Network Working Group
Internet-Draft
Obsoletes: 3490 (if approved)
Intended status: Standards Track
Expires: January 28, 2009

J. Klensin
July 27, 2008

Internationalized Domain Names in Applications (IDNA): Protocol
draft-ietf-idnabis-protocol-03.txt

Status of this Memo

Abstract

This document supplies the protocol definition for a revised and updated specification for internationalized domain names (IDNs). The rationale for these changes, the relationship to the older specification, and important terminology are provided in other documents. This document specifies the protocol mechanism, called Internationalizing 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.

Table of Contents

1. Introduction

This document supplies the protocol definition for a revised and updated specification for internationalized domain names. The rationale for these changes and relationship to the older specification and some new terminology is provided in other documents, notably [IDNA2008-Rationale].

IDNA works by allowing applications to use certain 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 depend on changes to any infrastructure. In particular, IDNA does not depend on any changes to DNS servers, resolvers, or protocol elements, because the ASCII name service provided by the existing DNS is entirely sufficient for IDNA.

IDNA is applied only to DNS labels. Standards for combining labels into fully-qualified domain names and parsing labels out of those names are covered in the base DNS standards [RFC1035]. An application may, of course, apply locally-appropriate conventions to the presentation forms of domain names as discussed in [IDNA2008-Rationale].

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

A good deal of the background material that appeared in RFC 3490 has been removed from this update. That material is either of historical interest only or has been covered from a more recent perspective in RFC 4690 [RFC4690] and [IDNA2008-Rationale].

[[anchor2: Note in Draft: This document still needs more specifics about how to perform some of the tests in the Registration and Lookup protocols described below. Those details will be supplied in a later revision, but the intent should be clear from the existing text.]]

1.1. Discussion Forum

[[anchor4: RFC Editor: please remove this section.]]

This work is being discussed in the IETF IDNABIS WG and on the mailing list idna-update@alvestrand.no

2. Terminology

General terminology applicable to IDNA, but with meanings familiar to

those who have worked with Unicode or other character set standards and the DNS, appears in [IDNA2008-Rationale]. Terminology that is an integral, normative, part of the IDNA definition, including the definitions of "ACE", appears in that document as well. Familiarity with the terminology materials in that document is assumed for reading this one. The reader of this document is assumed to be familiar with DNS-specific terminology as defined 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 conformance means adherence to the following requirements:

1. Whenever a domain name is put into an IDN-unaware domain name slot (see Section 2 and [IDNA2008-Rationale]), it MUST contain only ASCII characters (i.e., must be either an A-label or an LDH-label), or must be a label associated with a DNS application that is not subject to either IDNA or the historical recommendations for "hostname"-style names [RFC1034].

2. Comparison of labels MUST be done on the A-label form, using an ASCII case-insensitive comparison as with all comparisons of DNS labels.

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 is applicable to all domain names in all domain name slots except where it is explicitly excluded. It is not applicable to domain name slots which do not use the LDH syntax rules.

This implies that IDNA is applicable to many protocols that predate IDNA. Note that IDNs occupying domain name slots in those older protocols MUST be in A-label form until and unless those protocols and implementations of them are upgraded.

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.

There are currently no other exclusions on the applicability of IDNA to DNS resource records. Applicability depends entirely on the CLASS, and not on the TYPE except as noted below. This will remain true, even as new types are defined, unless there is a compelling reason for a new type that requires type-specific rules. The special naming conventions applicable to SRV records are examples of type-specific rules that are incompatible with IDNA coding. Hence the first two labels (the ones required to start in "_") on a record with TYPE SRV MUST NOT be A-labels or U-labels (while it would be possible to write a non-ASCII string with a leading underscore, conversion to an A-label would be impossible without loss of information because the underscore is not a letter, digit, or hyphen). Of course, those 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 (hostmaster@example.com would be represented as hostmaster.example.com). IDNA specifically 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 other standards, specifically those that specify the format of the SOA RR.

