Registry Fee Extension for the Extensible Provisioning Protocol (EPP)
draft-brown-epp-fees-07

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension mapping for registry fees.

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 October 10, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Table of Contents

1. Introduction .............................................. 3
   1.1. Conventions Used in This Document ................. 3
2. Migrating to Newer Versions of This Extension .......... 4
3. Extension Elements ......................................... 4
   3.1. Client Commands .................................... 4
   3.2. Currency Codes ................................... 5
   3.3. Validity Periods .................................. 5
   3.4. Fees and Credits ................................... 5
      3.4.1. Refunds ...................................... 6
      3.4.2. Grace Periods ................................. 6
      3.4.3. Correlation between Refundability and Grace Periods 7
      3.4.4. Applicability ................................ 7
   3.5. Account Balance ..................................... 7
   3.6. Credit Limit ....................................... 7
   3.7. Classification of Objects ......................... 8
4. Server Handling of Fee Information ...................... 8
5. EPP Command Mapping ........................................ 9
   5.1. EPP Query Commands ................................ 9
      5.1.1. EPP <check> Command .......................... 9
      5.1.1.1. Server Handling of <fee:class> Elements .... 12
      5.1.2. EPP Transfer Query Command ................... 13
   5.2. EPP Transform Commands .............................. 14
      5.2.1. EPP <create> Command .......................... 14
      5.2.2. EPP <delete> Command ......................... 17
      5.2.3. EPP <renew> Command ........................... 18
      5.2.4. EPP <transfer> Command ....................... 20
      5.2.5. EPP <update> Command .......................... 22
   5.3. Formal Syntax ...................................... 24
6. Security Considerations .................................... 29
7. IANA Considerations ....................................... 30
   7.1. XML Namespace ..................................... 30
   7.2. EPP Extension Registry ............................ 30
8. Implementation Status ..................................... 30
   8.1. RegistryEngine EPP Service ....................... 31
9. Acknowledgements .......................................... 31
10. Change History ........................................... 32
    10.1. Changes from 00 to 01 ............................ 32
    10.2. Changes from 01 to 02 ............................ 32
    10.3. Changes from 02 to 03 ............................ 33
    10.4. Changes from 03 to 04 ............................ 33
    10.5. Changes from 04 to 05 ............................ 34
    10.6. Changes from 05 to 06 ............................ 34
    10.7. Changes from 06 to 07 ............................ 35
11. Normative References ................................... 35
Authors’ Addresses .......................................... 36
1. Introduction

Historically, domain name registries have applied a simple fee structure for billable transactions, namely a basic unit price applied to domain <create>, <renew>, <transfer> and RGP [RFC3915] restore commands. Given the relatively small number of EPP servers to which EPP clients have been required to connect, it has generally been the case that client operators have been able to obtain details of these fees out-of-band by contacting the server operators.

Given the recent expansion of the DNS namespace, and the proliferation of novel business models, it is now desirable to provide a method for EPP clients to query EPP servers for the fees and credits associated with certain commands and specific objects.

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping provides a mechanism by which EPP clients may query the fees and credits associated with various billable transactions, and also obtain their current account balance.

1.1. Conventions Used in This Document

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 RFC 2119 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

"fee" is used as an abbreviation for "urn:ietf:params:xml:ns:fee-0.11". The XML namespace prefix "fee" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

(Note to RFC Editor: remove the following paragraph before publication as an RFC.)

The XML namespace prefix above contains a version number, specifically "0.11". This version number will increment with successive versions of this document, and will reach 1.0 if and when this document is published as an RFC. This permits clients to distinguish which version of the extension a server has implemented.
2. Migrating to Newer Versions of This Extension

(Note to RFC Editor: remove this section before publication as an RFC.)

Servers which implement this extension SHOULD provide a way for clients to progressively update their implementations when a new version of the extension is deployed.

Servers SHOULD (for a temporary migration period) provide support for older versions of the extension in parallel to the newest version, and allow clients to select their preferred version via the <svcExtension> element of the <login> command.

If a client requests multiple versions of the extension at login, then, when preparing responses to commands which do not include extension elements, the server SHOULD only include extension elements in the namespace of the newest version of the extension requested by the client.

When preparing responses to commands which do include extension elements, the server SHOULD only include extension elements for the extension versions present in the command.

3. Extension Elements

3.1. Client Commands

The <fee:command> element is used in the EPP <check> command to determine the fee which is applicable to the given command.

The element values permitted by the server is a matter of repository policy, but MUST include as a minimum the following values:

- "create" indicating a <create> command;
- "renew" indicating a <renew> command;
- "transfer" indicating a <transfer> command;

If the server supports the Registry Grace Period Mapping [RFC3915], then the server MUST also support the "restore" value.

The <fee:command> element MAY have an OPTIONAL "phase" attribute specifying a launch phase as described in [draft-ietf-eppext-launchphase]. It may also contain an OPTIONAL "subphase" attribute identifying the custom or sub-phase as described
3.2. Currency Codes

The `<fee:currency>` element is used to indicate which currency fees are charged in. This value of this element MUST be a three-character currency code from [ISO4217].

Note that ISO 4217 provides the special "XXX" code, which MAY be used if the server uses a non-currency based system for assessing fees, such as a system of credits.

The use of `<fee:currency>` elements in commands is OPTIONAL: if a `<fee:currency>` element is not present in a command, the server MUST determine the currency based on the client’s account settings which MUST be agreed by the client and server via an out-of-band channel. However, the `<fee:currency>` element MUST be present in responses.

Servers SHOULD NOT perform a currency conversion if a client uses an incorrect currency code. Servers SHOULD return a 2004 error instead.

3.3. Validity Periods

When querying for fee information using the `<check>` command, the `<fee:period>` element is used to indicate the units to be added to the registration period of objects by the `<create>`, `<renew>` and `<transfer>` commands. This element is derived from the `<domain:period>` element described in [RFC5731].

The `<fee:period>` element is OPTIONAL in `<check>` commands: if omitted, the server MUST determine the fee(s) using a validity period of 1 year. The `<fee:period>` element MUST be present in `<check>` responses.

3.4. Fees and Credits

Servers which implement this extension will include elements in responses which provide information about the fees and/or credits associated with a given billable transaction.

The `<fee:fee>` and `<fee:credit>` elements are used to provide this information. The presence of a `<fee:fee>` element in a response indicates a debit against the client’s account balance; a `<fee:credit>` element indicates a credit. A `<fee:fee>` element MUST have a non-negative value. A `<fee:credit>` element MUST have a negative value.

A server MAY respond with multiple `<fee:fee>` and `<fee:credit>` elements in the same response. In such cases, the net fee or credit
The following attributes are defined for the <fee:fee> element.
These are described in detail below:

description: an OPTIONAL attribute which provides a human-readable
description of the fee. Servers should provide documentation on
the possible values of this attribute, and their meanings.

refundable: an OPTIONAL boolean attribute indicating whether the fee
is refundable if the object is deleted.

grace-period: an OPTIONAL attribute which provides the time period
during which the fee is refundable.

applied: an OPTIONAL attribute indicating when the fee will be
deducted from the client’s account.

The <fee:credit> element can take a "description" attribute as
described above. No other attributes are defined for this element.

3.4.1. Refunds

<fee:fee> elements MAY have an OPTIONAL "refundable" attribute which
takes a boolean value. Fees may be refunded under certain
circumstances, such as when a domain application is rejected (as
described in [draft-ietf-eppext-launchphase]) or when an object is
deleted during the relevant Grace Period (see below).

If the "refundable" attribute is omitted, then clients SHOULD NOT
make any assumption about the refundability of the fee.

3.4.2. Grace Periods

[RFC3915] describes a system of "grace periods", which are time
periods following a billable transaction during which, if an object is
deleted, the client receives a refund.

The "grace-period" attribute MAY be used to indicate the relevant
grace period for a fee. If a server implements the Registry Grace
Period extension, it MUST specify the grace period for all relevant
transactions.

If the "grace-period" attribute is omitted, then clients SHOULD NOT
make any assumption about the grace period of the fee.
3.4.3. Correlation between Refundability and Grace Periods

If a <fee:fee> element has a "grace-period" attribute then it MUST also be refundable. If the "refundable" attribute of a <fee:fee> element is false then it MUST NOT have a "grace-period" attribute.

3.4.4. Applicability

Fees may be applied immediately upon receipt of a command from a client, or may only be applied once an out-of-band process (such as the processing of applications at the end of a launch phase) has taken place.

The "applied" attribute of the <fee:fee> element allows servers to indicate whether a fee will be applied immediately, or whether it will be applied at some point in the future. This attribute takes two possible values: "immediate" (which is the default) or "delayed".

3.5. Account Balance

The <fee:balance> element is an OPTIONAL element which MAY be included in server responses to transform commands. If present, it can be used by the client to determine the remaining credit at the server.

Whether or not the <fee:balance> is included in responses is a matter of server policy. However, if a server chooses to offer support for this element, it MUST be included in responses to all "transform" commands (i.e. <create>, <renew>, <update>, <delete>, <transfer op="request">).

The value of the <fee:balance> MAY be negative. A negative balance indicates that the server has extended a line of credit to the client (see below).

If a server includes a <fee:balance> element in response to transform commands, the value of the element MUST reflect the client's account balance after any fees or credits associated with that command have been applied.

3.6. Credit Limit

As described above, if a server returns a response containing a <fee:balance> with a negative value, then the server has extended a line of credit to the client. A server MAY also include a <fee:creditLimit> element in responses which indicates the maximum credit available to a client. A server MAY reject certain transactions if the absolute value of the <fee:balance> is equal to or exceeds the
value of the <fee:creditLimit> element.

Whether or not the <fee:creditLimit> is included in responses is a matter of server policy. However, if a server chooses to offer support for this element, it MUST be included in responses to all "transform" commands (ie <create>, <renew>, <update>, <delete>, <transfer op="request">).

3.7. Classification of Objects

Objects may be assigned to a particular class, category, or tier, each of which has a particular fee or set of fees associated with it. The <fee:class> element which appears in <check> responses is used to indicate the classification of an object.

If a server makes use of this element, it should provide clients with a list of all the values that the element may take via an out-of-band channel. Servers MUST NOT use values which do not appear on this list.

Servers which make use of this element MUST use a <fee:class> element with the value "standard" for all objects that are subject to the standard or default fee.

4. Server Handling of Fee Information

Depending on server policy, a client MAY be required to include the extension elements described in this document for certain transform commands. Servers must provide clear documentation to clients about the circumstances in which this extension must be used.

If a server receives a command from a client which does not include the extension elements required by the server for that command, then it MUST respond with a 2003 "Required parameter missing" error.

If the currency or total fee provided by the client do not agree with the server's own calculation of the fee for that command, then the server MUST reject the command with a 2004 "Parameter value range" error.

5. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in [RFC5730].
5.1. EPP Query Commands

This extension does not add any elements to the EPP <poll> or <info> commands or responses.

5.1.1. EPP <check> Command

This extension defines additional elements for the EPP <check> command.

The command MAY contain an <extension> element which MAY contain a <fee:check> element. The <fee:check> element contains the following child elements:

- A <fee:command> element;
- An OPTIONAL <fee:currency> element;
- An OPTIONAL <fee:period> element.
- An OPTIONAL <fee:class> element.

Example <check> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <check>
C:       <domain:check
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:name>example.net</domain:name>
C:         <domain:name>example.xyz</domain:name>
C:       </domain:check>
C:     </check>
C:     <extension>
C:       <fee:check xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
C:         <fee:command>create</fee:command>
C:         <fee:currency>USD</fee:currency>
C:       </fee:check>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

When the server receives a <check> command that includes the extension elements described above, its response MUST (subject to the exception described below) contain an <extension> element, which MUST
contain a child <fee:chkData> element. The <fee:chkData> element
MUST contain a <fee:cd> element for each object referenced in the
<check> element in the command.

The <fee:cd> element has an OPTIONAL "avail" attribute which is a
boolean. If the value of this attribute evaluates to false, this
indicates that the server cannot calculate the relevant fees, because
the object, command, currency, period or class is invalid according
to server policy.

The <fee:cd> contains the following child elements:

- A <fee:object> element, which contains a copy of the child element
  of the <check> element of the command, to which the fee
  information relates.

- A <fee:command> element, which contains the same command that
  appeared in the corresponding <fee:object> element. This element
  MAY have the OPTIONAL "phase" and "subphase" elements, which MUST
  match the same attributes in the corresponding <fee:object>
  element.

- A <fee:currency> element, which contains the same currency code
  that appeared in the <fee:currency> element of the command. If no
  <fee:currency> element appeared in the command, then the client’s
  default billing currency should be used.

- An OPTIONAL <fee:period> element, which contains the same unit
  that appeared in the <fee:currency> element of the command. If
  the value of the preceding <fee:command> element is "restore",
  this element MUST NOT be included. Otherwise it MUST be included.
  If no <fee:period> appeared in command (and the command is not
  "restore") then this element MUST have a value of 1 year.

- Zero or more <fee:fee> elements.

- Zero or more <fee:credit> elements.

- An OPTIONAL <fee:class> element.

- An OPTIONAL <fee:reason> element.

If no <fee:fee> elements are present in a <fee:cd> element, this
indicates that no fee will be assessed by the server for this
command.

If the "avail" attribute of the <fee:cd> element is false, then the
<fee:cd> element MUST NOT contain any <fee:fee> or <fee:credit> child
elements. If the "avail" attribute is true, then the <fee:cd> element MUST NOT contain a <fee:reason> element.

Example <check> response:

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
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:       <domain:chkData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:cd>
S:           <domain:name avail="1">example.com</domain:name>
S:         </domain:cd>
S:         <domain:cd>
S:           <domain:name avail="1">example.net</domain:name>
S:         </domain:cd>
S:         <domain:cd>
S:           <domain:name avail="1">example.xyz</domain:name>
S:         </domain:cd>
S:       </domain:chkData>
S:     </resData>
S:     <extension>
S:       <fee:chkData
S:         xmlns:fee="urn:ietf:params:xml:ns:fee-0.11"
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <fee:cd avail="1">
S:           <fee:object>
S:             <domain:name>example.com</domain:name>
S:           </fee:object>
S:           <fee:command>create</fee:command>
S:           <fee:currency>USD</fee:currency>
S:           <fee:period unit="y">1</fee:period>
S:           <fee:fee
S:             description="Registration Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:         </fee:cd>
S:         <fee:cd avail="1">
S:           <fee:object>
S:             <domain:name>example.com</domain:name>
S:           </fee:object>
S:           <fee:command>create</fee:command>
S:           <fee:currency>USD</fee:currency>
S:           <fee:period unit="y">1</fee:period>
S:           <fee:fee
S:             description="Registration Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:         </fee:cd>
S:       </fee:chkData>
S:   </response>
S: </epp>
5.1.1.1. Server Handling of <fee:class> Elements

Clients MAY include a <fee:class> in the <fee:check> element. There are three ways in which servers may handle this element:

1. If the server supports the concept of tiers or classes of objects, then the value of this element MUST be validated. If incorrect for the specified object, the "avail" attribute of the corresponding <fee:cd> element MUST be false.

2. If the server supports different "types" of object registrations (such as a "blocking" registration which does not resolve, or where a registry provides a value-added service that requires an opt-out to disable), then, as with the first model, the server MUST validate the value of the element. If the value is incorrect, the "avail" attribute of the corresponding <fee:cd> element MUST be false.

3. If the server supports neither of the above models, the element MUST be ignored.

Server operators must provide clear documentation to client operators which of the above models it supports.
5.1.2. EPP Transfer Query Command

This extension does not add any elements to the EPP <transfer> query command, but does include elements in the response, when the extension has been selected during a <login> command.

When the <transfer> query command has been processed successfully, the client selected the extension when it logged in, and the client is authorised by the server to view information about the transfer, the server MAY include in the <extension> section of the EPP response a <fee:trnData> element, which contains the following child elements:

- A <fee:currency> element.
- A <fee:period> element.
- Zero or more <fee:fee> elements containing the fees that will be charged to the gaining client.
- Zero or more <fee:credit> elements containing the credits that will be refunded to the losing client.

Servers SHOULD omit <fee:credit> when returning a response to the gaining client, and omit <fee:fee> elements when returning a response to the losing client.

If no <fee:trnData> element is included in the response, then no fee will be assessed by the server for the transfer.

Example <transfer> query response:
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1001">
      <msg>Command completed successfully; action pending</msg>
    </result>
    <resData>
      <domain:trnData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:trStatus>pending</domain:trStatus>
        <domain:reID>ClientX</domain:reID>
        <domain:reDate>2000-06-08T22:00:00.0Z</domain:reDate>
        <domain:acID>ClientY</domain:acID>
        <domain:acDate>2000-06-13T22:00:00.0Z</domain:acDate>
        <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
      </domain:trnData>
    </resData>
    <extension>
      <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
        <fee:currency>USD</fee:currency>
        <fee:period unit="y">1</fee:period>
        <fee:fee>5.00</fee:fee>
      </fee:trnData>
    </extension>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>

5.2. EPP Transform Commands

5.2.1. EPP <create> Command

This extension adds elements to both the EPP <create> command and response, when the extension has been selected during a <login> command.

When submitting a <create> command to the server, the client MAY include in the <extension> element a <fee:create> element which includes the following child elements:

- An OPTIONAL <fee:currency> element;
- One or more <fee:fee> elements.
When the `<create>` command has been processed successfully, and the client selected the extension when it logged in, and a fee was assessed by the server for the transaction, the server MUST include in the `<extension>` section of the EPP response a `<fee:creData>` element, which contains the following child elements:

- A `<fee:currency>` element;
- Zero or more `<fee:fee>` elements;
- Zero or more `<fee:credit>` elements;
- An OPTIONAL `<fee:balance>` element;
- An OPTIONAL `<fee:creditLimit>` element.

If no fee or credit has been assessed by the server for this transaction, a `<fee:creData>` element MUST NOT be included in the response.

Example `<create>` command:
Example <create> response:
5.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command, but does include elements in the response, when the extension has been selected during the <login> command.

When the <delete> command has been processed successfully, and the client selected the extension when it logged in, the server MAY include in the <extension> section of the EPP response a <fee:delData> element, which contains the following child elements:

- A <fee:currency> element;
- Zero or more <fee:credit> elements;
- An OPTIONAL <fee:balance> element;
- An OPTIONAL <fee:creditLimit> element.
If no credit has been assessed by the server for this transaction, a <fee:delData> element MUST NOT be included in the response.

Example <delete> response:

```xml
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
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:   </response>
S: </epp>
```

5.2.3. EPP <renew> Command

This extension adds elements to both the EPP <renew> command and response, when the extension has been selected during a <login> command.

When submitting a <renew> command to the server, the client MAY include in the <extension> element a <fee:renew> element which includes the following child elements:

- An OPTIONAL <fee:currency> element;
- One or more <fee:fee> elements.

When the <renew> command has been processed successfully, and the client selected the extension when it logged in, the server MAY include in the <extension> section of the EPP response a <fee:renData> element, which contains the following child elements:

- A <fee:currency> element;
Zero or more <fee:fee> elements;

Zero or more <fee:credit> elements;

An OPTIONAL <fee:balance> element;

An OPTIONAL <fee:creditLimit> element.

If no fee or credit has been assessed by the server for this transaction, a <fee:renData> element MUST NOT be included in the response.

Example <renew> command:

C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <renew>
C:       <domain:renew
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:curExpDate>2000-04-03</domain:curExpDate>
C:         <domain:period unit="y">5</domain:period>
C:       </domain:renew>
C:     </renew>
C:     <extension>
C:       <fee:renew xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:renew>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>

Example <renew> response:
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <domain:renData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
      </domain:renData>
    </resData>
    <extension>
      <fee:renData xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
        <fee:currency>USD</fee:currency>
        <fee:fee grace-period="P5D">5.00</fee:fee>
        <fee:balance>1000.00</fee:balance>
      </fee:renData>
    </extension>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>

5.2.4. EPP <transfer> Command

This extension adds elements to both the EPP <transfer> command and response, when the value of the "op" attribute of the <transfer> command element is "request", and the extension has been selected during the <login> command.

When submitting a <transfer> command to the server, the client MAY include in the <extension> element a <fee:transfer> element which includes the following child elements:

- An OPTIONAL <fee:currency> element;
- One or more <fee:fee> elements.

When the <transfer> command has been processed successfully, and the client selected the extension when it logged in, the server MAY include in the <extension> section of the EPP response a <fee:trnData> element, which contains the following child elements:
A <fee:currency> element;
zero or more <fee:fee> elements;
zero or more <fee:credit> elements;
An OPTIONAL <fee:balance> element;
An OPTIONAL <fee:creditLimit> element.

If no fee or credit has been assessed by the server for this transaction, a <fee:trnData> element MUST NOT be included in the response.

Example <transfer> command:

C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <transfer op="request">
C:       <domain:transfer
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:period unit="y">1</domain:period>
C:         <domain:authInfo>
C:           <domain:pw roid="JD1234-REP">2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:transfer>
C:     </transfer>
C:   </command>
C: </epp>

Example <transfer> response:
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
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:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2000-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2000-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee grace-period="P5D">5.00</fee:fee>
S:       </fee:trnData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

5.2.5. EPP <update> Command

This extension adds elements to both the EPP <update> command and response, when the extension has been selected during a <login> command.

When submitting a <update> command to the server, the client MAY include in the <extension> element a <fee:update> element which includes the following child elements:

- An OPTIONAL <fee:currency> element;
- One or more <fee:fee> elements.

When the <update> command has been processed successfully, and the client selected the extension when it logged in, the server MAY include in the <extension> section of the EPP response a <fee:upData>
element, which contains the following child elements:

- A `<fee:currency>` element;
- Zero or more `<fee:fee>` elements;
- Zero or more `<fee:credit>` elements;
- An OPTIONAL `<fee:balance>` element;
- An OPTIONAL `<fee:creditLimit>` element.

