Technical Reports

L2/09-024

Proposed Draft Unicode Technical Standard #46

UNICODE IDNA COMPATIBLE PREPROCESSSING

Version	1 (draft 1)
Authors	Mark Davis (markdavis@google.com), Michel Suignard
Date	2008-12-09
This Version	http://www.unicode.org/reports/tr46/tr46-1.html
Previous Version	n/a
Latest Version	http://www.unicode.org/reports/tr46/
Revision	1

Summary

This document provides a specification for an internationalized domain name preprocessing step that is intended for use with IDNAbis, the projected update for Internationalized Domain Names. The proposed specification maintains compatibility with IDNA2003 (the current version of Internationalized Domain Names), and consistently extends that mechanism for characters introduced in any later Unicode version.

At this point, IDNAbis is still in development, so this draft is based on the current draft of IDNAbis, and may change substantially as that draft changes.

Status

This is a draft document which may be updated, replaced, or superseded by other documents at any time.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 Standard (UTS)*** *is an independent specification. Conformance to the Unicode Standard does not imply conformance to any UTS.*

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. Introduction

Until 2003, domain names could only contain ASCII letters. The Internationalized Domain Name specifications adopted by the IETF in 2003 allow Unicode characters in domain names, as part of what are called IRIs (Internationalized Resource Identifiers). For example, one can now type in "http://bücher.de" into the address bar of any modern browser, and it will go to a corresponding site, even though the "ü" is not an ASCII character. Internally, this is handled by transforming the string into a case-folded and normalized (NFKC) form, then mapping it to a sequence of ASCII characters using a transformation known as Punycode. For this case, the Punycode value actually used to look up domain names on the wire is "http://xn--bcher-kva.de". The specifications for this are called the IDNA2003 specifications, which include: the IDNA base specification [RFC3490], Nameprep [RFC3491], Punycode [RFC3492], and Stringprep [RFC3454].

[Editorial Note: this introduction is too long. It needs to be condensed, possibly with extra material pushed to an appendix. The overall tone of the document also needs to become less informal, with removal of proposal-style language.]

Because of the transformation step in IDNA2003, it is possible to use "http://Bücher.de", as well as "HTTP://BÜCHER.DE", or "HTTP://BU¨CHER.DE" (where the ¨ represents a U+0308 (◌̈) COMBINING DIAERESIS), or many other variations.

There is a projected update of IDNA2003 which is called IDNAbis or IDNA2008. There are quite a number of changes between these two versions: the one

relevant to this document is that IDNAbis no longer has a required case folding and normalizing step. Instead, at a low level IDNAbis disallows any string that is not already case-folded and normalized. This means that strict adherence to IDNAbis, without any other action, would cause any of the above variant strings to fail. Thus typing "http://Bücher.de" would fail. (Domain names containing all ASCII characters, such as "Bucher.de", would continue to work even with case variations.) However, IDNAbis does allow for a preprocessing step, which is called a "local mapping".

Many user agents will have a requirement to interoperate compatibly with the prior IDNA2003 specification and/or operate in an environment that needs to allow lenient typing or parsing of internationalized domain names. This includes not only browsers, but email clients, word processors, IM clients, and many others. It is generally understood at the W3C that all attributes that take URLs should take full IRIs, not punycoded-URIs, so for example SVG, MathML, XLink, XML, et cetera, all take IRIs now, as does HTML5.

For example, here is a chart showing the behavior of some major browsers with links containing IDNs.

Browser Interpretation of IDNs

	Link	Firefox3, Safari	IE7	Comments
1	<a href="http://xn--bcher-kva.de">	works	works	Punycode version
2	<a href="http://bücher.de">	works	works	
3	<a href="http://Bücher.de">	works	works	
3a	<a href="http://Bücher.de">	works	works	The ü here is the decomposed form: U+0075 (u) LATIN SMALL LETTER U + U+0308 (¨) COMBINING DIAERESIS
4	<a href="http://B%C3%BCcher.de">	works	***doesn't***	%C3%BC is the percent-escaped UTF-8 version of ü; this is being implemented in current and upcoming versions of browsers...
5	<a href="http://Bücher ． com">	works	works	The dot is a U+FF0E (．) FULLWIDTH FULL STOP
6	<a href="http://Bücher ⒈ com">	works	works	The "⒈" are actually the single U+2488 (⒈) DIGIT ONE FULL STOP, so this maps to http://bücher1.de

Note that #6 is not formally provided for in IDNA2003, because the transformation is handled there on a label-by-label basis. However, this form is

commonly supported by browsers and other programs, probably because it is just simpler to apply the transformation to the whole host name rather than to each label individually.