4. Registration Protocol

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

4.1. Proposed label

The registrant submits a request for an IDN. The user typically produces the request string by the keyboard entry of a character

sequence in the local native character set (which might, of course, be Unicode). The registry MAY permit submission of labels in A-label form. If it does so, it SHOULD perform a conversion to a U-label, perform the steps and tests described below, and verify that the A-label produced by the step in Section 4.5 matches the one provided as input. If, for some reason, it does not, the registration MUST be rejected.
[[anchor9: Editorial: Should the sentences starting with "The registry" be moved to 4.3? I.e., would they be more in sequence there?]]

4.2. Conversion to Unicode and Normalization

Some system routine, or a localized front-end to the IDNA process, ensures that the proposed label is a Unicode string or converts it to one as appropriate. That string MUST be in Unicode Normalization Form C (NFC [Unicode-UAX15]).

As a local implementation choice, the implementation MAY choose to map some forbidden characters to permitted characters (for instance mapping uppercase characters to lowercase ones), displaying the result to the user, and allowing processing to continue. However, it is strongly recommended that, to avoid any possible ambiguity, entities responsible for zone files ("registries") accept registrations only for A-labels (to be converted to U-labels by the registry) or U-labels actually produced from A-labels, not forms expected to be converted by some other process.

4.3. Permitted Character and Label Validation

4.3.1. Rejection of Characters that are not Permitted

The Unicode string is checked to verify that no characters that IDNA does not permit in input appear in it. Those characters are identified in the "DISALLOWED" and "UNASSIGNED" lists that are discussed in [IDNA2008-Rationale]. The normative rules for producing that list and the initial version of it are specified in [IDNA2008-Tables]. Characters that are either DISALLOWED or UNASSIGNED MUST NOT be part of labels being processed for registration in the DNS.

4.3.2. Label Validation

The proposed label (in the form of a Unicode string, i.e., a putative U-label) is then examined, performing tests that require examination of more than one character.

4.3.2.1. Rejection of Confusing or Hostile Sequences in U-labels

The Unicode string MUST NOT contain "--" (two consecutive hyphens) in the third and fourth character positions.

4.3.2.2. Leading Combining Marks

The first character of the string is examined to verify that it is not a combining mark. If it is a combining mark, the string MUST NOT be registered.

4.3.2.3. Contextual Rules

Each code point is checked for its identification as characters requiring contextual processing for registration (the list of characters appears as the combination of CONTEXTJ and CONTEXTO in [IDNA2008-Tables]). If that indication appears, the table of contextual rules is checked for a rule for that character. If no rule is found, the proposed label is rejected and MUST NOT be installed in a zone file. If one is found, it is applied (typically as a test on the entire label or on adjacent characters). If the application of the rule does not conclude that the character is valid in context, the proposed label MUST BE rejected. (See the IANA Considerations: IDNA Context Registry section of [IDNA2008-Rationale] and Appendix A of this document.)

4.3.2.4. Labels Containing Characters Written Right to Left

Additional special tests for right-to-left strings are applied (See [IDNA2008-BIDI]. Strings that contain right to left characters that do not conform to the rule(s) identified there MUST NOT be inserted in zone files.
[[anchor15: If the bidi specification continues to specify checking more than one label, this subsection will need to be revised and/or moved to a separate "FQDN validation" section.]]

4.3.3. Registration Validation Summary

Strings that have been produced by the steps above, and whose contents pass the above tests, are U-labels.

To summarize, tests are made here for invalid characters, invalid combinations of characters, and for labels that are invalid even if the characters they contain are valid individually. For example, labels containing invisible ("zero-width") characters may be permitted in context with characters whose presentation forms are significantly changed by the presence or absence of the zero-width characters, while other labels in which zero-width characters appear

may be rejected.
[[anchor17: Should the example text be removed or moved? Note that I've been strongly encouraged to supply specific examples to reduce abstraction and questions about the appropriateness of the text. -JcK]]

4.4. Registry Restrictions

Registries at all levels of the DNS, not just the top level, are expected to establish policies about the labels that may be registered, and for the processes associated with that action. While exact policies are not specified as part of IDNA2008 and it is expected that different registries may specify different policies, there SHOULD be policies. These per-registry policies and restrictions are an essential element of the IDNA registration protocol even for registries (and corresponding zone files) deep in the DNS hierarchy. As discussed in [IDNA2008-Rationale], such restrictions have always existed in the DNS.

