Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 12, 2012

K. Davies
ICANN

March 11, 2012

Representing registration policy for IDNs using XML
draft-davies-idntables-01

Abstract

This memo describes a method of representing the registration policy that a zone administrator uses for registering Internationalised Domain Names using Extensible Markup Language (XML). These registry policies, commonly known as "IDN tables", are used to enforce and share policy on which specific code-points are permitted for registrations, and which alternative code-points are considered variants.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2012.

Table of Contents

1. Introduction

This memo describes how to use Extensible Markup Language (XML) to describe the list of permissible code points and variants used in a zone administrator's policies.

Historically, zone administrators - such as top-level domain registries - have published their policies using text and HTML based formats loosely based around the format used to describe a Language Variant Table in [RFC3743]. [RFC4290] further posts a "Model table format" that describes a similar set of functionality.

Through the first decade of IDN deployment, experience has shown that these table formats are difficult to consistently implement and compare due to their different formats. A more universal format, such as one using a structured XML format, will assist by improving machine-readability, consistency, and maintainability of IDN tables. It will also provide for more complex conditional implementation of variants that reflects the known requirements of current zone administrator policies.

2. Design Goals

The following items are explicit design goals of this format:

- MUST be in a format that can be implemented in a reasonably straightforward manner in software;
- The format SHOULD be able to be checked for formatting errors, such that common mistakes can be caught;
- An IDN Table MUST be able to express the set of valid code points that are allowed for registration under a specific zone administrator's policies;
- MUST be able to express computed alternatives to a given domain name based on a one-to-one, or one-to-many relationship. These computed alternatives are commonly known as "IDN variants";
- IDN Variants SHOULD be able to be tagged with specific categories, such that the categories can be used to support registry policy (such as whether to list the computed variant in the zone, or to merely block it from registration);
- IDN Variants MUST be able to stipulated based on contextual information. For example, specific variants may only be applicable when they follow another specific code point, or when the code point is displayed in a specific presentation form;
- The data contained within the table MUST be unambiguous, such that independent implementations that utilise the contents will arrive at the same results;
- IDN Tables SHOULD be suitable for comparison and re-use, such that one could easily compare the contents of two or more to see the differences, to merge them, and so on.
- As many existing IDN Tables are practicable SHOULD be able to be migrated to the new format with all applicable logic retained.

It is explicitly NOT the goal of this format to:

- Stipulate what code points should be listed in an IDN Table by a zone administrator. What registration policies are used for a particular zone is outside the scope of this memo.
- Stipulate what a consumer of an IDN Table must do when they determine a particular domain is valid or invalid; or arrive at a set of computed IDN variants. IDN Tables are only used to

describe rules for computing code points, but does not prescribe how registries and other parties utilise them.

3. IDN Table XML Format

3.1. Namespace

The XML Namespace URI is [TBD].

3.2. Basic structure

The basic XML framework of the document is as follows:

```
<?xml version="1.0"?>
<idntable xmlns="http://www.iana.org/idn-tables/0.1">
    ...
</idntable>
```

Within the "idntable" element rests two sub-elements. Firstly is a "meta" element that contains all meta-data associated with the IDN table, such as its authorship, what it is used for, implementation notes and references. This is followed by a "data" element that contains the substantive code-point data.

```
<?xml version="1.0"?>
<idntable xmlns="http://www.iana.org/idn-tables/0.1">
    <meta>
        ...
    </meta>
    <data>
        ...
    </data>
</idntable>
```

A document should contain exactly one "idntable" element, and within that optionally one "meta" element and exactly one "data" element.

3.3. The meta element

The "meta" element is used to express meta-data associated within the IDN table. It can be used to explain the author or relevant contact person, explain what the usage of the IDN table is, provide implementation notes as well as references. The data contained within is not required by software consuming the IDN table in order to calculate valid IDN labels, or to calculate variants.

3.3.1. The version element

The "version" element is used to uniquely identify each version of the table being represented. No specific format is required, but it is RECOMMENDED that it be a numerical positive integer, which is

incremented with each revision of the file.

