Internet Engineering Task Force (IETF)
Request for Comments: 5891
Obsoletes: 3490, 3491
Updates: 3492
Category: Standards Track
ISSN: 2070-1721

J. Klensin

August 2010

Internationalized Domain Names in Applications (IDNA): Protocol

Abstract

This document is the revised protocol definition for Internationalized Domain Names (IDNs). The rationale for changes, the relationship to the older specification, and important terminology are provided in other documents. This document specifies the protocol mechanism, called Internationalized Domain Names in Applications (IDNA), for registering and looking up IDNs in a way that does not require changes to the DNS itself. IDNA is only meant for processing domain names, not free text.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5891.

Table of Contents

1. Introduction

This document supplies the protocol definition for Internationalized Domain Names in Applications (IDNA), with the version specified here known as IDNA2008. Essential definitions and terminology for understanding this document and a road map of the collection of documents that make up IDNA2008 appear in a separate Definitions document [RFC5890]. Appendix A discusses the relationship between this specification and the earlier version of IDNA (referred to here as "IDNA2003"). The rationale for these changes, along with considerable explanatory material and advice to zone administrators who support IDNs, is provided in another document, known informally in this series as the "Rationale document" [RFC5894].

IDNA works by allowing applications to use certain ASCII [ASCII] string labels (beginning with a special prefix) to represent non-ASCII name labels. Lower-layer protocols need not be aware of this; therefore, IDNA does not change any infrastructure. In particular, IDNA does not depend on any changes to DNS servers, resolvers, or DNS protocol elements, because the ASCII name service provided by the existing DNS can be used for IDNA.

IDNA applies only to a specific subset of DNS labels. The base DNS standards [RFC1034] [RFC1035] and their various updates specify how to combine labels into fully-qualified domain names and parse labels out of those names.

This document describes two separate protocols, one for IDN registration (Section 4) and one for IDN lookup (Section 5). These two protocols share some terminology, reference data, and operations.

2. Terminology

As mentioned above, terminology used as part of the definition of IDNA appears in the Definitions document [RFC5890]. It is worth noting that some of this terminology overlaps with, and is consistent with, that used in Unicode or other character set standards and the DNS. Readers of this document are assumed to be familiar with the associated Definitions document and with the DNS-specific terminology in RFC 1034 [RFC1034].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

3. Requirements and Applicability

3.1. Requirements

IDNA makes the following requirements:

1. Whenever a domain name is put into a domain name slot that is not IDNA-aware (see Section 2.3.2.6 of the Definitions document [RFC5890]), it MUST contain only ASCII characters (i.e., its labels must be either A-labels or NR-LDH labels), unless the DNS application is not subject to historical recommendations for "hostname"-style names (see RFC 1034 [RFC1034] and Section 3.2.1).

2. Labels MUST be compared using equivalent forms: either both A-label forms or both U-label forms. Because A-labels and U-labels can be transformed into each other without loss of information, these comparisons are equivalent (however, in practice, comparison of U-labels requires first verifying that they actually are U-labels and not just Unicode strings). A pair of A-labels MUST be compared as case-insensitive ASCII (as with all comparisons of ASCII DNS labels). U-labels MUST be compared as-is, without case folding or other intermediate steps. While it is not necessary to validate labels in order to compare them, successful comparison does not imply validity. In many cases, not limited to comparison, validation may be important for other reasons and SHOULD be performed.

3. Labels being registered MUST conform to the requirements of Section 4. Labels being looked up and the lookup process MUST conform to the requirements of Section 5.

3.2. Applicability

IDNA applies to all domain names in all domain name slots in protocols except where it is explicitly excluded. It does not apply to domain name slots that do not use the LDH syntax rules as described in the Definitions document [RFC5890].

Because it uses the DNS, IDNA applies to many protocols that were specified before it was designed. IDNs occupying domain name slots in those older protocols MUST be in A-label form until and unless those protocols and their implementations are explicitly upgraded to be aware of IDNs and to accept the U-label form. IDNs actually appearing in DNS queries or responses MUST be A-labels.

IDNA-aware protocols and implementations MAY accept U-labels, A-labels, or both as those particular protocols specify. IDNA is not defined for extended label types (see RFC 2671 [RFC2671], Section 3).

3.2.1. DNS Resource Records

IDNA applies only to domain names in the NAME and RDATA fields of DNS resource records whose CLASS is IN. See the DNS specification [RFC1035] for precise definitions of these terms.