The string produced by the above steps 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.

4.5. Punycode Conversion

The resulting U-label is converted to an A-label (i.e., the encoding of that label according to the Punycode algorithm [RFC3492] with the ACE prefix added, i.e., the "xn--..." form).
[[anchor18: Explain why 3492 failures cannot occur or explain what to do if they do.]]

4.6. Insertion in the Zone

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

5. Domain Name Resolution (Lookup) Protocol

Resolution is conceptually different from registration and different tests are applied on the client. Although some validity checks are necessary to avoid serious problems with the protocol (see Section 5.5 ff.), the resolution-side tests are more permissive and rely heavily on the assumption that names that are present in the DNS are valid.

5.1. Label String Input

The user supplies a string in the local character set, typically by typing it or clicking on, or copying and pasting, a resource identifier, e.g., a URI [RFC3986] or IRI [RFC3987] from which the domain name is extracted. Or 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 next two are local matters, to be accomplished prior to actual invocation of IDNA, but at least these two steps must be accomplished in some way.

5.2. Conversion to Unicode

The string is converted from the local character set into Unicode, if it is not already Unicode. The exact nature of this conversion is beyond the scope of this document, but may involve normalization, as described in Section 4.2.

5.3. Character Changes in Preprocessing or the User Interface

The Unicode string MAY then be processed, in a way specific to the local environment, to make the result of the IDNA processing match user expectations. For instance, it would be reasonable, at this step, to convert all upper case characters to lower case, if this makes sense in the user's environment.

Other examples of processing for localization might be applied, if appropriate, at this point. They include interpreting various characters as separating domain name components from each other (label separators) because they either look like periods or are used to separate sentences, mapping different "width" forms of the same character into the one form permitted in labels[[anchor20: This needs clarification]], or giving special treatment to characters whose presentation forms are dependent only on placement in the label. Such localization changes are also outside the scope of this specification.

Recommendations for preprocessing for global contexts (i.e., when local considerations do not apply or cannot be used) and for maximum interoperability with labels that might have been specified under liberal readings of IDNA2003 are given in [IDNA2008-Rationale].

[[anchor21: The question of preprocessing remains controversial in the WG. One school of thought is that, for compatibility with IDNA2003, preprocessing should be standardized and required, with only one form permitted. Another sees important advantages in having the mappings between U-labels and A-labels be symmetric, unambiguous, and information-preserving. And a third believes that local mappings

will occur regardless of what we specify and that it is better to specify the protocol on that basis than to indirectly encourage local inventions. The first group (and perhaps others) believe that local mappings will be, to put it mildly, "very bad... for interoperability.]]

Because these transformations are local, it is important that domain names that might be passed between systems (e.g., in IRIs) be U-labels or A-labels and not forms that might be accepted locally as a consequence of this step. This step is not standardized as part of IDNA, and is not further specified here.

5.4. A-label Input

If the input to this procedure appears to be an A-label (i.e., it starts in "xn--"), the lookup application MAY attempt to convert it to a U-label and apply the tests of Section 5.5 and, of course, the conversion of Section 5.6 to that form. If the A-label is converted to a U-label then the processing specified in those two sections MUST yield an A-label identical to the original one. See also Section 6.1.

In general, that conversion and testing 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). 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.

5.5. Validation and Character List Testing

As with the registration procedure, the Unicode string is checked to verify that all characters that appear in it are valid for IDNA resolution input. As discussed above and in [IDNA2008-Rationale], the resolution check is more liberal than the registration one. Putative labels with any of the following characteristics MUST BE rejected prior to DNS lookup:

- Labels containing code points that are unassigned in the version of Unicode being used by the application, i.e., in the "Unassigned" Unicode category or the UNASSIGNED category of [IDNA2008-Tables].

- Labels that are not in NFC form.

- Labels containing prohibited code points, i.e., those that are assigned to the "DISALLOWED" category in the permitted character

table [IDNA2008-Tables].

- Labels containing code points that are shown in the permitted character table as requiring a contextual rule and that are flagged as requiring exceptional special processing on lookup ("CONTEXTJ" in the Tables) MUST conform to the rule, which MUST be present.

