L2/10-024

Technical Reports

Draft Unicode Technical Standard #46

UNICODE IDNA COMPATIBILITY PROCESSING

Version	5.2.0 (working draft 6a)
Authors	Mark Davis (markdavis@google.com), Michel Suignard
Date	2010-02-04
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.

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This is a draft document which may be updated, replaced, or superseded by other documents at any time.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 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 was introduced in 2003 for internationalized domain names (IDN). This system is called *Internationalizing Domain Names for Applications*, or IDNA for short. It consists of 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* [RFC3492].

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

In early 2010, a new version of IDNA was approved. Like IDNA2003, this version consists of a collection of RFCs and is called IDNA2008 [IDNA2008]. Despite the presence of "2008" in the name, it was actually approved in 2010.

[Review Note: While approved (except for BIDI) at the time of this writing, they have not yet been issued as RFCs.]

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 8, 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, such as all uppercase characters, all half-width or full-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 change the casing behavior: 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. In other scripts there are variations in characters similar to case in this respect. The IDNA2003 mapping is based on data specified by Unicode; this mapping was later formalized as the Unicode property [NFKC_CaseFold].

IDNA2008 does not require a mapping phase, but does *permit* one (called "Local Mapping" or "Custom Mapping"). There are no limitations on what the mapping can do to disallowed characters. Disallowed characters even include ASCII uppercase characters, if they occur in an IDN label. For more information on the permitted mappings, see the *Protocol* document of [IDNA2008], *Section 4.2 Permitted Character and Label Validation* and

Section 5.2 Conversion to Unicode. An implementation of IDNA2008 which uses the option of Custom Mapping can, in principle, allow any particular mapping. Such mappings can have 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:

[Review note: There will be a review of the final RFC numbers and titles for all IDNA2008 documents and fixes as necessary.]

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 that there is a dotless i in the result of the mapping illustrated in #4. This has the consequence that the mapped IDN resolves to a different location than the mapped IDN in #3.

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

[Review note: As of this time, the mapping document is not final. If the IDNA2010 mapping document is issued, the document will describe why it is not recommended, and that this document presents the recommended alternative for mapping. Otherwise it will delete references to that document.]

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 these characters are common in particular languages. Because of this common occurrence, a significant number of strings for domain names are affected in those languages. This set of characters is referred to as "Deviations". There are four of these, as shown in *Table 1, Deviation Characters*.

Table 1. Deviation Characters

Char	Example	IDNA2003 Result	IDNA2008 Result
ß 00DF	href="http://faß.de"	http://fass.de = http://fass.de	http://faß.de = http://xn--fa-hia.de
ς 03C2	href="http://βόλος.com"	http://βόλοσ.com = http://xn--nxasmq6b.com	http://βόλος.com = http://xn--nxasmm1c.com
ZWJ 200D	href="http://ශ්‍රී.com"	http://ශ්රී.com = http://xn--10cl1a0b.com	http://ශ්‍රී.com = http://xn--10cl1a0b760p.com
ZWNJ 200C	href="http://نامه‌ای.com"	http://نامهای.com = http://xn--mgba3gch31f.com	http://نامه‌ای.com = http://xn--mgba3gch31f060k.com

For more information on the rationale for the occurrence of these Deviations in IDNA2008, see the [IDN FAQ].

The differences in interpretation of Deviation characters results in the potential for security exploits. Consider a scenario involving http://www.sparkasse-gießen.de, a German IDN for "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 01.23.45.67*.*
2. She visits her 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 101.123.145.167, a spoof site.

> Alice ends up at the phishing site, supplies her bank password, and is her money is stolen. While the .DE registar (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. By contrast these are removed by the mapping in IDNA2003. In addition to this difference in mapping, these characters represent a special security concern because they are normally invisible. That is, the sequence "a<ZWJ>b" looks just like "ab". IDNA2008 provides a special category called CONTEXTJ for ZWJ and ZWNJ, and only permits them to occur in certain contexts: certain sequences of Arabic or Indic characters. 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 because of ZWJ and ZWNJ.

