Technical Reports

Working Draft Unicode Technical Standard #46

UNICODE IDNA COMPATIBILITY PROCESSING

Version	2 (draft 4)
Authors	Mark Davis (markdavis@google.com), Michel Suignard
Date	2009-11-03
This Version	http://www.unicode.org/reports/tr46/tr46-2.html
Previous Version	http://www.unicode.org/reports/tr46/tr46-1.html
Latest Version	http://www.unicode.org/reports/tr46/
Revision	2

Summary

This document provides a specification for processing that provides for compatibility between older and newer versions of internationalized domain names (IDN) for lookup in client software. It allows applications such as browsers and emailers to be able to handle both the original version of internationalized domain names(IDNA2003) and the newer version (IDNA2008) compatibly, avoiding possible interoperability and security problems.

*[**Review Note:** At this point, IDNA2008 is still in development, so this draft may change as IDNA2008 changes. The following is a substantial reorganization of version 2, draft 3 of this UTS; the changes previous to that version are not tracked with yellow highlighting.]*

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*This is a **draft** document which may be updated, replaced, or superseded by other documents at any time.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

One of the great strengths of domain names is universality. With http://Apple.com, you can get to Apple's website no matter where you are in the world, and no matter which browser you are using. With markdavis@google.com, you can send an email to an author of this specification, no matter which country you are in, and no matter which emailer you are using.

Initially, domain names were restricted to only handling ASCII characters. This is was a significant burden on people using other characters. Suppose, for example, that the domain name system had been invented by Greeks, and one could only use Greek characters in URLs. Rather than apple.com, one would have to write something like αππλε.κομ. An English speaker would not only have to be acquainted with Greek characters, but would also have to pick those Greek letters that would correspond to the desired English letters. One would have to guess at the spelling of particular words, because there are not exact matches between scripts.

A large majority of the world’s population faced this situation until recently, because their languages use non-ASCII characters.

1.1 IDNA2003

A system is now in place for internationalized domain names (IDN). This system is called *Internationalizing Domain Names for Applications*, or IDNA for short. It was introduced in 2003, in a series of RFCs collectively known as IDNA2003 [IDNA2003]. This system allows non-ASCII Unicode characters, which includes not only the characters needed for Latin-script languages other than English (such as Å, Ħ, or Þ), but also different scripts, such as Greek, Cyrillic, Tamil, or Korean.

The IDNA mechanism for allowing non-ASCII Unicode characters in domain names involves applying the following steps to each label in the domain name that contains Unicode characters:

1. Transforming (mapping) a Unicode string to remove case and other variant differences.
2. Checking the resulting mapped string for validity, according to certain rules.
3. Transforming the Unicode characters into a DNS-compatible ASCII string using a specialized encoding called *Punycode*.

For example, you can now type in http://Bücher.de into the address bar of any modern browser, and you will go to a corresponding site, even though the "ü" is not an ASCII character. This works because the IDN resolves to the Punycode string which is actually stored by the DNS for that site. Similarly, when a browser interprets a web page containing a link such as <a href="http://Bücher.de">, the appropriate site is reached. (In this document, when phrasing like "a browser interprets" is used, it refers both to domain names parsed out of URLs entered in an address bar and to those contained in links internal to HTML text.)

In this case, for the IDN Bücher.de, the Punycode value actually used for the domain names on the wire is http://xn--bcher-kva.de . The Punycode version is also typically transformed back into Unicode form for display. The resulting display string will be a string which has already been mapped according to the IDNA2003 rules. So in this example we end up with a display string that has been casefolded to lowercase:

http://Bücher.de → http://xn--bcher-kva.de → http://bücher.de

1.2 IDNA2008

There is a new version of IDNA under development. This version also consists of a collection of RFCs and is usually called IDNA2008 [IDNA2008]. The "2008" in that term does not reflect the actual date of approval, which is still pending and expected to occur in late 2009 or early 2010.

For the most common cases, the processing in IDNA2003 and IDNA2008 are identical. Both transform a Unicode domain name in a URL (like http://öbb.at) to the Punycode version (like http://xn--bb-eka.at). However, IDNA2008 does not maintain strict backwards compatibility with IDNA2003.

The main differences between the two are:

- **Additions.** Some IDNs are invalid in IDNA2003, but valid in IDNA2008.
- **Subtractions.** Some IDNs are valid in IDNA2003, but invalid in IDNA2008.
- **Deviations.** Some IDNs are valid in both, but resolve to different destinations.
- **Unpredictable Changes.** Some IDNs do not have predictable behavior in

applications implementing IDNA2008, due to the option of local mappings, as explained below. They may fail, or may have any of the above characteristics.

For more detail on the differences, see *Section 6.2 IDNA Comparison*.

1.3 Security Considerations

The cases of deviations and unpredictable changes introduced by the differences between IDNA2008 and IDNA2003 may cause both interoperability and security problems. They affect extremely common characters: all uppercase characters, all variant-width characters (commonly used in Japan, China, and Korea), and certain other characters like the German *eszett* (U+00DF ß LATIN SMALL LETTER SHARP S) and Greek *final sigma* (U+03C2 ς GREEK SMALL LETTER FINAL SIGMA).

IDNA2003 requires a mapping phase, which maps http://ÖBB.at to http://öbb.at (for example). Mapping typically involves mapping uppercase characters to their lowercase pairs, but it also involves other types of mappings between equivalent characters, such as mapping half-width *katakana* characters to normal (full-width) *katakana* characters in Japanese. The mapping phase in IDNA2003 was included to match the insensitivity of ASCII domain names. Users are accustomed to having both http://CNN.com and http://cnn.com work identically. They would not expect the addition of an accent to make a difference: they expect that if http://Bruder.com is the same as http://bruder.com, then of course http://Brüder.com is the same as http://brüder.com. There are variants similar to case in this respect used in other scripts. The IDNA2003 mapping is based on data specified by Unicode: what later became the Unicode property [NFKC_CaseFold].