The application of IDNA to DNS resource records depends entirely on the CLASS of the record, and not on the TYPE except as noted below. This will remain true, even as new TYPEs are defined, unless a new TYPE defines TYPE-specific rules. Special naming conventions for SRV records (and "underscore labels" more generally) are incompatible with IDNA coding as discussed in the Definitions document [RFC5890], especially Section 2.3.2.3. Of course, underscore labels may be part of a domain that uses IDN labels at higher levels in the tree.

3.2.2. Non-Domain-Name Data Types Stored in the DNS

Although IDNA enables the representation of non-ASCII characters in domain names, that does not imply that IDNA enables the representation of non-ASCII characters in other data types that are stored in domain names, specifically in the RDATA field for types that have structured RDATA format. For example, an email address local part is stored in a domain name in the RNAME field as part of the RDATA of an SOA record (e.g., hostmaster@example.com would be represented as hostmaster.example.com). IDNA does not update the existing email standards, which allow only ASCII characters in local parts. Even though work is in progress to define internationalization for email addresses [RFC4952], changes to the email address part of the SOA RDATA would require action in, or updates to, other standards, specifically those that specify the format of the SOA RR.

4. Registration Protocol

This section defines the model for registering an IDN. The model is implementation independent; any sequence of steps that produces exactly the same result for all labels is considered a valid implementation.

Note that, while the registration (this section) and lookup protocols (Section 5) are very similar in most respects, they are not identical, and implementers should carefully follow the steps described in this specification.

4.1. Input to IDNA Registration

Registration processes, especially processing by entities (often called "registrars") who deal with registrants before the request actually reaches the zone manager ("registry") are outside the scope of this definition and may differ significantly depending on local needs. By the time a string enters the IDNA registration process as described in this specification, it MUST be in Unicode and in Normalization Form C (NFC [Unicode-UAX15]). Entities responsible for zone files ("registries") MUST accept only the exact string for which registration is requested, free of any mappings or local adjustments. They MAY accept that input in any of three forms:

1. As a pair of A-label and U-label.

2. As an A-label only.

3. As a U-label only.

The first two of these forms are RECOMMENDED because the use of A-labels avoids any possibility of ambiguity. The first is normally preferred over the second because it permits further verification of user intent (see Section 4.2.1).

4.2. Permitted Character and Label Validation

4.2.1. Input Format

If both the U-label and A-label forms are available, the registry MUST ensure that the A-label form is in lowercase, perform a conversion to a U-label, perform the steps and tests described below on that U-label, and then verify that the A-label produced by the step in Section 4.4 matches the one provided as input. In addition, the U-label that was provided as input and the one obtained by conversion of the A-label MUST match exactly. If, for some reason, these tests fail, the registration MUST be rejected.

If only an A-label was provided and the conversion to a U-label is not performed, the registry MUST still verify that the A-label is superficially valid, i.e., that it does not violate any of the rules of Punycode encoding [RFC3492] such as the prohibition on trailing hyphen-minus, the requirement that all characters be ASCII, and so on. Strings that appear to be A-labels (e.g., they start with "xn--") and strings that are supplied to the registry in a context reserved for A-labels (such as a field in a form to be filled out), but that are not valid A-labels as described in this paragraph, MUST NOT be placed in DNS zones that support IDNA.

If only an A-label is provided, the conversion to a U-label is not performed, but the superficial tests described in the previous paragraph are performed, registration procedures MAY, and usually will, bypass the tests and actions in the balance of Section 4.2 and in Sections 4.3 and 4.4.

4.2.2. Rejection of Characters That Are Not Permitted

The candidate Unicode string MUST NOT contain characters that appear in the "DISALLOWED" and "UNASSIGNED" lists specified in the Tables document [RFC5892].

4.2.3. Label Validation

The proposed label (in the form of a Unicode string, i.e., a string that at least superficially appears to be a U-label) is then examined using tests that require examination of more than one character. Character order is considered to be the on-the-wire order. That order may not be the same as the display order.

4.2.3.1. Hyphen Restrictions

The Unicode string MUST NOT contain "--" (two consecutive hyphens) in the third and fourth character positions and MUST NOT start or end with a "-" (hyphen).

4.2.3.2. Leading Combining Marks

The Unicode string MUST NOT begin with a combining mark or combining character (see The Unicode Standard, Section 2.11 [Unicode] for an exact definition).

