core Internet Council of Registrars

IDN EPP Extension

for the .TRANSLATIONS TLD

CORE Internet Council of Registrars

Extension for Language Information

The CORE Registration System used to operate the .TRANSLATIONS TLD provides a proprietary EPP extension for internationalized domain names (IDNs).

1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) described in RFC 5730. This mapping is an extension of the domain name mapping described in RFC 5731. It is specified using the Extensible Markup Language (XML) and XML Schema notation.

This extension serves the purpose of supplying and querying information for internationalized domain names. In particular, the language or script used and domain name variants are addressed.

The CORE Registration System is capable with dealing the needs of the various registries by using different modes of operation. Each mode, however, requires slightly different input from the registrars and generate slightly different output to the registrars. In order to avoid a multitude of similar extensions, a unified extension has been created to cover all cases. In the following, all supported concepts are explained, although not all are relevant for the .TRANSLATIONS TLD. Specific notes indicate whether they apply to the .TRANSLATIONS TLD or not. The examples have been chosen in such a way that only relevant parts are contained.

The following subsections summarize details required for the understanding of the IDN extension. For further policy details, see "IDN Policies" and "Variant Policies" sections above.

1.1 Domain Names

Domain names consists of a sequence of characters of the Unicode character set. If the domain name uses only characters of the ASCII subset, it is called an ASCII domain name, otherwise, it is called an internationalized domain name, abbreviated as IDN. Registrable ASCII domain names are further limited to letters, digits and hyphens (abbreviated LDH), so they cannot contain symbols (like the percent sign) or punctuation (like the exclamation mark).

IDNs must generally conform to the IDNA2008 standard. For registrable names, further language and script restrictions apply. Within EPP, IDNs have to be encoded by the Punycode standard (typically identifiable by the "xn--" prefix), including, but not limited to the IDN extension.

1.2 Language and Script

ICANN requires that the registration of an IDN must be accompanied by the specification of a language or script. A "script" denotes a writing system, like the use of Latin, Cyrillic or Arabic characters or Asian ideographs. A "language" denotes in this context the use of a writing system for a specific human language. With the selection of a script or language, the characters that can be used to form a domain name are limited to those used by the respective script or language. The exact characters and rules are documented in the so-called IDN tables, published by the registries and the IANA[1], the tables applicable for the .TRANSLATIONS TLD are included further down in this documentation. The reason behind this is to mitigate the risks of so-called homonym attacks, which use identical or similar looking characters from different scripts or languages to spoof the users about the real identity of a domain name.

To specify the language or script, the extension provides the <lang> or <script> elements, respectively. The language identifier meets the requirements of RFC 5646 (which is based on the ISO 639-1 language identifiers), while the script identifier meets the requirements of ISO 15924. While client provided language and script identifiers are accepted in a case insensitive manner, they are always reported in the correct case. The .TRANSLATIONS TLD uses the <script> element.

The following IDN tags are supported by the .TRANSLATIONS TLD:

Name	**tag**	**type**	**supports variants**	**default**
none/ASCII	*(empty)*	none		x
Latin	Latn	script		

1.3 Variants

In some languages, there is more than one way to write a word. In Latin languages, accented characters can often be replaced by their base characters or ligatures can be replaced by the letters from which they derive, which was common practice back when mechanical typewriters and early computers could not represent such characters. It is still important today as not all protocols and systems are ready to deal with IDNs. For other languages in other scripts, similar replacements exists. For example, the Chinese differentiate between "traditional" and "simplified" ideographs, so that for many words multiple ideographs exist. The different ways to write a word needs to be distinguished from synonyms – two completely different words (often of different origins) meaning the same or something similar. Usually this is not covered by variants.

The variants mechanism is also a way to mitigate the homograph problem. The registry system does not allow two names, which are variants of each other, to

1 IANA, http://www.iana.org/domains/idn-tables

be registered by two different registrants. In order to achieve this, the registry system supports two modes: The first is called "attribute mode", the second is called "object mode".