IDNA2008 does not require a mapping phase, but does *permit* one (called "Local Mapping" or "Custom Mapping") with no limitation on what the mapping can do to disallowed characters (including even ASCII uppercase characters, if they occur in an IDN). For more information on the permitted mappings, see Section 4.3 and Section 5.3 in the Protocol document of [IDNA2008]. An implementation of IDNA2008 which uses custom mapping can, in principle, allow any mappings, with unpredictable results regarding the exact interpretation of the processed IDNs. For example, the following mappings show cases where IDNs are mapped to what would be considered completely different domain names by IDNA2003 rules:

1. Map http://ÖBB.at to http://öbb.at
2. Map http://ÖBB.at to http://oebb.at
3. Map http://TÜRKIYE.com to http://türkiye.com
4. Map http://TÜRKIYE.com to http://türkıye.com (note the dotless i)—and go to a different location than #3.

IDNA2008 does define a particular mapping, but it is not normative, and does not attempt to be compatible with IDNA2003. For more information, see the Mapping document in [IDNA2008].

1.3.1 Deviations

There are a few situations where the strict application of IDNA2008 will result in the resolution of IDNs to different IP addresses than in IDNA2003, unless the registry or registrant takes special action. This affects a relatively small number of characters, but some that are common in particular languages and will affect a significant number of strings in those languages. (For more information on why IDNA2008 does this, see the FAQ.) These are referred to as "Deviations"; the significant ones are listed below.

Code	Character	IDNA2008	IDNA2003	Example: IDNA2008	Example: IDNA2003
U+00DF	ß	ß	ss	http://faß.de	http://fass.de
U+03C2	ς	ς	σ	http://βόλος.com	http://βόλοσ.com
U+200D	ZWJ	ZWJ	*delete*	[TBD]	[TBD]
U+200C	ZWNJ	ZWNJ	*deleted*	[TBD]	[TBD]

These differences allow for security exploits. Consider http://www.sparkasse-gießen.de , which is for the "Gießen Savings and Loan".

1. Alice's browser supports IDNA2003. Under those rules, http://www.sparkasse-gießen.de is mapped to http://www.sparkasse-giessen.de , which leads to a site with the IP address *oo.kk.aa.yy*.
2. She visits a friend Bob, and checks her bank statement on his browser. His browser supports IDNA2008. Under those rules, http://www.sparkasse-gießen.de is also valid, but converts to a different Punycode domain name in http://www.xn--sparkasse-gieen-2ib.de . This can lead to a different site with the IP address *ee.vv.ii.ll*, a spoof site.

Alice ends up at the phishing site, supplies her bank password, and is robbed. While DENIC might have a policy about bundling all of the variants of ß together (so that they all have the same owner) it is not required of registries. It is quite unlikely that all registries will have or enforce such a bundling policy in all such cases.

There are two Deviations of particular concern. IDNA2008 allows ZWJ and ZWNJ characters in labels—these were removed by the IDNA2003 mapping. In addition to mapping differently, they represent a special security concern because they are normally invisible. That is, the sequence "a<ZWJ>b" looks just like "ab". IDNA2008 does provide a special category for characters like this (called CONTEXTJ), and only permits them in certain contexts (certain sequences of Arabic or Indic characters, for example). However, lookup applications are not required to check for these contexts, so overall security is dependent on registries having correct implementations. Moreover, those context restrictions do not catch all cases where distinct domain names have visually confusable appearances.

1.4 Unicode IDNA Compatibility Processing

To allow client-side applications to work around the incompatibilities between IDNA2003 and IDNA2008 for lookup, this document provides a standardized processing that allows conformant implementations to minimize the security and interoperability problems caused by the differences between IDNA2003 and

IDNA2008. This Unicode IDNA Compatibility Processing, also known as UTS46, extends IDNA2003 principles to Unicode 5.2 and beyond. It uses Unicode [NFKC_CaseFold] (the standard Unicode property) for mapping as described in this document. Thus it will allow http://ÖBB.at (mapping it to http://öbb.at). It also allows IDNs like http://√.com (which has an associated web page), although implementations may restrict the characters that they support based on security considerations, or flag the usage of such characters in a UI.

The result of the processing is a series of labels separated by U+002E (.) FULL STOP. For DNS lookup, the result of the Unicode IDNA Compatibility Processing is transformed by Punycoding each label that contains non-ASCII. It can then also be supplied to IDNA2008 lookup, which does not require checking of Punycode labels.

The Unicode IDNA Compatibility Processing produces results similar to the tactic of "try IDNA2008 then try IDNA2003". However, it avoids a dual lookup and has a much more cohesive approach, allowing browsers and other clients such as search engines to have a single processing step, without having to maintain two different implementations and multiple tables. It accounts for a number of edge cases that would cause problems, and provides a stable definition with predictable results that will remain absolutely backwards compatible over versions of Unicode. For a demonstration of differences between IDNA2003, IDNA2008, and the Unicode IDNA Compatibility Processing, see the IDNA demo.

The main goal of this document is to provide a compability mechanism for dealing with IDNA domain name lookup, not with IDNA registration. Note that neither the Unicode IDNA Compatibility Processing nor IDNA2008 address security problems associated with confusables (the so-called "paypal.com" problem). It is strongly recommended that UTR#36: Unicode Security Considerations [UTR36] be consulted for information on dealing with confusables.

1.5 Display of Internationalized Domain Names

For IDNA2003 applications, it has been customary to display the processed string to the user. This is helpful for security, since it reduces the opportunity for visual confusability. Thus, for example, http://googIe.com (with a capital I in place of the L) is revealed as http://googie.com. However, for the case of the Deviations, the distinction between the original and processed form is especially important. Thus in displaying domain names, it is strongly recommended that the Display Processing be applied. This is the same as the Unicode IDNA Compatibility Processing, except that it excludes the deviations: ß, ς, and joiners.

1.6 Notation

Sets of code pointsare defined by properties according to the syntax of *UTS#18: Unicode Regular Expressions* [UTS18] (with additional "+" signs added for clarity). Thus the set of combining marks is \p{gc=M}.

A *label* is a substring of a domain name that is bounded by the start or end of the string, or one of the following:

1. U+002E (.) FULL STOP

2. U+FF0E (．) FULLWIDTH FULL STOP
3. U+3002 (。) IDEOGRAPHIC FULL STOP
4. U+FF61 (｡) HALFWIDTH IDEOGRAPHIC FULL STOP

2 Conformance

The requirements for conformance on implementations of the **Unicode IDNA Compatibility Processing** are as follows:

C1 Given a version of Unicode and a Unicode String, a conformant implementation shall replicate the results given by applying the algorithm specified by Section 3, Processing for Lookup Processing.