If no fee or credit has been assessed by the server for this transaction, a `<fee:upData>` element MUST NOT be included in the response.

Example `<update>` command:

```xml
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <update>
      <domain:update xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:chg>
          <domain:registrant>sh8013</domain:registrant>
        </domain:chg>
      </domain:update>
    </update>
    <extension>
      <fee:update xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
        <fee:currency>USD</fee:currency>
        <fee:fee>5.00</fee:fee>
      </fee:update>
    </extension>
  </command>
  <clTRID>ABC-12345</clTRID>
</epp>
```

Example `<update>` response:
S: <xml version="1.0" encoding="utf-8" standalone="no"/>
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:       <fee:updData xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee>5.00</fee:fee>
S:       </fee:updData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

5.3. Formal Syntax

An EPP object mapping is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances.

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BEGIN
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:fee="urn:ietf:params:xml:ns:fee-0.11"
   xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
   xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
   targetNamespace="urn:ietf:params:xml:ns:fee-0.11"
   elementFormDefault="qualified">

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

<annotation>
   <documentation>Extensible Provisioning Protocol v1.0 extension schema for fee information.</documentation>
</annotation>

<!-- Child elements found in EPP commands and responses -->
<br> <element name="check" type="fee:checkType" />
<element name="chkData" type="fee:chkDataType" />
<element name="create" type="fee:transformCommandType" />
<element name="creData" type="fee:transformResultType" />
<element name="renew" type="fee:transformCommandType" />
<element name="renData" type="fee:transformResultType" />
<element name="transfer" type="fee:transformCommandType" />
<element name="trnData" type="fee:transferResultType" />
<element name="update" type="fee:transformCommandType" />
<element name="updData" type="fee:transformResultType" />
<element name="delData" type="fee:deleteDataType" />
</br>
<!-- client <check> command -->
<complexType name="checkType">
  <sequence>
</complexType>
<element name="command" type="fee:commandType" />
<element name="currency" type="fee:currencyType" minOccurs="0" />
<element name="period" type="domain:periodType" minOccurs="0" />
<element name="class" type="token" minOccurs="0" />
</sequence>
</complexType>

<!-- server <check> result -->
<complexType name="chkDataType">
  <sequence>
    <element name="cd" type="fee:objectCDType" maxOccurs="unbounded" />
  </sequence>
</complexType>

<complexType name="objectCDType">
  <sequence>
    <element name="object">
      <complexType>
        <sequence>
          <any namespace="##other" processContents="lax"/>
        </sequence>
      </complexType>
    </element>
    <element name="command" type="fee:commandType" />
    <element name="currency" type="fee:currencyType" />
    <element name="period" type="domain:periodType" minOccurs="0" maxOccurs="1" />
    <element name="fee" type="fee:feeType" minOccurs="0" maxOccurs="unbounded" />
    <element name="credit" type="fee:creditType" minOccurs="0" maxOccurs="unbounded" />
    <element name="class" type="token" minOccurs="0" />
    <element name="reason" type="token" minOccurs="0" />
  </sequence>
  <attribute name="avail" type="boolean" default="1" />
</complexType>

<!-- general transform (create, renew, update, transfer) command -->
<complexType name="transformCommandType">
  <sequence>

Brown & Frakes Expires October 10, 2016 [Page 26]
<element name="currency" type="fee:currencyType"
  minOccurs="0" />
<element name="fee" type="fee:feeType"
  maxOccurs="unbounded" />
<element name="credit" type="fee:creditType"
  minOccurs="0" maxOccurs="unbounded" />
</sequence>
</complexType>

<!--
general transform (create, renew, update) result
-->
<complexType name="transformResultType">
  <sequence>
    <element name="currency" type="fee:currencyType" />
    <element name="fee" type="fee:feeType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="credit" type="fee:creditType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="balance" type="fee:balanceType"
      minOccurs="0" />
    <element name="creditLimit" type="fee:creditLimitType"
      minOccurs="0" />
  </sequence>
</complexType>

<!--
transfer result
-->
<complexType name="transferResultType">
  <sequence>
    <element name="currency" type="fee:currencyType" />
    <element name="period" type="domain:periodType"
      minOccurs="0" />
    <element name="fee" type="fee:feeType"
      maxOccurs="unbounded" />
    <element name="credit" type="fee:creditType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

<!--
delete result
-->
<complexType name="deleteDataType">
<sequence>
  <element name="currency" type="fee:currencyType" />
  <element name="credit" type="fee:creditType"
    minOccurs="0" maxOccurs="unbounded" />
  <element name="balance" type="fee:balanceType"
    minOccurs="0" />
  <element name="creditLimit" type="fee:creditLimitType"
    minOccurs="0" />
</sequence>
</complexType>

<!--
  common types
-->
<simpleType name="currencyType">
  <restriction base="string">
    <pattern value="[A-Z]{3}" />
  </restriction>
</simpleType>

<complexType name="commandType">
  <simpleContent>
    <extension base="fee:commandTypeValue">
      <attribute name="phase" type="token" />
      <attribute name="subphase" type="token" />
    </extension>
  </simpleContent>
</complexType>

<simpleType name="commandTypeValue">
  <restriction base="token">
    <minLength value="3"/>
    <maxLength value="16"/>
  </restriction>
</simpleType>

<simpleType name="nonNegativeDecimal">
  <restriction base="decimal">
    <minInclusive value="0" />
  </restriction>
</simpleType>

<simpleType name="negativeDecimal">
  <restriction base="decimal">
    <maxInclusive value="0" />
  </restriction>
</simpleType>
6. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730], the EPP domain name mapping [RFC5731], and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.
7. IANA Considerations

7.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

URI: urn:ietf:params:xml:ns:fee-0.11

Registrant Contact: See the "Author’s Address" section of this document.

XML: See the "Formal Syntax" section of this document.

7.2. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: EPP Fee Extension

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: See the "Author’s Address" section of this document.

TLDs: any

IPR Disclosure: none

Status: active

Notes: none

8. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to
assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

8.1. RegistryEngine EPP Service

Organization: CentralNic

Name: RegistryEngine EPP Service

Description: Generic high-volume EPP service for gTLDs, ccTLDs and SLDs

Level of maturity: Deployed in CentralNic’s production environment as well as two other gTLD registry systems, and two ccTLD registry systems.

Coverage: All aspects of the protocol are implemented.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: https://www.centralnic.com

9. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions:

- James Gould of Verisign
- Luis Munoz of ISC
10. Change History

10.1. Changes from 00 to 01

1. Restore the <check> command extension; either <check> or <info> can be used.

2. added extension elements for <create>, <renew>, <transfer> and <update> so that the server can reject the command if the fee is incorrect.

10.2. Changes from 01 to 02

1. Use Internet-Draft version number rather than XML namespace version number in this section.

2. Support for multiple <fee:fee> and <fee:credit> elements.

3. Added the "description" attribute to <fee:fee> and <fee:credit> elements.

4. Added the <fee:balance> element.

5. Added the <fee:creditLimit> element.

6. Updated reference to [draft-ietf-eppext-launchphase].

7. Use <fee:command> instead of <fee:action>.

8. Use a single child element of <fee:chkData> instead of multiple elements for each domain. This also requires using a different
name (<fee:name>) for the domain name.

9. Added the "refundable" attribute to <fee:fee> elements.

10. Added the "grace-period" attribute to <fee:fee> elements.

10.3. Changes from 02 to 03

1. Added the "applied" attribute to <fee:fee> elements.

2. Simplified the wording in relation to when a server can return an error for extended <info> commands.

3. Added the <fee:period> element to transfer query responses.

4. Removed wording about how servers behave when receiving incorrect fee information from transform commands, and put it into a single section at the top of the document.

5. Allow servers to omit <fee:fee> elements from <fee:cd> elements if the command specified by the client is forbidden.

10.4. Changes from 03 to 04

1. Changed Intended Status to Standards Track.

2. As per suggestion from Michael Bauland, the <fee:period> element is no longer included in <check> and <info> responses for "restore" commands. It's still mandatory for all other commands.

3. Added summary of the attributes for the <fee:fee> element.

4. Clarified that the "refundable" and "grace-period" attributes of the <fee:fee> elements are dependant on each other and cannot appear on their own.

5. Removed the option of returning a 1001 response when the fee is incorrect.

6. Forbidden the inclusion of extension elements in transform responses if no fee/credit has been assessed.

7. Made the <fee:currency> element optional in transform commands.

8. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.
10.5. Changes from 04 to 05

1. Removed the extended <info> command. The <check> command is the only command that can be used now.

2. Introduced a mandatory-to-implement "standard" class for non-premium domains.

3. The decision was made to keep availability info in <check> responses as registrars have indicated that it is very useful as it avoids unnecessary round trips to the server.

4. Allow <fee:credit> elements to be present in <check> responses.

5. Allow the number of <fee:fee> which can appear in transform responses to be zero.

6. Removed the <fee:balance> and <fee:creditLimit> elements from transfer query responses. The reason is that these elements are defined as containing the values after the transform command has taken place - which means that it is not appropriate to include them in a query response.

7. Added Implementation Status section.

10.6. Changes from 05 to 06

1. The specification is now object-agnostic, but works with RFC5731 [RFC5731] domains by default.

2. Renamed the <fee:domain> element to <fee:object>. Added the "objURI" attribute.

3. Removed the default value for the "refundable" attribute of <fee:fee> elements, and added text about how clients should handle such cases. Added similar text to the documentation of the "grace-period" attribute.

4. Removed references to the defunct <info> command syntax.

5. "MUST" requirements regarding documentation have been changed to "must".

6. Created separate "Correlation between Refundability and Grace Periods" section describing how the "refundable" and "grace-period" attributes work together.
10.7. Changes from 06 to 07

1. Changed the syntax of the <check> form to be simpler: a single set of <fee:command>, <fee:currency> etc elements is applied to the objects specified in the main body of the <check> command.

2. Simplified the object-agnosticism to simply copy the element from the <check> command into the <fee:cd> element.

3. Added the "avail" attribute to the <fee:cd> element and added commentary about its semantics.

4. Added the <fee:reason> element to the <check> response so servers can indicate why fee information is not available.

11. Normative References


Authors’ Addresses

Gavin Brown
CentralNic Group plc
35-39 Moorgate
London, England EC2R 6AR
GB

Phone: +44 20 33 88 0600
Email: gavin.brown@centralnic.com
URI: https://www.centralnic.com

Jothan Frakes
Email: jothan@jothan.com
URI: http://jothan.com
Allocation Token Extension for the Extensible Provisioning Protocol (EPP)
draft-gould-allocation-token-04

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension for including an allocation token or code for allocating an object like a domain name to the client. The allocation token MAY be transferred out-of-band to a client to give them authorization to allocate an object using one of the EPP transform commands including create, update, and transfer.

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 November 24, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must
This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], for passing an allocation token one of the EPP transform commands including create, update, and transfer. The allocation token is known to the server to authorize a client that passes a matching allocation token with one of the supported EPP transform commands. It is up to server policy which EPP transform commands and which objects support the allocation token. The allocation token MAY
be returned to an authorized client for passing out-of-band to a client that uses it with an EPP transform command.

1.1. Conventions Used in This Document

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 RFC 2119 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this protocol.

"allocationToken-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:allocationToken-1.0". The XML namespace prefix "allocationToken" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Object Attributes

This extension adds additional elements to EPP object mappings like the EPP domain name mapping [RFC5731]. Only those new elements are described here.

2.1. Allocation Token

The Allocation Token is a simple XML "token" type. The exact format of the Allocation Token is up to server policy. The server MUST have the allocation token for each object to match against the allocation token passed by the client to authorize the allocation of the object. The same <allocationToken:allocationToken> element is used for all of the supported EPP transform commands as well as the info response. If an invalid allocation token is passed the server MUST return an EPP error result code of 2201.
An example `<allocationToken:allocationToken>` element with value of "abc123":

```xml
<allocationToken:allocationToken xmlns:allocationToken=
    "urn:ietf:params:xml:ns:allocationToken-1.0">
  abc123
</allocationToken:allocationToken>
```

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730].

3.1. EPP Query Commands

EPP provides three commands to retrieve object information: `<check>` to determine if an object is known to the server, `<info>` to retrieve detailed information associated with an object, and `<transfer>` to retrieve object transfer status information.

3.1.1. EPP `<check>` Command

This extension defines additional elements to extend the EPP `<check>` command of an object mapping like [RFC5731].

This extension allow clients to check the availability of an object with an allocation token, as described in Section 2.1. Clients can check if an object can be created using the allocation token. The allocation token is applied to all object names included in the EPP `<check>` command.
Example <check> command for the example.tld domain name using the <allocationToken:allocationToken> extension with the allocation token of ‘abc123’:

C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <domain:check
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.tld</domain:name>
C:      </domain:check>
C:    </check>
C:    <extension>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken=
C:          "urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

If the query was successful, the server replies with an <check> response providing availability status of queried object.
Example <check> domain response for a <check> command using the <allocationToken:allocationToken> extension:

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:     <domain:chkData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:       <domain:cd>
S:         <domain:name avail="0">example.tld</domain:name>
S:         <domain:reason>Invalid domain-token pair</domain:reason>
S:       </domain:cd>
S:     </domain:chkData>
S:   </resData>
S:   <trID>
S:     <clTRID>ABC-DEF-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S:   </trID>
S: </response>
S:</epp>
Example <check> command with the <allocationToken:allocationToken> extension for the example.tld and example2.tld domain names. Availability of example.tld and example2.tld domain names are based on the allocation token ‘abc123’:

C: <?xml version="1.0" encoding="UTF-8"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <check>
C:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:       <domain:name>example.tld</domain:name>
C:       <domain:name>example2.tld</domain:name>
C:   </domain:check>
C:   <extension>
C:     xmlns:allocationToken="urn:ietf:params:xml:ns:allocationToken-1.0">
C:       abc123
C:     </allocationToken:allocationToken>
C:   </extension>
C:  </command>
C: </epp>
Example <check> domain response for multiple domain names in the <check> command using the <allocationToken:allocationToken> extension:

```
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:     <domain:chkData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:       <domain:cd>
S:         <domain:name avail="0">example.tld</domain:name>
S:         <domain:reason>Invalid domain-token pair</domain:reason>
S:       </domain:cd>
S:       <domain:cd>
S:         <domain:name avail="1">example2.tld</domain:name>
S:       </domain:cd>
S:     </domain:chkData>
S:   </resData>
S:  <trID>
S:    <clTRID>ABC-DEF-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

This extension does not add any elements to the EPP <check> response described in the [RFC5730].

3.1.2. EPP <info> Command

This extension defines additional elements to extend the EPP <info> command of an object mapping like [RFC5731].

The EPP <info> command allows a client to request information on an existing object. Authorized clients MAY retrieve the allocation token (Section 2.1) along with the other object information using the <allocationToken:info> element that identifies the extension namespace. The <allocationToken:info> element is an empty element that serves as a marker to the server to return the <allocationToken:allocationToken> element, defined in Section 2.1, in the info response. If the client is not authorized to receive the allocation token (Section 2.1), the server MUST return an EPP error result code of 2201. If the client is authorized to receive the
allocation token (Section 2.1), but there is no allocation token (Section 2.1) associated with the object, the server MUST return an EPP error result code of 2303 object referencing the <allocationToken:info> element.

Example <info> command with the allocationToken:info extension for the example.tld domain name:

```xml
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <info>
C:       <domain:info
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
C:         xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0
domain-1.0.xsd">
C:         <domain:name>example.tld</domain:name>
C:       </domain:info>
C:     </info>
C:     <extension>
C:       <allocationToken:info
C:         xmlns:allocationToken=
C:         "urn:ietf:params:xml:ns:allocationToken-1.0/>
C:     </extension>
C:   </command>
C: <clTRID>ABC-12345</clTRID>
C: </epp>
```

If the query was successful, the server replies with an <allocationToken:allocationToken> element, as described in Section 2.1.
Example <info> domain response using the
<allocationToken:allocationToken> extension:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.tld</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="pendingCreate"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <allocationToken:allocationToken
S:        xmlns:allocationToken=
S:          "urn:ietf:params:xml:ns:allocationToken-1.0">
S:        abc123
S:      </allocationToken:allocationToken>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

3.1.3. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> query
command or <transfer> response described in the [RFC5730].
3.2. EPP Transform Commands

EPP provides five 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.

3.2.1. EPP <create> Command

This extension defines additional elements to extend the EPP <create> command of an object mapping like [RFC5731].

The EPP <create> command provides a transform operation that allows a client to create an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MUST contain a child <allocationToken:allocationToken> element, as defined in Section 2.1, that identifies the extension namespace for the client to be authorized to create and allocate the object. If the allocation token (Section 2.1) does not match the object’s allocation token (Section 2.1), the server MUST return an EPP error result code of 2201.
Example <create> command to create a domain object with an allocation token:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?><!--
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.tld</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken="urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:  </command>
C:</epp>
```

This extension does not add any elements to the EPP <create> response described in the [RFC5730].

3.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the [RFC5730].

3.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the [RFC5730].

3.2.4. EPP <transfer> Command

This extension defines additional elements to extend the EPP <transfer> request command of an object mapping like [RFC5731].
The EPP <transfer> request command provides a transform operation that allows a client to request the transfer of an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MUST contain a child <allocationToken:allocationToken> element, as defined in Section 2.1, that identifies the extension namespace for the client to be authorized to transfer and allocate the object. If the allocation token (Section 2.1) does not match the object’s allocation token (Section 2.1), the server MUST return an EPP error result code of 2201:

Example <transfer> request command to allocate the domain object with the allocation token:

```xml
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example1.tld</domain:name>
C:        <domain:period unit="y">1</domain:period>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:      </domain:authInfo>
C:    </domain:transfer>
C:    <extension>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken="urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:  </command>
C:</epp>
```

This extension does not add any elements to the EPP <transfer> response described in the [RFC5730].

3.2.5. EPP <update> Command

This extension defines additional elements to extend an extension of an empty EPP <update> command of an object mapping like [RFC5731]. An example of an extension of an empty EPP <update> command is the definition of the restore command within [RFC3915].
An extension of an empty EPP `<update>` command defines a new verb that transforms an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MUST contain a child `<allocationToken:allocationToken>` element, as defined in Section 2.1, that identifies the extension namespace for the client to be authorized to allocate the object. If the allocation token (Section 2.1) does not match the object’s allocation token (Section 2.1), the server MUST return an EPP error result code of 2201.:

