conformance to a Unicode algorithm does not require repeating the steps as described, but rather requires achieving the same outputs for the same inputs. This provides the necessary flexibility for implementations to pursue optimizations. Whether or not conformance to a given algorithm is required by Unicode conformance, implementations claiming to implement one of these algorithms must do so in conformance with its specification.

In many cases, the input to the algorithm is defined in terms of an ordered list of character property values: in other words, the results of the algorithm are identical for different input strings, as long as each input string maps to the same ordered list of character property values. Examples of such algorithms include the Unicode Bidirectional Algorithm [Bidi] and the Unicode Line Breaking Algorithm defined in [LineBreak]. In such cases, conformance claims can be tested separately for the mapping of characters to property values and for the operation on ordered lists of property values.

Unicode algorithms are specified to apply to all code points, and are usually expected to gracefully handle the case of receiving input from an up-level version of the standard. The Unicode Standard provides for the concept of a default value for properties, to improve the forward compatibility of Unicode algorithms. A default value applies to unassigned characters, or whenever no specific property value has been assigned, as described in [PropertyModel] In some cases, the use of specific default values is binding, in other cases it is merely recommended best practice.

3.7 Tailoring

Some algorithms provide explicit methods for tailoring, or customizing a general algorithm to the needs of a specific language, locality or application. Other algorithms simply describe the best default practice, and customization is assumed for any practical application. An example of this is the tailorable part of the line breaking algorithm in [LineBreak]. Whether or not conformance to a given algorithm is required by Unicode conformance, implementations claiming to

implement one of these algorithms must specify the tailoring or customization used.

3.8 Relation to 10646 Conformance

The Unicode Standard and ISO/IEC 10646 share the same repertoire of coded characters, including the character code position, character name and identity. However, the two standards differ in the precise terms of their conformance specifications. Any conformant Unicode implementation will conform to ISO/IEC 10646, but because the Unicode Standard imposes additional constraints on character semantics and transmittability, not all implementations that are compliant with ISO/IEC 10646 will be compliant with the Unicode Standard. For a detailed description see Appendix C, "Relationship to ISO/IEC 10646" of [Unicode].

4 Areas of Conformance

There are several broad areas of application where Unicode conformance makes specific types of requirements. Because not all applications and implementations cover all these areas, some aspects of Unicode conformance may not be applicable to them.

4.1 Representation

Representation covers all aspects of being able to express and transmit Unicode data. It is a requirement applicable to certain protocols (for example, XML), but might apply to the storage aspects of databases and other file formats as well. Conformant representation applies to correct use of encoding forms and encoding schemes, as well as the ability to represent all Unicode code points. In addition, issues related to [Normalization] are important.

4.2 Transcoding

Conformant transcoding between Unicode and all other so-called *legacy* character encodings, retains the identity of the transcoded characters. In addition, it may claim to retain a specific normalization form for the converted data. See [Normalization]. A separate Unicode Technical Standard, UTS #22: *Character Mapping Markup Language* [CharMapML], discusses many issues relevant to character transcoding and defines a format for expressing character mappings. Implementations may choose to conform to that format in order to be able to interchange mapping tables.

4.3 String Processing

String processing covers all operations on Unicode texts that can be carried out without considering layout and specifically without considering fonts. String processing encompasses a large variety of operations including, but not limited to text segmentation, text parsing, handling regular expressions, searching, and sorting, as well as creating formatted text representation of data types. For a number of these operations, model algorithms and other specifications exist to which an implementation may claim conformance, such as [UCA], [RegEx], [Boundaries], and [LineBreak].

4.4 Text Layout, including Display and Selection

Layout comprises all operations that go from backing store to displayed text. The same operations are run in reverse for text selection. These operations are dependent on font data, but are considered separately from fonts because the same implementation typically can work with a range of different fonts. Some operations, such as suppressing the display of certain ignorable code points, are typically handled by the layout system without involving fonts. Conformance issues for layout processes include reordering from logical to display ordering, as well as positional shape selection. For bidirectional reordering, conformance to [Bidi] is required. For positional shaping and script-specific layout, model algorithms exist, or are being developed for Arabic and Syriac, Devanagari, Tamil and other Indic Scripts, as well as Mongolian.