2 Unicode IDNA Compatibility Processing

To allow client-side applications to work around the incompatibilities between IDNA2003 and IDNA2008 for lookup, this document provides a Unicode algorithm for 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 is structured according to IDNA2003 principles, but extends those principles to Unicode 5.1 and later. In so doing, it also incorporates the repertoire extensions provided by IDNA2008.

The Unicode IDNA Compatibility Processing uses the standard Unicode mapping, [NFKC_CaseFold], for the mapping described in this document. As a result, the domain name in http://ÖBB.at is valid, and maps to http://öbb.at. It also allows domain names as in http://√.com (which has an associated web page), and are allowed in IDNA2003. Based on security considerations, implementations may restrict or flag (in a UI) domain names that include symbols and punctuation. For more information, see UTR#36: Unicode Security Considerations [UTR36].

The result of this Compatibility Processing is a series of labels, each separated by U+002E (.) FULL STOP. For DNS lookup, the result of the Compatibility Processing is transformed by Punycoding each label that contains non-ASCII.

Using the Unicode IDNA Compatibility Processing to transform an IDN into a form suitable for DNS lookup is comparable to the tactic of "try IDNA2008 then try IDNA2003". However, this approach avoids a dual lookup, which can be very problematic. It allows 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 in future versions of Unicode.

The Unicode IDNA Compatibility Processing does permit the IDNA2008 Subtractions (primarily symbols and punctuation). It also provides alternate mappings for the four Deviation characters. This is to allow for transition periods until the majority of major registries disallow the Subtractions and support either bundling or blocking for the Deviation characters that they support. Note that the term "registries" includes far more than top-level registries, such as for **.de** or **.com**. For example, **.blogspot.com** has more domain names registered than most top-level registries. There may be different policies in place for a registry and any of its subregistries. Thus there are millions of registries that have to be considered in a transition strategy, not hundreds.

In lookup software, transitions may be fine-grained: for example, it may be possible to transition to IDNA2008 Subtractions and/or Deviations for **.blogspot.com** at a given point but not for **.com**, or vice versa. If **.de** bundles or blocks the Deviation characters, then clients could transition Deviations for **.de**, but not for (say) **.blogspot.de**. The discussion of such transition strategies is outside of the scope of this document.

For a demonstration of differences between IDNA2003, IDNA2008, and the Unicode IDNA Compatibility Processing, see the [IDN_Demo]. For more detail on the differences, see *Section 8, IDNA Comparison*.

[Review note: The referenced links to UTR36 et UTS39 (et #31, et al) passim will be fixed.]

There are two slightly different compatibility mechanisms for dealing with two separate tasks: IDNA domain name *lookup* and IDNA domain name *display*. To this end, this document specifies two specific types of processing: Lookup Processing and Display Processing.

Note that neither the Unicode IDNA Compatibility Processing nor IDNA2008 address security problems associated with confusables (the so -called "paypal.com" problem). IDNA2008 does disallow certain symbols and punctuation characters that can be used for spoofing, such as spoofs of the slash character ("/"). These are, however, an extremely small fraction of the confusable characters used for spoofing. Moreover, confusable characters themselves account for a small proportion of fishing problems: most are cases like "secure-wellsfargo.com". For more information, see [Bortzmeyer] and the [IDN FAQ].

This document is not directed at the registration of IDNs, although registries may follow the specifications in this document, with additional repertoire restrictions of their choosing, to support a transitional period.

It is strongly recommended that *UTR#36: Unicode Security Considerations* [UTR36] and *UTS#39: Unicode Security Mechanisms* [UTR39] be consulted for information on dealing with confusables, both for client software and registries. In particular, [UTR39] provides information that can be used to drastically reduce the number of confusables when dealing with international domain names, much beyond what IDNA2008 does. See the [DemoConf].

