TANGO®
REGISTRY SERVICES

IDN EPP Extension

for the TANGO Registration System

Knipp Medien und Kommunikation GmbH

TANGO®
REGISTRY SERVICES

Table of Contents

1. Extension for Language Information

1.1 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 TANGO Registration System (for which this extension was designed) is capable of handling the needs of the various registries by using different modes of operation; in particular, it distinguishes between the two modes of handling variants, namely "Domain Variants as Attributes" ("attribute mode") and "Domain Variants as Objects" ("object mode"). 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, this unified extension has been created to cover all cases. In the following, all supported concepts are explained, although not all are relevant for all TLDs. Note that a single instance of the TANGO Registration System only supports one configured variant mode for all domains under management, i.e. mixing modes across domains is not possible. This EPP extension therefore assumes that either one mode or the other is in effect, but never both at the same time.

1.1.1 Domain Names

Domain names consist 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 e.g. the percent sign) or punctuation (like e.g. 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 this IDN extension.

1.1.2 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" in this context denotes 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 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.

1.1.3 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. Such a situation (different ways to write a word) needs to be distinguished from the concept of 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 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 associated with 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

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

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 contact and/or name servers with the existing domain name. However, the variant may have different other contacts, may use different name servers and different 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 remain unchanged. 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 extension, 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.

1.2 EPP Command Mapping

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

In the following, the respective root elements of the extension 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 extension is only used in relation to domain objects. It will not occur in commands that are related to host and contact objects.

See the "Examples" section below for XML examples covering all command mappings.

1.2.1 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 infor-

mation 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.

1.2.1.1 EPP <domain:check> Command

The <idn:check> element, if present, allows to specify the language or script with the help of the <lang> or <script> elements. 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 language or script, it is marked as unavailable with the reason of "Invalid".

In "object mode", the name is reported as "In use", if an identical domain exists. This includes existing variants. If the name represents a non-existing variant of an existing domain, the name is reported as "Blocked", unless this domain has been registered by the same registrar that is submitting the check command. In this case, "Registrable variant" is reported. This is means that by using the correct contact and/or name server data, the name may be registered.

In "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.

1.2.1.2 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 inquired domain's current language or script is reported via the <lang> or <script> elements.

In "object mode", any existing variant name can be queried. The <idn:infData> contains a list of all other variants that belong to the same bundle as the queried domain.

In "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; if this is attempted, 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 from the response if the domain is an ASCII domain name and does not have any variant names.

1.2.1.3 EPP <poll> Command

This extension does not add any element to the EPP <poll> command itself. However, an additional <idn:trnData> element is supplied in the <poll> response in case of poll messages related to the transfer of domain objects, but

only in the "object mode". It contains a list of the other domains in the bundle that will be transferred to the gaining registrar.

1.2.1.4 EPP <transfer> Query Command

This extension does not add any element to the EPP <transfer> query command itself. Similar to the dual use of the <domain:trnData> element, the <idn:trnData> element is also used in the responses of a <transfer> command using the "query" operation. The semantics are the same as in the transfer poll message, i.e. in "object mode", it reports the names of the other variants of the bundle that will be or have been transferred along with the queried domain.

1.2.2 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.

1.2.2.1 EPP <domain:create> Command

The create command, which allows the registration of domain objects, or, during 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 <lang> or <script> elements, the used language or script of the name and variants is defined. The specification of the language or script is optional. If omitted, an ASCII domain name is assumed.

In the "object mode" for variants, variants may not be specified as part of the <idn:create> extension. Instead, further variants must be registered via separate <domain:create> commands, sharing common attributes as described above. If such a variant has been created successfully and becomes part of an existing bundle, the other domains of that bundle are reported with the help of the <idn:creData> element in the response of the create command. Note that such variants are charged separately.

In "attribute mode", along with the domain name itself, an unlimited number of variants may be specified.

1.2.2.2 EPP <domain:delete> Command

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

In "object mode", if a domain object is being deleted that is part of a bundle with at least of another variant, it is done immediately. This means that in no case the domain enters the redemption grace period. Since the other variant(s) is/are still held by the same registrant, there is no risk that someone else "captures" (and potentially abuses) the released name. In case of an inadvertent deletion, the deleted variant can immediately be re-registered by the registrar and registrant.

While the handling regarding the redemption period differs in this fashion, the add and renew grace periods still apply.

1.2.2.3 EPP <domain:renew> Command

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

Note that, in "object mode", variants are charged separately and can be renewed with different extension periods. The auto-renewal is performed on each variant independent from each other when it is due.