An example of a typical first edition of a document:

```
<version>1</version>
```

A common alternative is to use a major-minor number scheme, where two decimal numbers are used to represent major and minor changes to the table. For example, "1.0" would be the first major release, "1.1" would be a minor update to that, and "2.0" would represent a major revision.

3.3.2. The date element

The "date" element is used to identify the date the table was written. The contents of this element MUST be a valid ISO 8601 date string as described in [RFC3339].

Example of a date:

```
<date>2009-11-01</date>
```

3.3.3. The language element

The "language" element signals that the table is associated with a specific language or script. The value of the language element must be a valid language tag as described in [RFC5646]. The tag may simply refer to a script if the table is not referring to a specific language. There may be multiple language elements for a table if the table spans multiple languages and/or scripts.

Example of an English language table:

```
<language>en</language>
```

If the table applies to a specific script, rather than a language, the "und" language tag should be used followed by the relevant [RFC5646] script subtag. For example, for a Cyrillic script table:

```
<language>und-Cyrl</language>
```

3.3.4. The domain element

This optional element refers to a domain to which this policy is applied.

```
<domain>example</domain>
```

There may be multiple <domain> tags used to reflect a list of domains.

3.3.5. The description element

The "description" element is a free-form element that contains any additional relevant description. Typically, this field contains authorship information, as well as additional context on how the table was formulated (such as with references), and how it has been applied.

The element has an optional "type" attribute, which refers to the media type of the enclosed data. If the description lacks a type field, it will be assumed to be plain text.

The description elements describe information relating to the IDN table that is useful for the user of the table in its interpretation. This may explain the history, the rationale, reference sources etc. It may also contain authorship information.

The "type" attribute may be used to specify the encoding within description element. The attribute should be a valid MIME type. If supplied, it will be assumed the contents is content of that encoding. Typical types would be "text/plain" or "text/html". "text/plain" will be assumed if no type attribute is specified.

3.4. The data element

The "data" element contains the code point data the comprises the registry policy. It describes registry policy using a series of XML elements that either represent individual code points, or ranges of code points.

The data may use the "char" and "range" elements to specify code points, and code ranges.

Discrete permissable code points or code point sequences may be stipulated with a "char" element, e.g.

```
<char cp="002D"/>
```

Ranges of permissable code points may be stipulated with a "range" element, e.g.

```
<range first-cp="0030" last-cp="0039"/>
```

Codepoints must be expressed in hexadecimal, i.e. according to the standard Unicode convention without the prefix "U+". The rationale

for not allowing other encoding formats, including native Unicode encoding in XML, is explored in [UAX42]. The XML conventions used in this format, including the element and attribute names, mirror this document where practical and reasonable to do so.

3.4.1. Sequences

A sequence of two or more code points may be specified in a table, when the exact sequence of code points is required to occur in order for the consituent elements to be eligible. This approach allows representation of policy where a specific code point is only eligible when preceded or followed by another code point. For example, in order to represent the eligibility of the MIDDLE DOT (U+00B7) only when both preceded and followed by the LATIN SMALL LETTER L (U+006C):

```
<char cp="006C 00B7 006C"/>
```

3.4.2. Variants

While most tables typically only determine code point eligibility, others additionally specify a mapping of code points to other code points, known as "variants". What constitutes a variant is a matter of policy, and varies for each implementation.

3.4.2.1. Basic variants

Variants are specified as one of more children of a "char" element.

For example, to map LATIN SMALL LETTER V (U+0076) as a variant of LATIN SMALL LETTER U (U+0075):

```
<char cp="0075">
  <var cp="0076"/>
</char>
```

A sequence of multiple code points can be specified as a variant of a single code point. For example, the sequence of LATIN SMALL LETTER O (U+006F) then LATIN SMALL LETTER E (U+0065) can be specified as a variant for an LATIN SMALL LETTER O WITH DIAERESIS (U+00F6) as follows:

```
<char cp="00F6">
  <var cp="006F 0065"/>
</char>
```

Variants are specified in only on direction. For symmetric variants, the inverse of the variant must be explicitly specified:

```
<char cp="006F 0065">
  <var cp="00F6"/>
</char>
```

It is not possible to specify variants for ranges.