2.1 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, because 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 for users. Thus in displaying domain names, it is recommended that the Display Processing be applied. This is the same as Lookup Processing, except that it excludes the deviations: ß, ς, and *joiners*.

Labels presented to a browser may or may not be in the display form preferred by a target site; for more information see the [IDN FAQ]. This specification defines a default display algorithm in *Section 4, Processing*.

Except for direct DNS lookup, the Display Processing should always be used. That preserves the four deviation characters in the original string. The Display Processing form is also used for support of Deviation characters after a transitional period.

2.2 Registries

This specification is primarily targeted at applications doing Lookup Processing for IDNs. There is, however, one strong recommendation for registries: *do not allow the registration of labels that are invalid according to Lookup Processing.* The registration of such a label would not be found by browsers and search engines following Unicode IDNA Compatibility Processing.

The label that is actually registered and inserted into a registry, is always a label that has been processed. For example, http://xn--bcher-kva.de which corresponds to http://bücher.de. However, it may be useful for a registry to also ask for "unprocessed" labels as part of the registration process, such as http://Bücher.de, so that they are aware of the registrant's intent. However, such unprocessed labels must be handled carefully:

- Storing the unprocessed label as the sequence of characters that the registrant really wanted to apply for.
- Processing the unprocessed label, and displaying the processed label to the registrant for confirmation.
- Proceeding with the regular registration process using *only* the processed label.

2.3 Notation

Sets of code points are defined using properties and the syntax of *UTS#18: Unicode Regular Expressions* [UTS18]. For example, the set of combining marks is represented by the syntax `\p{gc=M}`. An additional syntactic notation beyond the syntax of UTS#18 is used here: the "+" indicates the addition of elements to a set.

In this document, a *label* is a substring of a domain name. That substring is bounded on both sides by either the start or the end of the string, or any of the following characters, called *label-separators*:

1. U+002E (.) FULL STOP
2. U+FF0E (．) FULLWIDTH FULL STOP
3. U+3002 (。) IDEOGRAPHIC FULL STOP
4. U+FF61 (｡) HALFWIDTH IDEOGRAPHIC FULL STOP

Many people use the terms "domain names" and "host names" interchangeably. This document follows [RFC3490] in use of the term "domain name".

[Review note: bundling and blocking will be explained here, and a reference added on first use.]

3 Conformance

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

C1 Given a version of Unicode and a Unicode String, a conformant implementation of Lookup Processing shall replicate the results given by applying the Lookup Processing algorithm specified by *Section 4, Processing*.

C2 Given a version of Unicode and a Unicode String, a conformant implementation of Display Processing shall replicate the results given by applying the Display Processing algorithm specified by *Section 4, Processing*.

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

Any conformant implementation may also have *tighter* validity criteria than those imposed by *Section 6, Validity Criteria*. For example, an application could disallow or warn of domain name labels with certain characteristics. For example:

- labels with certain combinations of scripts (Safari)
- labels with characters outside of the user's specified languages (IE)
- labels with certain confusable characters (Firefox)
- labels that are detected by the Google Safe Browsing API [SafeBrowsing]
- labels that do not meet the validity requirements of IDNA2008, including BIDI well-formedness
- labels containing characters from *Table 4. Candidate Characters for Exclusion from Identifiers* and *Table 5. Recommended Scripts: Limited Use* from *UAX#31, Unicode Identifier and Pattern Syntax* [UAX31]
- labels that don't satisfy *Restriction Level 3, Moderately Restrictive* from *UTR#36: Unicode Security Considerations* [UTR36]

For more information, see UTR#36: *Unicode Security Considerations* [UTR36] and *UTS#39: Unicode Security Mechanisms* [UTR39].

4 Processing

The input to Unicode IDNA Compatibility Processing is a prospective *domain_name* string expressed in Unicode. The domain name consists of a sequence of labels with dot separators, such as "Bücher.de".

Note: For more information about the composition of a URL, see Section 3.5 of [STD13].