Firefox and IE7 both accept all of these forms (except #4, which is coming), and interpret them as equivalent. Because of that, and because of the substantial number of existing (and future) web pages that contain these formats, implementations will have little choice but to support a preprocessing step that allows for all of the forms, into the indefinite future. Note that this is not simply a typing or UI issue; IRIs are in existing HTML pages as hrefs, and also used in many other W3C protocols. These forms are also parsed out of plain text; for example, most email clients and IM clients parse for IRIs and add links to them. So an occurrence of "http://Bücher.de" in the middle of plain text will often be transformed as well.

IDNAbis allows for preprocessing (called "local mapping"), and even allows these to be different according to locale or application program. But it is undesirable to have different programs (browsers, email clients, et cetera) mapping these characters differently; that would cause a huge interoperability problem. For example, examine at what could happen to different strings in the following table under locale-specific mappings.

Possible Local Mapping Variations

	Link Text	Comments
1	http://schaffer.de	legal as is
2	http://schaeffer.de	legal as is
3	http://schäffer.de	legal as is
4	HTTP://SCHAFFER.DE	always legal, matches #1
5	HTTP://SCHÄFFER.DE	Could fail, or map to #1 for English, or #2 for German, et cetera
6	HTTP://SCHÆFFER.DE	Could map to #4 for English, or other languages without æ
7	HTTP://SCHÄffER.DE	Could fail, or map to #1. (The "ff" is U+FB00 (ff) LATIN SMALL LIGATURE FF)
8	HTTP://SCHÄF①FER.DE	Could fail, or map to #1. The ① here represents the normally invisible: U+00AD SOFT HYPHEN.
9	HTTP://①②SCHÄFFER③.DE	Could fail, or map to #1 or #2. The ①, ②, and ③ here represent the following normally invisible characters: ① U+E0065 TAG LATIN SMALL LETTER E, ② U+E006E TAG LATIN SMALL LETTER N, and ③ U+E007F CANCEL TAG.

An IDNA2008-conformant implementation could remap any of the items #4 to #9 in the **Link Text** column. using a local mapping—or not, in which case they would fail. It could remove the illegal characters in #8 to #9, or not remove them and have the lookup fail. It could map the ligature ff to ff, or not. It could even decide, based on local linguistic mappings, to map #5 and #6 to different valid

domain names, different than what IDNA2003 does. Such mappings could be based on the UI language of the client, or the language of the email, or the default system language, or other choices. That means on the same page, a browser might go to different places depending on what the user's language was.

With IDNA2003, in contrast, the mappings for all of these are completely determinant (with all but the first being allowed, and the last being disallowed).

This specification provides a common mapping that maintains compatibility with IDNA2003, accommodates the newer Unicode characters, provides stability over time, and thus allows for interoperability among all programs that use it.

Note that lower-level protocols, such as the SMTP envelope, should require the strict use of already-transformed IDNs, and thus not use the preprocessing specified here. Language-specific modifications to the preprocessing specified in this document are outside the scope of this document; they are, however, very strongly discouraged because of the problems they pose for interoperability.

Open Issues

- *The exact formulation for IDNAbis is not final yet, and we would not want to release a preprocessing specification until it is final.*

1.1 Requirements

- The preprocessing is compatible with IDNA2003, with a small set of documented differences for certain rare characters. See *Section 5, Differences from IDNA2003*.
- Where the preprocessing results in an "abort with error", the input is not interpreted as valid.
    - In a user-interface, this should be indicated with a warning; in other processing (such as search-engine parsing of a web page), lookup would fail.

2 Conformance

The requirements for conformance on implementations of the Unicode IDNA Compatible Preprocesssing are as follows:

C1 *Given a version of Unicode, and a Unicode String, a conformant implementation of Unicode IDNA Compatible Preprocessing shall replicate the results given by applying the algorithm specified by Section 3, Preprocessing.*

The algorithm is a *logical* specification, designed to be straightforward to describe. An actual implementation of the algorithm is free to change any part of the algorithm as long the result of preprocessing by the implementation would be the same as the result generated by the logical

algorithm.