Example use an extension of an empty `<update>` command to release a domain object with an allocation token:

```xml
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example1.tld</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <release:release
C:        xmlns:release="urn:ietf:params:xml:ns:release-1.0"/>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken=
C:          "urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

This extension does not add any elements to the EPP `<update>` response described in the [RFC5730].

### 4. Formal Syntax

One schema is presented here that is the EPP Allocation Token Extension schema.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.
4.1. Allocation Token Extension Schema

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

<schema targetNamespace="urn:ietf:params:xml:ns:allocationToken-1.0"
   xmlns:allocationToken="urn:ietf:params:xml:ns:allocationToken-1.0"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Allocation Token Extension.
    </documentation>
  </annotation>

  <!-- Element used in info command to get allocation token. -->
  <element name="info"/>

  <!-- Allocation Token used in transform commands and info response -->
  <element name="allocationToken"
    type="allocationToken:allocationTokenType"/>

  <complexType name="allocationTokenType">
    <simpleContent>
      <extension base="token"/>
    </simpleContent>
  </complexType>

  <!-- End of schema.-->
</schema>
END

5. IANA Considerations

5.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

URI: ietf:params:xml:ns:allocationToken-1.0

Registrant Contact: See the "Author’s Address" section of this document.
XML: See the "Formal Syntax" section of this document.

5.2. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Allocation Token Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track
Reference: (insert reference to RFC version of this document)
Registrant Name and Email Address: IESG, <iesg@ietf.org>
TLDs: Any
IPR Disclosure: None
Status: Active
Notes: None

6. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730] and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

7. Acknowledgements

The authors wish to acknowledge the original concept for this draft and the efforts in the initial versions of this draft by Trung Tran.

8. Normative References


Appendix A. Change History

A.1. Change from 00 to 01
1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.
2. Moved Change History to the back section as an Appendix.

A.2. Change from 01 to 02
1. Ping update.

A.3. Change from 02 to 03
1. Ping update.

A.4. Change from 03 to 04
1. Updated the authors for the draft.

Authors’ Addresses

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US

Email: jgould@verisign.com
URI: http://www.verisigninc.com
Sharon Wodjenski  
Neustar  
21575 Ridgetop Circle  
Sterling, VA  20166  
US  

Email: Sharon.Wodjenski@neustar.biz  
URI:  http://www.neustar.biz
Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension for notifying clients of operations on client sponsored objects that were not initiated by the client through EPP. These operations MAY include contractual or policy requirements including but not limited to regular batch processes, customer support actions, Uniform Domain-Name Dispute-Resolution Policy (UDRP) or Uniform Rapid Suspension (URS) actions, court directed actions, and bulk updates based on customer requests. Since the client is not directly involved or knowledgable of these operations, the extension is used along with an EPP object mapping to provide the resulting state of the post-operation object, and optionally a pre-operation object, with the operation meta-data of what, when, who, and why.

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 April 2, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ........................................ 3
   1.1. Conventions Used in This Document ............ 3
2. Object Attributes .................................. 3
   2.1. Operation .................................. 4
   2.2. Who .................................. 4
3. EPP Command Mapping ................................ 5
   3.1. EPP Query Commands .............................. 5
       3.1.1. EPP <check> Command ......................... 5
       3.1.2. EPP <info> Command .......................... 5
       3.1.3. EPP <transfer> Command ..................... 15
   3.2. EPP Transform Commands ........................ 15
       3.2.1. EPP <create> Command ......................... 15
       3.2.2. EPP <delete> Command ......................... 15
       3.2.3. EPP <renew> Command ........................ 15
       3.2.4. EPP <transfer> Command ...................... 15
       3.2.5. EPP <update> Command ......................... 15
4. Formal Syntax ....................................... 15
   4.1. Change Poll Extension Schema ................. 16
5. IANA Considerations ................................ 18
   5.1. XML Namespace ................................ 18
   5.2. EPP Extension Registry ......................... 19
6. Security Considerations ............................ 19
7. Acknowledgements ................................... 19
8. Normative References ................................ 19
Appendix A. Change History ............................ 20
   A.1. Change from 00 to 01 ............................ 20
   A.2. Change from 01 to 02 ............................ 20
   A.3. Change from 02 to 03 ............................ 20
   A.4. Change from 03 to 04 ............................ 20
   A.5. Change from 04 to 05 ............................ 20
Authors’ Addresses .................................... 20
1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], is used to notify clients of operations they are not directly involved in, on objects that the client sponsors. It is up to server policy to determine what transform operations and clients to notify. Using this extension clients can more easily keep their systems in-sync with the objects stored in the server. When a change occurs that a client needs to be notified of, a poll message can be inserted by the server for consumption by the client using the EPP <poll> command and response defined in [RFC5730]. The extension supports including a "before" operation poll message and an "after" operation poll message.

1.1. Conventions Used in This Document

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 RFC 2119 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this protocol.

"changePoll-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:changePoll-1.0". The XML namespace prefix "changePoll" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Object Attributes

This extension adds additional elements to EPP object mappings like the EPP domain name mapping [RFC5731]. Only those new elements are described here.
2.1.  Operation

An operation consists of any transform operation that impacts objects that the client sponsors and SHOULD be notified of. The <changePoll:operation> element defines the operation. The OPTIONAL "op" attribute is used to define a sub-operation or the name of a "custom" operation. The enumerated list of <changePoll:operation> values include:

"create"  Create operation as defined in [RFC5730].
"delete"  Delete operation as defined in [RFC5730]. If the delete operation results in an immediate purge of the object, then the "op" attribute MUST be set to "purge".
"renew"  Renew operation as defined in [RFC5730].
"transfer"  Transfer operation as defined in [RFC5730] with the OPTIONAL "op" attribute defining the transfer type with the possible values of "request", "approve", "cancel", and "reject".
"update"  Update operation as defined in [RFC5730].
"restore"  Restore operation as defined in [RFC3915] with the OPTIONAL "op" attribute defining the restore type with the possible values of "request" and "report".
"autoRenew"  Auto renew operation executed by the server.
"autoDelete"  Auto delete operation executed by the server. If the "autoDelete" operation results in an immediate purge of the object, then the "op" attribute MUST be set to "purge".
"autoPurge"  Auto purge operation executed by the server when removing the object after it had the "pendingDelete" status.
"custom"  Custom operation that uses the "op" attribute to define the custom operation name.

2.2.  Who

Who defines who executed the operation for audit purposes, and is represented using the <changePoll:who> element. The scheme used for the possible set of Who values is up to server policy. The server MAY identify Who based on:

"Identifier"  Unique user identifier of the user that executed the operation. An example is "ClientX".
"Name"  Name of the user that executed the operation. An example is "John Doe".
"Role"  Role of the user that executed operation. An example is "CSR" for a Customer Support Representative or "Batch" for a server batch.
3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730].

3.1. EPP Query Commands

EPP provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve detailed information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

This extension does not add any elements to the EPP <check> command or <check> response described in the [RFC5730].

3.1.2. EPP <info> Command

This extension does not add any elements to the EPP <info> command described in the [RFC5730].

This extension adds transaction detail of the operations to the EPP <info> poll response, as described in [RFC5730], of an EPP Object Mapping like [RFC5731]. Any transform operation to an object defined in an EPP Object Mapping, by a client other than the sponsoring client, MAY result in extending the <info> response of the object for inserting an EPP poll message with the operation detail. The sponsoring client will then receive the state of the object with operation detail like what, who, when, and why the object was changed. The <changePoll:changeData> element contains the operation detail along with an indication of whether the object reflects the state before or after the operation, using the OPTIONAL "state" attribute, with the possible values of "before" or "after", and with a default value of "after". The "state" attribute describes the state of the response data or <resData> block returned in the poll response. The server MAY support providing the "before" state and "after" state to the operation, by using one poll message for the "before" state and one poll message for the "after" state. When using the "before" state poll message, it MUST be inserted prior to the "after" state poll message. The <changePoll:changeData> element includes the operation detail with the following child elements:

- <changePoll:operation> Transform operation executed on the object as defined in Section 2.1.
- <changePoll:date> Date and time when the operation was executed.
- <changePoll:svTRID> Server transaction identifier of the operation.
<changePoll:who>  Who executed the operation as defined in Section 2.2.
<changePoll:caseId>  OPTIONAL case identifier associated with the operation. The required "type" attribute defines the type of case with an enumerated list of case types including:

  udrp  a Uniform Domain-Name Dispute-Resolution Policy (UDRP) case.
  urs  a Uniform Rapid Suspension (URS) case.
  custom  A custom case that is defined using the "name" attribute.
<changePoll:reason>  OPTIONAL reason for executing the operation. If present, this element contains the server-specific text to help explain the reason the operation was executed. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

Example poll <info> response with the <changePoll:changeData> extension for a URS lock transaction on the domain.example domain name, with the "before" state. The "before" state is reflected in the <resData> block:
Example poll <info> response with the <changePoll:changeData> extension for a URS lock transaction on the domain.example domain name, with the "after" state. The "after" state is reflected in the
<resData> block:
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg lang="en-US">Command completed successfully; ack to dequeue</msg>
    </result>
    <msgQ id="202" count="1">
      <qDate>2013-10-22T14:25:57.0Z</qDate>
      <msg>Registry initiated update of domain.</msg>
    </msgQ>
    <resData>
      <domain:infData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>domain.example</domain:name>
        <domain:roid>EXAMPLE1-REP</domain:roid>
        <domain:status s="serverUpdateProhibited"/>
        <domain:status s="serverDeleteProhibited"/>
        <domain:status s="serverTransferProhibited"/>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:clID>ClientX</domain:clID>
        <domain:crID>ClientY</domain:crID>
        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
        <domain:upID>ClientZ</domain:upID>
        <domain:upDate>2013-10-22T14:25:57.0Z</domain:upDate>
        <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
      </domain:infData>
    </resData>
    <extension>
      <changePoll:changeData
        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0" state="after">
        <changePoll:operation>update</changePoll:operation>
        <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
        <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
        <changePoll:who>URS Admin</changePoll:who>
        <changePoll:caseId type="urs">urs123</changePoll:caseId>
        <changePoll:reason>URS Lock</changePoll:reason>
      </changePoll:changeData>
    </extension>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
Example poll <info> response with the <changePoll:changeData> extension for a custom "sync" operation on the domain.example domain name, with the default "after" state. The "after" state is reflected in the <resData> block:
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 id="201" count="1">
S:         <qDate>2013-10-22T14:25:57.0Z</qDate>
S:         <msg>Registry initiated Sync of Domain Expiration Date</msg>
S:      </msgQ>
S:      <resData>
S:         <domain:infData
S:          xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:          <domain:name>domain.example</domain:name>
S:          <domain:roid>EXAMPLE1-REP</domain:roid>
S:          <domain:status s="ok"/>
S:          <domain:registrant>jd1234</domain:registrant>
S:          <domain:contact type="admin">sh8013</domain:contact>
S:          <domain:contact type="tech">sh8013</domain:contact>
S:          <domain:clID>ClientX</domain:clID>
S:          <domain:crID>ClientY</domain:crID>
S:          <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:          <domain:upID>ClientZ</domain:upID>
S:          <domain:upDate>2013-10-22T14:25:57.0Z</domain:upDate>
S:          <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:      </resData>
S:      <extension>
S:         <changePoll:changeData
S:          xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:          <changePoll:operation op="sync">custom</changePoll:operation>
S:          <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:          <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:          <changePoll:who>CSR</changePoll:who>
S:          <changePoll:reason lang="en">Customer sync request</changePoll:reason>
S:      </changePoll:changeData>
S:      </extension>
S:   </response>
S: </epp>
Example poll <info> response with the <changePoll:changeData> extension for a "delete" operation on the domain.example domain name that is immediately purged, with the default "after" state. The "after" state is reflected in the <resData> block:

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 id="200" count="1">
S:         <qDate>2013-10-22T14:25:57.0Z</qDate>
S:         <msg>Registry initiated delete of domain resulting in immediate purge.</msg>
S:      </msgQ>
S:      <resData>
S:        <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:          <domain:name>domain.example</domain:name>
S:          <domain:roid>EXAMPLE1-REP</domain:roid>
S:          <domain:clID>ClientX</domain:clID>
S:        </domain:infData>
S:      </resData>
S:      <extension>
S:        <changePoll:changeData xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:          <changePoll:operation op="purge">delete</changePoll:operation>
S:          <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:          <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:          <changePoll:who>ClientZ</changePoll:who>
S:          <changePoll:reason>Court order</changePoll:reason>
S:        </changePoll:changeData>
S:      </extension>
S:      <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54321-XYZ</svTRID>
S:      </trID>
S:   </response>
S:</epp>
Example poll <info> response with the <changePoll:changeData>
extension for an "autoPurge" operation on the domain.example domain
name that previously had the "pendingDelete" status, with the default
"after" state. The "after" state is reflected in the <resData>
block:

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 id="200" count="1">
S:         <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      </msgQ>
S:      <resData>
S:        <domain:infData
S:          xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:          <domain:name>domain.example</domain:name>
S:          <domain:roid>EXAMPLE1-REP</domain:roid>
S:          <domain:clID>ClientX</domain:clID>
S:        </domain:infData>
S:      </resData>
S:      <extension>
S:        <changePoll:changeData
S:          xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:          <changePoll:operation>autoPurge</changePoll:operation>
S:          <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:          <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:          <changePoll:who>Batch</changePoll:who>
S:          <changePoll:reason>Past pendingDelete 5 day period</changePoll:reason>
S:        </changePoll:changeData>
S:      </extension>
S:      <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54321-XYZ</svTRID>
S:      </trID>
S:   </response>
S:</epp>
Example poll `<info>` response with the `<changePoll:changeData>` extension for an "update" operation on the ns1.domain.example host, with the default "after" state. The "after" state is reflected in the `<resData>` block:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg>Command completed successfully; ack to dequeue</msg>
    </result>
    <msgQ id="201" count="1">
      <qDate>2013-10-22T14:25:57.0Z</qDate>
      <msg>Registry initiated update of host.</msg>
    </msgQ>
    <resData>
      <host:infData
        xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>ns1.domain.example</host:name>
        <host:roid>NS1_EXAMPLE1-REP</host:roid>
        <host:status s="linked"/>
        <host:status s="serverUpdateProhibited"/>
        <host:addr ip="v4">192.0.2.2</host:addr>
        <host:addr ip="v6">1080:0:0:8:800:200C:417A</host:addr>
        <host:clID>ClientX</host:clID>
        <host:crID>ClientY</host:crID>
        <host:crDate>2012-04-03T22:00:00.0Z</host:crDate>
        <host:upID>ClientY</host:upID>
        <host:upDate>2013-10-22T14:25:57.0Z</host:upDate>
      </host:infData>
      <extension>
        <changePoll:changeData
          xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
          <changePoll:operation>update</changePoll:operation>
          <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
          <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
          <changePoll:who>ClientZ</changePoll:who>
          <changePoll:reason>Host Lock</changePoll:reason>
        </changePoll:changeData>
      </extension>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```
3.1.3. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> query command or <transfer> response described in the [RFC5730].

3.2. EPP Transform Commands

EPP provides five 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.

3.2.1. EPP <create> Command

This extension does not add any elements to the EPP <create> command or <create> response described in the [RFC5730].

3.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the [RFC5730].

3.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the [RFC5730].

3.2.4. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the [RFC5730].

3.2.5. EPP <update> Command

This extension does not add any elements to the EPP <update> command or <update> response described in the [RFC5730].

4. Formal Syntax

One schema is presented here that is the EPP Change Poll Extension schema.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.
4.1. Change Poll Extension Schema

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
/schema targetNamespace="urn:ietf:params:xml:ns:changePoll-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <!-- Import common element types. -->
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>
  <import namespace="urn:ietf:params:xml:ns:epp-1.0"/>

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
    </documentation>
  </annotation>

  <!-- Change element. -->
  <element name="changeData" type="changePoll:changeDataType"/>

  <!-- Attributes associated with the change. -->
  <complexType name="changeDataType">
    <sequence>
      <element name="operation" type="changePoll:operationType"/>
      <element name="date" type="dateTime"/>
      <element name="svTRID" type="epp:trIDStringType"/>
      <element name="who" type="changePoll:whoType"/>
      <element name="caseId" type="changePoll:caseIdType" minOccurs="0"/>
      <element name="reason" type="eppcom:reasonType" minOccurs="0"/>
    </sequence>
    <attribute name="state" type="changePoll:stateType"
      default="after"/>
  </complexType>
Enumerated list of operations, with extensibility via "custom".

```xml
<simpleType name="operationEnum">
  <restriction base="token">
    <enumeration value="create"/>
    <enumeration value="delete"/>
    <enumeration value="renew"/>
    <enumeration value="transfer"/>
    <enumeration value="update"/>
    <enumeration value="restore"/>
    <enumeration value="autoRenew"/>
    <enumeration value="autoDelete"/>
    <enumeration value="autoPurge"/>
    <enumeration value="custom"/>
  </restriction>
</simpleType>
```

Enumerated of state of the object in the poll message.

```xml
<simpleType name="stateType">
  <restriction base="token">
    <enumeration value="before"/>
    <enumeration value="after"/>
  </restriction>
</simpleType>
```

Transform operation type

```xml
<complexType name="operationType">
  <simpleContent>
    <extension base="changePoll:operationEnum">
      <attribute name="op" type="token"/>
    </extension>
  </simpleContent>
</complexType>
```

Case identifier type