4.2.3.3. Contextual Rules

The Unicode string MUST NOT contain any characters whose validity is context-dependent, unless the validity is positively confirmed by a contextual rule. To check this, each code point identified as CONTEXTJ or CONTEXTO in the Tables document [RFC5892] MUST have a non-null rule. If such a code point is missing a rule, the label is invalid. If the rule exists but the result of applying the rule is negative or inconclusive, the proposed label is invalid.

4.2.3.4. Labels Containing Characters Written Right to Left

If the proposed label contains any characters from scripts that are written from right to left, it MUST meet the Bidi criteria [RFC5893].

4.2.4. Registration Validation Requirements

Strings that contain at least one non-ASCII character, have been produced by the steps above, whose contents pass all of the tests in Section 4.2.3, and are 63 or fewer characters long in ASCII-compatible encoding (ACE) form (see Section 4.4), are U-labels.

To summarize, tests are made in Section 4.2 for invalid characters, invalid combinations of characters, for labels that are invalid even if the characters they contain are valid individually, and for labels that do not conform to the restrictions for strings containing right-to-left characters.

4.3. Registry Restrictions

In addition to the rules and tests above, there are many reasons why a registry could reject a label. Registries at all levels of the DNS, not just the top level, are expected to establish policies about label registrations. Policies are likely to be informed by the local languages and the scripts that are used to write them and may depend on many factors including what characters are in the label (for example, a label may be rejected based on other labels already registered). See the Rationale document [RFC5894], Section 3.2, for further discussion and recommendations about registry policies.

The string produced by the steps in Section 4.2 is checked and processed as appropriate to local registry restrictions. Application of those registry restrictions may result in the rejection of some labels or the application of special restrictions to others.

4.4. Punycode Conversion

The resulting U-label is converted to an A-label (defined in Section 2.3.2.1 of the Definitions document [RFC5890]). The A-label is the encoding of the U-label according to the Punycode algorithm [RFC3492] with the ACE prefix "xn--" added at the beginning of the string. The resulting string must, of course, conform to the length limits imposed by the DNS. This document does not update or alter the Punycode algorithm specified in RFC 3492 in any way. RFC 3492 does make a non-normative reference to the information about the value and construction of the ACE prefix that appears in RFC 3490 or Nameprep [RFC3491]. For consistency and reader convenience, IDNA2008 effectively updates that reference to point to this document. That change does not alter the prefix itself. The prefix, "xn--", is the same in both sets of documents.

With the exception of the maximum string length test on Punycode output, the failure conditions identified in the Punycode encoding procedure cannot occur if the input is a U-label as determined by the steps in Sections 4.1 through 4.3 above.

4.5. Insertion in the Zone

The label is registered in the DNS by inserting the A-label into a zone.

5. Domain Name Lookup Protocol

Lookup is different from registration and different tests are applied on the client. Although some validity checks are necessary to avoid serious problems with the protocol, the lookup-side tests are more permissive and rely on the assumption that names that are present in the DNS are valid. That assumption is, however, a weak one because the presence of wildcards in the DNS might cause a string that is not actually registered in the DNS to be successfully looked up.

5.1. Label String Input

The user supplies a string in the local character set, for example, by typing it, clicking on it, or copying and pasting it from a resource identifier, e.g., a Uniform Resource Identifier (URI) [RFC3986] or an Internationalized Resource Identifier (IRI) [RFC3987], from which the domain name is extracted. Alternately, some process not directly involving the user may read the string from a file or obtain it in some other way. Processing in this step and the one specified in Section 5.2 are local matters, to be accomplished prior to actual invocation of IDNA.

5.2. Conversion to Unicode

The string is converted from the local character set into Unicode, if it is not already in Unicode. Depending on local needs, this conversion may involve mapping some characters into other characters as well as coding conversions. Those issues are discussed in the mapping-related sections (Sections 4.2, 4.4, 6, and 7.3) of the Rationale document [RFC5894] and in the separate Mapping document [IDNA2008-Mapping]. The result MUST be a Unicode string in NFC form.

5.3. A-label Input

If the input to this procedure appears to be an A-label (i.e., it starts in "xn--", interpreted case-insensitively), the lookup application MAY attempt to convert it to a U-label, first ensuring that the A-label is entirely in lowercase (converting it to lowercase

if necessary), and apply the tests of Section 5.4 and the conversion of Section 5.5 to that form. If the label is converted to Unicode (i.e., to U-label form) using the Punycode decoding algorithm, then the processing specified in those two sections MUST be performed, and the label MUST be rejected if the resulting label is not identical to the original. See Section 8.1 of the Rationale document [RFC5894] for additional discussion on this topic.