3. Preprocessing

The input to the preprocessing is a prospective *domain_name* string, which is a sequence of labels with dot separators, such as "Bücher.de". (For more about the parts of a URL, including the domain name, see http://tools.ietf.org/html/rfc3987). A series of steps transform this input *domain_name* string.

The preprocessing assumes that the input string is in Unicode. This may have involved converting escapes in an original domain name string to Unicode code points as necessary, depending on the environment in which it is being used. For example, this can include converting:

- HTML NCRs like 十 for U+5341 (十) CJK UNIFIED IDEOGRAPH-5341
- Javascript escapes like \u5341 for U+5341 (十) CJK UNIFIED IDEOGRAPH-5341
- URI/IRI %-escapes like %2e for U+002E (.) FULL STOP.

The preprocessing consists of the following steps, performed in order.

1. **Map** the *domain_name* string according to the IDNA Mapping Table (see below).
2. **Normalize** the *domain_name* string to Unicode Normalization Form C:
   - *domain_name = toNFC(domain_name)*
   - Note that because of the construction of the table, characters are limited to those already allowed by NFKC, so this is equivalent to toNFKC().
3. **Parse** the *domain_name* string into labels, using U+002E (.) FULL STOP as the label delimiter.
   - Note that the dot may have resulted from a mapping from other characters, such as U+2488 (⒈) DIGIT ONE FULL STOP or U+FF0E (．) FULLWIDTH FULL STOP. See Section 5.1.2, Remapped.
4. **Verify** that each label in the *domain_name* complies with IDNAbis.
   - *Abort with error if it does not comply*
   - Each label that contains only characters [\-a-zA-Z0-9] is an ASCII label. Each other label must conform to the IDNAbis specification.
   - IDNAbis has different requirements for two types of implementations: lookup and for registries. The verification will thus depend on the particular type of implemenation.
5. **Return** the string resulting from the successive application of the above steps, if there has been no error.

Note that the processing matches what is commonly done with label delimiters by browsers, whereby characters containing periods are transformed into the

NFKC format *before* labels are separated. These characters can be seen with the Unicode utilities using a regular expression:

- http://unicode.org/cldr/utility/list-unicodeset.jsp?a=[:toNFKC=/\./:]

Some of these characters would be effectively forbidden, because they would result in a sequence of two periods, and thus empty labels.

Note also that some browsers allow characters like "_" in domain names. Any such treatment is outside of the scope of this document.

4. IDNA Mapping Table

The IDNA Mapping Table provides a combined case folding and NFKC normalization, with some small modifications for IDNA2003 compatibility. The values in the table will remain stable for all future versions of Unicode; that is, no mappings will be changed, and any new mappings will only be added for newly assigned characters. There are more details in each section below.

Each version of Unicode will contain an updated version of this table: implementations will never need to actually use the algorithm for generating the tables—they can just pick up the data and use them in the Preprocessing algorithm.

Note that the way that the IDNA Mapping Table is constructed, in order to ensure that isNFKC(output), it is sufficient to do toNFC(output). That is, the extra changes that are in NFKC but not in NFC are already in the table. It is also necessary to do *at least* toNFC(output), because otherwise the text may have unordered combining marks and/or uncomposed character sequences.

The IDNA Mapping Table is stabilized in Unicode. That is, characters will only be added to the domain, never removed or changed in mapping—and the only characters that will be added are those that are newly assigned.

Editorial Note: we probably can't be quite this strict about stability, because IDNAbis (or a subsequent version) might make destabilizing changes.

4.1 Usage

The IDNA Mapping Table consists of a set of mappings from single code points to a sequence of zero or more other code points, also referred to as a 'table'. All code points that are not specifically entered into the table are mapped to themselves.

To use the table to map a string, walk through the string, one code point at a time. If there is a mapping entry for that code point, replace that code point with the result of the mapping entry. Otherwise retain the code point as is.

4.2. UCD Representation

The IDNA Mapping Table is represented in the Unicode Character Database

(UCD) via two properties. These properties will be updated with each version of Unicode.