C2 Given a version of Unicode and a Unicode String, a conformant implementation shall replicate the results given by applying the algorithm specified by Section 3, Processing for Display Processing.

These specifications are *logical* ones, designed to be straightforward to describe. An actual implementation is free to use different methods as long as the result is the same as the result generated by the logical algorithm.

Any conformant implementation may have *tighter* validity criteria than imposed by the Section 5, Validity Criteria. For example, an application could disallow or warn of domain name labels:

- with certain combinations of scripts, as Safari does
- with characters outside of the user's specified languages, as IE does
- with certain confusable characters, as Firefox does
- that are caught by the Google Safe Browsing API [SafeBrowsing]
- that do not meet the validity requirements of IDNA2008, including BIDI
- and so on

For more information, see UTR#36: *Unicode Security Considerations* [UTR36].

This specification is targeted at applications doing lookup. There is one recommendation for registries: to never allow the registration of labels that are invalid according to Lookup Processing.

[Review Note: We could also add a note saying that "it might still be very interesting for a registry to accept registration of "unprocessed" labels, if they really know what they are doing:
– Storing somewhere the unprocessed label as the sequence of characters that the registrant really wanted to apply for
– Preprocessing themselves and then feeding the regular registration process with the output of preprocessing.".]

3. Processing

The input to the Processing is a prospective *domain_name* string in Unicode, 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 Section 3.5 of [RFC1034]).

Preparation of the input *domain_name* string 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 numeric character references (NCRs) like &#x5341; for U+5341 (十) CJK UNIFIED IDEOGRAPH-5341
- Javascript escapes like \u5341 for U+5341 (十) CJK UNIFIED IDEOGRAPH-5341
- URI/IRI %-escapes like %C3%A0 for U+00E0 (à) LATIN SMALL LETTER A WITH GRAVE

The following series of steps, performed in order, successively alters the input *domain_name* string, and then outputs it (if there are no errors). The output of this processing is also a Unicode string, which can then be converted to a string containing Punycode labels ("asciified"). The processing is idempotent—applying the processing again to valid output will make no further changes. Where the processing results in an "abort with error", the processing fails and the input string is invalid.

There are two types of processing, Lookup Processing and Display Processing. They differ only in how the mapping table is used.

1. **Process** each code point in the *domain_name* string according to IDNA Mapping Table (Section 4), based on the status value:
   - **disallowed**: **Abort with an error.**
   - **mapped**: Replace the code point by the mapping value
   - **ignored**: Remove the code point
   - **display**: For Lookup Processing, replace the code point by the mapping value; for Display Processing, leave the character alone
   - **valid**: Leave the character alone
2. **Normalize** the *domain_name* string to Unicode Normalization Form C.
3. **Convert** any Punycode labels back into Unicode. **Abort with an error if such conversion fails.**
   - *domain_name_label = fromPunycode(domain_name_label)*
4. **Verify** that each label in the domain_name meets the validity criteria in Validity Criteria (Section 5). **Abort with an error if the validity criteria are not satisfied.**
5. **Return** the *domain_name* resulting from these steps if there has been no error.

Some browsers allow also characters like "_" in domain names. Any such extension is outside of the scope of this document.

The domain names that do not cause an error in the application of the above process are valid according to this specification. However, implementations are advised to apply additional tests to these labels such as those described in UTR#36: Unicode Security Considerations [UTR36], and take appropriate actions. For example, a label with mixed scripts or confusables may be called out in the UI.

[Review Note: Add a section of examples that illustrate each step.]

4. IDNA Mapping Table

For each code point in Unicode, the IDNA Mapping Table provide a status value and (optionally) a mapping value. The values are defined by the following data table:

- uts46-data-5.1.txt

[Review Note: The format of the table will be changed to match the following description.]

Each version of Unicode, starting at Unicode 5.1, will have an updated version of this table. A description of the derivation of these tables is in *Section 6, Mapping Table Derivation*. However, the data in the file are normative, not the description of the derivation. The tables will always be backwards compatible; if the description of the data generation needs to be changed in order to ensure that, it will be.

The files use the standard Unicode semicolon-delimited format. The first field is the hex value of the code point, and second is the status, and third field is a mapping result in hex, if applicable.

Status Values

- valid
- disallowed
- ignored (= mapped to empty string),
- mapped
- display

Examples:

```
0000..002C    ; disallowed                  #  NULL..COMMA
002D          ; valid                       #  HYPHEN-MINUS
...
0041          ; mapped ; 0061               #  LATIN CAPITAL LETTER A
...
00AD          ; ignored                     #  SOFT HYPHEN
...
```

[Review Note: Should tables in the format of a NamePrep profile [RFC3491] also be made available?]

5. Validity Criteria

Each of the following criteria must be satisfied for a label to be valid:

1. The label must contain at least one code point.
2. The label must not contain "--" (two U+002D (-) HYPHEN-MINUS characters) in the third and fourth positions, and must neither begin nor end with a U+002D (-) HYPHEN-MINUS character. [Review note: should we point to other sources for the first 2 of these?]
3. Each code point in the label must have certain status values according to the IDNA Mapping Table (Section 4):
   1. For Lookup Processing, the value must be "valid"
   2. For Display Processing, the value must be either "valid" or "display"
4. The label must not begin with a combining mark, that is: \p{gc=M}.

In addition, the label *should* meet the requirements for right-to-left characters specified in the Bidi document of [IDNA2008]. Any particular application may have tighter validity criteria, as discussed in Section 2, *Conformance*.

6 Mapping Table Derivation

The following describes the derivation of the mapping table. The data table is normative, however, not the description of the derivation. The derivation may also change in the future so as to maintain stability.