```xml
<complexType name="caseIdType">
  <simpleContent>
    <extension base="token">
      <attribute name="type" type="changePoll:caseTypeEnum" use="required"/>
      <attribute name="name" type="token"/>
    </extension>
  </simpleContent>
</complexType>
```
<complexType name="caseTypeEnum">
    <restriction base="token">
        <enumeration value="udrp"/>
        <enumeration value="urs"/>
        <enumeration value="custom"/>
    </restriction>
</complexType>

<complexType name="whoType">
    <restriction base="normalizedString">
        <minLength value="1"/>
        <maxLength value="255"/>
    </restriction>
</complexType>

5.  IANA Considerations

5.1.  XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

URI: urn:ietf:params:xml:ns:changePoll-1.0

Registrant Contact: See the "Author’s Address" section of this document.

XML: See the "Formal Syntax" section of this document.
5.2. EPP Extension Registry

The EPP extension described in this document should be registered by
the IANA in the EPP Extension Registry described in [RFC7451]. The
details of the registration are as follows:

Name of Extension: "Change Poll Extension for the Extensible
Provisioning Protocol (EPP)"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. Security Considerations

The mapping extensions described in this document do not provide any
security services beyond those described by EPP [RFC5730] and
protocol layers used by EPP. The security considerations described
in these other specifications apply to this specification as well.

7. Acknowledgements

The authors wish to acknowledge the original concept for this draft
and the efforts in the initial versions of this draft by Trung Tran.

Special suggestions that have been incorporated into this document
were provided by Michael Holloway.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
Internet-Draft                 changePoll                 September 2016


Appendix A. Change History

A.1. Change from 00 to 01

1. Added an optional caseId element that defines the case identifier from UDRP, URS, or custom case, based on feedback from Michael Holloway.

A.2. Change from 01 to 02

1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.
2. Moved Change History to the back section as an Appendix.

A.3. Change from 02 to 03

1. Fixed "before" state example to use the "before" state value based on feedback from Patrick Mevzek.

A.4. Change from 03 to 04

1. Updated the authors for the draft.

A.5. Change from 04 to 05

1. Ping update.

Authors' Addresses
Extensible Provisioning Protocol (EPP) and Registration Data Access Protocol (RDAP) Status Mapping
draft-gould-epp-rdap-status-mapping-03

Abstract

This document describes the mapping of the Extensible Provisioning Protocol (EPP) statuses with the statuses registered for use in the Registration Data Access Protocol (RDAP). This document identifies gaps in the mapping, and registers RDAP statuses to fill the gaps to ensure that all of the EPP RFC statuses are supported in RDAP.

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 November 24, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

This document maps the statuses defined in the Extensible Provisioning Protocol (EPP) RFCs to the list of statuses registered for use in the Registration Data Access Protocol (RDAP), in the RDAP JSON Values Registry [rdap-json-values].

The RDAP JSON Values Registry is described in section 10.2 of [RFC7483] and is available in the RDAP JSON Values Registry [rdap-json-values].

The EPP statuses used as the source of the mapping include section 2.3 of the EPP Domain Name Mapping [RFC5731], section 2.3 of the EPP Host Mapping [RFC5732], section 2.2 of the EPP Contact Mapping [RFC5733], and section 3.1 of EPP Grace Period Mapping [RFC3915].

Each EPP status MUST map to a single RDAP status to ensure that data in the Domain Name Registries (DNRs) that use EPP can be accurately presented in RDAP.

1.1. Conventions Used in This Document

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 RFC 2119 [RFC2119].
2. EPP to RDAP Status Mapping

Below is an alphabetically sorted list of EPP statuses from the EPP RFCs ([RFC5731], [RFC5732], [RFC5733], and [RFC3915]) mapped to the RDAP statuses registered in the RDAP JSON Values Registry [rdap-json-values], with the format <EPP Status> '=' <RDAP Status>, where a blank <RDAP Status> indicates a gap in the mapping.

addPeriod =
autoRenewPeriod =
clientDeleteProhibited =
clientHold =
clientRenewProhibited =
clientTransferProhibited =
clientUpdateProhibited =
inactive = inactive
linked = associated
ok = active
pendingCreate = pending create
pendingDelete = pending delete
pendingRenew = pending renew
pendingRestore =
pendingTransfer = pending transfer
pendingUpdate = pending update
rehabilitationPeriod =
renewPeriod =
serverDeleteProhibited =
serverRenewProhibited =
serverTransferProhibited =
serverUpdateProhibited =
serverHold =
transferPeriod =

The RDAP JSON Values Registry [rdap-json-values] does have a set of prohibited statuses including "renew prohibited", "update prohibited", "transfer prohibited", and "delete prohibited", but these statuses do not directly map to the EPP prohibited statuses. The EPP prohibited statuses reflect both what is prohibited ("renew", "update", "transfer", "delete") and who set ("client" or "server") and can clear the status. In the DNR, the client and server prohibited statuses are separate and RDAP MUST support the same separation.

Each of the EPP status values that don’t map directly to an RDAP status value is described below. Each EPP status value includes a proposed new RDAP status value and a description of the value. The RDAP status value is derived from the EPP status value by converting
the EPP camel case representation to lower case with a space character inserted between word boundaries.

addPeriod = add period;  For DNR that indicates if the object is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the registration.

autoRenewPeriod = auto renew period;  For DNR that indicates if the object is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the auto renewal.

clientDeleteProhibited = client delete prohibited;  For DNR that indicates the client requested that requests to delete the object MUST be rejected.

clientHold = client hold;  For DNR that indicates the client requested that the DNS delegation information MUST NOT be published for the object.

clientRenewProhibited = client renew prohibited;  For DNR that indicates the client requested that requests to renew the object MUST be rejected.

clientTransferProhibited = client transfer prohibited;  For DNR that indicates the client requested that requests to transfer the object MUST be rejected.

clientUpdateProhibited = client update prohibited;  For DNR that indicates the client requested that requests to update the object (other than to remove this status) MUST be rejected.

pendingRestore = pending restore;  For DNR that indicates a object is in the process of being restored after being in the redemptionPeriod state.

redemptionPeriod = redemption period;  For DNR that indicates a delete has been received, but the object has not yet been purged because an opportunity exists to restore the object and abort the deletion process.

renewPeriod = renew period;  For DNR that indicates if the object is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the renewal.

serverDeleteProhibited = server delete prohibited;  For DNR that indicates the server set the status so that requests to delete the object MUST be rejected.

serverRenewProhibited = server renew prohibited;  For DNR that indicates the server set the status so that requests to renew the object MUST be rejected.

serverTransferProhibited = server transfer prohibited;  For DNR that indicates the server set the status so that requests to transfer the object MUST be rejected.

serverUpdateProhibited = server update prohibited;  For DNR that indicates the server set the status so that requests to update the object (other than to remove this status) MUST be rejected.
serverHold = server hold; For DNR that indicates the server set the status so that DNS delegation information MUST NOT be published for the object.

transferPeriod = transfer period; For DNR that indicates if the domain name is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the transfer.

The resulting mapping after registering the new RDAP statuses is:

- addPeriod = add period
- autoRenewPeriod = auto renew period
- clientDeleteProhibited = client delete prohibited
- clientHold = client hold
- clientRenewProhibited = client renew prohibited
- clientTransferProhibited = client transfer prohibited
- clientUpdateProhibited = client update prohibited
- inactive = inactive
- linked = associated
- ok = active
- pendingCreate = pending create
- pendingDelete = pending delete
- pendingRenew = pending renew
- pendingRestore = pending restore
- pendingTransfer = pending transfer
- pendingUpdate = pending update
- redemptionPeriod = redemption period
- renewPeriod = renew period
- serverDeleteProhibited = server delete prohibited
- serverRenewProhibited = server renew prohibited
- serverTransferProhibited = server transfer prohibited
- serverUpdateProhibited = server update prohibited
- serverHold = server hold
- transferPeriod = transfer period

3. IANA Considerations

3.1. JSON Values Registry

The following values should be registered by the IANA in the RDAP JSON Values Registry described in [RFC7483]:

Value: add period

Type: status
Description: For DNR that indicates if the object is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the registration.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: auto renew period

Type: status

Description: For DNR that indicates if the object is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the auto renewal.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: client delete prohibited

Type: status

Description: For DNR that indicates the client requested that requests to delete the object MUST be rejected.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: client hold

Type: status

Description: For DNR that indicates the client requested that the DNS delegation information MUST NOT be published for the object.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: client renew prohibited

Type: status

Description: For DNR that indicates the client requested that requests to renew the object MUST be rejected.
Value: client transfer prohibited
Type: status
Description: For DNR that indicates the client requested that requests to transfer the object MUST be rejected.

Value: client update prohibited
Type: status
Description: For DNR that indicates the client requested that requests to update the object (other than to remove this status) MUST be rejected.

Value: pending restore
Type: status
Description: For DNR that indicates a object is in the process of being restored after being in the redemptionPeriod state.

Value: redemption period
Type: status
Description: For DNR that indicates a delete has been received, but the object has not yet been purged because an opportunity exists to restore the object and abort the deletion process.
Registrant Contact Information: epp-registry@verisign.com

Value: renew period
Type: status
Description: For DNR that indicates if the object is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the renewal.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: server delete prohibited
Type: status
Description: For DNR that indicates the server set the status so that requests to delete the object MUST be rejected.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: server renew prohibited
Type: status
Description: For DNR that indicates the server set the status so that requests to renew the object MUST be rejected.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: server transfer prohibited
Type: status
Description: For DNR that indicates the server set the status so that requests to transfer the object MUST be rejected.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: server update prohibited
Type: status

Description: For DNR that indicates the server set the status so that requests to update the object (other than to remove this status) MUST be rejected.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: server hold

Type: status

Description: For DNR that indicates the server set the status so that DNS delegation information MUST NOT be published for the object.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: transfer period

Type: status

Description: For DNR that indicates if the domain name is deleted by the registrar during this period, the registry provides a credit to the registrar for the cost of the transfer.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

4. Security Considerations

The mapping described in this document do not provide any security services beyond those described by RDAP [RFC7483].

5. Normative References


Appendix A.  Change History

A.1.  Change from 00 to 01

1.  Changed the mapping of "linked" to "associated" and removed the registration of "linked", based on feedback from Andrew Newton on the weirds mailing list.

A.2.  Change from 01 to 02

1.  Ping update.

A.3.  Change from 02 to 03

1.  Ping update.

Author’s Address

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA  20190
US

Email: jgould@verisign.com
URI:   http://www.verisigninc.com
Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for getting Internationalized Domain Name (IDN) Table information for the registration of IDNs, using the EPP domain name mapping, and optionally with the IDN mapping extension. An IDN Table defines the valid set of characters (code points) that can be used in a domain name. Code points may overlap across IDN Tables and the IDN Tables supported by the servers are up to server policy.

The IDN Table information can be used to validate an IDN prior to registration, can be cached by the client for pre-validation, can be used to select the best IDN Table for the IDN, and can be used to know if and what IDN Table Identifier to pass in a domain create.

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 April 18, 2019.
Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
   1.1. Conventions Used in This Document .................. 3
2. Object Attributes ............................................ 4
   2.1. IDN Table Identifier ................................. 4
   2.2. Domain Name ......................................... 4
3. EPP Command Mapping ......................................... 5
   3.1. EPP Query Commands ................................ 5
       3.1.1. EPP <check> Command ............................... 5
       3.1.1.1. Domain Check Form .............................. 5
       3.1.1.2. Table Check Form ............................... 8
       3.1.2. EPP <info> Command ................................ 9
       3.1.2.1. Domain Info Form ............................... 9
       3.1.2.2. Table Info Form ............................... 13
       3.1.2.3. List Info Form ................................ 17
       3.1.3. EPP <transfer> Command ......................... 19
   3.2. EPP Transform Commands ................................ 20
       3.2.1. EPP <create> Command ............................ 20
       3.2.2. EPP <delete> Command ............................ 20
       3.2.3. EPP <renew> Command ............................. 20
       3.2.4. EPP <transfer> Command ......................... 20
       3.2.5. EPP <update> Command ............................ 20
4. Formal Syntax ................................................. 20
   4.1. IDN Table Mapping Schema .............................. 21
5. IANA Considerations .......................................... 25
   5.1. XML Namespace ........................................ 26
   5.2. EPP Extension Registry ................................ 26
6. Implementation Status ........................................ 26
   6.1. Verisign EPP SDK ..................................... 27
7. Security Considerations ...................................... 27
8. References ................................................. 27
   8.1. Normative References ................................ 27
1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping provides Internationalized Domain Name (IDN) Table information for the registration of IDNs, using the EPP domain name mapping [RFC5731], and optionally with the IDN mapping extension [I-D.ietf-eppext-idnmap]. An IDN Table defines the valid set of characters (code points) that can be used in a domain name. Code points may overlap across IDN Tables and the IDN Tables supported by the servers are up to server policy. This mapping provides the information clients need to register IDNs across a variety of servers with differing IDN policies. The IDN Table Mapping can be used for the following:

"Validate IDN Domain Name" Validate that an IDN meets the server IDN policy. The validation can be done prior to submitting a domain create, per [RFC5731].

"Get IDN Tables Matching IDN Domain Name Along with Meta-Data" Since IDN Table code points may overlap, the mapping can be used to identify the matching set of IDN Tables (language or script), along with the IDN Table meta-data.

"Cache IDN Table Code Points" Clients can query for the complete list of IDN Tables and can get the IDN Table meta-data, based on server policy, to support pre-validation in the client.

"Get the IDN Table Identifier to Pass with a Domain Create" Each IDN Table includes a server unique IDN Table Identifier that may be used as the value of the <idn:table> element in the IDN mapping extension [I-D.ietf-eppext-idnmap]. A flag indicates whether the IDN mapping extension [I-D.ietf-eppext-idnmap] is needed for the domain name.

1.1. Conventions Used in This Document

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 RFC 2119 [RFC2119].
XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this protocol.

"idnTable-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:idnTable-1.0". The XML namespace prefix "idnTable" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Object Attributes

An EPP IDN Table object has attributes and associated values that can help in the registration of IDNs. This section describes each type in detail. The formal syntax for the attribute values described here can be found in the "Formal Syntax" section of this document and in the appropriate normative references.

2.1. IDN Table Identifier

The IDN Table Identifier is a server-defined unique value for an IDN Table object. It is represented using an <idnTable:table> element or an <idnTable:name> element. The IDN Table Identifier is used in the <check> and <info> commands and responses. The IDN Table Identifier MAY also be used as the value for the IDN mapping extension [I-D.ietf-eppext-idnmap] <idn:table> element with a domain create of an IDN object.

2.2. Domain Name

A Domain Name, as represented by an <idnTable:domain> element, is used to enable validating the code points against the server IDN Tables and IDN policies and for retrieving IDN Table information associated with the domain name. The Domain Name MUST be represented as either a U-label or A-label as defined in [RFC5890].

An OPTIONAL attribute "form" MAY be used to specify the representation. When present, the "form" attribute MUST be set to either "aLabel" for A-label or "uLabel" for U-label, depending on the chosen representation for the domain name. The default "form" attribute is "aLabel" for A-label.
3.  EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730].

3.1.  EPP Query Commands

EPP provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve detailed information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1.  EPP <check> Command

There are two forms of the EPP <check> command: the Domain Check Form (Section 3.1.1.1) and the Table Check Form (Section 3.1.1.2).

3.1.1.1.  Domain Check Form

The Domain Check Form is used to check the validity of the domain name against the server IDN Tables and IDN policies, return whether the IDN mapping extension [I-D.ietf-eppext-idnmap] is needed with a domain <create> command, and provide the matching list of IDN Table Identifiers. This mapping is not intended to determine the availability of the domain name.

In addition to the standard EPP command elements, the <check> command MUST contain an <idnTable:check> element that identifies the idnTable namespace. The <idnTable:check> element in the Domain Check Form contains the following child elements:

<idnTable:domain> One or more <idnTable:domain> elements that contain the fully qualified names of the domain objects, as defined in Section 2.2, to validate.
Example Domain Check Form <check> command with three IDNs:

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <check>
C:       <idnTable:check
C:         xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
C:         <idnTable:domain
C:           form="uLabel">idn1.example</idnTable:domain>
C:         <idnTable:domain
C:           form="aLabel">idn2.example</idnTable:domain>
C:         <idnTable:domain>idn3.example</idnTable:domain>
C:       </idnTable:check>
C:     </check>
C:   </command>
C: </epp>

When a <check> command has been processed successfully, the EPP <resData> element MUST contain a child <idnTable:chkData> element that identifies the idnTable namespace. The <idnTable:chkData> element in the Domain Check Form contain one or more <idnTable:domain> elements that contain the following child elements:

- **<idnTable:name>** The fully qualified name of the domain object, as defined in Section 2.2. The element MUST contain a "valid" attribute whose value indicates whether the domain name is valid according to the server IDN Tables and IDN policies. A value of "1" or "true" means that the domain name is valid according to the server IDN Tables and policies. A value of "0" or "false" means that the domain name is not valid according to the server IDN Tables and policies. The element MAY contain an "idnmap" attribute value that indicates whether the server requires the use of the IDN mapping extension [I-D.ietf-eppext-idnmap] with a domain create of the domain name. A value of "1" or "true" means that the IDN mapping extension [I-D.ietf-eppext-idnmap] is required using one of the <idnTable:table> values. A value of "0" or "false" means that the IDN mapping extension [I-D.ietf-eppext-idnmap] is not required.

- **<idnTable:reason>** OPTIONAL reason that the domain name is not valid. If present, this element contains server-specific text to help explain why the domain name is not valid. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).
<idnTable:table> Zero or more OPTIONAL <idnTable:table> elements that contain the server defined IDN Table Identifier, as defined in Section 2.1, that matches the code points of the <idnTable:name> element. The <idnTable:table> MAY be used as the value of the <idn:table> element in the IDN mapping extension [I-D.ietf-eppext-idnmap] for creating the IDN object or MAY be used as the value of the <idnTable:table> element of the Table Info Form (Section 3.1.2.2) <info> command, as described in Section 3.1.2, to retrieve more information about the IDN Table.

Example Domain Check Form <check> response with three IDNs:

```xml
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <idnTable:chkData xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
S:        <idnTable:domain>
S:          <idnTable:name valid="true">idn1.example</idnTable:name>
S:          <idnTable:table>CHI</idnTable:table>
S:        </idnTable:domain>
S:        <idnTable:domain>
S:          <idnTable:name valid="true" idnmap="true">idn2.example</idnTable:name>
S:          <idnTable:table>CHI</idnTable:table>
S:          <idnTable:table>JPN</idnTable:table>
S:        </idnTable:domain>
S:        <idnTable:domain>
S:          <idnTable:name valid="false">idn3.example</idnTable:name>
S:          <idnTable:reason>Commingled scripts</idnTable:reason>
S:        </idnTable:domain>
S:      </idnTable:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```
3.1.1.2. Table Check Form

The Table Check Form is used to check the existence of an IDN Table using the IDN Table Identifier represented by the <idnTable:table> element.

In addition to the standard EPP command elements, the <check> command MUST contain an <idnTable:check> element that identifies the idnTable namespace. The <idnTable:check> element in the Table Check Form contains the following child elements:

- <idnTable:table> One or more <idnTable:table> elements that contain the IDN Table Identifier, as defined in Section 2.1, to check for existence.

Example Table Check Form <check> command with three IDN Table Identifiers:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <check>
      <idnTable:check xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
        <idnTable:table>CHI</idnTable:table>
        <idnTable:table>JPN</idnTable:table>
        <idnTable:table>INVALID</idnTable:table>
      </idnTable:check>
    </check>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

When a <check> command has been processed successfully, the EPP <resData> element MUST contain a child <idnTable:chkData> element that identifies the idnTable namespace. The <idnTable:chkData> element in the Table Check Form contains the following child elements:

- <idnTable:table> One or more <idnTable:table> elements that contain the IDN Table Identifier, as defined in Section 2.1. The element MUST contain an "exists" attribute whose value indicates the existence of the IDN Table Identifier. A value of "1" or "true" means that the IDN Table Identifier exists. A value of "0" or "false" means that the IDN Table Identifier does not exist.
Example Table Check Form <check> response with three IDN Table Identifiers:

```xml
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:       <idnTable:chkData
S:         xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
S:         <idnTable:table exists="true">CHI</idnTable:table>
S:         <idnTable:table exists="true">JPN</idnTable:table>
S:         <idnTable:table exists="false">INVALID</idnTable:table>
S:     </idnTable:chkData>
S:     </resData>
S:   <trID>
S:     <clTRID>ABC-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S:   </trID>
S: </response>
S:</epp>
```

3.1.2. EPP <info> Command

There are three forms of the EPP <info> command: the Domain Info Form (Section 3.1.2.1), the Table Info Form (Section 3.1.2.2), and the List Info Form (Section 3.1.2.3).

3.1.2.1. Domain Info Form

The Domain Info Form is used to validate the domain name code points against the IDN Tables and IDN policies, and to return the matching IDN Table meta-data. The domain name, as defined in Section 2.2 can be provided as either a U-label or A-label.

In addition to the standard EPP command elements, the <info> command MUST contain an <idnTable:info> element that identifies the idnTable namespace. The <idnTable:info> element in the Domain Info Form contains the following child elements:
<idnTable:domain> The domain name, as defined in Section 2.2, to validate against the IDN Tables and IDN policies, and to retrieve the matching IDN Table meta-data.

Example Domain Info Form <info> command using a U-label Domain Name:

```xml
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <idnTable:info
C:       xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
C:        <idnTable:domain>idn1.example</idnTable:domain>
C:      </idnTable:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example Domain Info Form <info> command using an A-label Domain Name:

```xml
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <idnTable:info
C:       xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
C:        <idnTable:domain>xn--idn1.example</idnTable:domain>
C:      </idnTable:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <idnTable:infData> element that identifies the idnTable namespace. The <idnTable:infData> element in the Domain Info Form contains the <idnTable:domain> element that contains the following child elements:

- `<idnTable:name>` The fully qualified name of the domain object, as defined in Section 2.2. The element MUST contain a "valid" attribute whose value indicates whether the domain name is valid according to the server IDN Tables and IDN policies. A value of "1" or "true" means that the domain name is valid according to the server IDN Tables and policies. A value of "0" or "false" means that the domain name is not valid according to the server IDN Tables and policies. The element MAY contain an "idnmap"
attribute value that indicates whether the server requires the use of the IDN mapping extension [I-D.ietf-eppext-idnmap] with a domain create of the domain name. A value of "1" or "true" means that the IDN mapping extension [I-D.ietf-eppext-idnmap] is required using one of the <idnTable:table> <idnTable:name> values. A value of "0" or "false" means that the IDN mapping extension [I-D.ietf-eppext-idnmap] is not required.

=idnTable:uname> or <idnTable:aname>  OPTIONAL U-label or A-label form of the domain name value of the <idnTable:name> element in the opposite form, as defined in Section 2.2.

=idnTable:table>  Zero or more OPTIONAL <idnTable:table> elements that provide the IDN Table meta-data information. The <idnTable:table> element contains the following child elements:

=idnTable:name>  Server defined IDN Table Identifier, as defined in Section 2.1.

=idnTable:type>  The type of the IDN Table with the possible values of "language", to reflect a Language IDN Table, and "script", to reflect a Script IDN Table.

=idnTable:description>  Server defined description of the IDN Table. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

=idnTable:variantGen>  OPTIONAL boolean flag indicating that domains created using the IDN Table will have IDN variants generated. The management of variants is up to server policy.
Example Domain Info Form <info> response for a U-label Domain Name:

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <idnTable:infData
S:       xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
S:        <idnTable:domain>
S:          <idnTable:name valid="true">idn1.example
S:          </idnTable:name>
S:          <idnTable:aname>xn--idn1.example</idnTable:aname>
S:          <idnTable:table>
S:            <idnTable:name>THAI</idnTable:name>
S:            <idnTable:type>script</idnTable:type>
S:            <idnTable:description lang="en">Thai</idnTable:description>
S:            <idnTable:variantGen>false</idnTable:variantGen>
S:          </idnTable:table>
S:        </idnTable:domain>
S:      </idnTable:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
Example Domain Info Form <info> response for an A-label Domain Name:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:       <idnTable:infData
S:         xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
S:         <idnTable:domain>
S:           <idnTable:name valid="true" idnmap="true">
S:             xn--idn1.example
S:           </idnTable:name>
S:           <idnTable:uname>idn1.example</idnTable:uname>
S:           <idnTable:table>
S:             <idnTable:name>CHI</idnTable:name>
S:             <idnTable:type>language</idnTable:type>
S:             <idnTable:description>Chinese (CHI)</idnTable:description>
S:             <idnTable:variantGen>true</idnTable:variantGen>
S:           </idnTable:table>
S:           <idnTable:table>
S:             <idnTable:name>JPN</idnTable:name>
S:             <idnTable:type>language</idnTable:type>
S:             <idnTable:description>Japanese (JPN)</idnTable:description>
S:             <idnTable:variantGen>false</idnTable:variantGen>
S:           </idnTable:table>
S:         </idnTable:domain>
S:       </idnTable:infData>
S:     </resData>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

3.1.2.2. Table Info Form

The Table Info Form is used to retrieve information associated with an IDN Table object. The information provided meta-data about the IDN Table object.

In addition to the standard EPP command elements, the <info> command MUST contain an <idnTable:info> element that identifies the
idnTable namespace. The <idnTable:info> element in the Table Info Form contains the following child elements:

- <idnTable:table> Contains the IDN Table Identifier, as defined in Section 2.1, of the IDN Table object to be queried.

Example Table Info Form <info> command for the "CHI" IDN Table Identifier, which represents a Language IDN Table:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <idnTable:info xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
        <idnTable:table>CHI</idnTable:table>
      </idnTable:info>
    </info>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example Table Info Form <info> command for the "THAI" IDN Table Identifier, which represents a Script IDN Table:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <idnTable:info xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
        <idnTable:table>THAI</idnTable:table>
      </idnTable:info>
    </info>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <idnTable:infData> element that identifies the idnTable namespace. The <idnTable:infData> element in the Table Info Form contains the <idnTable:table> element that contains the following child elements:

- <idnTable:name> Server defined IDN Table Identifier, as defined in Section 2.1.
<idnTable:type> The type of the IDN Table with the possible values of "language", to reflect a Language IDN Table, and "script", to reflect a Script IDN Table.

<idnTable:description> Server defined description of the IDN Table. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

<idnTable:upDate> Contains the date and time that the IDN Table was created or last updated.

<idnTable:version> OPTIONAL server defined version number of the IDN Table.

<idnTable:effectiveDate> OPTIONAL effective date for the IDN Table.

<idnTable:variantGen> OPTIONAL boolean flag indicating that domains created using the IDN Table will have IDN variants generated. The management of variants is up to server policy.

<idnTable:url> OPTIONAL URL for downloading the IDN Table with the applicable set of code points and rules.
Example Table Info Form <info> response for the "CHI" IDN Table Identifier, which represents a Language IDN Table:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <idnTable:infData
S:       xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
S:        <idnTable:table>
S:           <idnTable:name>CHI</idnTable:name>
S:           <idnTable:type>language</idnTable:type>
S:           <idnTable:description lang="en">Chinese (CHI)</idnTable:description>
S:           <idnTable:upDate>2015-02-04T09:30:00.0Z</idnTable:upDate>
S:           <idnTable:version>1.0</idnTable:version>
S:           <idnTable:effectiveDate>2014-11-24</idnTable:effectiveDate>
S:           <idnTable:variantGen>true</idnTable:variantGen>
S:           <idnTable:url>
S:   https://www.iana.org/domains/idn-tables/tables/tld_chi_1.0.txt
S:           </idnTable:url>
S:        </idnTable:table>
S:      </idnTable:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
Example Table Info Form <info> response for the "THAI" IDN Table Identifier, which represents a Script IDN Table:

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <idnTable:infData
S:       xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
S:        <idnTable:table>
S:           <idnTable:name>THAI</idnTable:name>
S:           <idnTable:type>script</idnTable:type>
S:           <idnTable:description>Thai</idnTable:description>
S:           <idnTable:upDate>2014-08-16T09:20:00.0Z
S:           </idnTable:upDate>
S:           <idnTable:version>1.0</idnTable:version>
S:           <idnTable:effectiveDate>2014-11-24
S:           </idnTable:effectiveDate>
S:           <idnTable:variantGen>false</idnTable:variantGen>
S:           <idnTable:url>
S:            https://www.iana.org/domains/idn-tables/tables/tld_thai_1.0.txt
S:           </idnTable:url>
S:        </idnTable:table>
S:      </idnTable:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

3.1.2.3. List Info Form

The List Info Form is used to retrieve the list of IDN Tables supported by the server. The list of IDN Table Identifiers MAY be used to query for the IDN Table information using the Table Info Form (Section 3.1.2.2).

In addition to the standard EPP command elements, the <info> command MUST contain an <idnTable:info> element that identifies the idnTable namespace. The <idnTable:info> element in the List Info Form contains the following child elements:
<idnTable:list> Empty element used as a marker to the server of the List Info Form to retrieve the list of IDN Tables.