- Labels containing other code points that are shown in the permitted character table as requiring a contextual rule ("CONTEXTO" in the tables), but for which no such rule appears in the table of rules. With the exception in the rule immediately above, applications resolving DNS names or carrying out equivalent operations are not required to test contextual rules, only to verify that a rule exists.

- Labels whose first character is a combining mark. [[anchor23: Note in Draft: this definition may need to be further tightened.]]

In addition, the application SHOULD apply the following test. The test may be omitted in special circumstances, such as when the resolver application knows that the conditions are enforced elsewhere, because an attempt to resolve such strings will almost certainly lead to a DNS lookup failure. 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 -- then DNS resolution failure alone. In any event, resolvers should avoid looking up labels that are invalid under that test.
[[anchor24: Should this be a MUST? Pro: this is the only remaining SHOULD (true?), the test is relatively straightforward, and it helps avoid visual ambiguity. Con: the "special circumstances" that might justify doing something different are explained above.]]

- Verification that the string is compliant with the requirements for right to left characters, specified in [IDNA2008-BIDI].

For all other strings, the resolver 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. A resolver that declines to look up a string that conforms to the above rules is not in conformance with this protocol.

5.6. Punycode Conversion

The validated string, a U-label, is converted to an A-label using the Punycode algorithm with the ACE prefix added.

5.7. DNS Name Resolution

The A-label is looked up in the DNS, using normal DNS procedures.

6. Name Server Considerations

6.1. Processing Non-ASCII Strings

Existing DNS servers do not know the IDNA rules for handling non-ASCII forms of IDNs, and therefore need to be shielded from them. All existing channels through which names can enter a DNS server database (for example, master files (as described in RFC 1034) and DNS update messages [RFC2136]) are IDN-unaware because they predate IDNA. Other sections of this document provide the needed shielding by ensuring that internationalized domain names entering DNS server databases through such channels have already been converted to their equivalent ASCII A-label forms.

Because of the design of the algorithms in Section 4 and Section 5 (a domain name containing only ASCII codepoints can not be converted to an A-label), there can not be more than one A-label form for any given U-label.

The current update to the definition of the DNS protocol [RFC2181] explicitly allows domain labels to contain octets beyond the ASCII range (0000..007F), and this document does not change that. Note, however, that there is no defined interpretation of octets 0080..00FF as characters. If labels containing these octets are returned to applications, unpredictable behavior could result. The A-label form, which cannot contain those characters, is the only standard representation for internationalized labels in the current DNS protocol.

6.2. DNSSEC Authentication of IDN Domain Names

DNS Security [RFC2535] is a method for supplying cryptographic verification information along with DNS messages. Public Key Cryptography is used in conjunction with digital signatures to provide a means for a requester of domain information to authenticate the source of the data. This ensures that it can be traced back to a trusted source, either directly or via a chain of trust linking the source of the information to the top of the DNS hierarchy.

IDNA specifies that all internationalized domain names served by DNS servers that cannot be represented directly in ASCII must use the A-label form. Conversion to A-labels must be performed prior to a zone being signed by the private key for that zone. Because of this ordering, it is important to recognize that DNSSEC authenticates a domain name containing A-labels or conventional LDH-labels, not U-labels. In the presence of DNSSEC, no form of a zone file or query response that contains a U-label may be signed or the signature validated.

One consequence of this for sites deploying IDNA in the presence of DNSSEC is that any special purpose proxies or forwarders used to transform user input into IDNs must be earlier in the resolution flow than DNSSEC authenticating nameservers for DNSSEC to work.

6.3. Root and other DNS Server Considerations

IDNs in A-label form will generally be somewhat longer than current domain names, so the bandwidth needed by the root servers is likely to go up by a small amount. Also, queries and responses for IDNs will probably be somewhat longer than typical queries historically, so EDNS0 [RFC2671] support may be more important (otherwise, queries and responses may be forced to go to TCP instead of UDP).

7. Security Considerations

The general security principles and issues for IDNA appear in [IDNA2008-Rationale]. The comments below are specific to this pair of protocols, but should be read in the context of that material and the definitions and specifications, identified there, on which this one depends.