3.4.2.2. Null variants

To specify a null variant, which is a variant string that maps to no codepoint, use an empty cp attribute. For example, to mark a string with a ZERO WIDTH NON-JOINER (U+200C) to the same string without the ZERO WIDTH NON-JOINER:

```
<char cp="200C">
  <var cp=""/>
</char>
```

3.4.2.3. Conditional variants

At its basis, generation of variants are conditional on a specific code point or set of code points. However, in some instances registries perform control based on other attributes that can not solely be determined based on simple code point comparisons. For example, in some tables utilising the Arabic script, the Arabic contextual form is a determinant in which variants are used. The contextual form can not be derived solely from the code point, as the code point is the same for the various forms.

The IDN table provides for conditioning generation variants on specific instances as follows, using the "when" attribute.

arabic-initial Based on context, the code point would be presented in its Arabic Initial form.

arabic-isolated Based on context, the code point would be presented in its Arabic Isolated form.

arabic-medial Based on context, the code point would be presented in its Arabic Medial form.

arabic-final Based on context, the code point would be presented in its Arabic Final form.

For example, to mark ARABIC LETTER ALEF WITH WAVY HAMZA BELOW (U+0673) as a variant of ARABIC LETTER ALEF WITH HAMZA BELOW (U+0625), but only when it appears in isolated or final forms:

```
<char cp="0625">
  <var cp="0673" when="arabic-isolated"/>
  <var cp="0673" when="arabic-final"/>
</char>
```

3.5. Example table

A sample complete XML IDN table is as follows.

```
<?xml version="1.0"?>
<idntable xmlns="http://www.iana.org/idn-tables/0.1">
    <meta>
         <version>1</version>
         <date>2010-01-01</date>
         <language>sv</language>
         <domain>example</domain>
         <description type="text/html">
         <![CDATA[
             This language table was developed with the
             <a href="http://swedish.example/">Swedish
             examples institute</a>.
         ]]>
         </description>
    </meta>
    <data>
        <range first-cp="0061" last-cp="007A"/>
        <char cp="00E4"/>
    </data>
</idntable>
```

4. Processing a label against a table

4.1. Determining eligibility for a label

In order to use a table to test a specific IDN table for membership in the table, a consumer of an IDN table must iterate through each code point within a given U-label, and test that each code point is a member of the IDN table. If any code point is not a member of the IDN Table, it shall be deemed as not eligible in accordance with the table.

A code point is deemed a member of the table when it is listed with the <char> element, and all necessary condition listed in "when" attributes are correctly satisfied.

4.2. Determining variants for a label

For a given eligible label, the set of variants is deemed to be each possible permutation of <var> elements, whereby all "when" attributes are correctly satisfied for each code point in the given permutation.

5. Conversion between other formats

5.1. RFC 3743 Language Variant Table

All attributes can be retained in conversion from an [RFC3743] language variant table to this XML format.

This XML format can be converted to the format described in [RFC3743], with the following caveats:

- Version numbers not expressed as integers will not satisfy the ABNF formatting for [RFC3743].
- Much of the additional meta data can not be expressed in the text format (although can be supplied as comments in the text file).
- The [RFC3743] format only allows for two variant classes, those that are preferred and those that are regular. Other distinctions will be lost.
- No ability to retain conditional variants.

5.2. RFC 4290 Model Table Format

All attributes can be retained in conversion from the [RFC4290] model table format to this XML format.

Tables similarly can be converted to the format described in [RFC4290] with the same caveats as the [RFC3743] format, and additionally the inability to classify variants into groups such as "preferred".

6. IANA Considerations

This document does not specify any IANA actions.

7. Security Considerations

There are no security considerations for this memo.

8. References

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3743] Konishi, K., Huang, K., Qian, H., and Y. Ko, "Joint Engineering Team (JET) Guidelines for Internationalized Domain Names (IDN) Registration and Administration for Chinese, Japanese, and Korean", RFC 3743, April 2004.

[RFC4290] Klensin, J., "Suggested Practices for Registration of Internationalized Domain Names (IDN)", RFC 4290, December 2005.