1.2.2.4 EPP <domain:transfer> Command

In the "object mode" for variants, the transfer command with all its operations ("request", "cancel", "approve", "reject") applies to all variants within a bundle. This means that if a transfer for a domain is started by the gaining registrar and approved by the losing registrar, all other variants in the bundle get transferred, too. The <idn:trnData> is included in the responses to inform the gaining and losing registrar about the names of these variants, in order to allow them to track these names in their own databases and also to make the gaining registrar aware of the costs, as the transfer of each variant is charged separately. During transfer, all variants will show the "pendingTransfer" status in the response to the <domain:info> command.

Note that, in any case, this is still a single transfer, meaning that poll messages will only be generated for the requested domain name. However, as a convenience, any other name of the bundle may be used in subsequent operations (e.g. for the approval), but this is not recommended.

In "attribute mode", the extension element <idn:trnData> is not used, as it applies only to the "object mode" variants model. There is no special behaviour of the transfer command regarding IDNs in "attribute mode".

1.2.2.5 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 <lang> or <script> elements, the language or script 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 language or script and a desired variant could not be registered.

A precondition for the successful execution of the change of the language or script is that the domain name and, in "object mode", the current variants or, in "attribute mode", the final variants (i.e. after applying any supplied variant additions or deletions within the same extension) are valid names for the new language or script.

In "attribute mode", 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.

In "object mode", if the domain is part of a bundle with other variants, the update command may have side effects on them. All changes of domain attributes that are marked as being shared in the table above are applied to the bundled domains as well. To allow registrars to keep their database in sync with these changes, the names of these affected domains are reported in the response using the <idn:updData> element. If only attributes are changed that are marked as individual per variant, the list of domains is omitted.

1.3 Formal Syntax (Schema Definition)

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

<schema targetNamespace="http://xmlns.tango-rs.net/epp/idn-1.0"
    xmlns:idn="http://xmlns.tango-rs.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>
```

1.4 Examples

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

1.4.1 EPP <check> Command

1.4.1.1 Example <check> command with language 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>xn--grn-ioa.tld</name>
```

```
C:      </check>
C:    </check>
C:    <extension>
C:      <check xmlns="http://xmlns.tango-rs.net/epp/idn-1.0">
C:        <lang>de</lang>
C:      </check>
C:    </extension>
C:    <clTRID>0168899153_1332496264546</clTRID>
C:  </command>
C:</epp>
```

1.4.2 EPP <info> Command

1.4.2.1 Example <info> response with language 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>xn--grn-ioa.tld</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.tango-rs.net/epp/idn-1.0">
S:        <lang>de</lang>
S:        <variants/>
S:      </infData>
S:    </extension>
S:    <trID>
S:      <svTRID>ZYX-99958</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

1.4.2.2 Example <info> response with language 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>xn--grn-ioa.tld</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.tango-rs.net/epp/idn-1.0">
S:        <lang>de</lang>
S:        <variants>
S:          <nameVariant>xn--grn-60a.tld</nameVariant>
S:          <nameVariant>xn--grn-eoa.tld</nameVariant>
S:        </variants>
S:      </infData>
S:    </extension>
S:    <trID>
S:      <svTRID>ZYX-99958</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

1.4.3 EPP <poll> Command

1.4.3.1 Example <poll> response for domain transfer with variants in "object mode":

```
S:<?xml version='1.0' encoding='UTF-8'?>
S:<epp xmlns='urn:ietf:params:xml:ns:epp-1.0'>
S:  <response>
S:    <result code='1301'>
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ count='12' id='4'>
S:      <qDate>2014-06-26T16:05:10.357Z</qDate>
```

```
S:    </msgQ>
S:    <resData>
S:      <trnData xmlns='urn:ietf:params:xml:ns:domain-1.0'>
S:        <name>xn--grn-ioa.tld</name>
S:        <trStatus>pending</trStatus>
S:        <reID>REG-123</reID>
S:        <reDate>2014-06-26T16:05:10.357Z</reDate>
S:        <acID>REG-234</acID>
S:        <acDate>2014-07-01T16:05:10.357Z</acDate>
S:        <exDate>2015-05-01T22:00:00.000Z</exDate>
S:      </trnData>
S:    </resData>
S:    <extension>
S:      <trnData xmlns="http://xmlns.tango-rs.net/epp/idn-1.0">
S:        <variants>
S:          <nameVariant>xn--grn-60a.tld</nameVariant>
S:          <nameVariant>xn--grn-eoa.tld</nameVariant>
S:        </variants>
S:      </trnData>
S:    </extension>
S:    <trID>
S:      <svTRID>ZYX-99958</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

1.4.4 EPP <create> Command

1.4.4.1 Example <create> command with language 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>xn--grn-ioa.tld</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.tango-rs.net/epp/idn-1.0">
C:        <lang>de</lang>
C:      </create>
C:    </extension>
C:    <clTRID>abc-00042</clTRID>
C:  </command>
C:</epp>
```