The input *domain_name* string must have had all escaped Unicode code points converted to Unicode code points. For example, U+5341 (十) CJK

UNIFIED IDEOGRAPH-5341 could have been escaped as any of the following:

- &#x5341; an HTML numeric character reference (NCR)
- \u5341 a Javascript escapes
- %E5%8D%81 a URI/IRI %-escape

The following steps, performed in order, successively alter the input *domain_name* string and then output it as a converted Unicode string. The output is a converted Unicode string, plus a flag to indicate whether there was an error. Even if an error occurs, the conversion of the string is performed as much as is possible.

1. For each code point in the *domain_name* string, lookup the status value in *Section 5, IDNA Mapping Table*, and take the following actions:
   - **disallowed**: Record that there was an error.
   - **ignored**: Remove the code point from the string. This is equivalent to mapping the code point to an empty string.
   - **mapped**: Replace the code point in the string by the value for the mapping in *Section 5, IDNA Mapping Table*.
   - **deviation**:
     - For Lookup Processing, replace the code point in the string by the value for the mapping in *Section 5, IDNA Mapping Table*.
     - For Display Processing, leave the code point unchanged in the string.
   - **valid**: Leave the code point unchanged in the string.
2. Normalize the *domain_name* string to Unicode Normalization Form C.
3. For each label in the *domain_name* string
   1. If the label starts with "xn--", attempt to convert the rest of the label to Unicode according to *Punycode* [RFC3492].
   2. If that conversion fails, record that there was an error.

   3. If that conversion succeeds, replace the original label in the string by the results of the conversion.
4. For each label in the domain_name string, verify that it meets the validity criteria in *Section 6, Validity Criteria*. If any of the validity criteria are not satisfied, record that there was an error.

Any input *domain_name* string that does record that there was an error in the application of these steps is valid according to this specification. Conversely, if an input *domain_name* string causes an error, then that input input *domain_name* string is not valid. The processing is idempotent—reapplying the processing to the output will make no further changes. For examples, see *Table 2, Examples of Lookup Processing*.

Implementations may make further modifications to the display string in display to the user. For example, it is recommended that invalid characters be replaced by a U+FFFD so as to make them visible to the user. Similarly, labels that are invalid according to steps 3 or 4 may be marked by the insertion of a U+FFFD or other visual device.

There are two types of processing: Lookup Processing and Display Processing. These differ only in how the deviation code points in the mapping table are handled. The result of Lookup processing can be converted to a domain name string containing Punycode labels ("asciified").

Note: Some browsers allow also characters such as *underscore* ("_") in domain names. Any such extension is outside of the scope of this document.

For those familiar with [RFC3490], operations corresponding to ToASCII and ToUnicode are implemented as follows.

- ToASCII:
  - Apply the Lookup Processing. This may record an error.
  - Break the result into labels at U+002E FULL STOP.
  - Convert each label with non-ASCII characters into Punycode [RFC3492]. This may record an error. If any converted label contains a U+002E FULL STOP, record an error.

  - If requested (an option in [RFC3490]), check that the result is a valid DNS domain name (checking for length restrictions, etc.) according to *Domain names - concepts and facilities* [STD13] and [STD3]. This may record an error. Note that with the provided mapping table, non-LDH ASCII characters will be rejected regardless of whether this option is chosen.
  - If an error was recorded, then the operation failed, and no DNS lookup should be done.
- ToUnicode:
  - Apply the Display Processing. Note that unlike RFC3490 ToASCII, this always signals whether or not there was an error.
  - Like RFC3490, this will always produce a converted Unicode string, even if there was an error.

[Review note: These requirements are somewhat more strict in certain ways than RFC3490 (or IDNA2008). For example, raw Punycode is always validated, where RFC3490 will just pass on failed Punycode in ToUnicode, and IDNA2008 doesn't require checking raw Punycode in lookup at all (a sizable hole in its validity checking).]