~~It is common to these scripts that~~ All these scripts occasionally require the use of *join controls* (ZWJ or ZWNJ) ~~is occasionally required~~ to override default rendering of certain character sequences to achieve a specific appearance ~~which is~~ required by orthographic rules. This is different from purely stylistic variations in rendered forms.

Conformance requires a relation between specific constructs in the writing system and corresponding character code sequences, so that these constructs can be interchanged reliably; therefore the conformance requirements include support for both the generic rendering and these types of basic and required orthographic variations.

In contrast, the requirements of high-end typography for any scripts typically exceed the basic rendering specifications put forth in the Unicode Standard. Conformance to the Unicode standard does not limit or prescribe high-end typographical features that an implementation can support.

4.5 Fonts

The Unicode Standard does not standardize the ~~actual~~ appearance of characters, but instead ~~intends that they should~~ expects them to be depicted within a customary range of design interpretations. Conformance to the Unicode Standard therefore primarily refers to those tables in the fonts that correlate character codes with the glyphs in the font, for example 'cmap' tables, and to claims of "coverage" of the Unicode repertoire by fonts.

4.6 Input

Conformance-related issues for character input consist of coverage of Unicode repertoire, conversion of input to Unicode character values for storage, and consistency with the text models required for particular scripts and text layout. The entities here are mostly IMEs and keyboards (drivers).

5 Levels of Support

Unicode Technical Standard #18: *Unicode Regular Expressions* [RegEx] is an example of a standard that has well-defined levels of conformance. Each implementation can claim conformance to a specific level, and each level has

specific conformance requirements. By contrast, conformance to the Unicode Standard is not organized into such discrete levels. However, there are some areas where the standard allows limited, or partial support of some requirements.

5.1 Repertoire Coverage

The Unicode Standard explicitly does not require that all implementations support all Unicode characters. Any implementation may support an arbitrary subset of Unicode characters, and in fact, may support different sets of characters for different operations.

However, for certain algorithms, any implementation that claims conformance is required to support the full range of Unicode code points covered by that algorithm. For example, an implementation of normalization, or a UTF-8 converter is required to support the entire range of Unicode code points.

> **Note:** An implementation may define an algorithm, such as identifier matching, that uses normalization as part of the algorithm but also restricts the allowable set of input characters. In that case, any implementation of that algorithm is free to use a limited implementation of normalization because the limit on the input makes it impossible to distinguish between a full and limited implementation of normalization.

An implementation may support a certain repertoire of characters, but may not support all sequences of characters from that repertoire. For example, an implementation may support combining marks, but may not be able to render combining sequences that contain more than one combining mark.

> **Note:** Some algorithms are specified in a way that they apply to all Unicode characters and all possible sequences. A complete and conformant implementation of such an algorithm would support all possible sequences.

Unlike some other standards, [Unicode] does not provide a formal method for specifying sub-repertoires, nor announcement techniques that would indicate the support for specific sub-repertoires.

5.2 Full Support

Conformance in a given area is not necessarily the same as full support for that area, as conformance requirements in many cases are minimal requirements. Exceptions are certain well-defined areas such as encoding forms or normalization that have few or no options and few or no levels.

5.3 Levels of Support Defined

While conformance means not violating any of the conformance clauses applicable to an area, it does not necessarily require that a specific feature be implemented or that it be implemented in its most fancy realization.

Given the aim of the Unicode Standard of universal coverage and of universal applicability for text encoding, ~~it is not expected that most~~few implementations will attempt to exhaustively support the entire repertoire of the standard. Support

of a limited repertoire is typical, and in many cases will deliver better results to users. For example, for most purposes, a font supporting a particular set of characters or scripts with high quality is going to be more useful than a font supporting most or all Unicode characters with minimal quality.

The Unicode Standard does not define particular levels of support.

> **Note:** Some Unicode algorithms require that the entire range of code points be supported by a conformant implementation. Examples include [Normalization] and encoding form conversion, for example conversion between UTF-8 and UTF-32. An exception would be a situation where another feature of an implementation restricts the available input—in such cases an implementation that acts only on a subset of code points would be conformant if evaluated *in the context of a conformant overall implementation.*

6 Interoperability

6.1 Inter-level Compatibility Issues

Conformant implementations will have to interact with both down-level and up-level implementations. This creates particular issues. The Unicode conformance requirements are structured to encourage implementations to passively support data containing characters assigned in future versions of the standard.