[RFC5564] El-Sherbiny, A., Farah, M., Oueichek, I., and A. Al-Zoman, "Linguistic Guidelines for the Use of the Arabic Language in Internet Domains", RFC 5564, February 2010.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

[UAX42] Unicode Consortium, "Unicode Character Database in XML".

Appendix A. RelaxNG Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar ns="http://www.iana.org/idn-tables/0.1"
  xmlns="http://relaxng.org/ns/structure/1.0">
  <define name="language-tag">
    <text/>
  </define>
  <define name="domain-name">
    <text/>
  </define>
  <define name="code-point">
    <text/>
  </define>
  <define name="variant-condition">
    <text/>
  </define>
  <define name="point-single">
    <element name="char">
      <attribute name="cp">
        <ref name="code-point"/>
      </attribute>
      <zeroOrMore>
        <ref name="point-variant"/>
      </zeroOrMore>
    </element>
  </define>
  <define name="point-variant">
    <element name="var">
      <attribute name="cp">
        <ref name="code-point"/>
      </attribute>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="when">
          <ref name="variant-condition"/>
        </attribute>
      </optional>
    </element>
  </define>
  <define name="point-multiple">
    <element name="range">
      <attribute name="first-cp">
        <ref name="code-point"/>
      </attribute>
```

```
        <attribute name="last-cp">
          <ref name="code-point"/>
        </attribute>
        <text/>
      </element>
    </define>
    <define name="points">
      <oneOrMore>
        <choice>
          <ref name="point-single"/>
          <ref name="point-multiple"/>
        </choice>
      </oneOrMore>
    </define>
    <start>
      <ref name="idn-table"/>
    </start>
    <define name="idn-table">
      <element name="idntable">
        <optional>
          <ref name="meta-section"/>
        </optional>
        <ref name="data-section"/>
      </element>
    </define>
    <define name="meta-section">
      <element name="meta">
        <zeroOrMore>
          <choice>
            <optional>
              <element name="version">
                <text/>
              </element>
            </optional>
            <optional>
              <element name="date">
                <text/>
              </element>
            </optional>
            <zeroOrMore>
              <element name="language">
                <ref name="language-tag"/>
              </element>
            </zeroOrMore>
            <zeroOrMore>
              <element name="domain">
                <ref name="domain-name"/>
              </element>
```

```
            </zeroOrMore>
            <zeroOrMore>
              <element name="description">
                <attribute name="type"/>
                <text/>
              </element>
            </zeroOrMore>
          </choice>
        </zeroOrMore>
      </element>
    </define>
    <define name="data-section">
      <element name="data">
        <ref name="points"/>
      </element>
    </define>
  </grammar>
```

Appendix B. Acknowledgements

This format builds upon the work on documenting IDN tables by a number of other parties, most significantly that of the the Joint Engineering Team published as [RFC3743], and [RFC5564] published by the Arabic-language community.

Contributions that have helped shape this document have been contributed by Francisco Arias, Mark Davis, Nicholas Ostler, Thomas Roessler, Steve Sheng and Andrew Sullivan.

Appendix C. Editorial Notes

This appendix to be removed prior to final publication.

C.1. Known Issues and Future Work

- An optional mechanism for explicitly nominating the registry action associated with a computed variant could be added. For example, an "action" attribute to the <var> element could specify one of the following: "allocate", "block", "delegate", "mirror" or "withhold". Each of these actions would need to be formally defined.

- The tables may benefit from a unique identifier, such as an "id" attribute on the <idntables> element.

- A method of specifying the origin URI for a table, and an expiration or refresh policy, as meta-data may be a useful way to declare how the table will be updated.

- A more formal step-wise description of how variants are computed, including the methodology for assessing contextual rules, needs to be supplied.

C.2. Sample tables and running code

Some sample tables using this format, as well as a basic implementation of this specification, is posted at https://github.com/kjd/idntables

C.3. Change History

-00 Initial draft.

Author's Address

Kim Davies
Internet Corporation for Assigned Names and Numbers
4676 Admiralty Way
Suite 330
Marina del Rey, CA 90292
US

Phone: +1 310 823 9358
Email: kim.davies@icann.org
URI: http://www.iana.org/