In the "attribute mode", for a registered domain name, variant names can be added and removed. These variant names are an attribute of the domain object, i.e. integral data of it, comparable to the contact and hosts references, the DNSSEC data and the status values. All variants are published in the DNS using the same name server set and DNSSEC keys as the original domain name.

In the "object mode", each variant is a separate domain object. To create a new variant, a new domain:create EPP command must be issued. To prevent the registration by a different registrant, such a variant may only be created by the same registrar and the variant must share some attributes like registrant, administrative and/or name servers with the existing domain name. However, the variant may have different other contacts, may use other name servers and other DNSSEC data. The variants and the original domain name form a so-called *bundle*. The original domain name has, contrary to the "attribute mode", no special role. For example, if the original domain is deleted, the variants survive. Changes to the registrant (and administrative) contacts applied to one domain of the bundle are mirrored to all other domains of the bundle. Similarly, if a transfer on one domain is performed, the other domains of the bundle are transferred as well.

Throughout the extensions, the domain names of variants are represented by <nameVariant> elements, each containing one name in Punycode notation, as noted above. The <nameVariant> elements are wrapped in a single <variants> element for all commands and response extensions except the <idn:update> extension. An omitted <variants> element has the same meaning as a <variants> element containing no variants. For the <idn:update> extension, the <nameVariant> elements directly appear within the <add> and <rem> elements. There is no special order defined on variants, they may be submitted or reported in an arbitrary order.

The .TRANSLATIONS TLD supports variants in the "attribute mode". Per domain object, up to 10 variants may be added.

2. EPP Command Mapping

This section deals with the specific command mappings for the .TRANSLATIONS TLD EPP extension for IDNs.

In the following, the respective root elements of the extensions are mentioned. If used, they must be placed or expected within the optional <extension> element at the proper location in the XML document representing the EPP command or response, as described in RFC 5730. Note that the use of the "idn:" XML namespace prefix is for documentation purposes only. Conforming to the "Namespaces in XML 1.1" standard, EPP and the registry implementation take only the associated namespace URI into account, and not the prefix itself. So actually any prefix or even the default namespace may be used in requests and must be expected in responses.

The IDN extenion is only used in relation to domain objects. It will not occur in commands that are related to host and contact objects.

2.1 EPP Query Commands

There are four EPP commands to retrieve object information: <check> to find out whether an object is known to the server, <info> to ask for detailed information associated with an object, <poll> to discover and retrieve queued service messages for individual clients and <transfer> to get transfer status information for an object.

2.1.1 EPP <domain:check> Command

The <idn:check> element, if present, allows to specify the script with the help of the <script> element. It applies to all names submitted. No extension is added to the response of the check command.

If a given name is not suitable for the given script, it is marked as unavailable with the reason of "Invalid".

Using the "attribute mode", the name is reported as "In use", if an identical domain name exists. If the name represents a variant of an existing domain, it is reported as "Blocked", independent of whether the variant actually exists or not.

2.1.2 EPP <domain:info> Command

The IDN extension does not provide an element for the info command. However, the response may contain an <idn:infData> element providing additional information.

The current script is reported via the <script> element.

As the registry system uses the "attribute mode", the provided domain name must be the object name. It is not possible to use a variant name, neither existing or non-existing, in which case an "object does not exist" error is generated. The <idn:infData> contains a list of all existing variants that have been defined for the domain.

Note that the extension is omitted if the domain is an ASCII domain name and does not have any variant names.

2.1.3 EPP <poll> Command

This extension does not add any element to the EPP <poll> command itself.

2.1.4 EPP <transfer> Query Command

This extension does not add any element to the EPP <transfer> query command itself.

2.2 EPP Transform Commands

There are five EPP commands to transform objects: <create> to create an instance of an object, <delete> to delete an instance of an object, <renew> to extend the validity period of an object, <transfer> to manage object sponsorship changes and <update> to change information associated with an object.

2.2.1 EPP <domain:create> Command

The create command, which allows the registration of domain objects, or, during the registry sunrise and landrush phases, the application for domain objects, can be augmented by an <idn:create> element in the extension section of the command. This element can carry a language/script identifier as well as a list of variants, as described in the following.