6.1.1 Down-level Compatibility

For several important properties, [Unicode] provides explicit support for implementations that need to be compatible with a down-level version of the relevant algorithm. This is usually done by guaranteeing the stability of property assignments [Stability]. In some cases, specific properties are introduced that isolate an algorithm from changes in a character's General Category. For an example, see the section on backwards compatibility in UAX #31: *Unicode Identifier and Pattern Syntax* [Identifier].

6.1.2 Up-level Compatibility

For most properties, there is a single default value that down-level systems can apply to unassigned characters when present in data sent from up-level systems. Where the fallback represented by such default value would give particularly poor results, the [UCD] or [Unicode] provide for several ranges with different default values. Such default values increase the chance that an actual property assigned to a new character will be the same as the default value for its code point in the down-level version of [Unicode]. An example are the [Bidi] properties, which default to strong right-to-left for areas of the code space earmarked for RTL scripts.

A common implementation technique is to use dynamic assignment of implementation specific default values, based on the actual property values of characters surrounding an unassigned code point. Such interpolation of character

properties can further increase the chance that any given code point is treated compatibly by a down-level system. At the same time, it can increase performance by creating longer contiguous runs of code points with the same property.

6.2 Repertoire Matching

It is generally not helpful to tag data created by an implementation with the version level of Unicode supported by that implementation. Because the repertoire of that version of Unicode is far larger than the actual set of characters used in the data, a large part of text data created and interchanged worldwide can be represented in *all* versions of Unicode. Therefore, the version level of the implementation bears little relation to the repertoire needed to cover the data.

Most implementations will not equally support the entire repertoire of Unicode characters for a given version. In fact, there is no conformance requirement to support any specific part of the repertoire. Therefore, even if the version level of a receiving implementation is higher than that of the creating implementation there is no guarantee that both support the repertoire covered by the data, or support it equally well.

[Unicode] defines no method for enumerating or identifying common sub-repertoires of the standard, but ISO/IEC 10646 does so. Implementations can use the [DerivedAge] for each character code to avoid sending character codes to a down-level system which lacks a definition for them. Because character coding is strictly additive, implementations receiving data can easily identify characters that are not defined in the version of the standard to which they conform and take appropriate action. In many cases, appropriate action consists of passing through such data, or treating them as characters possessing default properties. (See UTR #23: *The Unicode Character Property Model* [PropertyModel] for more details on default properties).

6.3 Matching Areas and Levels of Conformance between Implementations and Components

A mere matching of version numbers between an implementation and components it relies on will not be sufficient, because components may subset the repertoire they support or choose a different level of conformance, where available.

Appendix 1 - ECMA Report on the Meaning of Conformance to Standards

This appendix contains observations on the meaning of conformance to standards excerpted and abridged with permission from a September 1983 report by ECMA. These observations highlight the differences between traditional standards, for example for industrial products based on mechanical processes, on the one hand, and software related standards, for example for computer languages, on the other hand. From the text it becomes clear that the issues related to conformance discussed in this report are neither novel, nor unique to the Unicode Standard. In a few instances emphasis has been added.

[...]

THE MEANING OF CONFORMANCE TO STANDARDS

September 1983

[...]

It is helpful to distinguish very clearly between conformance and certification.

Conformance to a standard may be claimed for a computer product (hardware or software) if:

> i) the standard is intended to cover entities of the class to which the product belongs, and
>
> ii) the product satisfies all the mandatory clauses including chosen alternative clauses of the standard, if any, and
>
> iii) the product satisfies any chosen optional clauses, and
>
> iv) a clear declaration is provided of the standard alternatives and/or options to which the product is claimed to conform.

Certification

Certification means the issue of a document declaring that a particular (sample of a) product has been checked for conformance to a standard or standards. Certification is, as is known, mandatory in some countries for some standards e.g. related to human safety or connections to PTT services.