This memo describes procedures for registering and looking up labels that are not compatible with the preferred syntax described in the base DNS specifications (STD13 [RFC1034] [RFC1035] and Host Requirements [RFC1123]) because they contain non-ASCII characters. These procedures depend on the use of a special ASCII-compatible encoding form that contains only characters permitted in host names by those earlier specifications. The encoding is specified in [RFC3492]. No security issues such as string length increases or new allowed values are introduced by the encoding process or the use of these encoded values, apart from those introduced by the ACE encoding itself.

Domain names (or portions of them) are sometimes compared against a set domains to be given special treatment if a match occurs, e.g., treated as more privileged than others or blocked in some way. In

such situations it is especially important that the comparisons be done properly, as specified in requirement 2 of Section 3.1. For labels already in ASCII form (i.e., are LDH-labels or A-labels), the proper comparison reduces to the same case-insensitive ASCII comparison that has always been used for ASCII labels.

The introduction of IDNA means that any existing labels that start with the ACE prefix would be construed as A-labels, at least until they failed one of the relevant tests, whether or not that was the intent of the zone administrator or registrant. There is no evidence that this has caused any practical problems since RFC 3490 was adopted, but the risk still exists in principle.

8. IANA Considerations

IANA actions for this version of IDNA are specified in [IDNA2008-Rationale].

9. Change Log

[[anchor30: RFC Editor: Please remove this section.]]

9.1. Changes between Version -00 and -01 of draft-ietf-idnabis-protocol

- Corrected discussion of SRV records.
- Several small corrections for clarity.
- Inserted more "open issue" placeholders.

9.2. Version -02

- Rewrote the "conversion to Unicode" text in Section 5.2 as requested on-list.
- Added a comment (and reference) about EDNS0 to the "DNS Server Conventions" section, which was also retitled.
- Made several editorial corrections and improvements in response to various comments.
- Added several new discussion placeholder anchors and updated some older ones.

9.3. Version -03

- Trimmed change log, removing information about pre-WG drafts.
- Incorporated a number of changes suggested by Marcos Sanz in his note of 2008.07.17 and added several more placeholder anchors.
- Several minor editorial corrections and improvements.
- "Editor" designation temporarily removed because the automatic posting machinery does not accept it.

10. 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 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.

11. Acknowledgements

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, [RFC4690], and [IDNA2008-Rationale] were particularly important.

Specific textual changes were incorporated into this document after suggestions from Stephane Bortzmeyer, Mark Davis, and others.

12. References

12.1. Normative References

[IDNA2008-BIDI]
Alvestrand, H. and C. Karp, "An updated IDNA criterion for right-to-left scripts", July 2008, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-bidi/>.

[IDNA2008-Rationale]
Klensin, J., Ed., "Internationalizing Domain Names for Applications (IDNA): Issues, Explanation, and Rationale", July 2008, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-rationale>.

[IDNA2008-Tables]
Faltstrom, P., "The Unicode Codepoints and IDNA", July 2008, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-tables/>.

A version of this document is available in HTML format at http://stupid.domain.name/idnabis/draft-ietf-idnabis-tables-02.html

[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.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[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.

[Unicode-PropertyValueAliases]
The Unicode Consortium, "Unicode Character Database: PropertyValueAliases", March 2008, <http://www.unicode.org/Public/UNIDATA/PropertyValueAliases.txt>.

[Unicode-RegEx]
The Unicode Consortium, "Unicode Technical Standard #18: Unicode Regular Expressions", May 2005, <http://www.unicode.org/reports/tr18/>.

[Unicode-Scripts]
The Unicode Consortium, "Unicode Standard Annex #24: Unicode Script Property", February 2008, <http://www.unicode.org/reports/tr24/>.

[Unicode-UAX15]
The Unicode Consortium, "Unicode Standard Annex #15: Unicode Normalization Forms", 2006, <http://www.unicode.org/reports/tr15/>.

12.2. 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.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[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.

[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.

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 5.0", 2007.

Boston, MA, USA: Addison-Wesley. ISBN 0-321-48091-0

Appendix A. The Contextual Rules Registry

[[anchor38: Note in Draft: The WG seems to be concluding that this material should actually be in the Tables document, possibly with some additional material added from Rationale. Unless there are objections and consensus on some other plan, that move will be made with -03 of this document. Regardless of where they are placed, the WG will still need to review the specific content of the rules. In this version of the document, the table remains something of a illustrative placeholder, not a final specification.]]