Implementations are advised to apply additional tests to these labels such as those described in *UTR#36: Unicode Security Considerations* [UTR36] and *UTS#39: Unicode Security Mechanisms* [UTR39], and take appropriate actions. For example, a label with mixed scripts or confusables may be called out in the UI.

Table 2. Examples of Lookup Processing

Input	Step 1	Step 2	Step 3	Step 4	Comment
Bloß.de	bloss.de	=	=	**valid**	maps uppercase and eszett
u¨.com	=	ü.com	=	**valid**	normalizes u + *umlaut*
xn--tda.com	xn--tda.com	=	ü.com	**valid**	xn--tda = ü

xn--u-ccb.com	=	=	ü.com	***error***	xn--u-ccb = u + *umlaut*
a⒈com	***error***				**⒈ is disallowed**
xn--a-ecp.ru	xn--a-ecp.ru	=	a⒈.ru	***error***	xn--a-ecp = a⒈
xn--a.pt	xn--a.pt	=	***error***		invalid Punycode
日本語。ＪＰ	日本語.jp	=	=	**valid**	mapping full width characters
☐.us	=	=	=	**valid**	post Unicode 3.2 character

4.1 Implementation Notes

There are a number of optimizations can be applied to this processing. These optimizations can improve performance, reduce table size, make use of existing NFKC transform mechanisms, and so on. For example:

- There is an NFC check in *Section 6, Validity Criteria*. However, it only needs to be applied to labels that were converted from Punycode into Unicode in Step 3.
- A simple way to do much of the validity checking in *Section 6, Validity Criteria* is to simply reapply Steps 1 and 2, and verify that the result does not change.
- Because the four label separators are all mapped to U+002E (.) FULL STOP by Step 1, the parsing of labels in Steps 3 and 4 only need to detect U+002E (.) FULL STOP, and not the other label separators defined in IDNA [RFC3490].

5 IDNA Mapping Table

For each code point in Unicode, the IDNA Mapping Table provides a status value. If this status value is **mapped** or **deviation**, the table also supplies a mapping value for that code point. A table is provided for each version of Unicode starting with Unicode 5.1, in versioned directories under [IDNA-Table]. Each table for a version of the Unicode Standard will always be backwards compatible with previous versions of the table: only characters with the status value **disallowed** may change in status or mapping value.

A description of the derivation of these tables is provided in *Section 7, Mapping Table Derivation*. As for derived properties in the Unicode Character Database, the description of the derivation is informative. Only the data in IDNA Mapping Table is normative for the application of this specification.

The files use a semicolon-delimited format similar to those in the Unicode Character Database. The first field is the code point; the second field is the status value; and the third field is the mapping value. Code points are expressed in hexadecimal. The status values are one of the following five values: **valid**, **disallowed**, **ignored**, **mapped**, and **deviation**.

Example:

```
0000..002C    ; disallowed                # NULL..COMMA
002D          ; valid                     # HYPHEN-MINUS
...
0041          ; mapped      ; 0061        # LATIN CAPITAL LETTER A
...
00AD          ; ignored                   # SOFT HYPHEN
...
00DF          ; deviation   ; 0073 0073   # LATIN SMALL LETTER SHARP S
...
```

6 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 a U+002D HYPHEN-MINUS character in both the third position and fourth positions.
3. The label must neither begin nor end with a U+002D HYPHEN-MINUS character.
4. The label must be in Unicode Normalization Form NFC.
5. The label must not contain a U+002E (.) FULL STOP.
6. Each code point in the label must only have certain status values according to *Section 5, IDNA Mapping Table*:
    1. For Lookup Processing, each value must be **valid**.

    2. For Display Processing, each value must be either **valid** or **deviation.**

7. The label must not begin with a combining mark, that is: General_Category=Mark.

In addition, the label *should* meet the requirements for right-to-left characters specified in the Bidi document of [IDNA2008], and for the CONTEXTJ requirements in the Protocol document of [IDNA2008]. It is strongly recommended that *UTR#36: Unicode Security Considerations* [UTR36] and *UTS#39: Unicode Security Mechanisms* [UTR39] be consulted for information on dealing with confusables, and for characters that should be excluded from identifiers. Note that the recommended exclusions are a superset of those in [IDNA2008].