Produce an base mapping value

1. Map each character to its NFKC_CaseFold value [NFKC_CaseFold].
2. Map the following label separator characters to U+002E (.) FULL STOP
   1. U+FF0E (．) FULLWIDTH FULL STOP
   2. U+3002 (。) IDEOGRAPHIC FULL STOP
   3. U+FF61 (｡) HALFWIDTH IDEOGRAPHIC FULL STOP

Derive the base valid set

The definition is based on IDNA2003; when restricted to Unicode 3.2 characters, this set closely follows the characters allowed in IDNA2003.

Formal Sets	Descriptions
`[ \P{Changes_When_NFKC_Casefolded}`	Start with characters that are NFKC Case folded (excluding uppercase, for example).
`- \p{c} - \p{z}`	Remove Control Characters and Whitespace
`- \p{Block=Ideographic_Description_Characters}`	Remove ideographic description characters
`- \p{ascii} - [\u1806 \uFFFC \uFFFD]`	Remove ASCII and three special characters: U+1806 (᠆) MONGOLIAN TODO SOFT HYPHEN U+FFFC ( ) OBJECT REPLACEMENT

	CHARACTER U+FFFD (�) REPLACEMENT CHARACTER
`+ [\u002Da-zA-Z1-0]`	Add back all the valid ASCII

Specify the base exclusion set

The exclusion set consists of characters that were valid in IDNA2003, but would be mapped differently by later versions of Unicode. For more information, see the FAQ.

- Case Exclusions
  - 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
- Normalization Exclusions (CJK Compatibility Characters)
  - U+2F868, U+2F874, U+2F91F, U+2F95F, U+2F9BF
- Default Ignorable Exclusions
  - U+3164 ( ) HANGUL FILLER
  - U+FFA0 ( ) HALFWIDTH HANGUL FILLER
  - U+115F ( ) HANGUL CHOSEONG FILLER
  - U+1160 ( ) HANGUL JUNGSEONG FILLER
  - U+17B4 ( ) KHMER VOWEL INHERENT AQ
  - U+17B5 ( ) KHMER VOWEL INHERENT AA
- Full Stop Exclusions
  - U+2024 (․) ONE DOT LEADER ..U+2026 (…) HORIZONTAL ELLIPSIS
  - U+2488 (⒈) DIGIT ONE FULL STOP ..U+249B (⒛) NUMBER TWENTY FULL STOP
  - U+33C2 (㏂) SQUARE AM
  - U+33C7 (㏇) SQUARE CO
  - U+33D8 (㏘) SQUARE PM
  - U+FE19 (︙) PRESENTATION FORM FOR VERTICAL HORIZONTAL ELLIPSIS
  - U+FE30 (︰) PRESENTATION FORM FOR VERTICAL TWO DOT LEADER
  - U+FE52 (﹒) SMALL FULL STOP

[Review Note: recheck the above list programmatically]

Produce a status and mapping

For each code point:

1. If the code point is a Deviation character
   - the status is "display", and the mapping value is the base mapping value.
2. Otherwise, if the code point is in the base exclusion set, or if any code point in its base mapping value is not in the base valid set
   - the status is "disallowed", and there is no mapping value.
3. Otherwise, if the base mapping value is an empty string,
   - the status is "ignored" and there is no mapping value.
4. Otherwise, if the base mapping value is the same as the code point,
   - the status is "valid", and there is no mapping value.
5. Otherwise,
   - the status is "mapping" and the mapping value is the base mapping value.

6.1 IDNA2008 Characters

For comparison, the following describes the set of allowed characters defined by IDNA2008. This set corresponds to the union of the PVALID, CONTEXTJ, and CONTEXTO characters with rules defined by the Tables document of [IDNA2008]. This is only presented for comparison, and has no bearing on validity of this specification.

Formal Sets	Descriptions
`[ \P{Changes_When_NFKC_Casefolded}`	Start with characters that are NFKC Case folded (as in IDNA2003)
`- \p{c} - \p{z}`	Remove Control Characters and Whitespace (as in IDNA2003)
`- \p{s} - \p{p} - \p{nl} - \p{no} - \p{me}`	Remove Symbols, Punctuation, non-decimal Numbers, and Enclosing Marks
`- \p{HST=L} - \p{HST=V} - \p{HST=V}`	Remove characters used for archaic Hangul (Korean)

`- \p{block=Combining_Diacritical_Marks_For_Symbols}`
`- \p{block=Musical_Symbols}`
`- \p{block=Ancient_Greek_Musical_Notation}`
	Remove three blocks of technical or archaic symbols.

`- [\u0640 \u07FA \u302E \u302F \u3031-\u3035 \u303B]`	Remove certain exceptions: U+0640 (ـ) ARABIC TATWEEL U+07FA (ߺ) NKO LAJANYALAN U+302E (〮) HANGUL SINGLE DOT TONE MARK U+302F (〯) HANGUL DOUBLE DOT TONE MARK U+3031 (〱) VERTICAL KANA REPEAT MARK U+3032 (〲) VERTICAL KANA REPEAT WITH VOICED SOUND MARK ..

	U+3035 (〵) VERTICAL KANA REPEAT MARK LOWER HALF U+303B (〻) VERTICAL IDEOGRAPHIC ITERATION MARK

`+ [\u00B7 \u0375 \u05F3 \u05F4 \u30FB]`
`+ [\u002D \u06FD \u06FE \u0F0B \u3007]`
	Add certain exceptions:
	U+00B7 (·) MIDDLE DOT
	U+0375 (͵) GREEK LOWER NUMERAL SIGN
	U+05F3 (׳) HEBREW PUNCTUATION GERESH
	U+05F4 (״) HEBREW PUNCTUATION GERSHAYIM
	U+30FB (・) KATAKANA MIDDLE DOT
	plus
	U+002D (-) HYPHEN-MINUS
	U+06FD (۽) ARABIC SIGN SINDHI AMPERSAND
	U+06FE (۾) ARABIC SIGN SINDHI POSTPOSITION MEN
	U+0F0B (་) TIBETAN MARK INTERSYLLABIC TSHEG
	U+3007 (〇) IDEOGRAPHIC NUMBER ZERO

`+ [\u00DF \u03C2]`
`+ \p{JoinControl}]`
	Add special exceptions (Deviations):
	U+00DF (ß) LATIN SMALL LETTER SHARP S
	U+03C2 (ς) GREEK SMALL LETTER FINAL SIGMA
	U+200C ( ) ZERO WIDTH NON-JOINER
	U+200D ( ) ZERO WIDTH JOINER

[Review Note: Once IDNA2008 is final, the exact list of characters will be aligned.]