Conversion from the A-label and testing that the result is a U-label SHOULD be performed if the domain name will later be presented to the user in native character form (this requires that the lookup application be IDNA-aware). If those steps are not performed, the lookup process SHOULD at least test to determine that the string is actually an A-label, examining it for the invalid formats specified in the Punycode decoding specification. Applications that are not IDNA-aware will obviously omit that testing; others MAY treat the string as opaque to avoid the additional processing at the expense of providing less protection and information to users.

5.4. Validation and Character List Testing

As with the registration procedure described in Section 4, the Unicode string is checked to verify that all characters that appear in it are valid as input to IDNA lookup processing. As discussed above and in the Rationale document [RFC5894], the lookup check is more liberal than the registration one. Labels that have not been fully evaluated for conformance to the applicable rules are referred to as "putative" labels as discussed in Section 2.3.2.1 of the Definitions document [RFC5890]. Putative U-labels with any of the following characteristics MUST be rejected prior to DNS lookup:

- Labels that are not in NFC [Unicode-UAX15].
- Labels containing "--" (two consecutive hyphens) in the third and fourth character positions.
- Labels whose first character is a combining mark (see The Unicode Standard, Section 2.11 [Unicode]).
- Labels containing prohibited code points, i.e., those that are assigned to the "DISALLOWED" category of the Tables document [RFC5892].
- Labels containing code points that are identified in the Tables document as "CONTEXTJ", i.e., requiring exceptional contextual rule processing on lookup, but that do not conform to those rules. Note that this implies that a rule must be defined, not null: a

character that requires a contextual rule but for which the rule is null is treated in this step as having failed to conform to the rule.

- Labels containing code points that are identified in the Tables document as "CONTEXTO", but for which no such rule appears in the table of rules. Applications resolving DNS names or carrying out equivalent operations are not required to test contextual rules for "CONTEXTO" characters, only to verify that a rule is defined (although they MAY make such tests to provide better protection or give better information to the user).

- Labels containing code points that are unassigned in the version of Unicode being used by the application, i.e., in the UNASSIGNED category of the Tables document.

  This requirement means that the application must use a list of unassigned characters that is matched to the version of Unicode that is being used for the other requirements in this section. It is not required that the application know which version of Unicode is being used; that information might be part of the operating environment in which the application is running.

In addition, the application SHOULD apply the following test.

- Verification that the string is compliant with the requirements for right-to-left characters specified in the Bidi document [RFC5893].

This test may be omitted in special circumstances, such as when the lookup application knows that the conditions are enforced elsewhere, because an attempt to look up and resolve such strings will almost certainly lead to a DNS lookup failure except when wildcards are present in the zone. However, applying the test is likely to give much better information about the reason for a lookup failure -- information that may be usefully passed to the user when that is feasible -- than DNS resolution failure information alone.

For all other strings, the lookup application MUST rely on the presence or absence of labels in the DNS to determine the validity of those labels and the validity of the characters they contain. If they are registered, they are presumed to be valid; if they are not, their possible validity is not relevant. While a lookup application may reasonably issue warnings about strings it believes may be problematic, applications that decline to process a string that conforms to the rules above (i.e., does not look it up in the DNS) are not in conformance with this protocol.

5.5. Punycode Conversion

The string that has now been validated for lookup is converted to ACE form by applying the Punycode algorithm to the string and then adding the ACE prefix ("xn--").

5.6. DNS Name Resolution

The A-label resulting from the conversion in Section 5.5 or supplied directly (see Section 5.3) is combined with other labels as needed to form a fully-qualified domain name that is then looked up in the DNS, using normal DNS resolver procedures. The lookup can obviously either succeed (returning information) or fail.

6. Security Considerations

Security Considerations for this version of IDNA are described in the Definitions document [RFC5890], except for the special issues associated with right-to-left scripts and characters. The latter are discussed in the Bidi document [RFC5893].

In order to avoid intentional or accidental attacks from labels that might be confused with others, special problems in rendering, and so on, the IDNA model requires that registries exercise care and thoughtfulness about what labels they choose to permit. That issue is discussed in Section 4.3 of this document which, in turn, points to a somewhat more extensive discussion in the Rationale document [RFC5894].