Any particular application *may* have tighter validity criteria, as discussed in *Section 3, Conformance.*

7 Mapping Table Derivation

The following describes the derivation of the mapping table. Step 1 defines a base mapping value; Steps 2-4 define three sets of characters. These are all used in Step 5 to produce the mapping and status values for the table. Step 5 also removes characters whose NFD form would be invalid.

If a Unicode property were to change in a future version in a way that would affect backwards compatibility, a grandfathering clause will be added to maintain compatibility. For more information on compatibility, see *Section 5, IDNA Mapping Table*.

Step 1: Produce a base mapping value

1. Map the following label separator characters to U+002E (.) FULL STOP
    - U+FF0E (．) FULLWIDTH FULL STOP
    - U+3002 (。) IDEOGRAPHIC FULL STOP
    - U+FF61 (｡) HALFWIDTH IDEOGRAPHIC FULL STOP

2. Map each other character to its NFKC_CaseFold value [NFKC_CaseFold].

Step 2: Specify the base valid set

The base valid set is defined by the sequential list of additions and subtractions in *Table 3, Base Valid Set*. This definition is based on the principles of IDNA2003. When applied to the repertoire of Unicode 3.2 characters, this produces a set which is closely aligned with IDNA2003.

Table 3. Base Valid Set

Formal Sets	Descriptions
`[ \P{Changes_When_NFKC_Casefolded}`	Start with characters that are NFKC Case folded (excluding uppercase, for example). Note that \P means the inverse of \p, so these are the characters that *don't* change when individually NFKC_CaseFolded.
`- \p{c} - \p{z}`	Remove Control Characters and Whitespace
`- \p{Block=Ideographic_Description_Characters}`	Remove ideographic description characters
`- \p{ascii}`	Remove ASCII
`+ [\u002D\u002Ea-zA-Z0-9]`	Add back all the valid ASCII, plus U+002E (.) FULL STOP

Step 3: Specify the base exclusion set

The exclusion set consists of characters that have a different mapping in IDNA2003 than the base mapping value specified in Step 1, or that are disallowed in IDNA2003. For more information, see the [IDN FAQ]. For this version, the exclusion set consists of the following:

- 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 (□) 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
  - U+1806 (᠆) MONGOLIAN TODO SOFT HYPHEN
  - U+FFFC ( ) OBJECT REPLACEMENT CHARACTER
  - U+FFFD (�) REPLACEMENT CHARACTER

• Vietnamese Tone Marks
  - U+0340 (̀) COMBINING GRAVE TONE MARK
    U+0341 (́) COMBINING ACUTE TONE MARK

• Bidi Format characters
  - U+200E ( ) LEFT-TO-RIGHT MARK..U+202E ( ) RIGHT-TO-LEFT OVERRIDE

• Invisible operators
  - U+2061 ( ) FUNCTION APPLICATION..U+2063 ( ) INVISIBLE SEPARATOR

• Musical Symbols
  - U+1D173 ( ) MUSICAL SYMBOL BEGIN BEAM..U+1D17A ( ) MUSICAL SYMBOL END PHRASE

• Format Characters **(deprecated)**

-
    - U+206A ( ) INHIBIT SYMMETRIC SWAPPING..U+206F ( ) NOMINAL DIGIT SHAPES
- Tags **(deprecated)**
    - U+E0001 ( ) LANGUAGE TAG
    - U+E0020 ( ) TAG SPACE..U+E007F ( ) CANCEL TAG

Step 4: Specify the deviation set

This is the set of characters that deviate between IDNA2003 and IDNA2008.