The process of certification implies some form of product examination and checking. Any body can issue a certificate. If the certificate is issued by the supplier of the product, the procedure is called "self-certification"; if by an independent body, neither supplier nor purchaser, it is called "third party certification".

The status and validity of a certificate clearly depend on the reputation and/or authority of its issuer and his evident competence to declare the conformance of the product, whether from a knowledge or inspection of its design, results of tests or otherwise. To be of value to a user, a certificate of product conformance obviously demands some assurance that all samples of a product actually supplied thereafter, conform as strictly as the test sample.

Ideally, conformance should result from the product design; it should be a matter of technically ascertainable facts that can be established unambiguously by reproducible tests or checks. **However, establishing conformance quite unambiguously by tests or checks, is, in some cases, beyond the state of the art**.

For a wide range of standards, suppliers themselves are obviously well placed to make statements with varying degrees of commitment as to the conformance of their products to particular standards.

[...]

[The] choice [of whether or not third party tests are advisable] is entirely

dependent on a variety of customer considerations, among which are government regulations. However, ... third party tests do need considerable expertise, whereas such expertise as well as knowledge of the design is in most cases available at suppliers' level. Thus third party tests can add considerable administrative complexity and often delay, to say nothing of significantly adding to cost.

2. PRINCIPLES OF CONFORMANCE

Any organization publishing standards in any field does so on the assumption that they will benefit both suppliers and users of products....

Historically, the earliest standards were probably physical standards for weights and measures. Obviously, these benefited both merchants and their customers; the customers avoided short measure and honest merchants giving full measure were not driven out of business by less scrupulous competitors. Broadly similar considerations apply today to most standards, including computer standards.

Innumerable standards are now relied upon in business and industry, for example, to ensure interchangeability of parts in producing and repairing machinery and plant in the factory and in the home. Not merely must the parts fit so that mechanism can be assembled (and spare parts procured), they must be made of suitable materials so that they do not wear out too fast or fail, say to provide the correct degree of friction, as in brake lining. In such cases, a hierarchy of standards is needed, starting with the primary ones for length mass and time, then going on to more complex matters such as the form of screw threads and including others defining methods for measurement of hardness, say of escapement parts in a watch. The actual design of any particular watch would then also call for a set of manufacturers' internal standards defining the dimensions and other characteristics of its parts. Checking the conformance of any component part will involve checking its conformance within defined limits of tolerance to each clause of each of the relevant standards.

Obviously many such general considerations will apply to the setting of standards for computer products. For instance, consider the set of standards required to make sure that it is possible to interchange data on magnetic tape between two computer systems. Clearly it is necessary to stipulate (with acceptable tolerance levels) the physical characteristics of the tape, its width, thickness and magnetic properties and also the dimensions and material of the reel on which the tape is wound. Then the limits of size, position and magnetic strength of the recorded elements on the tape must be defined. Given conformance with these criteria, one could feel confident that individual magnetic elements recorded on a tape by one tape transport could be successfully read by another.

However, unless there is also agreement about the meaning of certain recorded bit combinations, it may still not be possible even to stop the tape correctly between blocks. Alternatively, if one simply relies on the absence of recorded signals in inter-block gaps, then those gaps must be of defined

sizes and (in general) free from spurious signals and so on. Again, there must be some agreed way of recognizing the start and end of the tape to avoid breaking it or unwinding it completely from the reel.

Further, if a computer attached to a transport reading an interchange tape is to make sense of the elements recorded on the tape, the bit combinations in the recorded elements must conform to some agreed code. There must also be information on the tape to identify both the tape itself and the data written on it, in other words a label. So certain elements of a procedural protocol for the reading and writing processes must be incorporated into a magnetic tape interchange standard.

Thus, the successful interchange of data, on magnetic tape say, is likely to depend on conformance not merely to one but to a whole set of particular specialized standards for character codes, for recording these on magnetic tape, for labeling or identifying the tape, for the format of the data and so on. These in turn rely for their validity on certain primary standards, not only of physical measurement, e.g. of length or magnetic properties, but also of sets of characters in the case of codes. As was stressed above, this is so well known and even obvious as to be commonly forgotten or ignored.