1. A contributory property, Other_Idna_Mapping, which contains the exceptional values.
2. A derived property, Idna_Mapping (Idna_Maps), which contains the full mapping table.

During the span of time between the first release of IDNAbis, and the release of the subsequent version of Unicode, the tables will initially be made available outside of the UCD.

5. Differences from IDNA2003

The following are differences from IDNA2003. Section 5.1 Joiner Characters and Section 5.2 Other Special Cases are special exceptions required for compatibility with IDNA2008. The other exceptions are because of changes in Unicode between version 3.2 and version 5.1. They could have been carried forward as exceptions, but their usage is so rare that they are simply documented here.

For security and interoperability, applications may need to generate two versions of domain names, one with the exceptions of Section 5.1 Joiner Characters and Section 5.2 Other Special Cases, and one without. This would allow determining whether there is a divergence in a registry between the two forms.

To use a online demo of IDNA2003 mappings, see http://demo.icu-project.org/icu-bin/idnbrowser.

5.1 Joiner Characters

The following characters were mapped to nothing (deleted) in IDNA2003. They are allowed in IDNA2008, and thus unchanged by this mapping.

U+200C ZERO WIDTH NON-JOINER

U+200D ZERO WIDTH JOINER

5.2 Other Special Cases

The following characters were case-folded in IDNA2003. They are allowed in IDNA2008 (despite being mapped in IDNA2003), and thus unchanged by this mapping.

U+00DF (ß) LATIN SMALL LETTER SHARP S

U+03C2 (ς) GREEK SMALL LETTER FINAL SIGMA

5.3 Post Unicode 3.2 Case Pairs

These are characters that did not have corresponding lowercase characters in Unicode 3.2, but had lowercase characters added later. Unicode has since

stabilized case folding, so that this will not happen in the future. That is, case pairs will be assigned in the same version of Unicode—so any newly assigned character will either have a case folding in that version of Unicode, or it will never have a case folding in the future.

U+04C0 (Ӏ) CYRILLIC LETTER PALOCHKA

U+10A0 (Ⴀ) GEORGIAN CAPITAL LETTER AN

…U+10C5 (Ⴥ) GEORGIAN CAPITAL LETTER HOE

U+2132 (Ⅎ) TURNED CAPITAL F

U+2183 (Ↄ) ROMAN NUMERAL REVERSED ONE HUNDRED

5.4 Post Unicode 3.2 Normalization Mappings

These are characters whose normalizations changed after Unicode 3.2 (all of them were in Unicode 4.0.0). See Corrigendum #4: Five Unihan Canonical Mapping Errors. While the set of characters that are normalized to different values has been stable in Unicode, the results have not been. As of Unicode 5.1, normalization is completely stabilized, so these are the only such characters.

U+2F868 (?) CJK COMPATIBILITY IDEOGRAPH-2F868 → U+2136A (?) CJK UNIFIED IDEOGRAPH-2136A

U+2F874 (?) CJK COMPATIBILITY IDEOGRAPH-2F874 → U+5F33 (?) CJK UNIFIED IDEOGRAPH-5F33

U+2F91F (?) CJK COMPATIBILITY IDEOGRAPH-2F91F → U+43AB (?) CJK UNIFIED IDEOGRAPH-43AB

U+2F95F (?) CJK COMPATIBILITY IDEOGRAPH-2F95F → U+7AAE (?) CJK UNIFIED IDEOGRAPH-7AAE

U+2F9BF (?) CJK COMPATIBILITY IDEOGRAPH-2F9BF → U+4D57 (?) CJK UNIFIED IDEOGRAPH-4D57

6 Mapping Table Construction

The IDNA Mapping Table is constructed as specified in this section, for each version of Unicode. Post Unicode 5.0, case folding and normalization are always backwards compatible. The only issue for any new release of Unicode is whether any unassigned characters needed to be added to the exception table.

Like all Unicode properties, it is the data in the Unicode Character Database that is normative. This is simply a description of how that data is generated.

Informally, the table construction is done by mapping each Unicode character by applying case folding and then normalization to Unicode Normalization Form KD (NFKD). There are some exceptional mappings that provide for compatibility with

IDNA2003, and allow for special handling of future assigned characters. Those are listed in Section 5.3 Special Cases. Note that unassigned (reserved) code points never need an entry in the IDNA Mapping Table; they do not need to be included because their presence will cause an error anyway in the preprocessing.