6.2 IDNA Comparison

The following table provides an illustration of the differences between the three specifications. It omits all code points unassigned in U5.2, and all ASCII, since those are the same for all three forms. The Count column shows the number of characters in each bucket. The Differences column calls out some illustrative character differences: sets with ... are abbreviated. Characters marked * for UTS46 are not modified in display.

[Review Note: the following table is to be updated as follows:

- Reorder the list to group similar characters.
- Regenerate the figures, since they have changed.

- Under the comments, include one or two illustrative characters, and a description of the features of that row, and a link to the full list.

]

Count	Unicode Version	IDNA2003	UTS46	IDNA2008	Comments
432	v3.2	Disallowed	Disallowed	Disallowed	
25	v3.2	Ignored	Ignored	Disallowed	[\u034F\u180B-\u180D\u200B\u2060
2	v3.2	Ignored	Ignored*	Valid	[\u200C\u200D]
2	v3.2	Mapped	Disallowed	Disallowed	[\u3164\uFFA0]
4,619	v3.2	Mapped	Mapped	Disallowed	… a 𝐚 𝑎 ⓐ A 𝐀 𝐴 Ⓐ ª ÁÀĂẮẰẴẲÂẤẦ ẪẨǍÅǺÄ
2	v3.2	Mapped	Mapped*	Valid	[ßς]
4	v3.2	Valid	Disallowed	Disallowed	[\u17B4\u17B5\u115F\u1160]
41	v3.2	Valid	Mapped	Disallowed	[ƆⱻIƸ–ƅͰᏧ–ҤƆQ–ԲЧQ–ԷҰΧᏂ₰]
3,258	v3.2	Valid	Valid	Disallowed	…₵ЭⱠȸ℗℘℞℟℣ʒ⊂⊄⊃⊅⊆⊈⊇ –☯☸–♬– – ❵➔➘–⇨⇨–⇨♭–♯㉄〒▬☺¤¢£¥…
86,676	v3.2	Valid	Valid	Valid	
14	v4.0–5.1	Disallowed	Disallowed	Disallowed	
241	v4.0–5.1	Disallowed	Ignored	Disallowed	
473	v4.0–5.1	Disallowed	Mapped	Disallowed	
1,226	v4.0–5.1	Disallowed	Valid	Disallowed	
3,538	v4.0–5.1	Disallowed	Valid	Valid	

7 Testing

[Review Note: The intent is to supply conformance test files for each Unicode version starting with Unicode 5.1, so that implementations can test their implementations against a set of data.]

8 Background

[Review Note: Some or all of this material is probably best moved to the Unicode FAQ, and just referenced from here, while some is appropriate for inclusion here. What is left here, if anything, would need to be modified to remove duplication of material.]

For compatibility in the foreseeable future, special steps need to be taken with Deviations. While some steps could be taken by top-level domain registries to mitigate the above problems (the so-called "bundle" option), there are a very large number of lower level domains that are under the control of millions of other organizations. For example, the domain names under "blogspot.com", such as http://café.blogspot.com, are controlled by the company that has registered "blogspot". For IDNA2008 to avoid problems, no registries—at whatever level—would allow two IDNs that correspond according to the Deviations table to resolve to different IP addresses. So blogspot.com would need to disallow registration of both the registration of http://gefäss.blogspot.com and of http://gefäß.blogspot.com, to prevent problems (and of other cases like the normally-invisible ZWJ and ZWNJ). However, applications cannot depend on all such registries behaving correctly, because the odds are high that at least some (and likely very many) of the many thousands of registries will not check for this. Thus the burden is primarily on applications handling IDNs to prevent the situation.

The worst of all possible cases is an implementation of IDNA2008 that uses Custom mappings. Unfortunately, there appears to be no good way to prevent security problems with these implementations, because it is impossible to anticipate what such implementations would do. Such an implementation is not limited to just the above four Deviations for exploits—it could remap even characters like "Ö" to "oe" or arbitrary other characters. Because there is no way to predict what it will do, there are no effective countermeasures for security.

Note that IDNA2008 does not make any appreciable difference in reducing problems with visually-confusable characters (so-called *homographs*). Thus programmers still need to be aware of those issues as detailed in UTR#36: Unicode Security Considerations [UTR36], including the mechanisms for detecting potentially visually-confusable characters are found in the associated UTS#39: Unicode Security Mechanisms [UTS39].

8.1 Handling Deviations

Because of the Deviations, even the strict application of IDNA2008 leads to significant new security issues. The Unicode Technical Committee and invited experts considered at length various options for dealing with Deviations. Among those options that were considered but rejected were:

1. Dual lookups, checking for differences.
   - One problem with this approach is that the IP lookup may return spurious differences, because a website may return different IP addresses for load-balancing.
2. Not mapping deviations if the registry is *trusted.* A trusted registry is one that is complies with this specification, and bundles all allowed Deviations with their mappings.
   - For example, http://www.sparkasse-gießen.de (if the registry for "de" bundles Deviations) would be unaltered, but that http://www.sparkasse-gießen.com would be mapped to http://www.sparkasse-giessen.com (if the "com" registry does not bundle Deviations) before any lookup. Note that this also applies to lower-level registries. The URL http://www.sparkasse-gießen.blogspot.de would be remapped to http://www.sparkasse-gießen.blogspot.de *unless* the registry for "blogspot.de" is trusted.

In the end, the consensus in the committee was that the distinction between deviations ({ss, ß, SS}, {σ, ς, Σ}, and joiners) was most important for display. In particular, it is strongly recommended that any registry that allows for both forms *should* always bundle them to avoid security problems. And those registries that didn't bundle would cause problems. Thus the conclusion was that the distinction between deviations did not need to be maintained in lookup, because lookup would always work with registries that are handling the deviations correctly, and would avoid security problems with the registries that didn't.