But it is very important to be clear what may properly be said for instance about conformance to data interchange standards. In other words what it is that is actually required to conform.

A question as to whether a particular magnetic tape transport conforms to a certain set of data interchange standards, can thus be answered only by a statement about its capability of recording and reading tapes in accordance with those standards, since the standards in fact say nothing explicitly about tape transport as such.

When checking for conformance to a magnetic tape interchange standard, one should look upon the transport, the computer driving it, and the computer software together as equivalent to a piece of test equipment. It is convenient, indeed it may be essential, to make use of a transport in determining conformance to a tape interchange standard, although the transport itself is not the subject of that standard. This type of distinction will be seen to be even more important in later discussion of conformance to other kind of standard. What is implied is that when either drafting or reading a standard one must be very clear as to precisely what entities are and what entities are not affected directly by that standard.

3. SOME PROBLEMS OF CONFORMANCE

In this section some of the problems that have arisen in establishing conformance are examined. Ideally, any standard ought to be written in such a way that conformance can be established, if not unambiguously, at least to the joint satisfaction of a typical vendor and a typical purchaser. Conformance would be checked with one hundred per cent certainty either by a set of physical measurements, by direct inspection or by performing certain functional tests that give yes or no answers, or some combination of the three. But not all standards are amenable to this approach. For example,

particular difficulties have arisen in defining conformance, notably in connection with programming language standards. An early example of such a standard would apply only to the language, and would consist of definitions of what could legitimately be said in that language and how it was to be expressed. Typically, it would prescribe a set of commands that could be written and made to apply to certain defined classes of variable or operand, what those commands meant and what the proper results of executing the commands should be. In effect, such language definitions postulated a hypothetical machine that could execute directly programs written in the language, though usually such a machine did not exist.

The difficulties that arose in deciding conformance led to the introduction of conformance clauses in language specifications. These clauses often stated both the conditions to be satisfied by a program purporting to be written in the language and by an implementation in the form of a compiler/computer combination.

If one were asked whether a language standard can or ought to be applicable to a compiler, one should consider the precision with which it is possible to determine whether the compiler, together with the hardware on which it is designed to run, precisely emulates the hypothetical machine implied by the standard, command by command. In the strictest sense, this cannot be done by testing. High-level commands are written in terms of general operands described by symbols, whereas any real (emulating) machine operates logically and arithmetically on bit patterns. A correspondence between the transformation of actual bit patterns and the transformation of the ideal operands is difficult to define precisely and unambiguously.

For this reason if for no other, it is usually impractical to verify exactly the operations of individual commands. Moreover, unless the emulation is by an interpretive process one command at a time, the sequence of machine level instructions generated by a compiler for each high level command will depend on the sequence of other high level commands in which it is embedded. It is thus possible in practice to verify only the results of several commands taken as a sequence on selected ranges of operands. The combinations of sets of machine, level sequences of instructions and of values of operands are limitless, and only a small fraction of all possible command sequences can be explored in the testing. Experience has shown that extensive testing is necessary if troublesome misinterpretations of a language standard are to be detected.

Other more mundane reasons why it may not be completely straightforward to determine whether a given product conforms to a specific standard include:

i) the drafting of the standard itself may not be clear or complete, (e.g. the test limits have not been defined), so that its very interpretation is contentious,

ii) no satisfactory method of testing conformance was (or could have been) laid down so that technical disputes arise in interpreting test results,

iii) the product in question fails to embody every aspect of the standard, even though it conforms in respect of those aspects it does embody,

iv) the product does conform in every defined respect but it incorporates additional features of a similar kind to those specified and which could, for example, affect interchanges of data or programs, interworking between equipments or interchange of units of equipment,

v) the standard left options to the implementor that mean that a desired form of interworking can be frustrated.

The first two of these reasons imply some weaknesses in standards that could possibly have been eliminated in the drafting process and that in future are more likely to be avoided now that stress is being placed on the importance of doing so. However, as has been explained in connection with programming languages, comprehensive testing for conformance may be strictly impossible, so that while the position can certainly be improved, absolute conformance may not be a valid concept in every instance.