Formally, the construction of the IDNA Mapping Table is specified as:

For each code point X:

1. If X is in the IDNA Preprocessing Exceptions Table, add the mapping found in that table
2. Else add a mapping from X to toNFKC(toCaseFold(toNFKC(X)))

Notes:

- toCaseFold and isCaseFolded are defined in the Unicode Standard 5.0, *Section 3.13 Default Case Algorithms*, page 125, rule R4 and definition D127
  - (also http://www.unicode.org/versions/Unicode5.0.0/ch03.pdf#G34078)
- toNFKC and isNFKC are defined in Unicode Standard 5.0, UAX#15, *Section X2 Notation*, page 1339.
  - (also http://www.unicode.org/reports/tr15/#Notation)

The following is an exhaustive list of the items in the IDNA Preprocessing Exceptions Table. The notation [:xxx:] means a Unicode property value. A mapping is expressed as X → Y, where X is a single code point, and Y is a sequence of zero or more other code points.

This set is stabilized. That is, characters will only be added to the set, never removed or changed in mapping—and the only characters that will be added are those that are newly assigned.

6.1. Removed (X → "")

These are specific mappings as part of IDNA2003, plus natural property extensions for post Unicode 3.2.

The following characters are removed in the mapping; that is, they map to the empty string:

- U+00AD SOFT HYPHEN
- U+034F COMBINING GRAPHEME JOINER
- U+1806 (‐) MONGOLIAN TODO SOFT HYPHEN
- U+200B ZERO WIDTH SPACE
- U+2060 WORD JOINER
- U+FEFF ZERO WIDTH NO-BREAK SPACE
- Variation Selectors

- Default-Ignorable Code Points minus Unassigned minus Join Controls (http://unicode.org/cldr/utility/list-unicodeset.jsp?a=[[:di:]-[:unassigned:]-[:Join_ This allows text that has default ignorable characters (thus invisible to the user) to be removed in processing.

In UnicodeSet notation, this is:

[\u034F\u200B-\u200D\u2060\uFEFF\u00AD
[:variation_selector:][:di:]-[:unassigned:]-[:Join_C:]]

The Join Controls were mapped to nothing (deleted) in IDNA2003. They are allowed in IDNAbis in limited contexts.

- U+200C ZERO WIDTH NON-JOINER
- U+200D ZERO WIDTH JOINER

[Issue: best compatibility would be to delete them in all the contexts in which they are disallowed in IDNAbis. That, however, would mean having to track the CONTEXT rules for IDNAbis, which would make for an unstable mapping.]

6.2. Remapped (X → Y)

The following are specific mappings as part of IDNA2003, having to do with label separators:

Map U+3002 (。) IDEOGRAPHIC FULL STOP (and anything mapped to it by toNFKC) to U+002E (.) FULL STOP.

That is:

- U+3002 (。) IDEOGRAPHIC FULL STOP → U+002E (.) FULL STOP
- U+FF61 (。) HALFWIDTH IDEOGRAPHIC FULL STOP → U+002E (.) FULL STOP

Note: Like IDNA2003, the set of exceptions is quite limited. Only those characters that are treated as full-stops in CJK character sets are mapped. This does not include all characters that function like full stops, nor does the mapping include are characters that look like full stops but are not full stops. Note that because the preprocessing is done to the entire domain_name string, in some cases a dot may result from the NFKC decomposition of a character like U+2488 (⒈) DIGIT ONE FULL STOP.

6.3 Special Cases (X → X)

The following characters are not altered by the mapping: they are treated as special cases by IDNAbis.

U+200C ZERO WIDTH NON-JOINER

U+200D ZERO WIDTH JOINER

U+00DF (ß) LATIN SMALL LETTER SHARP S

U+03C2 (ς) GREEK SMALL LETTER FINAL SIGMA

6.4 Grandfathered (X → Y)

If any future version of Unicode were to have property assignments that would cause the mapping for a previously-existing character to change, then the old mapping will be retained by being added as an exceptional case. This list is currently empty.

Acknowledgements

[TBD].

References

[TBD].
(http://tools.ietf.org/id/idnabis)

Modifications

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