8.2 Handling Label Separators

The **Split** 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. This allows the domain name to be transformed in a single pass, rather than label by label. Some of the input characters are effectively forbidden, because they would result in a sequence of two periods, and thus empty labels. The exact list of characters can be seen with the Unicode utilities using a regular expression:

- http://unicode.org/cldr/utility/list-unicodeset.jsp?a=\p{toNFKC=/\./}

The question also arises as to how to handle escaped periods (such as %2E) and characters like U+2488 (⒈) DIGIT ONE FULL STOP or U+FF0E (．) FULLWIDTH FULL STOP that decompose to sequences that include period. While %2E is outside of the scope of this document, it is useful to see how both of these are handled in current browsers:

Input	http://à%2Ecom	%2E		http://à⒈com	⒈
Internet Explorer	http://xn--0ca.com/	= "."		http://xn--1-rfa.com/	= "1."
Firefox	http://www.xn--.com-hta.com/	≠ "."		http://xn--1-rfa.com/	= "1."
Safari / Chrome	http://xn--0ca.com/	= "."		http://xn--1.com-qqa/	≠ "1."

There are three possible behaviors for characters like U+2488 (⒈) DIGIT ONE FULL STOP:

1. The dot behaves like a label separator.
2. The character is rejected
3. The dot is included in the label (the garbled punycode seen above in the ≠ cases).

The conclusion of the committee was that the best behavior was #2, to forbid all characters (other than the 4 label separators) that contained a FULL STOP in their compatibility decompositions.

9 FAQ

[Review Note: Some or all of this material is probably best moved to the Unicode FAQ, and just referenced from here, while some is appropriate for inclusion here. What is left here, if anything, would need to be modified to remove duplication of material.]

Q. What are examples of where the different categories of IDNA implementation behave differently?

A. Here is a table that illustrates the differences, where 2003 is the current behavior in applications now.

- **Yes** indicates that the URL would be valid;
- **No** that it wouldn't be; and
- **??** that it might or might not be, depending on the exact behavior of a Custom Mapping. Such a Custom mapping might be from Ö to ö, or it might be from Ö to oe, or might not map at all. Because they are unpredictable, they are marked with ??.

URL	2003	UTS46	Strict IDNA2008	IDNA2008 with Custom Mapping	Comments
http://öbb.at	Yes	Yes	Yes	Yes	Simple characters
http://ÖBB.at	Yes	Yes	No	??	Case mapping
http://√.com	Yes	Yes	No	??	Symbol
http://faß.de	Yes	Yes	Yes*	Yes*	* Deviation (different resulting IP address)
http://ԛәлп.com	No	Yes	Yes	Yes	New Unicode (version 5.1) U+051B (ԛ) *cyrillic qa*

Q. How much of a problem is this actually if support for symbols like √ were just dropped immediately?

A. IDNA2008 removes many characters that were valid under IDNA2003, because it makes most symbols and punctuation illegal. So while http://√.com is valid in an IDNA2003 implementation; it would fail on a strict IDNA2008 implementation. This affects about 3,000 characters, mostly rarely used ones. A small percentage of those are security risks because of confusability. The vast majority are unproblematic: for example, having http://I♥NY.com doesn't cause security problems. IDNA2008 also has additional tests that are based on the context in which characters are found, but they apply to few characters, and don't provide any appreciable increase in security.

Q. Doesn't the removal of symbols and punctuation in IDNA2008 help security?

A. Surprisingly, not really. It doesn't do anything about the most frequent sources of spoofing; look-alike characters that are both letters, like "http://paypal.com" with a Cyrillic "a". If a symbol that can be used to spoof a letter X is removed, but there is another letter that can spoof X is retained, there is no net benefit. Weighted by frequency, according to data at Google the removal of symbols and punctuation in IDNA2008 reduces opportunities for spoofing by only about 0.000016%. In another study at Google of 1B web pages, the top 277 confusable URLs used confusable letters or numbers, not symbols or punctuation. The 278th page had a confusable URL with × (U+00D7 MULTIPLICATION SIGN - by far the most common confusable); but that page could could be even better spoofed with х (U+0445 CYRILLIC SMALL LETTER HA), which normally has precisely the same displayed shape as "x".

Q. What are the main advantages of IDNA2008?

[Review Note: Is it worth listing the advantages and disadvantages of IDNA2008?]

A. The main advantages are:

- Major improvement in updating to Unicode 5.2
- Major improvement in making process of updating to future Unicode versions (mostly) automatic
- Significant improvement in allowing needed sequences (combining marks at end of bidi label).
- Significant improvements to the BIDI rules:
  - restricting sequences that lead to "bidi label hopping". (While these new bidi rules go a long way towards reducing this problem, they do not eliminate it because they do not check for inter-label situations.)
- Improvements in some user's expectations for display of Deviations: sigma, sharp s, joiners.
- Improvement in clarifying that what people register is the unmapped form.

Q. What are the disadvantages of IDNA2008?

A. If IDNA2003 had not existed, then there would be few disadvantages to IDNA2008. Given that IDNA2003 does exist, and is widely deployed, the main disadvantages are:

- Major interoperability/security issue with Deviations and Unpredictables
- Significant interoperability issue by not continuing IDNA2003 mappings
- Significant increase in complexity, reducing the likelihood of correct implementation
  - For example, there are new contextual rules that are fairly complicated to implement, and are not in a machine-readable format. Without a comprehensive test suite and/or reference implementations to test against, it is fairly likely that there will be incompatibilities.
- Small interoperability issues caused by excluding symbols, punctuation
  - While there are many such characters, they are relatively rare.
- More fragile in that future Unicode versions require a manual step to avoid instabilities
  - That is, if Unicode version X changes properties in such a way as to add or remove characters from PVALID, it requires a manual step to retain the previous status.
- No requirements for stability: that all labels valid under Version X (>= 2008) must also be valid under all future versions.

Q. What is "bidi label hopping"?

A. It is where bidi reordering causes characters from one label to appear to be part of another label. For example, "B1.2d" in a right-to-left paragraph (where B stands for an Arabic or Hebrew letter) would display as "1.2dB". For more information, see the Unicode bidi demo.