With the help of the <script> element, the used script of the name and variants is defined. The specification of the script is optional. If omitted, an ASCII domain name is assumed.

Along with the domain name itself, up to 10 variants may be specified.

2.2.2 EPP <domain:delete> Command

There are no extension elements for the domain delete command and response.

2.2.3 EPP <domain:renew> Command

There are no extension elements for the renew delete command and response.

2.2.4 EPP <domain:transfer> Command

For the .TRANSLATIONS TLD, the extension element <idn:trnData> is not used. It applies only to instance that use the "object mode" variants model. There is no special behaviour of the transfer command regarding IDNs.

2.2.5 EPP <domain:update> Command

For the domain update command, extension elements exist for both the command and response, <idn:update> and <idn:updData> respectively. The latter is only used for variants in the "object mode".

Using the <script> element, the language of an existing domain name and its variants can be changed at a later point in time. This can be useful if it turns out that the domain has been originally registered using a wrong script and a desired variant could not be registered. A precondition for the successful execution of the change of the script is that the domain name the final variants (i.e. after applying any supplied variant additions or deletions within the same extension) are valid names for the new script.

By supplying variant names in the <add> and <rem> section, variants associated with the domains can be added and removed, respectively. The variants added must be valid variants of the domain name. After processing the changes, the limit of 10 variants may not be exceeded.

3. Formal Syntax (Schema Definition)

```
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="http://xmlns.corenic.net/epp/idn-1.0"
    xmlns:idn="http://xmlns.corenic.net/epp/idn-1.0"
    xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      domain name extension schema for internationalised domain names
      processing.
    </documentation>
  </annotation>

  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
    schemaLocation="eppcom-1.0.xsd"/>


  <!-- child elements found in EPP commands -->

  <element name="check" type="idn:checkType"/>
  <element name="create" type="idn:createType"/>
  <element name="update" type="idn:updateType"/>


  <!-- child elements of the <check> command -->

  <complexType name="checkType">
    <sequence>
      <group ref="idn:idnTagGroup"/>
    </sequence>
  </complexType>


  <!-- child elements of the <create> command -->

  <complexType name="createType">
    <sequence>
      <group ref="idn:idnTagGroup" minOccurs="0"/>
      <element name="variants" type="idn:variantListType" minOccurs="0"/>
    </sequence>
  </complexType>
```

```
<!-- child elements of the <update> command -->

<complexType name="updateType">
  <sequence>
    <element name="add" type="idn:variantListType" minOccurs="0"/>
    <element name="rem" type="idn:variantListType" minOccurs="0"/>
    <element name="chg" minOccurs="0">
      <complexType>
        <sequence>
          <group ref="idn:idnTagGroup" minOccurs="0"/>
        </sequence>
      </complexType>
    </element>
  </sequence>
</complexType>


<!-- child response elements -->

<element name="creData" type="idn:bundleDataType"/>
<element name="updData" type="idn:bundleDataType"/>
<element name="trnData" type="idn:bundleDataType"/>
<element name="infData" type="idn:respDataType"/>


<!-- response elements -->

<complexType name="respDataType">
  <sequence>
    <group ref="idn:idnTagGroup"/>
    <element name="variants" type="idn:variantListType" minOccurs="0"/>
  </sequence>
</complexType>


<!-- bundle information -->

<complexType name="bundleDataType">
  <sequence>
    <element name="variants" type="idn:variantListType"/>
  </sequence>
</complexType>


<!-- common types -->

<!-- type that allows an empty string only -->

<simpleType name="emptyTokenType">
  <restriction base="token">
    <length value="0"/>
  </restriction>
</simpleType>


<!-- a script code according to ISO 15924, additionally allowing an
     empty string -->

<simpleType name="scriptType">
  <union memberTypes="idn:emptyTokenType">
    <simpleType>
      <restriction base="token">
        <minLength value="3"/>
        <maxLength value="4"/>
      </restriction>
    </simpleType>
  </union>
  </simpleType>


  <!-- a language code according to RFC 5646, additionally allowing an
       empty string -->

  <simpleType name="languageType">
    <union memberTypes="idn:emptyTokenType language">
    </union>
  </simpleType>


  <!-- elements that represent either a language or a script -->

  <group name="idnTagGroup">
    <choice>
      <element name="lang" type="idn:languageType"/>
      <element name="script" type="idn:scriptType"/>
    </choice>
  </group>


  <!-- a list of variants of a domain name -->

  <complexType name="variantListType">
    <sequence>
      <element name="nameVariant" type="eppcom:labelType"
        minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

</schema>
```

4. Examples

In the following examples, "C:" represents lines sent by an EPP client and "S:" represents lines returned by the .TRANSLATIONS Registry EPP server.