Example List Info Form <info> command:

```
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <info>
C:       <idnTable:info xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
C:         <idnTable:list/>
C:       </idnTable:info>
C:     </info>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <idnTable:infData> element that identifies the idnTable namespace. The <idnTable:infData> element in the List Info Form contains the <idnTable:list> element that contains the following child elements:

- <idnTable:table> Zero or more <idnTable:table> elements that contain the following child elements:
  - <idnTable:name> Server defined IDN Table Identifier, as defined in Section 2.1, that is supported by the server. The <idnTable:table> element value MAY be used as the value of the <idnTable:table> element value in the Table Info Form (Section 3.1.2.2) <info> command to retrieve the information on the IDN Table.
  - <idnTable:upDate> Contains the date and time that the IDN Table was created or last updated. This element can be used to determine whether a client-side cache needs to be refreshed for the IDN Table using the Table Info Form (Section 3.1.2.2).
Example List Info Form <info> response that contains three IDN Table Identifiers:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <idnTable:infData
S:       xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0">
S:        <idnTable:list>
S:           <idnTable:table>
S:              <idnTable:name>CHI</idnTable:name>
S:              <idnTable:upDate>2015-02-04T09:30:00.0Z</idnTable:upDate>
S:           </idnTable:table>
S:           <idnTable:table>
S:              <idnTable:name>JPN</idnTable:name>
S:              <idnTable:upDate>2015-01-01T09:40:00.0Z</idnTable:upDate>
S:           </idnTable:table>
S:           <idnTable:table>
S:              <idnTable:name>THAI</idnTable:name>
S:              <idnTable:upDate>2014-08-16T09:20:00.0Z</idnTable:upDate>
S:           </idnTable:table>
S:        </idnTable:list>
S:      </idnTable:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S: </epp>

3.1.3. EPP <transfer> Command

Transfer semantics do not apply to IDN table objects, so there is no mapping defined for the EPP <transfer> command.
3.2. EPP Transform Commands

EPP provides five 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.

3.2.1. EPP <create> Command

Create semantics do not apply to IDN table objects, so there is no mapping defined for the EPP <create> command.

3.2.2. EPP <delete> Command

Delete semantics do not apply to IDN table objects, so there is no mapping defined for the EPP <delete> command.

3.2.3. EPP <renew> Command

Renewal semantics do not apply to IDN table objects, so there is no mapping defined for the EPP <renew> command.

3.2.4. EPP <transfer> Command

Transfer semantics do not apply to IDN table objects, so there is no mapping defined for the EPP <transfer> command.

3.2.5. EPP <update> Command

Update semantics do not apply to IDN table objects, so there is no mapping defined for the EPP <update> command.

4. Formal Syntax

One schema is presented here that is the EPP IDN Table Mapping schema.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.
4.1. IDN Table Mapping Schema

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:idnTable-1.0"
    xmlns:idnTable="urn:ietf:params:xml:ns:idnTable-1.0"
    xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      IDN Table Mapping.
    </documentation>
  </annotation>

  <!-- imports -->
  <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="idnTable:checkType"/>
  <element name="info" type="idnTable:infoType"/>

  <!-- Form of the domain name. -->
  <simpleType name="domainFormType">
    <restriction base="token">
      <enumeration value="aLabel"/>
      <enumeration value="uLabel"/>
    </restriction>
  </simpleType>

  <!-- Domain label element. -->
  <complexType name="domainLabelType">
    <simpleContent>
      <extension base="eppcom:labelType">
        <attribute name="form" type="idnTable:domainFormType" default="aLabel"/>
      </extension>
    </simpleContent>
  </complexType>
</schema>
<complexType name="checkType">
  <sequence>
    <choice>
      <element name="table" type="eppcom:minTokenType" maxOccurs="unbounded"/>
      <element name="domain" type="idnTable:domainLabelType" maxOccurs="unbounded"/>
    </choice>
  </sequence>
</complexType>

<!--
Child elements of info command.
-->  
<complexType name="infoType">
  <sequence>
    <choice>
      <element name="table" type="eppcom:minTokenType"/>
      <element name="domain" type="idnTable:domainLabelType"/>
      <element name="list"/>
    </choice>
  </sequence>
</complexType>

<!--
Child response elements.
-->  
<element name="chkData" type="idnTable:chkDataType"/>
<element name="infData" type="idnTable:infDataType"/>

<!--
Child elements of check response.
-->  
<complexType name="chkDataType">
  <sequence>
    <choice>
      <element name="table" type="idnTable:chkTableType" maxOccurs="unbounded"/>
      <element name="domain" type="idnTable:chkDomainType" maxOccurs="unbounded"/>
    </choice>
  </sequence>
</complexType>
<complexType name="chkTableType">
    <simpleContent>
        <extension base="eppcom:minTokenType">
            <attribute name="exists" type="boolean" use="required"/>
        </extension>
    </simpleContent>
</complexType>

<!-- Domain name check response element -->
<complexType name="domainNameType">
    <simpleContent>
        <extension base="eppcom:labelType">
            <attribute name="valid" type="boolean" use="required"/>
            <attribute name="idnmap" type="boolean" default="true"/>
        </extension>
    </simpleContent>
</complexType>

<!-- Domain check response element -->
<complexType name="chkDomainType">
    <sequence>
        <element name="name" type="idnTable:domainNameType"/>
        <choice>
            <element name="reason" type="eppcom:reasonType"/>
            <element name="table" type="eppcom:minTokenType" maxOccurs="unbounded"/>
        </choice>
    </sequence>
</complexType>

<!-- Child elements of info response. -->
<complexType name="infDataType">
    <sequence>
        <choice>
            <element name="table" type="idnTable:infTableType"/>
            <element name="domain" type="idnTable:infDomainType"/>
            <element name="list" type="idnTable:infListType"/>
        </choice>
    </sequence>
</complexType>
Table types

<complexType name="tableTypeEnumType">
  <restriction base="token">
    <enumeration value="language"/>
    <enumeration value="script"/>
  </restriction>
</complexType>

<complexType name="descriptionType">
  <simpleContent>
    <extension base="token">
      <attribute name="lang" type="language"/>
    </extension>
  </simpleContent>
</complexType>

Table info response information

<complexType name="infTableType">
  <sequence>
    <element name="name" type="eppcom:minTokenType"/>
    <element name="type" type="idnTable:tableTypeEnumType"/>
    <element name="description" type="idnTable:descriptionType"/>
    <element name="upDate" type="dateTime"/>
    <element name="version" type="token" minOccurs="0"/>
    <element name="effectiveDate" type="date" minOccurs="0"/>
    <element name="variantGen" type="boolean" minOccurs="0"/>
    <element name="url" type="anyURI" minOccurs="0"/>
  </sequence>
</complexType>

Domain info response information

<complexType name="infDomainType">
  <sequence>
    <element name="name" type="idnTable:domainNameType"/>
    <choice minOccurs="0">
      <element name="uname" type="eppcom:labelType"/>
      <element name="aname" type="eppcom:labelType"/>
    </choice>
  </sequence>
</complexType>
5. IANA Considerations
5.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

URI: urn:ietf:params:xml:ns:idnTable-1.0

Registrant Contact: See the "Author’s Address" section of this document.

XML: See the "Formal Syntax" section of this document.

5.2. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Extensible Provisioning Protocol (EPP) Internationalized Domain Name (IDN) Table Mapping"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 7942 [RFC7942] before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF.
Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942 [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

6.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-gould-idn-table.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com


7. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730] and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

8. References

8.1. Normative References
8.2. Informative References


Appendix A. Change History

A.1. Change from 00 to 01

1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.
A.2. Change from 01 to 02

1. Removed support for returning the idnTable:codePoint and idnTable:codeRange elements in the table info response based on the feedback at the Registration Operations Workshop (ROW), held on March 22, 2015, prior to IETF-92.
2. Fixed info command samples that referenced the <check> element instead of the <info> element.

A.3. Change from 02 to 03

1. Ping update.

A.4. Change from 03 to 04

1. Ping update.

A.5. Change from 04 to 05

1. Ping update.

A.6. Change from 05 to 06

1. Moved RFC 7451 to an informational reference based on a check done by the Idnits Tool.

A.7. Change from 06 to 07

1. Added the Implementation Status section.

Authors’ Addresses

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US

Email: jgould@verisign.com
URI: http://www.verisigninc.com
Francisco Obispo
Uniregistry Corp.
3-110 Governors Square
Grand Cayman, Grand Cayman KY1-1108
KY

Email: fobispo@uniregistry.com
URI: http://www.uniregistry.com/

Luis Enrique Munoz
Uniregistry Corp.
3-110 Governors Square
Grand Cayman, Grand Cayman KY1-1108
KY

Email: fobispo@uniregistry.com
URI: http://www.uniregistry.com/
Internationalized Domain Name Mapping Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-eppext-idnmap-02

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension mapping for the provisioning of Internationalized Domain Names (IDN) stored in a shared central repository. This mapping extends the EPP domain name mapping to provide additional features required to implement registrations of domain names in character sets other than ASCII.

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 July 9, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

The EPP protocol provides a complete description of EPP command and response structures. A thorough understanding of the base protocol specification is necessary to understand the mapping described in this document.

This document is written in consideration with the Guidelines for Extending the Extensible Provisioning Protocol as defined in [RFC3735].

To comply with the Guidelines for the Implementation of Internationalized Domain Names [1], it is required to associate each label to be registered with a single script, as defined by the code division of the Unicode code chart. This requirement imposes a challenge for registries using the EPP protocol, since there is no such field currently in the domain name mapping to allow for this information to be exchanged.

In addition, registries intending to comply with the recommendation of section 4.1 [RFC5891] of the IDNA2008 protocol, which implies the verification of both the name in ASCII Compatible Encoding and Unicode form, will be able to do so using this extension.

This extension adds two additional data element to the EPP Domain Name mapping, to allow for association of a domain name to an IDN
table identifier, and a the domain name in Unicode Normalization Form C (NFC [2]).

2. Conventions Used in This Document

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 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case representation presented in order to develop a conforming specification.

"idn-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:idn-1.0". The XML namespace prefix "idn" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in [RFC5730].

3.1. EPP Query Commands

This extension does not add any elements to the EPP <check>, <poll>, or <transfer> commands or responses.

3.1.1. EPP <info> Command

This extension does not add any elements to the EPP <info> command, but does include elements in the response, when the extension has been selected during a <login> command.

Example <info> command:
When the info command has been processed successfully, and the domain name is an IDN, the server must include in the <extension> section of the EPP response an <idn:data> element with the following elements:

- A <idn:table> element that contains the IDN table identifier.
- A <idn:uname> element that contains the domain name in Unicode NFC form.

Example <info> response for an authorized client:
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:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--espaol-zwa.example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:hostObj>ns1.example.com</domain:hostObj>
S:        <domain:hostObj>ns1.example.net</domain:hostObj>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:          <domain:hostObj>ns1.example.net</domain:hostObj>
S:        </domain:ns>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientX</domain:upID>
S:        <domain:upDate>1999-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <idn:data xmlns:idn="urn:ietf:params:xml:ns:idn-1.0">
S:        <idn:table>es</idn:table>
S:        <idn:uname>espa#x1f1;ol.example.com</idn:uname>
S:      </idn:data>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
3.2. EPP Transform Commands

This extension does not add any elements to the EPP <delete>, <renew>, or <transfer> commands or responses.

3.2.1. EPP <create> Command

This extension defines additional elements for the EPP <create> command.

If the domain name is an IDN, the EPP command MUST contain an <extension> element, which MUST contain a child <idn:data> element with the following child elements:

- A <idn:table> element that contains the IDN table identifier as provided by the server.
- An optional <idn:uname> element that contains the domain name to be registered in Unicode NFC.

Example <create> command:
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <create>
C:     <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:     <domain:name>xn--espaol-zwa.example.com</domain:name>
C:     <domain:period unit="y">2</domain:period>
C:     <domain:ns>
C:       <domain:hostObj>ns1.example.net</domain:hostObj>
C:       <domain:hostObj>ns2.example.net</domain:hostObj>
C:     </domain:ns>
C:     <domain:registrant>jd1234</domain:registrant>
C:     <domain:contact type="admin">sh8013</domain:contact>
C:     <domain:contact type="tech">sh8013</domain:contact>
C:     <domain:authInfo>
C:       <domain:pw>2fooBAR</domain:pw>
C:     </domain:authInfo>
C:   </create>
C:   <extension>
C:   <idn:data xmlns:idn="urn:ietf:params:xml:ns:idn-1.0">
C:     <idn:table>es</idn:table>
C:     <idn:uname>espa\xF1ol.example.com</idn:uname>
C:   </idn:data>
C:   </extension>
C: </command>
C:</epp>

The server MUST validate the name using the procedure described in section 4.2 of [RFC5891].

If the validation of the IDN name failed because it contained a code point not available in the specified IDN table, the server MUST return an EPP error 2306.

In the specific case that the <domain:name> provided did not map to the provided <idn:uname>, the server MUST respond with an EPP error 2005.

3.3. Formal Syntax

An EPP object mapping is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances.
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
    xmlns:idn="urn:ietf:params:xml:ns:idn-1.0"
    targetNamespace="urn:ietf:params:xml:ns:idn-1.0"
    elementFormDefault="qualified">
    <annotation>
        <documentation>
            Extensible Provisioning Protocol v1.0 domain name extension
            schema for IDN Table selection.
        </documentation>
    </annotation>
    <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
        schemaLocation="eppcom-1.0.xsd"/>
    <!-- Child elements found in IDN -->
    <element name="data" type="idn:idnDataType"/>
    <complexType name="idnDataType">
        <sequence>
            <element name="table" type="eppcom:minTokenType"/>
            <element name="uname" type="eppcom:labelType"
                minOccurs="0"/>
        </sequence>
    </complexType>
    <!-- End of schema. -->
</schema>

4. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas
conforming to a registry mechanism described in [RFC3688]. Two URI
assignments have been registered by the IANA.

Registration request for the contact namespace:

URI: urn:ietf:params:xml:ns:idn-1.0

Registrant Contact: See the "Author’s Address" section of this
document.

XML: None. Namespace URIs do not represent an XML specification.

Registration request for the contact XML schema:

URI: urn:ietf:params:xml:schema:idn-1.0

Registrant Contact: See the "Author’s Address" section of this
document.
5. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730] the EPP domain name mapping [RFC5731], and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

6. References

6.1. Normative References


6.2. Informational References


6.3. URIs


Authors' Addresses

Francisco Obispo
Uniregistry Corp.
3-110 Governors Square
Grand Cayman, Grand Cayman KY1-1108
KY

Phone: +194990334499
Email: fobispo@uniregistry.com
URI:   http://www.uniregistry.com/
Luis Enrique Munoz
Uniregistry Corp.
3-110 Governors Square
Grand Cayman, Grand Cayman  KY1-1108
KY

Phone: +19499034226
Email: fobispo@uniregistry.com
URI: http://www.uniregistry.com/
**Key Relay Mapping for the Extensible Provisioning Protocol**

**draft-ietf-eppext-keyrelay-12**

**Abstract**

This document describes an Extensible Provisioning Protocol (EPP) mapping for a key relay object that relays DNSSEC key material between EPP clients using the poll queue defined in RFC5730.

This key relay mapping will help facilitate changing the DNS operator of a domain while keeping the DNSSEC chain of trust intact.

**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 December 2, 2016.

**Copyright Notice**

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
carefully, as they describe your rights and restrictions with respect to this document.  Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.  Introduction ............................................. 3
   1.1.  Conventions Used in This Document ................. 3
   1.2.  Secure Transfer of DNSSEC Key Material .......... 3
2.  Object Attributes ....................................... 4
   2.1.  DNSSEC Key Material ................................. 5
       2.1.1.  <keyRelayData> element ....................... 5
3.  EPP Command Mapping .................................... 5
   3.1.  EPP Query Commands ................................ 5
       3.1.1.  EPP <check> Command ......................... 6
       3.1.2.  EPP <info> Command ......................... 6
       3.1.3.  EPP <transfer> Command ...................... 9
   3.2.  EPP Transform Commands ............................ 9
       3.2.1.  EPP <create> Command ......................... 9
       3.2.2.  EPP <delete> Command ....................... 11
       3.2.3.  EPP <renew> Command ......................... 11
       3.2.4.  EPP <transfer> Command ...................... 12
       3.2.5.  EPP <update> Command ....................... 12
4.  Formal Syntax ........................................... 12
5.  IANA Considerations ..................................... 13
   5.1.  XML Namespace ....................................... 13
   5.2.  XML Schema ......................................... 13
   5.3.  EPP Extension Registry ............................. 14
6.  Security Considerations ................................ 14
7.  Acknowledgements ....................................... 15
8.  References ............................................... 15
   8.1.  Normative References ............................... 15
   8.2.  Informative References ............................. 15
Appendix A.  Changelog ...................................... 16
   A.1.  draft-gieben-epp-keyrelay-00 .................... 16
   A.2.  draft-gieben-epp-keyrelay-01 .................... 16
   A.3.  draft-gieben-epp-keyrelay-02 .................... 16
   A.4.  draft-gieben-epp-keyrelay-03 .................... 16
   A.5.  draft-ietf-eppext-keyrelay-00 ................... 17
   A.6.  draft-ietf-eppext-keyrelay-01 ................... 17
   A.7.  draft-ietf-eppext-keyrelay-02 ................... 17
   A.8.  draft-ietf-eppext-keyrelay-03 ................... 17
   A.9.  draft-ietf-eppext-keyrelay-04 ................... 17
   A.10. draft-ietf-eppext-keyrelay-05 ................... 18
   A.11. draft-ietf-eppext-keyrelay-06 ................... 18
   A.12. draft-ietf-eppext-keyrelay-07 ................... 18

1. Introduction

There are certain transactions initiated by a DNS-operator that require an authenticated exchange of information between DNS-operators. Often, there is no direct channel between these parties or it is non-scalable and insecure.

One such transaction is the exchange of DNSSEC key material when changing the DNS operator for DNSSEC signed zones. We suggest that DNS-operators use the administrative EPP channel to bootstrap the delegation by relaying DNSSEC key material for the zone.

In this document we define an EPP extension to sent DNSSEC key material between EPP clients. This allows DNS operators to bootstrap automatically, reliably and securely the transfer of a domain name while keeping the DNSSEC chain of trust intact.

1.1. Conventions Used in This Document

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].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client, and "S:" represents lines returned by a protocol server. Indentation and white space in examples is provided only to illustrate element relationships and is not a mandatory feature of this protocol.

1.2. Secure Transfer of DNSSEC Key Material

Exchanging DNSSEC key material in preparation of a domain name transfer is one of the phases in the lifecycle of a domain name [I-D.koch-dnsop-dnssec-operator-change].
DNS-operators need to exchange DNSSEC key material before the registration data can be changed to keep the DNSSEC chain of trust intact. This exchange is normally initiated through the gaining registrar.

The gaining and losing DNS operators could talk directly to each other (the ~ arrow in Figure 1) to exchange the DNSKEY, but often there is no trusted path between the two. As both can securely interact with the registry over the administrative channel through the registrar, the registry can act as a relay for the key material exchange.

The registry is merely used as a relay channel. Therefore it is up to the losing DNS-operator to complete the intended transaction. The registry SHOULD have certain policies in place that require the losing DNS operator to cooperate with this transaction, however this is beyond this document. This document focuses on the EPP protocol syntax.