[[anchor39: The next appendix sketches out an alternate way to present this information. See the notes there.]]

As discussed in the IANA Considerations section of [IDNA2008-Rationale], a registry of rules that define the contexts in which particular PROTOCOL-VALID characters, characters associated with a requirement for Contextual Information, are permitted. These rules are expressed as tests on the label in which the characters appear (all, or any part of, the label may be tested). [[anchor40: Probably the IANA registry spec should be moved directly from Rationale to Tables -- see above.]]

For each character specified as requiring a contextual rule, a rule MAY be established with the following data elements:

1. The code point associated with the character.

2. The name of the character.

3. An indication as to whether the code point requires the rule be processed at lookup time (this indication is equivalent to the difference between "CONTEXTJ" and "CONTEXTO" in the tables document [IDNA2008-Tables]).

4. A prose description of the contextual rule.

5. A description of the contextual rule using Unicode Regular Expression notation [Unicode-RegEx]. Only a Level 1

implementation is needed for the expressions below, which also make reference to the Unicode Script definition [Unicode-Scripts] and the Unicode Property Value Aliases list [Unicode-PropertyValueAliases]. Note that in these regular expressions, the label is taken to be an entire line, i.e., "^" refers to the beginning of the label and "$" refers to the end of the label.

These regular expressions are used as tests. The contextual requirement is met if there is a match for the regular expression and not met if there is no match.

[[anchor41: Patrik and I (JcK) would like to find a way to state these rules that does not require the reader and implementer to understand what we believe to be a fairly exotic element of the Unicode specification. See the second Appendix for a possible alternative. Suggestions welcome.]]

6. An optional comment preceded by "#"

Should there be any conflict between the two statements of a rule, the regular expression form MUST be considered normative until the registry can be corrected.

The rules for the characters listed in the Tables document as exception cases or Join_Controls and for which rules are being defined at this time appear below.

[[anchor42: Note in draft: This table is not complete and the rule entries below are temporarily only examples.]]

002D; HYPHEN-MINUS; F;
Must not appear at the beginning or end of a label;
Regular expression:
[^^]\u002D|\u002D[^$] ;
Note that there are some additional prohibitions in the specification on consecutive hyphens in anything but a valid A-label.

200C; ZERO WIDTH NON-JOINER; T;
Between two characters from the same script only. The script must be one in which the use of this character causes significant visual transformation of one or both of the adjacent characters;
Regular expression:
[\p(Script:Deva)\p(Script:Tamil)]\u200C[\p(Script:Deva)\p(Script:Tamil)] ;
[[anchor43: That script list is _not_ complete and, in particular, more Indic scripts certainly need to be listed. It also does not

correctly express the "same script" restriction mentioned in the prose, since it only tests adjacent characters.]] This character is also required for Arabic script. The minimal restriction is \p(Joining_Type:L)\p(Joining_Type:T)*\u200C\p(Joining_Type:T)*\p(Joining_Type:R) ;
; more narrow restrictions may be suggested by the Arabic script group.

200D; ZERO WIDTH JOINER; T;
Between two characters from the same script only. The script must be one in which the use of this character causes significant visual transformation of one or both of the adjacent characters;
Regular expression:
[\p(Script:Deva)\p(Script:Tamil)]+
\u200D[\p(Script:Deva)\p(Script:Tamil)]+ ;
[[anchor44: That script list is _not_ complete and, in particular, more Indic scripts certainly need to be listed. It also does not correctly express the "same script" restriction mentioned in the prose, since it only tests adjacent characters. This character is not required for Arabic script.]]