The last three reasons however relate to decisions by the designer or supplier of a product intended to conform to, or to interwork with, other products in accordance with some standardized protocols or procedures. The level of standardization achieved may be entirely adequate and acceptable in some contexts, but not others. It is also clear that both the force and the value of statements of conformance depend on the extent to which the standard was drafted with a clear intention of making conforming products identical, interchangeable or interworkable. Very often the variation in the drafters' intentions here has been enormous. In the case of programming language standards, the drafters were clearly unable to meet the objective of completely standard entities (programs or compilers) that would interwork with no difficulty at all.

In these circumstances, it becomes specially important that statements made by different suppliers are on similar lines, so that one does not claim conformance in circumstances where another would not feel justified in doing so. ... But it is equally important that users understand clearly what manufacturers' statements about conformance mean, and if confirmed, what benefits they as users can expect to derive from the level of standardization implied.

[...]

5. SIGNIFICANCE OF CONFORMANCE CLAIMS

In this section are given some idealized meanings of conformance to standards in each of the above categories. It is accepted that a claim to conform may be substantiated by the passing of certain checks or tests. Ultimately the claim rests on the product having been correctly designed to conform. **For many reasons, such ideal meanings may not always be applicable today.** Consideration of deviations from these ideals does, however, lead to practical recommendations.

[...]

5.5 Significance of a Claim to Conform to Standards of the Proposed Classes

The following are statements of the ideal, not necessarily of current practice.

5.51 Class I -Physical Standards including Safety Standards
Conformance involves checking to mandatory clauses of the standard; the result of each check gives a clear indication of pass or fail and there is no need to do tests or checks in any particular sequence or combination. The results will be the same. Conformance means passing all tests or checks.

5.5.2 Class II -Codes and Formats
The standard will distinguish mandatory from optional features and define both the tests or checks applicable as well as the category or categories of object covered by the standards. Checking will involve establishing:

i) conformance with all mandatory features including declared alternatives,

ii) that in respect of each optional feature, a declaration has been made of the option selected,

iii) conformance to chosen options.

In the Unicode context, an alternative would be the choice of an encoding form, while an option might be whether or not to support a particular script, or to provide tailoring where permissible.

5.5.3 Class III -Programming languages
A programming language standard defines a programming language and not its compilers which are means by which programs written in that language are converted for execution. The standard does not define a compiler explicitly though in practice the problem is to test compilers.

One can envisage high-level test programs or suite of test programs so devised that all features of a language are systematically exercised. If these test programs when submitted to a compiler produced a machine level program that ran and produced the intended results, one could infer that the compiler was (in general) capable of compiling source code written in accordance with the language standard.

For compilers the term "validation" is used increasingly to mean checking to see that a compiler will properly handle programs written in a given high-level language. The issue is not one of testing for absolute conformance since that is not achievable. Obviously a validation process will not often give an unequivocal result, more a quality rating. The only unequivocal result would be an abject failure. The reasons for this were explained in section 3 above.

To test a claim that a program has been written in a specified programming language, the program could be submitted to a compiler known to be effective for that language and for a particular computer. If the compiler a)

compiled the program, and b) the program ran and gave the intended results for each of a set of carefully chosen input parameters, it might then reasonably be deduced that the program submitted was correctly written in that language.

5.5.4 Class IV –Communication Protocols
Establishment of conformance rules for standards for communication protocols has been handicapped by:

- the new architectural approach needed,
- the consequent lack of working experience,
- the inherent complexity of the functions to be performed.

Thus procedures for conformance testing are immature and, indeed, the very meaning of conformance in this context is still under discussion.

[...]

The Unicode Standard has features of both Class III (if one views a programming language as example of an algorithmic transformations of input to output) and Class IV (for example, the character encoding forms, Unicode Normalization Algorithm, and Unicode Bidirectional Algorithm are all more or less concerned with the consistency of interchange of text data).