Q. Are the "local" mappings just a UI issue?

A. No, not if what is meant is that they are only involved in interactions with the address bar.

Examples:

- Alice sees that a URL works in her browser (say http://faß.de or http://TÜRKIYE.com). She sends it to Bob in an email. Bob clicks on the link in his email, and doesn't find a site or goes to a wrong (and potentially malicious) site, because his browser maps to http://fass.de or http://türkiye.com while Alice's maps to http://faß.de or http://türkıye.com.
- Alice creates a web page, using <a href=" http://faß.de"> (or http://TÜRKIYE.com). Bob clicks on the link in his email, and doesn't find a site or goes to a wrong (and potentially malicious) site.
- Alice is in a IM chat with Bob. She copies in http://faß.de (or http://TÜRKIYE.com) and hits return. Bob clicks on the link he sees in his chat window. Bob clicks on the link in his email, and doesn't find a site or goes to a wrong (and potentially malicious) site.
- Alice sends a Word document to Bob with a link in it...
- Alice creates a PDF document...
- ...

Q. Do the Custom exploits require unscrupulous registries?

A. No. The exploits do not require unscrupulous registries—it only requires that registries do not police every URL that they register for possible spoofing behavior.

The custom mappings matter to security, because entering the same URL on two different browsers may go to two different IP addresses (whenever the two browsers have different custom mappings). The same thing could happen within an emailer that is parsing for URLs, and then opening a browser. And for that matter, there is nothing that prevents two different browsers from applying those custom mappings to URLs within a page, for example, to a URL in href="...".

Q. Why does IDNA2003 map final sigma (ς) to sigma (σ), map eszett (ß) to "ss", and delete ZWJ/ZWNJ?

A. This was following the Unicode Standard. These characters are anomalous: the uppercase of ς is Σ, the same as the uppercase of σ. Note that the text "ΒὀλοΣ.com", which appears on http://Βὀλος.com, illustrates this: the normal case mapping of Σ is to σ. If σ and ς were not treated as case variants in Unicode, there wouldn't be a match between ΒὀλοΣ and Βὀλος.

Similarly, the standard uppercase of ß is "SS", the same as the uppercase of "ss". Note, for example, that on http://www.uni-giessen.de, Gießen is spelled with ß, but in the top left corner spelled with GIESSEN. The situation is even more complicated:

- In Switzerland, "ss" is uniformly used instead of ß.
- The recent spelling reform in Germany and Austria changed whether ß or ss is used in many words. For example, http://Schloß.de was the spelling before 1996, and http://Schloss.de is "correct" after.

- Recently, in Unicode 5.1, an uppercase version of ß was added (ẞ), because it is attested in some cases. It is unknown, however, whether it will ever become the preferred uppercase. Unicode now treats all of these as a single equivalence class for case-insensitive matching: {ss, ß, SS, ẞ}. See also the Unicode FAQ.
- Both the German and Austrian NICs favored keeping the mapping from ß to "ss".

For full case insensitivity (with transitivity), {ss, ß, SS} and {σ, ς, Σ} need to be treated as equivalent, with one of each set chosen as the representative in the mapping. That is what is done in the Unicode Standard, which was followed by IDNA2003.

ZWJ and ZWNJ are normally invisible, which allows them to be used for a variety of spoofs. Invisible characters (like these and soft-hyphen) are allowed on input in IDNA2003, but deleted so that they do not allow spoofs.

While these are full parts of the orthographies of the languages in question, neither IDNA2003 nor IDNA2008 ever claimed that all parts of every language's orthographies are representable in domain names. There are trivial examples even in English, like the word ***can't*** (vs *cant*) or ***Wendy's/Arby's Group*** (NYSE WEN), which use standard English orthography but cannot be represented faithfully in a domain name .

The Unicode IDNA Compatibility Processing deals with the Deviations by using a different display format that preserves these distinctions.

Q. Why allow ZWJ/ZWNJ at all?

During the development of Unicode, the ZWJ and ZWNJ were intended only for presentation —that is, they would make no difference in the semantics of a word. Thus the IDNA2003 mapping should and does delete them. That result, however, should never really be seen by users—it should be just a transient form used for comparison. Unfortunately, the way IDN works this "comparison format" (with transformations of eszett, final sigma, and deleted ZWJ/NJ) ends up being visible to the user, unless a display format is used that differs from the format used to transform for lookup.

For example, there are words such as the name of the country of Sri Lanka, which require preservation of these joiners (in this case, ZWJ) in order to appear correct to the end users when the URL comes back from the DNS server.

Q. Aren't the problems with eszett and final sigma just the same as with l, I, and 1?

A. No, The eszett and sigma are fundamentally different than I,l, and 1. With the following (using a digit 1), all browsers will go to the same location, whether they old or new:

> http://goog1e.com

With the following, browsers that use IDNA2003 will go to a different location than browsers that use a strict version of IDNA2008, *unless* the registry for xx puts into place a bundle strategy.

http://gießen.xx

The same goes for Greek *sigma*, which is a more common character in Greek than the *eszett* is in German.

Q. Why doesn't IDNA2008 (or for that matter IDNA2003) restrict allowed domains on the basis of language?

A. It is extremely difficult to restrict on the basis of language, because the letters used in a particular language are not well defined. The "core" letters typically are, but many others are typically accepted in loan words, and have perfectly legitimate commercial and social use.

It is a bit easier to maintain a bright line based on script differences between characters: every Unicode character has a defined script (or is Common/Inherited). Even there it is problematic to have that as a restriction. Some languages (Japanese) require multiple scripts. And in most cases, mixtures of scripts are harmless. One can have http://SONY日本.com with no problems at all—while there are many cases of "homographs" (visually confusable characters) within the same script that a restriction based on script doesn't deal with.

The rough consensus among the IETF IDNA working group is that script/language mixing restrictions are not appropriate for the lowest-level protocol. So in this respect, IDNA2008 is no different than IDNA2003. IDNA doesn't try to attack the homograph problem, because it is too difficult to have a bright line. Effective solutions depend on information or capabilities outside of the protocol's control, such as language restrictions appropriate for a particular registry, the language of the user looking at this URL, the ability of a UI to display suspicious URLs with special highlighting, and so on.