1.4.4.2 Example <create> response with domain name variants in "object mode":

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <creData xmlns="urn:ietf:params:xml:ns:domain-1.0">
S:        <name>xn--grn-ioa.tld</name>
S:        <crDate>2014-07-02T11:48:45.479Z</crDate>
S:        <exDate>2015-07-02T11:48:45.479Z</exDate>
S:      </creData>
S:    </resData>
S:    <extension>
S:      <creData xmlns="http://xmlns.tango-rs.net/epp/idn-1.0">
S:        <variants>
S:          <nameVariant>xn--grn-60a.tld</nameVariant>
S:          <nameVariant>xn--grn-eoa.tld</nameVariant>
S:        </variants>
S:      </creData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>1404301725479-7</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

1.4.4.3 Example <create> command with language tag and domain name variants in "attribute mode":

```
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>xn--grn-ioa.tld</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.tango-rs.net/epp/idn-1.0">
C:        <lang>de</lang>
C:        <variants>
C:          <nameVariant>xn--grn-60a.tld</nameVariant>
C:          <nameVariant>xn--grn-eoa.tld</nameVariant>
C:        </variants>
C:      </create>
```

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

1.4.5 EPP <transfer> Command

1.4.5.1 Example <transfer> response to a request starting a transfer with variants in "object mode":

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <trnData xmlns="urn:ietf:params:xml:ns:domain-1.0">
S:        <name>xn--grn-ioa.tld</name>
S:        <trStatus>pending</trStatus>
S:        <reID>REG-123</reID>
S:        <reDate>2014-07-02T11:16:14.652Z</reDate>
S:        <acID>REG-234</acID>
S:        <acDate>2014-07-07T11:16:14.652Z</acDate>
S:        <exDate>2018-06-27T07:17:38.652Z</exDate>
S:      </trnData>
S:    </resData>
S:    <extension>
S:      <trnData xmlns:idn="http://xmlns.tango-rs.net/epp/idn-1.0">
S:        <variants>
S:          <nameVariant>xn--grn-60a.tld</nameVariant>
S:          <nameVariant>xn--grn-eoa.tld</nameVariant>
S:        </variants>
S:      </trnData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>1404299774652-1</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

1.4.5.2 Example <transfer> response to a query of a transfer with variants in "object mode":

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <trnData xmlns="urn:ietf:params:xml:ns:domain-1.0">
S:        <name>xn--grn-ioa.tld</name>
S:        <trStatus>pending</trStatus>
S:        <reID>REG-123</reID>
S:        <reDate>2014-07-02T11:16:14.652Z</reDate>
S:        <acID>REG-234</acID>
```

```
S:        <acDate>2014-07-07T11:16:14.652Z</acDate>
S:        <exDate>2018-06-27T07:17:38.652Z</exDate>
S:      </trnData>
S:    </resData>
S:    <extension>
S:      <trnData xmlns="http://xmlns.tango-rs.net/epp/idn-1.0">
S:        <variants>
S:          <nameVariant>xn--grn-60a.tld</nameVariant>
S:          <nameVariant>xn--grn-eoa.tld</nameVariant>
S:        </variants>
S:      </trnData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>1404299774652-1</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

1.4.6 EPP <update> Command

1.4.6.1 Example <update> command changing the language 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>xn--grn-ioa.tld</name>
C:        <chg/>
C:      </update>
C:    </update>
C:    <extension>
C:      <update xmlns="http://xmlns.tango-rs.net/epp/idn-1.0">
C:        <chg>
C:          <lang>de</lang>
C:        </chg>
C:      </update>
C:    </extension>
C:    <clTRID>abc-00042</clTRID>
C:  </command>
C:</epp>
```

1.4.6.2 Example <update> response changing value affecting the variants in "object mode":

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <updData idn="http://xmlns.tango-rs.net/epp/idn-1.0">
S:        <variants>
S:          <nameVariant>xn--grn-60a.tld</nameVariant>
S:          <nameVariant>xn--grn-eoa.tld</nameVariant>
```

```
S:        </variants>
S:      </updData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>1404302604587-8</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

1.4.6.3 Example <update> command adding and removing domain name variants in "attribute mode":

```
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>xn--grn-ioa.tld</name>
C:        <chg/>
C:      </update>
C:    </update>
C:    <extension>
C:      <update xmlns="http://xmlns.tango-rs.net/epp/idn-1.0">
C:        <add>
C:          <nameVariant>xn--grn-5na.tld</nameVariant>
C:          <nameVariant>xn--grn-9na.tld</nameVariant>
C:        </add>
C:        <rem>
C:          <nameVariant>xn--grn-60a.tld</nameVariant>
C:          <nameVariant>xn--grn-eoa.tld</nameVariant>
C:        </rem>
C:      </update>
C:    </extension>
C:    <clTRID>abc-00042</clTRID>
C:  </command>
C:</epp>
```