References

[10646] International Organization for Standardization. *Information Technology--Universal Multiple-Octet Coded Character Set (UCS)*. (ISO/IEC 10646:2003).
For availability see http://www.iso.org

[14651] International Organization for Standardization. *Information Technology--International String ordering and comparison--Method for comparing character strings and description of the common template tailorable ordering.* (ISO/IEC 14651:2001).
For availability see http://www.iso.org

[Bidi] Unicode Standard Annex #9: ~~*The*~~ *Unicode Bidirectional Algorithm*
http://www.unicode.org/reports/tr9/

[Boundaries] Unicode Standard Annex #29: ~~*Text Boundaries*~~*Unicode Text Segmentation*
http://www.unicode.org/reports/tr29/

[CharMapML] Unicode Technical Standard #22: *Character Mapping Markup Language (CharMapML)*,
http://www.unicode.org/reports/tr22/

[Charts] The online code charts can be found at
http://www.unicode.org/charts/
An index to characters names with links to the corresponding chart is found at: http://www.unicode.org/charts/charindex.html

[DerivedAge] The version for which a given character was added to the Unicode Standard is listed in:
http://www.unicode.org/Public/UNIDATA/DerivedAge.txt

[ECMA1983] *The Meaning of Conformance to Standards*, ECMA TR/18, September 1983, ECMA, Geneva.

[Errata] Updates and errata to the Unicode Standard, as well as other technical standards developed by the Unicode Consortium can be found at http://www.unicode.org/errata/

[Feedback] Reporting Errors and Requesting Information Online http://www.unicode.org/reporting.html

[FAQ] Unicode Frequently Asked Questions http://www.unicode.org/faq/ *For answers to common questions on technical issues.*

[Glossary] Unicode Glossary http://www.unicode.org/glossary/ *For explanations of terminology used in this and other documents.*

[Identifier] Unicode Standard Annex #31: *Unicode Identifier and Pattern Syntax*, http://www.unicode.org/reports/tr31/

[LineBreak] Unicode Standard Annex #14: *Unicode Line Breaking ~~Properties~~ Algorithm* http://www.unicode.org/reports/tr14/

[Normalization] Unicode Standard Annex #15: *Unicode Normalization Forms* http://www.unicode.org/reports/tr15/

[Property Model] Unicode Technical Report #23: *The Unicode Character Property Model,* http://www.unicode.org/reports/tr23/

[RegEx] Unicode Technical Standard #18: *Unicode Regular Expressions*, http://www.unicode.org/reports/tr18/

[Reports] Unicode Technical Reports http://www.unicode.org/reports/ *For information on the status and development process for technical reports, and for a list of technical reports.*

[Stability] Unicode ~~Stability Policy for~~ Character Encoding Stability Policy~~and Character Properties~~ http://www.unicode.org/~~standard~~policies/stability_policy.html

[UCA] Unicode Technical Standard #10: *Unicode Collation Algorithm*, http://www.unicode.org/reports/tr10/

[UCD] Unicode Character Database, http://www.unicode.org/ucd/ *For an overview of the Unicode Character Database and a list of its associated files*

[Unicode] The Unicode Standard *For the latest version see:* http://www.unicode.org/versions/latest/. *For the last major version see:* The Unicode Consortium. *The Unicode Standard, Version 5.0*. (Boston, MA, Addison-Wesley, 200~~6~~7. 0-321-48091-0) *or online as* http://www.unicode.org/versions/Unicode5.0.0/

[UTC] The Unicode Technical Committee, see http://www.unicode.org/consortium/utc.html for more information on procedures ~~etc~~.

~~[UTR32] Unicode Technical Report #32: *Assessing Unicode Support*, http://www.unicode.org/reports/tr32/~~

[Versions] Versions of the Unicode Standard, http://www.unicode.org/standard/versions/
For information on version numbering, and citing and referencing the Unicode Standard, the Unicode Character Database, and Unicode Technical Reports.

Acknowledgements

Thanks to Dr. Julie Allen for extensive copy-editing. Thanks to ECMA for giving permission to reproduce excerpts of their report on the meaning of conformance to standards.

Modifications

The following summarizes modifications from the previous version of this document.

Revision 4

- Proposed update. [KW]
- Minor edits to the text.
- Corrected titles of UAXs and UTRs.
- Removed reference to withdrawn UTR #32.

Revision 3

- Updated for publication as Unicode Technical Report. [AF]

Revision 2

- Updated to Draft version incorporating input from reviewers. [AF]

Revision 1

- Initial proposed Draft. [AF]