7. IANA Considerations

IANA actions for this version of IDNA are specified in the Tables document [RFC5892] and discussed informally in the Rationale document [RFC5894]. The components of IDNA described in this document do not require any IANA actions.

8. Contributors

While the listed editor held the pen, the original versions of this document represent the joint work and conclusions of an ad hoc design team consisting of the editor and, in alphabetic order, Harald Alvestrand, Tina Dam, Patrik Faltstrom, and Cary Karp. This document draws significantly on the original version of IDNA [RFC3490] both conceptually and for specific text. This second-generation version would not have been possible without the work that went into that first version and especially the contributions of its authors Patrik Faltstrom, Paul Hoffman, and Adam Costello. While Faltstrom was

actively involved in the creation of this version, Hoffman and Costello were not and should not be held responsible for any errors or omissions.

9. Acknowledgments

This revision to IDNA would have been impossible without the accumulated experience since RFC 3490 was published and resulting comments and complaints of many people in the IETF, ICANN, and other communities (too many people to list here). Nor would it have been possible without RFC 3490 itself and the efforts of the Working Group that defined it. Those people whose contributions are acknowledged in RFC 3490, RFC 4690 [RFC4690], and the Rationale document [RFC5894] were particularly important.

Specific textual changes were incorporated into this document after suggestions from the other contributors, Stephane Bortzmeyer, Vint Cerf, Lisa Dusseault, Paul Hoffman, Kent Karlsson, James Mitchell, Erik van der Poel, Marcos Sanz, Andrew Sullivan, Wil Tan, Ken Whistler, Chris Wright, and other WG participants and reviewers including Martin Duerst, James Mitchell, Subramanian Moonesamy, Peter Saint-Andre, Margaret Wasserman, and Dan Winship who caught specific errors and recommended corrections. Special thanks are due to Paul Hoffman for permission to extract material to form the basis for Appendix A from a draft document that he prepared.

10. References

10.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC5893] Alvestrand, H., Ed. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, August 2010.

[Unicode-UAX15] The Unicode Consortium, "Unicode Standard Annex #15: Unicode Normalization Forms", September 2009, <http://www.unicode.org/reports/tr15/>.

10.2. Informative References

[ASCII] American National Standards Institute (formerly United States of America Standards Institute), "USA Code for Information Interchange", ANSI X3.4-1968, 1968. ANSI X3.4-1968 has been replaced by newer versions with slight modifications, but the 1968 version remains definitive for the Internet.

[IDNA2008-Mapping] Resnick, P. and P. Hoffman, "Mapping Characters in Internationalized Domain Names for Applications (IDNA)", Work in Progress, April 2010.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC4690] Klensin, J., Faltstrom, P., Karp, C., and IAB, "Review and Recommendations for Internationalized Domain Names (IDNs)", RFC 4690, September 2006.

[RFC4952] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 4952, July 2007.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, August 2010.

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 5.0", 2007. Boston, MA, USA: Addison-Wesley. ISBN 0-321-48091-0. This printed reference has now been updated online to reflect additional code points. For code points, the reference at the time this document was published is to Unicode 5.2.

Appendix A. Summary of Major Changes from IDNA2003

1. Update base character set from Unicode 3.2 to Unicode version agnostic.

2. Separate the definitions for the "registration" and "lookup" activities.

3. Disallow symbol and punctuation characters except where special exceptions are necessary.

4. Remove the mapping and normalization steps from the protocol and have them, instead, done by the applications themselves, possibly in a local fashion, before invoking the protocol.

5. Change the way that the protocol specifies which characters are allowed in labels from "humans decide what the table of code points contains" to "decision about code points are based on Unicode properties plus a small exclusion list created by humans".

6. Introduce the new concept of characters that can be used only in specific contexts.

7. Allow typical words and names in languages such as Dhivehi and Yiddish to be expressed.

8. Make bidirectional domain names (delimited strings of labels, not just labels standing on their own) display in a less surprising fashion, whether they appear in obvious domain name contexts or as part of running text in paragraphs.

9. Remove the dot separator from the mandatory part of the protocol.

10. Make some currently valid labels that are not actually IDNA labels invalid.

Author’s Address

John C Klensin
1770 Massachusetts Ave, Ste 322
Cambridge, MA 02140
USA

Phone: +1 617 245 1457
EMail: john+ietf@jck.com