```
+--------------------+    DNSKEY     +--------------------+
| gaining DNS operator|    --------> | losing DNS operator |
+--------------------+    +--------------------+
     ^
    |    |
+--------------------+    +--------------------+
| gaining registrar |    | registrar of record |
+--------------------+    +--------------------+
     V
     V
EPP keyrelay    |    ^ EPP poll
    +-----------------------------+
    |        registry            |
    +-----------------------------+
```

Figure 1: Transfer of DNSSEC key material.

There is no distinction in the EPP protocol between Registrars and DNS-operators, there is only mention of an EPP client and EPP server. Therefore the term EPP client will be used for the interaction with the EPP server for relaying DNSSEC key material.

2. Object Attributes
2.1. DNSSEC Key Material

The DNSSEC key material is represented in EPP by a <keyRelayData> element.

2.1.1. <keyRelayData> element

The <keyRelayData> contains the following elements:

- One REQUIRED <keyData> element that contains the DNSSEC key material as described in [RFC5910], Section 4
- An OPTIONAL <expiry> element that describes the expected lifetime of the relayed key(s) in the zone. When the <expiry> element is provided the losing DNS operator SHOULD remove the inserted key material from the zone after the expire time. This may be because the transaction that needed the insertion should either be completed or abandoned by that time. If a client receives a key relay object that has been sent previously it MUST update the expire time of the key material. This enables the clients to update the lifetime of the key material when a transfer is delayed.

The <expiry> element MUST contain exactly one of the following child elements:

* <absolute>: The DNSSEC key material is valid from the current date and time until it expires on the specified date and time. If a date in the past is provided this MUST be interpreted as a revocation of a previously sent key relay object.

* <relative>: The DNSSEC key material is valid from the current date and time until the end of the specified duration. If a period of zero is provided this MUST be interpreted as a revocation of a previously sent key relay object.

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mapping described here is specifically for use in this key relay mapping.

3.1. EPP Query Commands

EPP provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve...
detailed information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

Check semantics do not apply to key relay objects, so there is no mapping defined for the EPP <check> command and the EPP <check> response.

3.1.2. EPP <info> Command

Info command semantics do not apply to the key relay objects, so there is no mapping defined for the EPP <info> Command.

The EPP <info> response for key relay objects is used in the EPP poll response, as described in [RFC5730]. The key relay object created with the <create> command, described in Section 3.2.1 is inserted into the receiving client’s poll queue. The receiving client will receive the key relay object using the EPP <poll> command, as described in [RFC5730].

When a <poll> command has been processed successfully for a key relay poll message, the EPP <resData> element MUST contain a child <keyrelay:infData> element that is identified by the keyrelay namespace. The <keyrelay:infData> element contains the following child elements:

- A REQUIRED <name> element containing the domain name for which the DNSSEC key material is relayed.
- A REQUIRED <authInfo> element that contains authorization information associated with the domain object ([RFC5731], Section 3.2.1).
- One or more REQUIRED <keyRelayData> elements containing data to be relayed, as defined in Section 2.1. A server MAY apply a server policy that specifies the number of <keyRelayData> elements that can be incorporated. When a server policy is violated, a server MUST respond with an EPP result code 2308 "Data management policy violation".
- An OPTIONAL <crDate> element that contains the date and time of the submitted <create> command.
- An OPTIONAL <reID> element that contains the identifier of the client that requested the key relay.
o An OPTIONAL <acID> element that contains the identifier of the client that SHOULD act upon the key relay.

Example <poll> response:
3.1.3. EPP <transfer> Command

Transfer semantics do not apply to key relay objects, so there is no mapping defined for the EPP <transfer> command.

3.2. EPP Transform Commands

EPP provides five 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.

3.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create a key relay object that includes the domain name and DNSSEC key material to be relayed. When the <create> command is validated, the server MUST insert an EPP <poll> message, using the key relay info response (See Section 3.1.2), in the receiving client’s poll queue that belongs to the registrar on record of the provided domain name.

In addition to the standard EPP command elements, the <create> command MUST contain a <keyrelay:create> element that is identified by the keyrelay namespace. The <keyrelay:create> element contains the following child elements:

- A REQUIRED <keyrelay:name> element containing the domain name for which the DNSSEC key material is relayed.
- A REQUIRED <authInfo> element that contains authorization information associated with the domain object ([RFC5731], Section 3.2.1).
- One or more REQUIRED <keyrelay:keyRelayData> element containing data to be relayed, as defined in Section 2.1

Example <create> commands:

Note that in the provided example the second <keyrelay:keyRelayData> element has a period of zero and thus represents the revocation of a previously sent key relay object (see Section 2.1.1).
When a server has successfully processed the <create> command it MUST respond with a standard EPP response. See [RFC5730], Section 2.6.

Example <create> response:
When a server cannot process the <create> command due to the server policy it MUST return an EPP 2308 error message. This might be the case when the server knows that the receiving client does not support keyrelay transactions. See [RFC5730], Section 2.6.

Example <create> response:

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="2308">
S:      <msg>Data management policy violation</msg>
S:    </result>
S:    <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>

3.2.2.  EPP <delete> Command

Delete semantics do not apply to key relay objects, so there is no mapping defined for the EPP <delete> command and the EPP <delete> response.

3.2.3.  EPP <renew> Command

Renew semantics do not apply to key relay objects, so there is no mapping defined for the EPP <renew> command and the EPP <renew> response.
3.2.4. EPP <transfer> Command

Transfer semantics do not apply to key relay objects, so there is no mapping defined for the EPP <transfer> command and the EPP <transfer> response.

3.2.5. EPP <update> Command

Update semantics do not apply to key relay objects, so there is no mapping defined for the EPP <update> command and the EPP <update> response.

4. Formal Syntax

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:keyrelay-1.0"
   xmlns:keyrelay="urn:ietf:params:xml:ns:keyrelay-1.0"
   xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
   xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1"
   xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">
  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0 protocol extension schema for relaying DNSSEC key material.
    </documentation>
  </annotation>
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
  <import namespace="urn:ietf:params:xml:ns:secDNS-1.1" />
  <import namespace="urn:ietf:params:xml:ns:domain-1.0" />
  <element name="keyRelayData" type="keyrelay:keyRelayDataType" />
  <element name="infData" type="keyrelay:infDataType" />
  <element name="create" type="keyrelay:createType" />
  <complexType name="createType">
    <sequence>
      <element name="name" type="eppcom:labelType" />
      <element name="authInfo" type="domain:authInfoType" />
      <element name="keyRelayData" type="keyrelay:keyRelayDataType" maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <complexType name="infDataType">
    <sequence>
      <element name="keyRelayData" type="keyrelay:keyRelayDataType" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</schema>
```
<sequence>
  <element name="name" type="eppcom:labelType" />
  <element name="authInfo" type="domain:authInfoType" />
  <element name="keyRelayData" type="keyrelay:keyRelayDataType" maxOccurs="unbounded"/>
  <element name="crDate" type="dateTime" />
  <element name="reID" type="eppcom:clIDType" />
  <element name="acID" type="eppcom:clIDType" />
</sequence>
</complexType>

<complexType name="keyRelayDataType">
  <sequence>
    <element name="keyData" type="secDNS:keyDataType" />
    <element name="expiry" type="keyrelay:keyRelayExpiryType" minOccurs="0" />
  </sequence>
</complexType>
</complexType>

<complexType name="keyRelayExpiryType">
  <choice>
    <element name="absolute" type="dateTime" />
    <element name="relative" type="duration" />
  </choice>
</complexType>

5. IANA Considerations

5.1. XML Namespace

This document uses URNs to describe a XML namespace conforming to the registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

URI: urn:ietf:params:xml:ns:keyrelay-1.0

Registrant Contact: See the "Author’s Address" section of this document.

XML: See the "Formal Syntax" section of this document.

5.2. XML Schema

This document uses URNs to describe a XML schema conforming to the registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:
5.3. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Key Relay Mapping for the Extensible Provisioning Protocol"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, iesg@ietf.org

TLDs: Any

IPR Disclosure: https://datatracker.ietf.org/ipr/2393/

Status: Active

Notes: None

6. Security Considerations

A server SHOULD NOT perform any transformation on data under server management when processing a <keyrelay:create> command. The intent of this command is to put DNSSEC key material on the poll queue of another client. To make sure that this EPP extension is interoperable with the different server policies that already have implemented EPP this extension it is not classified as must not.

Any EPP client can use this mechanism to put data on the message queue of another EPP client, allowing for the potential of a denial of service attack. However this can, and should be detected by the server. A server MAY set a server policy which limits or rejects a <keyrelay:create> command if it detects the mechanism is being abused.

For the <keyrelay:keyRelayData> data a correct <domain:authInfo> element should be used as an indication that putting the key material on the receiving EPP clients poll queue is authorized by the _registrant_ of that domain name. The authorization of EPP clients
to perform DNS changes is not covered in this document as it depends on registry specific policy.

A client that uses this mechanism to send DNSSEC key material to another client could verify through DNS that the DNSSEC key material is added to the authoritative zone of the domain. This check can be used to verify that the DNSSEC key material has traveled end-to-end from the gaining DNS operator to the losing DNS operator. This check does not tell anything about the DNSSEC chain of trust and can merely be used as a verification of a successful transfer of the DNSSEC key material.

7. Acknowledgements

We like to thank the following individuals for their valuable input, review, constructive criticism in earlier revisions or support for the concepts described in this document:


8. References

8.1. Normative References


8.2. Informative References
Appendix A. Changelog

[This section should be removed by the RFC editor before publishing]

A.1. draft-gieben-epp-keyrelay-00
   1. Initial document.

A.2. draft-gieben-epp-keyrelay-01
   1. Style and grammar changes;
   2. Added an expire element as per suggestion by Klaus Malorny;
   3. Make the authInfo element mandatory and make the registry check
      it as per feedback by Klaus Malorny and James Gould.

A.3. draft-gieben-epp-keyrelay-02
   1. Added element to identify the relaying EPP client as suggested by
      Klaus Malorny;
   2. Corrected XML for missing and excess clTRID as noted by Patrick
      Mevzek;
   3. Added clarifications for the examples based on feedback by
      Patrick Mevzek;
   4. Reviewed the consistency of using DNS operator versus registrar
      after review comments by Patrick Faltstrom and Ed Lewis.

A.4. draft-gieben-epp-keyrelay-03
   1. Style and grammar changes
   2. Corrected acknowledgement section
   3. Corrected XML for Expire element to not be mandatory but only
      occur once.
A.5.  draft-ietf-eppext-keyrelay-00

1.  Added feedback from Seth Goldman and put him in the acknowledgement section.

2.  IDnits formatting adjustments

A.6.  draft-ietf-eppext-keyrelay-01

1.  Introducing the <relay> command, and thus separating the data and the command.

2.  Updated the Introduction, describing the general use of relay vs the intended use-case of relaying DNSSEC key data.

3.  Restructuring the document to make it more inline with existing EPP extensions.

A.7.  draft-ietf-eppext-keyrelay-02

1.  Updated the XML structure by removing the <relay> command based on WG feedback

2.  Updated the wording

A.8.  draft-ietf-eppext-keyrelay-03

1.  Updated the document title in the EPP Extension Registry section

2.  Restored Acknowledgement section, thanks to Marco Davids

3.  Incorporated feedback from Patrick Mevzek

A.9.  draft-ietf-eppext-keyrelay-04

1.  Incorporated feedback from James Gould

2.  Added additional text when server is aware that receiving clients do not support keyrelay transactions or DNSSEC as suggested by Kees Monshouwer.

3.  Added additional text for supporting key revocation as suggested by Kees Monshouwer

4.  Updated some of the wording

5.  Fix the usage of multiple keys in a create message
A.10.  draft-ietf-eppext-keyrelay-05
    1.  Review comments after WG last call
A.11.  draft-ietf-eppext-keyrelay-06
    1.  Review comments by Ulrich Wisser during IESG writeup
A.12.  draft-ietf-eppext-keyrelay-07
    1.  fixed changelog
A.13.  draft-ietf-eppext-keyrelay-08
    1.  fixed issue with authinfo
    2.  fixed issue with relative period in example xml
A.14.  draft-ietf-eppext-keyrelay-09
    1.  fixed issue with naming
A.15.  draft-ietf-eppext-keyrelay-10
    1.  removed 4 spaces
A.16.  draft-ietf-eppext-keyrelay-11
    1.  Processed editorial changes from AD review
    2.  Processed comments made during IETF last call
A.17.  draft-ietf-regext-keyrelay-00
    1.  Processed comments made during IESG review

Authors’ Addresses

Rik Ribbers
SIDN
Meander 501
Arnhem 6825 MD
NL

Email: rik.ribbers@sidn.nl
URI: https://www.sidn.nl/
Marc Groeneweg
SIDN
Meander 501
Arnhem 6825 MD
NL
Email: marc.groeneweg@sidn.nl
URI:   https://www.sidn.nl/

Miek Gieben
Email: miek@miek.nl

Antoin Verschuren
Email: ietf@antoin.nl
Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)  
draft-ietf-eppext-launchphase-07  

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension mapping for the provisioning and management of domain name registrations and applications during the launch of a domain name registry.

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 May 5, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
   1.1. Conventions Used in This Document ...................... 4
2. Object Attributes ............................................. 5
   2.1. Application Identifier ................................. 5
   2.2. Validator Identifier .................................... 5
   2.3. Launch Phases ........................................... 6
   2.4. Status Values .......................................... 6
      2.4.1. State Transition .................................. 8
   2.5. Poll Messaging ......................................... 9
   2.6. Mark Validation Models ................................. 12
      2.6.1. <launch:codeMark> element ......................... 13
      2.6.2. <mark:mark> element ............................... 14
      2.6.3. Digital Signature .................................. 14
         2.6.3.1. <smd:signedMark> element ..................... 14
         2.6.3.2. <smd:encodedSignedMark> element ............... 14
3. EPP Command Mapping ........................................... 14
   3.1. EPP <check> Command .................................... 15
      3.1.1. Claims Check Form ................................ 15
      3.1.2. Availability Check Form .......................... 18
      3.1.3. Trademark Check Form .............................. 20
   3.2. EPP <info> Command ..................................... 23
   3.3. EPP <create> Command ................................... 26
      3.3.1. Sunrise Create Form ................................ 26
      3.3.2. Claims Create Form ................................ 32
      3.3.3. General Create Form ................................ 35
      3.3.4. Mixed Create Form ................................ 36
      3.3.5. Create Response ................................... 38
   3.4. EPP <update> Command .................................. 39
   3.5. EPP <delete> Command .................................. 40
   3.6. EPP <renew> Command .................................. 41
   3.7. EPP <transfer> Command ................................. 42
4. Formal Syntax ................................................ 42
   4.1. Launch Schema .......................................... 42
5. IANA Considerations .......................................... 49
   5.1. XML Namespace .......................................... 49
   5.2. EPP Extension Registry ................................ 50
6. Implementation Status ....................................... 50
   6.1. Verisign EPP SDK ....................................... 51
   6.2. Verisign Consolidated Top Level Domain (CTLD) SRS .... 51
   6.3. Verisign .COM / .NET SRS ................................ 52
   6.4. REnGen v3.7 ............................................ 52
   6.5. RegistryEngine EPP Service ............................ 52
   6.6. Neustar EPP SDK ........................................ 53
1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping specifies a flexible schema that can be used to implement several common use cases related to the provisioning and management of domain name registrations and applications during the launch of a domain name registry.

It is typical for domain registries to operate in special modes during their initial launch to facilitate allocation of domain names, often according to special rules. This document uses the term "launch phase" and the shorter form "launch" to refer to such a period.

The EPP domain name mapping [RFC5731] is designed for the steady-state operation of a registry. During a launch period, the model in place may be different from what is defined in the EPP domain name mapping [RFC5731]. For example, registries often accept multiple applications for the same domain name during the "Sunrise" launch phase, referred to as a Launch Application. A Launch Registration
refers to a registration made during a launch phase when the server uses a "first-come, first-served" model. Even in a "first-come, first-served" model, additional steps and information might be required, such as trademark information. In addition, the [I-D.ietf-eppext-tmch-smd] defines a registry interface for the Trademark Claims or "claims" launch phase that includes support for presenting a Trademark Claims Notice to the Registrant. This document proposes an extension to the domain name mapping in order to provide a uniform interface for the management of Launch Applications and Launch Registrations in launch phases.

1.1. Conventions Used in This Document

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 RFC 2119 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this protocol.

"launch-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:launch-1.0". The XML namespace prefix "launch" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

"signedMark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:signedMark-1.0" that is defined in [I-D.ietf-eppext-tmch-smd]. The XML namespace prefix "smd" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

"mark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:mark-1.0" that is defined in [I-D.ietf-eppext-tmch-smd]. The XML namespace prefix "mark" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.
2. Object Attributes

This extension adds additional elements to the EPP domain name mapping [RFC5731]. Only those new elements are described here.

2.1. Application Identifier

Servers MAY allow multiple applications, referred to as a Launch Application, of the same domain name during its launch phase operations. Upon receiving a valid request to create a Launch Application, the server MUST create an application object corresponding to the request, assign an application identifier for the Launch Application, set the [RFC5731] pendingCreate status, and return the application identifier to the client with the <launch:applicationID> element. In order to facilitate correlation, all subsequent launch operations on the Launch Application MUST be qualified by the previously assigned application identifier using the <launch:applicationID> element.

If the <domain:create> command processes a request synchronously without the use of an intermediate Launch Application, then an application identifier MAY not be needed.

2.2. Validator Identifier

The Validator Identifier is the unique identifier for a Trademark Validator that validates marks and has a repository of validated marks. The OPTIONAL "validatorID" attribute is used to define the Validator Identifier of the Trademark Validator. Registries MAY support more than one Third Party Trademark Validator. The Internet Corporation for Assigned Names and Numbers (ICANN) Trademark Clearinghouse (TMCH) is the default Trademark Validator and is reserved the Validator Identifier of "tmch". If the ICANN TMCH is not used or multiple Trademark Validators are used, the Validator Identifier MUST be defined using the "validatorID" attribute.

The Validator Identifier MAY be related to one or more issuer identifiers of the <mark:id> element and the <smd:id> element defined in [I-D.ietf-eppext-tmch-smd]. Both the Validator Identifier and the Issuer Identifier used MUST be unique. The list of validator identifiers and the relationship to issuer identifiers is out of scope for this document.

The Validator Identifier MAY define a non-Trademark Validator that supports a form of claims.
2.3. Launch Phases

The server MAY support multiple launch phases sequentially or simultaneously. The `<launch:phase>` element MUST be included by the client to define the target launch phase of the command. The server SHOULD validate the phase and MAY validate the sub-phase of the `<launch:phase>` element against the active phase and OPTIONAL sub-phase of the server on a create command, and return an EPP error result code of 2306 if there is a mismatch.

The following launch phase values are defined:

- **sunrise**: The phase during which trademark holders can submit registrations or applications with trademark information that can be validated by the server.
- **landrush**: A post-Sunrise phase when non-trademark holders are allowed to register domain names with steps taken to address a large volume of initial registrations.
- **claims**: The Trademark Claims phase, as defined in the TMCH Functional Specification [I-D.ietf-eppext-tmch-func-spec], in which a Claims Notice must be displayed to a prospective registrant of a domain name that matches trademarks.
- **open**: A post-launch phase that is also referred to as "steady state". Servers MAY require additional trademark protection during this phase.
- **custom**: A custom server launch phase that is defined using the "name" attribute.

For extensibility, the `<launch:phase>` element includes an OPTIONAL "name" attribute that can define a sub-phase, or the full name of the phase when the `<launch:phase>` element has the "custom" value. For example, the "claims" launch phase could have two sub-phases that include "landrush" and "open".

Launch phases MAY overlap to support the "claims" launch phase, defined in the TMCH Functional Specification [I-D.ietf-eppext-tmch-func-spec], and to support a traditional "landrush" launch phase. The overlap of the "claims" and "landrush" launch phases SHOULD be handled by setting "claims" as the `<launch:phase>` value and setting "landrush" as the sub-phase with the "name" attribute. For example, the `<launch:phase>` element SHOULD be `<launch:phase name="landrush">claims</launch:phase>`.

2.4. Status Values

A Launch Application or Launch Registration object MAY have a launch status value. The `<launch:status>` element is used to convey the launch status pertaining to the object, beyond what is specified in
the object mapping. A Launch Application or Launch Registration MUST set the [RFC5731] "pendingCreate" status if a launch status is supported and the launch status is not one of the final statuses, including the "allocated" and "rejected" statuses.

The following status values are defined using the required "s" attribute:

pendingValidation: The initial state of a newly-created application or registration object. The application or registration requires validation, but the validation process has not yet completed.

validated: The application or registration meets relevant registry rules.

invalid: The application or registration does not validate according to registry rules. Server policies permitting, it may transition back into "pendingValidation" for revalidation, after modifications are made to ostensibly correct attributes that caused the validation failure.

pendingAllocation: The allocation of the application or registration is pending based on the results of some out-of-band process (for example, an auction).

allocated: The object corresponding to the application or registration has been provisioned. Is a possible end state of an application or registration object.

rejected: The application or registration object was not provisioned. Is a possible end state of an application or registration object.

custom: A custom status that is defined using the "name" attribute.

Each status value MAY be accompanied by a string of human-readable text that describes the rationale for the status applied to the object. The OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

For extensibility the <launch:status> element includes an OPTIONAL "name" attribute that can define a sub-status or the full name of the status when the status value is "custom". The server SHOULD NOT use the "custom" status value.

Status values MAY be skipped. For example, an application or registration MAY immediately start at the "allocated" status or an application or registration MAY skip the "pendingAllocation" status. If the launch phase does not require validation of a request, an application or registration MAY immediately skip to "pendingAllocation".
2.4.1. State Transition

![State Transition Diagram]

Figure 1
2.5. Poll Messaging

A Launch Application MUST and a Launch Registration MAY be handled as a domain name of [RFC5731] in "pendingCreate" status, with the launch status values defined in Section 2.4. As a Launch Application or Launch Registration transitions between the status values defined in Section 2.4, the server SHOULD insert poll messages, per [RFC5730], for the applicable intermediate statuses, including the "pendingValidation", "validated", "pendingAllocation", and "invalid" statuses, using the <domain:infData> element with the <launch:infData> extension. The <domain:infData> element MAY contain non-mandatory information, like contact and name server information. Also, further extensions that would normally be included in the response of a <domain:info> command, per [RFC5731], MAY be included. For the final statuses, including the "allocated" and "rejected" statuses, the server MUST insert a <domain:panData> poll message, per [RFC5731], with the <launch:infData> extension.
The following is an example poll message for a Launch Application that has transitioned to the "pendingAllocation" state.

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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="5" id="12345">
S:      <qDate>2013-04-04T22:01:00.0Z</qDate>
S:      <msg>Application pendingAllocation.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:          ...
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>abc123</launch:applicationID>
S:        <launch:status s="pendingAllocation"/>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
The following is an example <domain:panData> poll message for an "allocated" Launch Application.

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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="5" id="12345">
S:      <qDate>2013-04-04T22:01:00.0Z</qDate>
S:      <msg>Application successfully allocated.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:panData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name paResult="1">domain.example</domain:name>
S:        <domain:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </domain:paTRID>
S:        <domain:paDate>2013-04-04T22:00:00.0Z</domain:paDate>
S:      </domain:panData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>abc123</launch:applicationID>
S:        <launch:status s="allocated"/>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
The following is an example <domain:panData> poll message for an "allocated" Launch Registration.

```
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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="5" id="12345">
S:      <qDate>2013-04-04T22:01:00.0Z</qDate>
S:      <msg>Registration successfully allocated.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:panData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name paResult="1">domain.example</domain:name>
S:        <domain:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </domain:paTRID>
S:        <domain:paDate>2013-04-04T22:00:00.0Z</domain:paDate>
S:      </domain:panData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:status s="allocated"/>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

2.6. Mark Validation Models

A server MUST support at least one of the following models for validating trademark information:

- **code** Use of a mark code by itself to validate that the mark matches the domain name. This model is supported using the `<launch:codeMark>` element with just the `<launch:code>` element.
- **mark** The mark information is passed without any other validation element. The server will use some custom form of validation to
validate that the mark information is authentic. This model is supported using the `<launch:codeMark>` element with just the `<mark:mark>` (Section 2.6.2) element.

**code with mark:** A code is used along with the mark information by the server to validate the mark utilizing an external party. The code represents some form of secret that matches the mark information passed. This model is supported using the `<launch:codeMark>` element that contains both the `<launch:code>` and the `<mark:mark>` (Section 2.6.2) elements.

**signed mark:** The mark information is digitally signed as described in the Digital Signature (Section 2.6.3) section. The digital signature can be directly validated by the server using the public key of the external party that created the signed mark using its private key. This model is supported using the `<smd:signedMark>` (Section 2.6.3.1) and `<smd:encodedSignedMark>` (Section 2.6.3.2) elements.

More than one `<launch:codeMark>`, `<smd:signedMark>` (Section 2.6.3.1), or `<smd:encodedSignedMark>` (Section 2.6.3.2) element MAY be specified. The maximum number of marks per domain name is up to server policy.

### 2.6.1. `<launch:codeMark>` element

The `<launch:codeMark>` element that is used by the "code", "mark", and "code with mark" validation models, has the following child elements:

- `<launch:code>`: OPTIONAL mark code used to validate the `<mark:mark>` (Section 2.6.2) information. The mark code is be a mark-specific secret that the server can verify against a third party. The OPTIONAL "validatorID" attribute is the Validator Identifier (Section 2.2) whose value indicates which Trademark Validator that the code originated from, with no default value.
- `<mark:mark>`: OPTIONAL mark information with child elements defined in the Mark (Section 2.6.2) section.

The following is an example `<launch:codeMark>` element with both a `<launch:code>` and `<mark:mark>` (Section 2.6.2) element.