- 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

Step 5: Produce the status and mapping values for the table

For each code point:

1. If the code point is in the deviation set
    - the status is **deviation** and the mapping value is the base mapping value for that code point
2. Otherwise, if (a) the code point is in the base exclusion set, or if (b) 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 in the table
3. Otherwise, if the base mapping value is an empty string
    - the status is **ignored** and there is no mapping value in the table
4. Otherwise, if the base mapping value is the same as the code point
    - the status is **valid** and there is no mapping value in the table
5. Otherwise,
    - the status is **mapped** and the mapping value is the base mapping value for that code point

After processing all code points, iterate through the **valid** set of characters and remove any whose canonical decompositions (NFD) are not wholy in the **valid** set. Recursively apply this action until there are no removals. In Unicode 5.2, the set of characters so removed consist of the following:

- U+2260 (≠) NOT EQUAL TO
- U+226E (≮) NOT LESS-THAN
- U+226F (≯) NOT GREATER-THAN

Note that characters such as U+2488 (⒈) DIGIT ONE FULL STOP are disallowed by substep (2a).

8 IDNA Comparison

Table 4, IDNA Comparisons illustrates the differences between the three specifications in terms of valid character repertoire. It omits the ASCII-repertoire code points, all code points unassigned in Unicode 5.2, as well as control characters, private-use characters, and surrogate code points. It also includes labels separators that are valid or mapped. The table has separate groupings for Unicode 3.2 (the only characters valid in IDNA2003) and beyond. It also separates buckets where UTS46 and IDNA2008 behave the same from those where they behave differently.

Each row in the table defines a bucket of code points that share a pattern of behavior across the three specifications. The columns provide the following information:

- The Count column shows the number of characters in each bucket.
- The IDNA2003, UTS46, and IDNA2008 columns show the status of the characters in each bucket for the respective specifications.
  - Deviations are modified in lookup processing, but not modified in display processing; see *Section 4, Processing*.
  - IDNA2003 allows unassigned code points in lookup but not registration. These are in the section of the table marked "Unicode v4.0-5.2", and marked as LookupValid.

  - IDNA2008 has several statuses. Characters that are DISALLOWED or UNASSIGNED in IDNA2008 are marked as Disallowed, while characters that are CONTEXTJ, CONTEXTO, and PVALID are marked as Valid.
- The Comments and Samples column describes the nature of the correlation between the specifications and provides illustrative characters.

Table 4. IDNA Comparisons

Count	IDNA2003	UTS46	IDNA2008	Comments and Samples
Unicode v3.2 (IDNA2003 = UTS46 = IDNA2008)				
86,676	Valid	Valid	Valid	*Valid in all three systems* U+00E0 (à) LATIN SMALL LETTER A WITH GRAVE
433	Disallowed	Disallowed	Disallowed	*Disallowed in all three systems* U+FF01 (！) FULLWIDTH EXCLAMATION MARK
Unicode v3.2 (IDNA2003 ≠ UTS46 = IDNA2008)				
48	Valid	Disallowed	Disallowed	*Mappings changed after v3.2* U+2132 (Ⅎ) TURNED CAPITAL F
8	Mapped	Disallowed	Disallowed	*Mappings changed after v3.2* U+2F868 (㛼) CJK COMP.
Unicode v3.2 (IDNA2003 = UTS46 ≠ IDNA2008)				
4,638	Mapped / Ignored	Mapped / Ignored	Disallowed	*Case and compatibility variants, default ignorables* U+00C0 (À) LATIN CAPITAL LETTER A WITH GRAVE
3,254	Valid	Valid	Disallowed	*Punctuation, Symbols, etc.* U+2665 (♥) BLACK HEART SUIT

4	Mapped / Ignored	Mapped / Ignored	Valid	***Deviations*** 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
Unicode v4.0-5.2 (UTS46 = IDNA2008)				
9,705	LookupValid	Valid	Valid	U+0221 (ȡ) LATIN SMALL LETTER D WITH CURL
52	LookupValid	Disallowed	Disallowed	U+0602 (؂) ARABIC FOOTNOTE MARKER
Unicode v4.0-5.2 (UTS46 ≠ IDNA2008)				
1,592	LookupValid	Valid	Disallowed	U+2615 (□) HOT BEVERAGE
791	LookupValid	Mapped / Ignored	Disallowed	U+023A (Ⱥ) LATIN CAPITAL LETTER A WITH STROKE

A more detailed online listing of differences is found at [DemoIDNChars] and [DemoIDN].