4.1 EPP <check> Command

4.1.1 Example <check> command with script tag:

```
C:<?xml version="1.0" encoding="UTF-8"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <check xmlns="urn:ietf:params:xml:ns:domain-1.0">
C:        <name>example..TRANSLATIONS</name>
C:      </check>
C:    </check>
C:    <extension>
C:      <check xmlns="http://xmlns.corenic.net/epp/idn-1.0">
C:        <script>Latn</script>
C:      </check>
C:    </extension>
C:    <clTRID>0168899153_1332496264546</clTRID>
C:  </command>
```

```
C:</epp>
```

4.2 EPP <info> Command

4.2.1 Example <info> response with script tag and empty list of variants:

```
S:<?xml version='1.0' encoding='UTF-8'?>
S:<epp xmlns='urn:ietf:params:xml:ns:epp-1.0'>
S:  <response>
S:    <result code='1000'>
S:      <msg lang='en-US'>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <infData xmlns='urn:ietf:params:xml:ns:domain-1.0'>
S:        <name>example..TRANSLATIONS</name>
S:        <roid>D123456789</roid>
S:        <status s='active'/>
S:        <registrant>abc123</registrant>
S:        <contact type='admin'>def456</contact>
S:        <contacttype='tech'>ghi789</contact>
S:        <ns>
S:          <hostObj>ns1.example.net</hostObj>
S:          <hostObj>ns2.example.net</hostObj>
S:        </ns>
S:        <clID>registrar</clID>
S:        <crID>registrar</crID>
S:        <crDate>2010-09-08T07:06:05.0Z</crDate>
S:        <exDate>2012-09-08T23:59:59.0Z</exDate>
S:        <authInfo>
S:          <pw>secret</pw>
S:        </authInfo>
S:      </infData>
S:    </resData>
S:    <extension>
S:      <infData xmlns="http://xmlns.corenic.net/epp/idn-1.0">
S:        <script>Latn</script>
S:        <variants/>
S:      </infData>
S:    </extension>
S:    <trID>
S:      <svTRID>ZYX-99958</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

4.2.2 Example <info> response with script tag and domain name variants:

```
S:<?xml version='1.0' encoding='UTF-8'?>
S:<epp xmlns='urn:ietf:params:xml:ns:epp-1.0'>
S:  <response>
S:    <result code='1000'>
S:      <msg lang='en-US'>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <infData xmlns='urn:ietf:params:xml:ns:domain-1.0'>
S:        <name>example..TRANSLATIONS</name>
```

```
S:          <roid>D123456789-COM</roid>
S:          <status s='active'/>
S:          <registrant>abc123</registrant>
S:          <contact type='admin'>def456</contact>
S:          <contacttype='tech'>ghi789</contact>
S:          <ns>
S:            <hostObj>ns1.example.net</hostObj>
S:            <hostObj>ns2.example.net</hostObj>
S:          </ns>
S:          <clID>registrar</clID>
S:          <crID>registrar</crID>
S:          <crDate>2010-09-08T07:06:05.0Z</crDate>
S:          <exDate>2012-09-08T23:59:59.0Z</exDate>
S:          <authInfo>
S:            <domain:pw>secret</pw>
S:          </authInfo>
S:        </infData>
S:      </resData>
S:      <extension>
S:        <infData xmlns="http://xmlns.corenic.net/epp/idn-1.0">
S:          <script>Latn</script>
S:          <variants>
S:            <nameVariant>dummy..TRANSLATIONS</nameVariant>
S:            <nameVariant>wrong..TRANSLATIONS</nameVariant>
S:          </variants>
S:        </infData>
S:      </extension>
S:      <trID>
S:        <svTRID>ZYX-99958</svTRID>
S:      </trID>
S:    </response>
S:</epp>
```

4.3 EPP <create> Command

4.3.1 Example <create> command with script tag and empty list of variants:

```
C:<?xml version="1.0" encoding="UTF-8"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <create xmlns="urn:ietf:params:xml:ns:domain-1.0">
C:        <name>example..TRANSLATIONS</name>
C:        <period unit="y">1</period>
C:        <ns>
C:          <hostObj>ns1.example.net</hostObj>
C:          <hostObj>ns2.example.net</hostObj>
C:        </ns>
C:        <registrant>abc123</registrant>
C:        <contact type="admin">def456</contact>
C:        <contact type="tech">ghi789</contact>
C:        <authInfo>
C:          <pw>secret42</pw>
C:        </authInfo>
C:      </create>
C:    </create>
C:    <extension>
C:      <create xmlns="http://xmlns.corenic.net/epp/idn-1.0">
```

```
C:        <script>Latn</script>
C:      </create>
C:    </extension>
C:    <clTRID>abc-00042</clTRID>
C:  </command>
C:</epp>
```

4.3.2 Example <create> command with script tag and domain name variants:

```
C:<?xml version="1.0" encoding="UTF-8"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <create xmlns="urn:ietf:params:xml:ns:domain-1.0">
C:        <name>example..TRANSLATIONS</name>
C:        <period unit="y">1</period>
C:        <ns>
C:          <hostObj>ns1.example.net</hostObj>
C:          <hostObj>ns2.example.net</hostObj>
C:        </ns>
C:        <registrant>abc123</registrant>
C:        <contact type="admin">def456</contact>
C:        <contact type="tech">ghi789</contact>
C:        <authInfo>
C:          <pw>secret42</pw>
C:        </authInfo>
C:      </create>
C:    </create>
C:    <extension>
C:      <create xmlns="http://xmlns.corenic.net/epp/idn-1.0">
C:        <script>Latn</script>
C:        <variants>
C:          <nameVariant>dummy..TRANSLATIONS</nameVariant>
C:          <nameVariant>wrong..TRANSLATIONS</nameVariant>
C:        </variants>
C:      </create>
C:    </extension>
C:    <clTRID>abc-00042</clTRID>
C:  </command>
C:</epp>
```

4.4 EPP <update> Command

4.4.1 Example <update> command changing the script of a domain:

```
C:<?xml version="1.0" encoding="UTF-8"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <update xmlns="urn:ietf:params:xml:ns:domain-1.0">
C:        <name>example..TRANSLATIONS</name>
C:        <chg/>
C:      </update>
C:    </update>
C:    <extension>
C:      <update xmlns="http://xmlns.corenic.net/epp/idn-1.0">
```

```
C:          <chg>
C:            <script>Latn</script>
C:          </chg>
C:        </update>
C:      </extension>
C:      <clTRID>abc-00042</clTRID>
C:    </command>
C:</epp>
```

4.4.2 Example <update> command adding and removing domain name variants:

```
C:<?xml version="1.0" encoding="UTF-8"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:    <command>
C:      <update>
C:        <update xmlns="urn:ietf:params:xml:ns:domain-1.0">
C:          <name>example..TRANSLATIONS</name>
C:          <chg/>
C:        </update>
C:      </update>
C:      <extension>
C:        <update xmlns="http://xmlns.corenic.net/epp/idn-1.0">
C:          <add>
C:            <nameVariant>silly..TRANSLATIONS</nameVariant>
C:            <nameVariant>right..TRANSLATIONS</nameVariant>
C:          </add>
C:          <rem>
C:            <nameVariant>dummy..TRANSLATIONS</nameVariant>
C:            <nameVariant>wrong..TRANSLATIONS</nameVariant>
C:          </rem>
C:        </update>
C:      </extension>
C:      <clTRID>abc-00042</clTRID>
C:    </command>
C:</epp>
```