00B7; MIDDLE DOT; F;
Between two 'l' (U+006C) characters only, used to permit the Catalan character ela geminada to be expressed;
Regular expression:
\u006C\u00B7\u006C ;

0375; GREEK LOWER NUMERAL SIGN (KERAIA); F;
Greek script only. Might be further restricted to specific following characters;
Regular expression:
\u0375\p(Script:Greek) ;

02B9; MODIFIER LETTER PRIME; F;;;
Permitted only in contexts in which GREEK LOWER NUMERAL SIGN, U+0375, is permitted. GREEK NUMERAL SIGN, U+0374, and the Lower Numeral Sign (U+0375) are indicators for numeric use of letters in older Greek writing systems. U+02B9 is relevant because normalization maps U+0374 into it.;
Regular expression:
\p(Script:Greek)\u02B9\p(Script:Greek) ;
[[anchor45: The test is that the adjacent characters be in the Greek script. It is not clear whether this is sufficient. The requirement for a preceding Greek letter may not be necessary. More input needed.]]

0483; COMBINING CYRILLIC TITLO; F;
Cyrillic script only. Might be further restricted to permit only a preceding list of characters.
Regular expression:
\p(Script:Cyrillic)\u0483 ;

05F3; HEBREW PUNCTUATION GERESH; F;
The script of the preceding character and the subsequent character, if any, MUST be Hebrew;
Regular expression:
\p(Script:Hebrew)\u05F3\p(Script:Hebrew)? ;

05F4; HEBREW PUNCTUATION GERSHAYIM; F
The script of the preceding character and the subsequent character, if any, MUST be Hebrew;
Regular expression:
\p(Script:Hebrew)\u05F4\p(Script:Hebrew)? ;

3005; IDEOGRAPHIC ITERATION MARK; F;
MUST NOT be at the beginning of the label, and the previous character MUST be in Han Script;
Regular expression:
\p(Script:Hani)\u3005 ;

303B; VERTICAL IDEOGRAPHIC ITERATION MARK; F;
MUST NOT be at the beginning of the label, and the previous character MUST be in Han Script;
Regular expression:
\p(Script:Hani)\u303B ;

30FB; KATAKANA MIDDLE DOT; F;
Adjacent characters MUST be Katakana;
Regular expression:
\p(Script:Kana)\u30FB\p(Script:Kana) ;

While the information above is to be used to initialize the registry, IANA should treat the table format in this Appendix simply as an initial, tentative, suggestion. Subject to review and comment from the IESG and any Expert Reviewers, IANA is responsible for, and should develop, a format for that registry, or a copy of it maintained in parallel, that is convenient for retrieval and machine processing and publish the location of that version.

Appendix B. Contextual Rules Registry - Alternate Syntax

[[anchor46: This Appendix is temporary. It illustrates, for discussion, a possible way of presenting the Contextual Rules as a

procedural pseudocode rule set rather than as a regular expression or property list and also shows a bit of the layout suggested by Mark Davis. Each entry consists of the name for identification, followed by an informal description, the code point, and the rule set. Note that the two appendices are alternate forms of the same information; only one should be moved to Tablss; the other will be deleted.]]

[[anchor47: The grammatical rules and operations for the pseudocode below are left as an exercise for the reader in this draft. Note however that the "Before" and "After" operations, by themselves, match anything including null, i.e., BeforeScript would match any script if the character was the first one in the label. Obviously, if something satisfies all of the rules, then it is contextually valid. If any of them yield "False" than it isn't. If we decide to go in this direction, we should form a small ad hoc committee to either sort that out or possibly convert it to standard Prolog.]]

B.1. HYPHEN-MINUS

Code point: 002D

Overview: Must appear at the beginning or end of a label.

Lookup: False

Rule Set:

```
If FirstChar .eq.  True Then False;
If LastChar .eq.  Then False;
Else True;
```

Comment: Note that there are some additional prohibitions in the specification on consecutive hyphens in anything but a valid A-label.

B.2. ZERO WIDTH NON-JOINER

Code point: 200C

Overview: Between two characters from the same script only. The script must be one in which the use of this character causes significant visual transformation of one or both of the adjacent characters.

Lookup: True

Rule Set:

If BeforeScript .eq. (Deva | Tamil | Arabic) Then
If AfterScript .eq. (Deva | Tamil | Arabic) Then True;
Else False;

[[anchor50: That script list is _not_ complete and, in particular, more Indic scripts certainly need to be listed. It also does not correctly express the "same script" restriction mentioned in the prose, since it only tests adjacent characters.]]