[Review Note: The rows with counts above will be numbered for reference. An explanation of the differences and implications for implementers will be added here. Recommendations: everyone should do a/b. Avoid c/d : problematic even in 2003 because implementations handled them differently. efg (mapping same), ...]

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.

[Bortzmeyer]	http://www.bortzmeyer.org/idn-et-phishing.html (machine translated at http://translate.google.com/translate?u=http%3A%2F%2Fwww.bortzmeyer.org%2Fidn-et-phishing.html)
[DemoConf]	http://unicode.org/cldr/utility/confusables.jsp
[DemoIDN]	http://unicode.org/cldr/utility/idna.jsp
[DemoIDNChars]	http://unicode.org/cldr/utility/list-unicodeset.jsp?a=\p{age%3D3.2}-\p{cn}-\p{cs}-\p{co}&abb=on&g=uts46+idna+idna2008
[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: • IDNA [RFC3490] • Nameprep [RFC3491] • Punycode [RFC3492] • Stringprep [RFC3454].
[IDNA2008]	The draft IDNA2008 specification is defined by a cluster of IETF RFCs: • Internationalized Domain Names for Applications (IDNA): Definitions and

Document Framework – http://tools.ietf.org/html/draft-ietf-idnabis-defs

- Internationalized Domain Names in Applications (IDNA): Protocol – http://tools.ietf.org/html/draft-ietf-idnabis-protocol
- The Unicode code points and IDNA – http://tools.ietf.org/html/draft-ietf-idnabis-tables
- Right-to-left scripts for IDNA – http://tools.ietf.org/html/draft-ietf-idnabis-bidi

There are also two informative documents:

- Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale – http://tools.ietf.org/html/draft-ietf-idnabis-rationale
- Mapping Characters in IDNA – http://tools.ietf.org/html/draft-ietf-idnabis-mappings

For more information, see http://tools.ietf.org/id/idnabis.

*[**Review Note**: Once IDNA2008 is final, and the final formal titles and RFC numbers will be used in references in the text.]*

[IDNA-Table] http://www.unicode.org/Public/idna

[IDN-Demo] http://unicode.org/cldr/utility/idna.jsp

[IDN-FAQ] http://www.unicode.org/faq/idn.html

[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.
[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/
[STD3]	Braden, R., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, and "Requirements for Internet Hosts -- Application and Support", STD 3, RFC 1123, October 1989. http://www.rfc-editor.org/std/std3.txt
[STD13]	Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034 and "Domain names - implementation and specification", STD 13, RFC 1035, November 1987. http://www.rfc-editor.org/std/std13.txt

[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.

Revision 2

- **Draft 6**
- Made modifications resulting from UTC discussion.
- Made it clear that Subtraction and Deviation support is transitional.
- Some other rewording after the approval of IDNA2008. Final wording will await assignment of RFC numbers.
- Added explanation of correspondances to ToASCII and ToUnicode.
- Modified the error handling to make it more flexible, and always produce a (determinant) converted string.
- Added some open issues.
- **Draft 5 and previous**
- Draft UTS posted for public review
- Incorporated Editorial Committee modifications
- Changed title
- Major restructuring as result of UTC discussion.
- Added notation section, draft data file
- Made it clear that applications can choose to have tighter validity criteria.
- Renumbered sections
- Fixed references.

- Added comparison table of IDNA2003, UTS46, and IDNA2008 in Section 8
- Radical simplification as directed by the UTC.

Revision 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.