Responsible registries can apply such restrictions. For example, a country-level registry can decide on a restricted set of characters appropriate for that country's languages. Application software can also take certain precautions—MSIE, Safari, and Chrome all display domain names in Unicode only if the user's language(s) typically use the scripts in those domain names. For more information on the kinds of techniques that implementations can use on the Unicode web site, see UTR#36: Unicode Security Considerations [UTR36].

Q. Are there differences in mapping between UTS46 and IDNA2003?

No. There are however some cases where IDNA2003 maps characters and UTS46 makes the characters disallowed. For a detailed table of mapping differences, see section 6.1 IDNA2008 Characters.

In particular, there are collections of characters that would have changed mapping according to NFKC_CaseFold after Unicode 3.2, unless they were specifically excluded. All of these characters are extremely rare, and do not require any special handling.

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

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

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.

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). 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

Q. How do implementations handle normalization for IDNA2003?

There were two corrigenda to normalization issued after 3.2. Formally speaking, an implementation applying IDNA2003 would disregard these corrigenda, but browsers do not consistently implement this behavior. In practice this makes no difference, since the characters and character sequences involved are not found except in specially-devised test cases, so it is understandable that systems may not want to maintain the extra code necessary to duplicate the broken Unicode 3.2 behavior.

Corrigendum #4: Five Unihan Canonical Mapping Errors

Corrigendum #4 deals with the 5 characters above.

Example

- 2F868 (㛼) = xn--g22n
    - 3.2 normalization → xn--j74i = 2136A (𡍪)
    - 5.2 normalization → xn--snl = 36FC (㛼)

Example Behavior

- IE/Chrome/Safari – 3.2
- FF – 5.2

Corrigendum #5: Normalization Idempotency

Corrigendum #5 deals with with a subtle algorithmic problem.

Example

- 1100 0300 1161 0323 (ᄀ̀ᅡ̣) = xn--ksa4ez54cela
  - 3.2 normalization → xn--ksa4ez795d = AC00 0300 0323 (가̣̀)
→ xn--ksa3e0795d = AC00 0323 0300 (가̣̀)
  - 5.2 normalization → xn--ksa4ez54cela = 1100 0300 1161 0323 (ᄀ̀ᅡ̣)

Example Behavior

- IE – 5.2
- Chrome/Safari – 3.2
- FF – 3.2 -- applied twice

Unicode has since stabilized normalization, so such changes will not happen in the future.

Acknowledgements

For their contributions of ideas or text to this specification, thanks to Matitiahu Allouche, Peter Constable, Craig Cummings, Martin Dürst, Peter Edberg, Deborah Goldsmith, Laurentiu Iancu, Gervase Markham, Simon Montagu, Lisa Moore, Eric Muller, Murray Sargent, Markus Scherer, Jungshik Shin, Shawn Steele, Erik van der Poel, Chris Weber, and Ken Whistler. The specification builds upon [IDNA2008], developed in the IETF Idnabis working group, especially contributions from Matitiahu Allouche, Harald Alvestrand, Vint Cerf, Martin J. Dürst, Lisa Dusseault, Patrik Fältström, Paul Hoffman, Cary Karp, John Klensin, and Peter Resnick, and also upon [IDNA2003], authored by Marc Blanchet, Adam Costello, Patrik Fältström, and Paul Hoffman.

References

References not listed here may be found in http://www.unicode.org/reports/tr41/#UAX41.

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

[IDNA2003] The IDNA2003 specification is defined by a cluster of IETF RFCs: the IDNA base specification [RFC3490], Nameprep [RFC3491], Punycode [RFC3492], and Stringprep [RFC3454].

[IDNA2008] http://tools.ietf.org/id/idnabis

[NFKC_CaseFold] The Unicode property specified in [UAX44], and defined by the data in DerivedNormalizationProps.txt (search for "NFKC_CaseFold").

[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.

[RFC1034] P. Mockapetris. "DOMAIN NAMES - CONCEPTS AND FACILITIES", RFC1034, November 1987
http://tools.ietf.org/html/rfc1034

[RFC3454] P. Hoffman, M. Blanchet. "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.
http://ietf.org/rfc/rfc3454.txt

[RFC3490] Faltstrom, P., Hoffman, P. and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.
http://ietf.org/rfc/rfc3490.txt

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.
http://ietf.org/rfc/rfc3491.txt

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.
http://ietf.org/rfc/rfc3492.txt

[SafeBrowsing] http://code.google.com/apis/safebrowsing/

[Unicode] The Unicode Standard
For the latest version see:
http://www.unicode.org/versions/latest/.

[Versions] Versions of the Unicode Standard
http://www.unicode.org/versions/
For details on the precise contents of each version of the Unicode Standard, and how to cite them.

Modifications

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

Version 2

- Draft 4
- Changed title
- Draft 3
- Added notation section, draft data file (uts46-data-5.1.txt)

- Made it clear that applications can choose to have tighter validity criteria.
- Fixed the names of Sections 5.1 and 5.2
- Added a review note on how this could be extended to registries.
- Draft 2
- Small changes in wording (not typically marked with yellow).
- Additional review notes.
- Removed active links from URLs and domain names: replaced by special style.
- Fixed references.
- Added table of period behavior in 8.2
- Added comparison table of IDNA2003, UTS46, and IDNA2008 in section 5.2
- Draft 1
- Draft UTS posted for public review.
- Radical simplification as directed by the UTC.

Version 1

- Proposed Draft UTS posted for public review.
- Fixed a number of typos and problems pointed out by Marcos (mostly not noted in the text).
- Added draft security and FAQ sections.
- Replaced the introduction, and shortened the document overall; with theNFKC_CaseFolded property, the mapping is considerably simpler.
- Added specifications for the Hybrid and Compatibility implementations, including the two Modes, based on the additional material from the UTC in early 2008.
- Removed the Hybrid variant, and added a discussion of tactics for deviations.

Copyright © 2008-2009 Unicode, Inc. All Rights Reserved. The Unicode Consortium makes no expressed or implied warranty of any kind, and assumes no liability for errors or omissions. No liability is assumed for incidental and consequential damages in connection with or arising out of the use of the information or programs contained or accompanying this technical report. The Unicode Terms of Use apply.

Unicode and the Unicode logo are trademarks of Unicode, Inc., and are registered in some jurisdictions.