This character is also required for Arabic script. The minimal restriction (in regex form) is
\p(Joining_Type:L)\p(Joining_Type:T)*\u200C\p(Joining_Type:
T)*\p(Joining_Type:R) ;
; more narrow restrictions may be suggested by the Arabic script group.

B.3. ZERO WIDTH JOINER

Code point: 200D

Overview: Between two characters from the same script only. The script must be one in which the use of this character causes significant visual transformation of one or both of the adjacent characters.

Lookup: True

Rule Set:

If BeforeScript .eq. (Deva | Tamil | Arabic) Then
If AfterScript .eq. (Deva | Tamil | Arabic) Then True;
Else False;

[[anchor52: The script list for this character is _not_ complete and, in particular, more Indic scripts certainly need to be listed. It also does not correctly express the "same script" restriction mentioned in the prose, since it only tests adjacent characters. This character is not required for Arabic script.]]

B.4. MIDDLE DOT

Code point: 00B7

Overview: Between 'l' (U+006C) characters only, used to permit the Catalan character ela geminada to be expressed

Lookup: False

Rule Set:

```
If BeforeChar .eq. \006C Then
If AfterChar .eq. \006C Then True;
Else False;
```

B.5. GREEK LOWER NUMERAL SIGN (KERAIA)

Code point: 0375

Overview: Greek script only. Might be further restricted to specific following characters

Lookup: False

Rule Set:

```
If AfterScript .eq.  Greek Then True;
Else False;
```

B.6. MODIFIER LETTER PRIME

Code point: 02B9

Overview: Permitted only in contexts in which GREEK LOWER NUMERAL SIGN, U+0375, is permitted. GREEK NUMERAL SIGN, U+0374, and the Lower Numeral Sign (U+0375) are indicators for numeric use of letters in older Greek writing systems. U+02B9 is relevant because normalization maps U+0374 into it.

Lookup: False

Rule Set:

```
BeforeScript If .eq.  Greek Then
If AfterScript .eq.  Greek Then True;
Else False;
```

Comment: [[anchor56: The test is that the adjacent characters be in the Greek script. It is not clear whether this is sufficient. The requirement for a preceding Greek letter may not be necessary. More input needed.]]

B.7. COMBINING CYRILLIC TITLO

Code point: 0483

Overview: Cyrillic script only. Might be further restricted to permit only a preceding list of characters.

Lookup: False

Rule Set:

```
If BeforeScript .eq. Cyrillic Then
If AfterScript .eq. Cyrillic Then True;
Else False;
```

B.8. HEBREW PUNCTUATION GERESH

Code point: 05F3

Overview: The script of the preceding character and the subsequent character, if any, MUST be Hebrew.

Lookup: False

Rule Set:

```
If FirstChar .eq. True then False;
Else If BeforeScript .eq. Hebrew Then
If AfterScript .eq. Hebrew Then True;
Else False;
```

B.9. HEBREW PUNCTUATION GERSHAYIM

Code point: 05F4

Overview: The script of the preceding character and the subsequent character, if any, MUST be Hebrew.

Lookup: False

Rule Set:

```
If FirstChar .eq. True then False;
Else If BeforeScript .eq. Hebrew Then
If AfterScript .eq. Hebrew Then True;
Else False;
```

B.10. IDEOGRAPHIC ITERATION MARK;

Code point: 3005

Overview: MUST NOT be at the beginning of the label, and the previous character MUST be in Han Script.

Lookup: False

Rule Set:

```
If FirstChar .eq.  True Then False;
Else If BeforeScript .eq.  Han Then True;
Else False;
```

B.11. VERTICAL IDEOGRAPHIC ITERATION MARK

Code point: 303B

Overview: MUST NOT be at the beginning of the label, and the previous character MUST be in Han Script.

Lookup: False

Rule Set:

```
If FirstChar .eq.  True Then False;
Else If BeforeScript .eq.  Han Then True;
Else False;
```

B.12. KATAKANA MIDDLE DOT

Code point: 30FB

Overview: Adjacent characters MUST be Katakana.

Lookup: False

Rule Set:

```
If FirstChar .eq.  True Then False;
Else If BeforeScript .eq.  Kana Then
If AfterScript .eq.  Kana Then True;
Else False;
```

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

Full Copyright Statement

Intellectual Property