```xml
<launch:codeMark>
  <launch:code validatorID="sample">
    49FD46E6C4B5C5D4AC</launch:code>
  <mark:mark xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
    ...
  </mark:mark>
</launch:codeMark>
```
2.6.2.  <mark:mark> element

A <mark:mark> element describes an applicant’s prior right to a given domain name that is used with the "mark", "mark with code", and the "signed mark" validation models. The <mark:mark> element is defined in [I-D.ietf-eppext-tmch-smd]. A new mark format can be supported by creating a new XML schema for the mark that has an element that substitutes for the <mark:abstractMark> element from [I-D.ietf-eppext-tmch-smd].

2.6.3.  Digital Signature

Digital signatures MAY be used by the server to validate either the mark information, when using the "signed mark" validation model with the <smd:signedMark> (Section 2.6.3.1) element or the <smd:encodedSignedMark> (Section 2.6.3.2) element.

2.6.3.1.  <smd:signedMark> element

The <smd:signedMark> element contains the digitally signed mark information. The <smd:signedMark> element is defined in [I-D.ietf-eppext-tmch-smd]. A new signed mark format can be supported by creating a new XML schema for the signed mark that has an element that substitutes for the <smd:abstractSignedMark> element from [I-D.ietf-eppext-tmch-smd].

2.6.3.2.  <smd:encodedSignedMark> element

The <smd:encodedSignedMark> element contains an encoded form of the digitally signed <smd:signedMark> (Section 2.6.3.1) element. The <smd:encodedSignedMark> element is defined in [I-D.ietf-eppext-tmch-smd]. A new encoded signed mark format can be supported by creating a new XML schema for the encoded signed mark that has an element that substitutes for the <smd:encodedSignedMark> element from [I-D.ietf-eppext-tmch-smd].

3.  EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in the Launch Phase Extension.

This mapping is designed to be flexible, requiring only a minimum set of required elements.

While it is meant to serve several use cases, it does not prescribe any interpretation by the client or server. Such processing is
typically highly policy-dependent and therefore specific to implementations.

Operations on application objects are done via one or more of the existing EPP verbs defined in the EPP domain name mapping [RFC5731]. Registries MAY choose to support a subset of the operations.

3.1. EPP <check> Command

There are three forms of the extension to the EPP <check> command: the Claims Check Form (Section 3.1.1), the Availability Check Form (Section 3.1.2), and the Trademark Check Form (Section 3.1.3). The <launch:check> element "type" attribute defines the form, with the value of "claims" for the Claims Check Form (Section 3.1.1), with the value of "avail" for the Availability Check Form (Section 3.1.2), and with the value of "trademark" for the Trademark Check Form (Section 3.1.3). The default value of the "type" attribute is "claims". The forms supported by the server is determined by server policy. The server MUST return an EPP error result code of 2307 if it receives a check form that is not supported.

3.1.1. Claims Check Form

The Claims Check Form defines a new command called the Claims Check Command that is used to determine whether or not there are any matching trademarks, in the specified launch phase, for each domain name passed in the command, that requires the use of the "Claims Create Form" on a Domain Create Command. The availability check information defined in the EPP domain name mapping [RFC5731] MUST NOT be returned for the Claims Check Command. This form is the default form and MAY be explicitly identified by setting the <launch:check> "type" attribute to "claims".

Instead of returning whether the domain name is available, the Claims Check Command will return whether or not at least one matching trademark exists for the domain name, that requires the use of the "Claims Create Form" on a Domain Create Command. If there is at least one matching trademark that exists for the domain name, a <launch:claimKey> element is returned. The client MAY then use the value of the <launch:claimKey> element to obtain information needed to generate the Trademark Claims Notice from Trademark Validator based on the Validator Identifier (Section 2.2). The unique notice identifier of the Trademark Claims Notice MUST be passed in the <launch:noticeID> element of the extension to the Create Command (Section 3.3).

The <domain:name> elements in the EPP <check> command of EPP domain name mapping [RFC5731] define the domain names to check for matching
trademarks. The <launch:check> element contains the following child elements:

<launch:phase> Contains the value of the active launch phase of the server. The server SHOULD validate the value against the active server launch phase.

Example Claims Check command using the <check> domain command and the <launch:check> extension with the "type" explicitly set to "claims", to determine if "domain1.example", "domain2.example", and "domain3.example" require claims notices during the "claims" launch phase:

C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <check>
C:    <domain:check
C:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>domain1.example</domain:name>
C:      <domain:name>domain2.example</domain:name>
C:      <domain:name>domain3.example</domain:name>
C:    </domain:check>
C:   </check>
C:   <extension>
C:    <launch:check
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:     type="claims">
C:      <launch:phase>claims</launch:phase>
C:    </launch:check>
C:   </extension>
C:   <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

If the <check> command has been processed successfully, the EPP <response> MUST contain an <extension> <launch:chkData> element that identifies the launch namespace. The <launch:chkData> element contains the following child elements:

<launch:phase> The phase that mirrors the <launch:phase> element included in the <launch:check>.
<launch:cd> One or more <launch:cd> elements that contain the following child elements:

<launch:name> Contains the fully qualified name of the queried domain name. This element MUST contain an "exists" attribute
whose value indicates if a matching trademark exists for the
domain name that requires the use of the "Claims Create Form"
on a Domain Create Command. A value of "1" (or "true") means
that a matching trademark does exist and that the "Claims
Create Form" is required on a Domain Create Command. A value
of "0" (or "false") means that a matching trademark does not
exist or that the "Claims Create Form" is NOT required on a
Domain Create Command.
<launch:claimKey> Zero or more OPTIONAL claim keys that MAY be
passed to a third-party trademark validator such as the
Trademark Clearinghouse (TMCH) for querying the information
needed to generate a Trademark Claims Notice. The
<launch:claimKey> is used as the key for the query in place
of the domain name to securely query the service without
using a well-known value like a domain name. The OPTIONAL
"validatorID" attribute is the Validator Identifier
(Section 2.2) whose value indicates which Trademark Validator
to query for the Claims Notice information, with the default
being the ICANN TMCH. The "validatorID" attribute MAY
reference a non-trademark claims clearinghouse identifier to
support other forms of claims notices.
Example Claims Check response when a claims notice is not required for the domain name domain1.example, a claims notice is required for the domain name domain2.example in the "tmch", and a claims notice is required for the domain name domain3.example in the "tmch" and "custom-tmch", for the "claims" launch phase:

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:     <launch:chkData
S:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:      <launch:phase>claims</launch:phase>
S:      <launch:cd>
S:        <launch:name exists="0">domain1.example</launch:name>
S:        <launch:cd>
S:          <launch:name exists="1">domain2.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJ1Nr2vDsAzasdff7EasdfgjX4R0000000002
S:          </launch:claimKey>
S:        </launch:cd>
S:      </launch:cd>
S:      <launch:cd>
S:        <launch:name exists="1">domain3.example</launch:name>
S:        <launch:claimKey validatorID="tmch">
S:          2013041500/2/6/9/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
S:        </launch:claimKey>
S:        <launch:claimKey validatorID="custom-tmch">
S:          20140423200/1/2/3/rJ1Nr2vDsAzasdff7EasdfgjX4R0000000002
S:        </launch:claimKey>
S:      </launch:cd>
S:    </launch:chkData>
S:  </extension>
S:  <trID>
S:   <clTRID>ABC-12345</clTRID>
S:   <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>

3.1.2. Availability Check Form

The Availability Check Form defines additional elements to extend the EPP <check> command described in the EPP domain name mapping [RFC5731]. No additional elements are defined for the EPP <check>
The EPP <check> command is used to determine if an object can be provisioned within a repository. Domain names may be made available only in unique launch phases, whilst remaining unavailable for concurrent launch phases. In addition to the elements expressed in the <domain:check>, the command is extended with the <launch:check> element that contains the following child elements:

- `<launch:phase>` The launch phase to which domain name availability should be determined.

Example Availability Check Form command using the <check> domain command and the <launch:check> extension with the "type" set to "avail", to determine the availability of two domain names in the "idn-release" custom launch phase:

```xml
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <check>
C:    <domain:check
C:      xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>domain1.example</domain:name>
C:      <domain:name>domain2.example</domain:name>
C:    </domain:check>
C:   </check>
C:   <extension>
C:    <launch:check
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:     type="avail">
C:      <launch:phase name="idn-release">custom</launch:phase>
C:    </launch:check>
C:   </extension>
C:   <clTRID>ABC-12345</clTRID>
C: </command>
C:</epp>
```

The Availability Check Form does not define any extension to the response of an <check> domain command. After processing the command, the server replies with a standard EPP response as defined in the EPP domain name mapping [RFC5731].
3.1.3. Trademark Check Form

The Trademark Check Form defines a new command called the Trademark Check Command that is used to determine whether or not there are any matching trademarks for each domain name passed in the command, independent of the active launch phase of the server and whether the "Claims Create Form" is required on a Domain Create Command. The availability check information defined in the EPP domain name mapping [RFC5731] MUST NOT be returned for the Claims Check Command. This form MUST be identified by setting the <launch:check> "type" attribute to "trademark".

Instead of returning whether the domain name is available, the Trademark Check Command will return whether or not at least one matching trademark exists for the domain name. If there is at least one matching trademark that exists for the domain name, a <launch:claimKey> element is returned. The client MAY then use the value of the <launch:claimKey> element to obtain Trademark Claims Notice information from Trademark Validator based on the Validator Identifier (Section 2.2).

The <domain:name> elements in the EPP <check> command of EPP domain name mapping [RFC5731] define the domain names to check for matching trademarks. The <launch:check> element does not contain any child elements with the "Trademark Check Form":

Example Trademark Check command using the <check> domain command and the <launch:check> extension with the "type" set to "trademark", to determine if "domain1.example", "domain2.example", and "domain3.example" have any matching trademarks:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <check>
      <domain:check xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>domain1.example</domain:name>
        <domain:name>domain2.example</domain:name>
        <domain:name>domain3.example</domain:name>
      </domain:check>
    </check>
    <extension>
      <launch:check xmlns:launch="urn:ietf:params:xml:ns:launch-1.0" type="trademark"/>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

If the <check> command has been processed successfully, the EPP <response> MUST contain an <extension> <launch:chkData> element that identifies the launch namespace. The <launch:chkData> element contains the following child elements:

- <launch:cd> One or more <launch:cd> elements that contain the following child elements:
  - <launch:name> Contains the fully qualified name of the queried domain name. This element MUST contain an "exists" attribute whose value indicates if a matching trademark exists for the domain name. A value of "1" (or "true") means that a matching trademark does exist. A value of "0" (or "false") means that a matching trademark does not exist.
  - <launch:claimKey> Zero or more OPTIONAL claim keys that MAY be passed to a third-party trademark validator such as the Trademark Clearinghouse (TMCH) for querying the information needed to generate a Trademark Claims Notice. The <launch:claimKey> is used as the key for the query in place of the domain name to securely query the service without using a well-known value like a domain name. The OPTIONAL "validatorID" attribute is the Validator Identifier...
(Section 2.2) whose value indicates which Trademark Validator to query for the Claims Notice information, with the default being the ICANN TMCH. The "validatorID" attribute MAY reference a non-trademark claims clearinghouse identifier to support other forms of claims notices.

Example Trademark Check response when no matching trademarks are found for the domain name domain1.example, matching trademarks are found for the domain name domain2.example in the "tmch", matching trademarks are found for domain name domain3.example in the "tmch" and "custom-tmch", for the "claims" launch phase:

```xml
S:<!DOCTYPE epp SYSTEM "urn:ietf:params:xml:ns:epp-1.0.dtd">
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:     <launch:chkData
S:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:      <launch:cd>
S:        <launch:name exists="0">domain1.example</launch:name>
S:      </launch:cd>
S:      <launch:cd>
S:        <launch:name exists="1">domain2.example</launch:name>
S:        <launch:claimKey validatorID="tmch">
S:          2013041500/2/6/9/rJ1NrD092vDsAzf7EzgjX4R000000001
S:        </launch:claimKey>
S:      </launch:cd>
S:      <launch:cd>
S:        <launch:name exists="1">domain3.example</launch:name>
S:        <launch:claimKey validatorID="tmch">
S:          2013041500/2/6/9/rJ1NrD092vDsAzf7EzgjX4R000000001
S:        </launch:claimKey>
S:        <launch:claimKey validatorID="custom-tmch">
S:          20140423200/1/2/3/rj1NkrVdsAzasdf7EasdfgjX4R000000002
S:        </launch:claimKey>
S:      </launch:cd>
S:    </extension>
S:    <trID>
S:     <clTRID>ABC-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```
3.2. EPP <info> Command

This extension defines additional elements to extend the EPP <info> command and response to be used in conjunction with the EPP domain name mapping [RFC5731].

The EPP <info> command is used to retrieve information for a launch phase registration or application. The Application Identifier (Section 2.1) returned in the <launch:creData> element of the create response (Section 3.3) is used for retrieving information for a Launch Application. A <launch:info> element is sent along with the regular <info> domain command. The <launch:info> element includes an OPTIONAL "includeMark" boolean attribute, with a default value of "false", to indicate whether or not to include the mark in the response. The <launch:info> element contains the following child elements:

- <launch:phase> The phase during which the application or registration was submitted or is associated with. Server policy defines the phases that are supported.
- <launch:applicationID> OPTIONAL application identifier of the Launch Application.

Example <info> domain command with the <launch:info> extension to retrieve information for the sunrise application for domain.example and application identifier "abc123":

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <domain:info xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>domain.example</domain:name>
      </domain:info>
    </info>
    <extension>
      <launch:info xmlns:launch="urn:ietf:params:xml:ns:launch-1.0" includeMark="true">
        <launch:phase>sunrise</launch:phase>
        <launch:applicationID>abc123</launch:applicationID>
      </launch:info>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```
Example `<info>` domain command with the `<launch:info>` extension to retrieve information for the sunrise registration for domain.example:

```
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <info>
C:       <domain:info
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>domain.example</domain:name>
C:       </domain:info>
C:     </info>
C:     <extension>
C:       <launch:info
C:         xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:         <launch:phase>sunrise</launch:phase>
C:       </launch:info>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

If the query was successful, the server replies with a `<launch:infData>` element along with the regular EPP `<resData>`. The `<launch:infData>` contains the following child elements:

- `<launch:phase>` The phase during which the application was submitted, or is associated with, that matches the associated `<info>` command `<launch:phase>`.
- `<launch:applicationID>` OPTIONAL Application Identifier of the Launch Application.
- `<launch:status>` OPTIONAL status of the Launch Application using one of the supported status values (Section 2.4).
- `<mark:mark>` Zero or more `<mark:mark>` (Section 2.6.2) elements.
Example `<info>` domain response using the `<launch:infData>` extension with the mark information:

```xml
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="pendingCreate"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>abc123</launch:applicationID>
S:        <launch:status s="pendingValidation"/>
S:        <mark:mark
S:          xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
S:            ...
S:        </mark:mark>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```
3.3. EPP <create> Command

There are four forms of the extension to the EPP <create> command that include the Sunrise Create Form (Section 3.3.1), the Claims Create Form (Section 3.3.2), the General Create Form (Section 3.3.3), and the Mixed Create Form (Section 3.3.4). The form is dependent on the supported launch phases (Section 2.3) as defined below.

sunrise The EPP <create> command with the "sunrise" launch phase is used to submit a registration with trademark information that can be verified by the server with the <domain:name> value. The Sunrise Create Form (Section 3.3.1) is used for the "sunrise" launch phase.

landrush The EPP <create> command with the "landrush" launch phase MAY use the General Create Form (Section 3.3.3) to explicitly specify the phase and optionally define the expected type of object to create.

claims The EPP <create> command with the "claims" launch phase is used to pass the information associated with the presentation and acceptance of the Claims Notice. The Claims Create Form (Section 3.3.2) is used and the General Create Form (Section 3.3.3) MAY be used for the "claims" launch phase.

open The EPP <create> command with the "open" launch phase is undefined but the form supported is up to server policy. Use of the Claims Create Form (Section 3.3.2) MAY be used to pass the information associated with the presentation and acceptance of the Claims Notice if required for the domain name.

custom The EPP <create> command with the "custom" launch phase is undefined but the form supported is up to server policy.

3.3.1. Sunrise Create Form

The Sunrise Create Form of the extension to the EPP domain name mapping [RFC5731] includes the verifiable trademark information that the server uses to match against the domain name to authorize the domain create. A server MUST support one of four models in Claim Validation Models (Section 2.6) to verify the trademark information passed by the client.

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element has an OPTIONAL "type" attribute that defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created. The <launch:create> element contains the following child elements:

- <launch:phase> The identifier for the launch phase.
<launch:codeMark> or <smd:signedMark> or <smd:encodedSignedMark>

<launch:codeMark> Zero or more <launch:codeMark> elements. The
<launch:codeMark> child elements are defined in the
<launch:codeMark> element (Section 2.6.1) section.
<smd:signedMark> Zero or more <smd:signedMark> elements. The
<smd:signedMark> child elements are defined in the
<smd:signedMark> element (Section 2.6.3.1) section.
<smd:encodedSignedMark> Zero or more <smd:encodedSignedMark>
elements. The <smd:encodedSignedMark> child elements are
defined in the <smd:encodedSignedMark> element
(Section 2.6.3.2) section.
The following is an example `<create>` domain command using the
`<launch:create>` extension, following the "code" validation model,
with multiple sunrise codes:

```
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <create>
C:       <domain:create
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>domain.example</domain:name>
C:         <domain:registrant>jd1234</domain:registrant>
C:         <domain:contact type="admin">sh8013</domain:contact>
C:         <domain:contact type="tech">sh8013</domain:contact>
C:         <domain:authInfo>
C:           <domain:pw>2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:create>
C:     </create>
C:     <extension>
C:       <launch:create
C:         xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:         <launch:phase>sunrise</launch:phase>
C:         <launch:codeMark>
C:           <launch:code validatorID="sample1">49FD46E6C4B45C55D4AC</launch:code>
C:         </launch:codeMark>
C:         <launch:codeMark>
C:           <launch:code>49FD46E6C4B45C55D4AD</launch:code>
C:         </launch:codeMark>
C:         <launch:codeMark>
C:           <launch:code validatorID="sample2">49FD46E6C4B45C55D4AE</launch:code>
C:         </launch:codeMark>
C:       </launch:create>
C:     </extension>
C:   </command>
C: </epp>
```
The following is an example <create> domain command using the
<launch:create> extension, following the "mark" validation model,
with the mark information:

```
C:<xml version="1.0" encoding="UTF-8" standalone="no"/>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domainone.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <mark:mark xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:            ...
C:          </mark:mark>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:  </command>
C:</epp>
```
The following is an example <create> domain command using the <launch:create> extension, following the "code with mark" validation model, with a code and mark information:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>domain.example</domain:name>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
    <extension>
      <launch:create xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
        <launch:phase>sunrise</launch:phase>
        <launch:codeMark>
          <launch:code validatorID="sample">49FD46E6C4B45C55D4AC</launch:code>
          <mark:mark xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
            ...
          </mark:mark>
        </launch:codeMark>
      </launch:create>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```
The following is an example <create> domain command using the
<launch:create> extension, following the "signed mark" validation
model, with the signed mark information for a sunrise application:

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domainone.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:       type="application">
C:        <launch:phase>sunrise</launch:phase>
C:        <smd:signedMark id="signedMark"
C:          xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:          ...
C:        </smd:signedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
The following is an example <create> domain command using the <launch:create> extension, following the "signed mark" validation model, with the base64 encoded signed mark information:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>domainone.example</domain:name>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
    <extension>
      <launch:create xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
        <launch:phase>sunrise</launch:phase>
        <smd:encodedSignedMark xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
          ...
        </smd:encodedSignedMark>
      </launch:create>
    </extension>
  </command>
</epp>
```

3.3.2. Claims Create Form

The Claims Create Form of the extension to the EPP domain name mapping [RFC5731] includes the information related to the registrant’s acceptance of the Claims Notice.

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element has an OPTIONAL "type" attribute that defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created. The <launch:create> element contains the following child elements:
<launch:phase> Contains the value of the active launch phase of the server. The server SHOULD validate the value against the active server launch phase.

<launch:notice> One or more <launch:notice> elements that contain the following child elements:

<launch:noticeID> Unique notice identifier for the Claims Notice. The <launch:noticeID> element has an OPTIONAL "validatorID" attribute is the Validator Identifier (Section 2.2) whose value indicates which Trademark Validator is the source of the claims notice, with the default being the ICANN TMCH.

<launch:notAfter> Expiry of the claims notice.

<launch:acceptedDate> Contains the date and time that the claims notice was accepted.
The following is an example <create> domain command using the
<launch:create> extension with the <launch:notice> information for
the "tmch" and the "custom-tmch" validators, for the "claims" launch
phase:

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>claims</launch:phase>
C:        <launch:notice>
C:          <launch:noticeID validatorID="tmch">
C:            370d0b7c9223372036854775807</launch:noticeID>
C:          <launch:notAfter>2014-06-19T10:00:00.0Z</launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:00.0Z</launch:acceptedDate>
C:        </launch:notice>
C:        <launch:notice>
C:          <launch:noticeID validatorID="custom-tmch">
C:            470d0b7c9223654313275808</launch:noticeID>
C:          <launch:notAfter>2014-06-19T10:00:00.0Z</launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:30.0Z</launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
C:  </command>
C: </epp>
3.3.3. General Create Form

The General Create Form of the extension to the EPP domain name mapping [RFC5731] includes the launch phase and optionally the object type to create. The OPTIONAL "type" attribute defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created.

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element contains the following child elements:

<launch:phase> Contains the value of the active launch phase of the server. The server SHOULD validate the value against the active server launch phase.

The following is an example <create> domain command using the <launch:create> extension for a "landrush" launch phase application:

```xml
<C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C: <command>
C:  <create>
C:   <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:     <domain:name>domain.example</domain:name>
C:     <domain:registrant>jd1234</domain:registrant>
C:     <domain:contact type="admin">sh8013</domain:contact>
C:     <domain:contact type="tech">sh8013</domain:contact>
C:     <domain:authInfo>
C:       <domain:pw>2fooBAR</domain:pw>
C:     </domain:authInfo>
C:   </domain:create>
C: </create>
C: <extension>
C:  <launch:create xmlns:launch="urn:ietf:params:xml:ns:launch-1.0" type="application">
C:    <launch:phase>landrush</launch:phase>
C:  </launch:create>
C: </extension>
C: <clTRID>ABC-12345</clTRID>
C: </command>
C:</epp>
```
3.3.4. Mixed Create Form

The Mixed Create Form supports a mix of the create forms, where for example the Sunrise Create Form (Section 3.3.1) and the Claims Create Form (Section 3.3.2) MAY be supported in a single command by including both the verified trademark information and the information related to the registrant’s acceptance of the Claims Notice. The server MAY support the Mixed Create Form. The "custom" launch phase SHOULD be used when using the Mixed Create Form.
The following is an example <create> domain command using the
<launch:create> extension, with using a mix of the Sunrise Create
Form (Section 3.3.1) and the Claims Create Form (Section 3.3.2) by
including both a mark and a notice:

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domainone.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:      </domain:create>
C:      <extension>
C:        <launch:create
C:          xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:          type="application">
C:          <launch:phase name="non-tmch-sunrise">custom</launch:phase>
C:          <launch:codeMark>
C:            <mark:mark
C:              xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:              ...
C:            </mark:mark>
C:          </launch:codeMark>
C:        </launch:phaseID validatorID="tmch">
C:          49FD46E6C4B45C55D4AC
C:        </launch:noticeID>
C:        <launch:notAfter>2012-06-19T10:00:10.0Z</launch:notAfter>
C:        <launch:acceptedDate>2012-06-19T09:01:30.0Z</launch:acceptedDate>
C:      </extension>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C: </epp>
3.3.5. Create Response

If the create was successful, the server MAY reply with the
<launch:creData> element along with the regular EPP <resData> to
indicate the server generated Application Identifier (Section 2.1),
when multiple applications of a given domain name are supported;
otherwise no extension is included with the regular EPP <resData>.
The <launch:creData> element contains the following child elements:

- **<launch:phase>** The phase of the application that mirrors the
  <launch:phase> element included in the <launch:create>.
- **<launch:applicationID>** The application identifier of the
  application.

An example response when multiple overlapping applications are
supported by the server:

```xml
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <domain:creData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:crDate>2010-08-10T15:38:26.623854Z</domain:crDate>
S:      </domain:creData>
S:    </resData>
S:    <extension>
S:      <launch:creData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>2393-9323-E08C-03B1</launch:applicationID>
S:      </launch:creData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```
3.4. EPP <update> Command

This extension defines additional elements to extend the EPP <update> command to be used in conjunction with the domain name mapping.

A client MUST NOT pass the extension on an EPP <update> command to a server that does not support launch applications. A server that does not support launch applications during its launch phase MUST return an EPP error result code of 2102 when receiving an EPP <update> command with the extension.

Registry policies permitting, clients may update an application object by submitting an EPP <update> command along with a <launch:update> element to indicate the application object to be updated. The <launch:update> element contains the following child elements:

<launch:phase> The phase during which the application was submitted or is associated with.
<launch:applicationID> The application identifier for which the client wishes to update.
The following is an example <update> domain command with the <launch:update> extension to add and remove a name server of a sunrise application with the application identifier "abc123":

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:add>
C:          <domain:ns>
C:            <domain:hostObj>ns2.domain.example</domain:hostObj>
C:          </domain:add>
C:          <domain:rem>
C:            <domain:ns>
C:              <domain:hostObj>ns1.domain.example</domain:hostObj>
C:            </domain:rem>
C:      </domain:update>
C:    </update>
C:    <extension>
C:    <launch:update
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:      <launch:phase>sunrise</launch:phase>
C:      <launch:applicationID>abc123</launch:applicationID>
C:    </launch:update>
C:  </extension>
C:  <clTRID>ABC-12345</clTRID>
C: </command>
C:</epp>
```

This extension does not define any extension to the response of an <update> domain command. After processing the command, the server replies with a standard EPP response as defined in the EPP domain name mapping [RFC5731].

3.5. EPP <delete> Command

This extension defines additional elements to extend the EPP <delete> command to be used in conjunction with the domain name mapping.

A client MUST NOT pass the extension on an EPP <delete> command to a server that does not support launch applications. A server that does not support launch applications during its launch phase MUST return
an EPP error result code of 2102 when receiving an EPP <delete> command with the extension.

Registry policies permitting, clients MAY withdraw an application by submitting an EPP <delete> command along with a <launch:delete> element to indicate the application object to be deleted. The <launch:delete> element contains the following child elements:

<launch:phase> The phase during which the application was submitted or is associated with.
<launch:applicationID> The application identifier for which the client wishes to delete.

The following is an example <delete> domain command with the <launch:delete> extension:

```xml
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <delete>
C:    <domain:delete xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>domain.example</domain:name>
C:    </domain:delete>
C:   </delete>
C:   <extension>
C:    <launch:delete xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:      <launch:phase>sunrise</launch:phase>
C:      <launch:applicationID>abc123</launch:applicationID>
C:    </launch:delete>
C:   </extension>
C:   <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

This extension does not define any extension to the response of a <delete> domain command. After processing the command, the server replies with a standard EPP response as defined in the EPP domain name mapping [RFC5731].

3.6. EPP <renew> Command

This extension does not define any extension to the EPP <renew> command or response described in the EPP domain name mapping [RFC5731].
3.7. EPP <transfer> Command

This extension does not define any extension to the EPP <transfer> command or response described in the EPP domain name mapping [RFC5731].

4. Formal Syntax

One schema is presented here that is the EPP Launch Phase Mapping schema.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

4.1. Launch Schema

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema

targetNamespace="urn:ietf:params:xml:ns:launch-1.0"
xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"
xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
xmlns="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified">

<!--
Import common element types.
-->
<import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>
<import namespace="urn:ietf:params:xml:ns:mark-1.0"/>
<import namespace="urn:ietf:params:xml:ns:signedMark-1.0"/>

<annotation>
  <documentation>
    Extensible Provisioning Protocol v1.0
domain name extension schema
    for the launch phase processing.
  </documentation>
</annotation>

<!--
Child elements found in EPP commands.
-->
<element name="check" type="launch:checkType"/>
<element name="info" type="launch:infoType"/>
<element name="create" type="launch:createType"/>
<element name="update" type="launch:idContainerType"/>
<element name="delete" type="launch:idContainerType"/>

<!--
Common container of id (identifier) element
-->
<complexType name="idContainerType">
  <sequence>
    <element name="phase"
      type="launch:phaseType"/>
    <element name="applicationID"
      type="launch:applicationIDType"/>
  </sequence>
</complexType>

<!--
 Definition for application identifier
 -->
 <simpleType name="applicationIDType">
   <restriction base="token"/>
 </simpleType>

 <!--
 Definition for launch phase. Name is an optional attribute used to extend the phase type. For example, when using the phase type value of &lt;custom&gt;, the name can be used to specify the custom phase.
 -->
 <complexType name="phaseType">
   <simpleContent>
     <extension base="launch:phaseTypeValue">
       <attribute name="name" type="token"/>
     </extension>
   </simpleContent>
 </complexType>

 <!--
 Enumeration of for launch phase values.
 -->
 <simpleType name="phaseTypeValue">
   <restriction base="token">
     <enumeration value="sunrise"/>
     <enumeration value="landrush"/>
     <enumeration value="claims"/>
     <enumeration value="open"/>
     <enumeration value="custom"/>
   </restriction>
 </simpleType>

 <!--
 Definition for the sunrise code
 -->
 <simpleType name="codeValue">
   <restriction base="token">
     <minLength value="1"/>
   </restriction>
 </simpleType>

 <complexType name="codeType">
   <simpleContent>
     <extension base="launch:codeValue">
       <attribute name="validatorID" type="launch:validatorIDType" use="optional"/>
     </extension>
   </simpleContent>
 </complexType>
</extension>
</complexContent>
</complexType>

<!--
Definition for the notice identifier
-->
<complexType name="noticeIDValue">
 <restriction base="token">
  <minLength value="1"/>
 </restriction>
</simpleType>

<complexType name="noticeIDType">
 <complexContent>
  <extension base="launch:noticeIDValue">
   <attribute name="validatorID" type="launch:validatorIDType" use="optional"/>
  </extension>
 </complexContent>
</complexType>

<!--
Definition for the validator identifier
-->
<complexType name="validatorIDType">
 <complexContent>
  <restriction base="token">
   <minLength value="1"/>
  </restriction>
 </complexContent>
</complexType>

<!--
Possible status values for sunrise application
-->
<complexType name="statusValueType">
 <complexContent>
  <restriction base="token">
   <enumeration value="pendingValidation"/>
   <enumeration value="validated"/>
   <enumeration value="invalid"/>
   <enumeration value="pendingAllocation"/>
   <enumeration value="allocated"/>
   <enumeration value="rejected"/>
   <enumeration value="custom"/>
  </restriction>
 </complexContent>
</complexType>

<!--
Status type definition
-->
<complexType name="statusType">
    <simpleContent>
        <extension base="normalizedString">
            <attribute name="s" type="launch:statusValueType" use="required"/>
            <attribute name="lang" type="language" default="en"/>
            <attribute name="name" type="token"/>
        </extension>
    </simpleContent>
</complexType>

<complexType name="codeMarkType">
    <sequence>
        <element name="code" type="launch:codeType" minOccurs="0"/>
        <element ref="mark:abstractMark" minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="createType">
    <sequence>
        <element name="phase" type="launch:phaseType"/>
        <choice minOccurs="0">
            <element name="codeMark" type="launch:codeMarkType" maxOccurs="unbounded"/>
            <element ref="smd:abstractSignedMark" maxOccurs="unbounded"/>
            <element ref="smd:encodedSignedMark" maxOccurs="unbounded"/>
        </choice>
        <element name="notice" type="launch:createNoticeType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="type" type="launch:objectType"/>
</complexType>

<!-- codeMark Type that contains an optional code with mark information. -->

<!-- Child elements for the create command -->

<!--
Type of launch object
-->  
<simpleType name="objectType">
    <restriction base="token">
        <enumeration value="application"/>
        <enumeration value="registration"/>
    </restriction>
</simpleType>

<!-- Child elements of the create notice element. -->
<complexType name="createNoticeType">
    <sequence>
        <element name="noticeID" type="launch:noticeIDType"/>
        <element name="notAfter" type="dateTime"/>
        <element name="acceptedDate" type="dateTime"/>
    </sequence>
</complexType>

<!-- Child elements of check (Claims Check Command). -->
<complexType name="checkType">
    <sequence>
        <element name="phase" type="launch:phaseType" minOccurs="0"/>
    </sequence>
    <attribute name="type" type="launch:checkFormType" default="claims"/>
</complexType>

<!-- Type of check form (claims check or availability check) -->
<complexType name="checkFormType">
    <restriction base="token">
        <enumeration value="claims"/>
        <enumeration value="avail"/>
        <enumeration value="trademark"/>  
    </restriction>
</complexType>
5. IANA Considerations

5.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688].

Registration request for the launch namespace:

URI: urn:ietf:params:xml:ns:launch-1.0
Registrant Contact: See the "Author’s Address" section of this document.
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the launch XML schema:

URI: urn:ietf:params:xml:schema:launch-1.0
5.2. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 6982 [RFC6982] before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982 [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982 [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable
experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit.

6.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-ietf-eppext-launchphase.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com


6.2. Verisign Consolidated Top Level Domain (CTLD) SRS

Organization: Verisign Inc.

Name: Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS)

Description: The Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS) implements the server-side of draft-ietf-eppext-launchphase for a variety of Top Level Domains (TLD’s).

Level of maturity: Production

Coverage: The "signed mark" Mark Validation Model, the Claims Check Form for the EPP <check> Command, the Sunrise and Claims Forms for the EPP <create> Command of Launch Registrations and Launch Applications. For Launch Applications the Poll Messaging, the EPP <info> Command, the EPP <update> Command, and the EPP <delete> Command is covered.

Licensing: Proprietary

Contact: jgould@verisign.com
6.3. Verisign .COM / .NET SRS

Organization: Verisign Inc.

Name: Verisign .COM / .NET Shared Registry System (SRS)

Description: The Verisign Shared Registry System (SRS) for .COM, .NET and other IDN TLD’s implements the server-side of draft-ietf-eppext-launchphase.

Level of maturity: Operational Test Environment (OTE)

Coverage: The "signed mark" Mark Validation Model, the Claims Check Form for the EPP <check> Command, the Sunrise and Claims Forms for the EPP <create> Command of Launch Registrations.

Licensing: Proprietary

Contact: jgould@verisign.com

6.4. REngin v3.7

Organization: Domain Name Services (Pty) Ltd

Name: REngin v3.7

Description: Server side implementation only

Level of maturity: Production

Coverage: All features from version 12 have been implemented

Licensing: Proprietary Licensing with Maintenance Contracts

Contact: info@dnservices.co.za

URL: https://www.registry.net.za and soon http://dnservices.co.za

6.5. RegistryEngine EPP Service

Organization: CentralNic

Name: RegistryEngine EPP Service

Description: Generic high-volume EPP service for gTLDs, ccTLDs and SLDs
Level of maturity: Deployed in CentralNic’s production environment as well as two other gTLD registry systems, and two ccTLD registry systems.

Coverage: Majority of elements including TMCH sunrise, landrush and TM claims as well as sunrise applications validated using codes.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: https://www.centralnic.com

6.6. Neustar EPP SDK

Organization: Neustar

Name: Neustar EPP SDK

Description: The Neustar EPP SDK includes client implementation of draft-ietf-eppext-launchphase in both Java and C++.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: trung.tran@neustar.biz

6.7. gTLD Shared Registry System

Organization: Stichting Internet Domeinnaamregistratie Nederland (SIDN)

Name: gTLD Shared Registry System

Description: The gTLD SRS implements the server side of the draft-ietf-eppext-launchphase.

Level of maturity: (soon) Production

Coverage: The following parts of the draft are supported:

- Signed mark validation model using Digital Signature (Section 2.6.3)
- Claims Check Form (Section 3.1.1)
- Sunrise Create Form (Section 3.3.1)
Claims Create Form (Section 3.3.2)

The parts of the document not described here are not implemented.

Licensing: Proprietary

Contact: rik.ribbers@sidn.nl

7. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730], the EPP domain name mapping [RFC5731], and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

Updates to, and deletion of an application object must be restricted to clients authorized to perform the said operation on the object.

As information contained within an application, or even the mere fact that an application exists may be confidential. Any attempt to operate on an application object by an unauthorized client MUST be rejected with an EPP 2201 (authorization error) return code. Server policy may allow <info> operation with filtered output by clients other than the sponsoring client, in which case the <domain:infData> and <launch:infData> response SHOULD be filtered to include only fields that are publicly accessible.

8. Acknowledgements

The authors wish to acknowledge the efforts of the leading participants of the Community TMCH Model that led to many of the changes to this document, which include Chris Wright, Jeff Neuman, Jeff Eckhaus, and Will Shorter.

Special suggestions that have been incorporated into this document were provided by Jothan Frakes, Keith Gaughan, Seth Goldman, Michael Holloway, Jan Jansen, Rubens Kuhl, Ben Levac, Gustavo Lozano, Klaus Malorny, Alexander Mayrhofer, Patrick Mevzek, James Mitchell, Francisco Obispo, Mike O’Connell, Bernhard Reutner-Fischer, Trung Tran, Ulrich Wisser and Sharon Wodjenski.

9. Normative References

[I-D.ietf-eppext-tmch-func-spec]
[I-D.ietf-eppext-tmch-smd]


Appendix A. Change History

A.1. Change from 00 to 01

1. Changed to use camel case for the XML elements.
2. Replaced "cancelled" status to "rejected" status.
3. Added the child elements of the <claim> element.
4. Removed the XML schema and replaced with "[TBD]".

A.2. Change from 01 to 02

1. Added support for both the ICANN and ARI/Neustar TMCH models.
2. Changed the namespace URI and prefix to use "launch" instead of "launchphase".
3. Added definition of multiple claim validation models.
4. Added the `<launch:signedClaim>` and `<launch:signedNotice>` elements.
5. Added support for Claims Info Command

A.3. Change from 02 to 03

1. Removed XSI namespace per Keith Gaughan’s suggestion on the provreg list.
2. Added extensibility to the `launch:status` element and added the pendingAuction status per Trung Tran’s feedback on the provreg list.
3. Added support for the Claims Check Command, updated the location and contents of the `signedNotice`, and replaced most references of Claim to Mark based on the work being done on the ARI/Neustar launch model.

A.4. Change from 03 to 04

1. Removed references to the ICANN model.
2. Removed support for the Claims Info Command.
3. Removed use of the `signedClaim`.
4. Revised the method for referring to the `signedClaim` from the XML Signature using the IDREF URI.
5. Split the `launch-1.0.xsd` into three XML schemas including `launch-1.0.xsd`, `signeMark-1.0.xsd`, and `mark-1.0.xsd`.
6. Split the "claims" launch phase to the "claims1" and "claims2" launch phases.
7. Added support for the `encodedSignedMark` with base64 encoded `signedMark`.
8. Changed the elements in the `createNoticeType` to include the `noticeID`, `timestamp`, and the `source` elements.
9. Added the `class` and `effectiveDate` elements to `mark`.

A.5. Change from 04 to 05

1. Removed reference to `<smd:zone>` in the `<smd:signedMark>` example.
2. Incorporated feedback from Bernhard Reutner-Fischer on the provreg mail list.
3. Added missing launch XML prefix to `applicationIDType` reference in the `idContainerType` of the Launch Schema.
5. Updated note on replication of the EPP contact mapping elements in the Mark Contact section.
A.6. Change from 05 to 06

1. Removed the definition of the mark-1.0 and signedMark-1.0 and replaced with reference to draft-lozano-smd, that contains the definition for the mark, signed marked, and encoded signed mark.
2. Split the <launch:timestamp> into <launch:generatedDate> and <launch:acceptedDate> based on feedback from Trung Tran.
3. Added the "includeMark" optional attribute to the <launch:info> element to enable the client to request whether or not to include the mark in the info response.
4. Fixed state diagram to remove redundant transition from "invalid" to "rejected"; thanks Klaus Malorny.

A.7. Change from 06 to 07

1. Proof-read grammar and spelling.
2. Changed "pendingAuction" status to "pendingAllocation", changed "pending" to "pendingValidation" status, per proposal from Trung Tran and seconded by Rubens Kuhl.
3. Added text related to the use of RFC 5731 pendingCreate to the Application Identifier section.
4. Added the Poll Messaging section to define the use of poll messaging for intermediate state transitions and pending action polling messaging for final state transitions.

A.8. Change from 07 to 08

1. Added support for use of the launch statuses and poll messaging for Launch Registrations based on feedback from Sharon Wodjenski and Trung Tran.
2. Incorporated changes based on updates or clarifications in draft-lozano-tmch-func-spec-01, which include:
   1. Removed the unused <launch:generatedDate> element.
   2. Removed the <launch:source> element.
   3. Added the <launch:notAfter> element based on the required <tmNotice:notAfter> element.

A.9. Change from 08 to 09

1. Made <choice> element optional in <launch:create> to allow passing just the <launch:phase> in <launch:create> per request from Ben Levac.
2. Added optional "type" attribute in <launch:create> to enable the client to explicitly define the desired type of object (application or registration) to create to all forms of the create extension.
3. Added text that the server SHOULD validate the <launch:phase> element in the Launch Phases section.
4. Add the "General Create Form" to the create command extension to support the request from Ben Levac.
5. Updated the text for the Poll Messaging section based on feedback from Klaus Malorny.
6. Replaced the "claims1" and "claims2" phases with the "claims" phase based on discussion on the provreg list.
7. Added support for a mixed create model (Sunrise Create Model and Claims Create Model), where a trademark (encoded signed mark, etc.) and notice can be passed, based on a request from James Mitchell.
8. Added text for the handling of the overlapping "claims" and "landrush" launch phases.
9. Added support for two check forms (claims check form and availability check form) based on a request from James Mitchell. The availability check form was based on the text in draft-rbp-application-epp-mapping.

A.10. Change from 09 to 10
1. Changed noticeIDType from base64Binary to token to be compatible with draft-lozano-tmch-func-spec-05.
2. Changed codeType from base64Binary to token to be more generic.
3. Updated based on feedback from Alexander Mayrhofer, which include:
   1. Changed "extension to the domain name extension" to "extension to the domain name mapping".
   2. Changed use of 2004 return code to 2306 return code when phase passed mismatches active phase and sub-phase.
   3. Changed description of "allocated" and "rejected" statuses.
   4. Moved sentence on a synchronous <domain:create> command without the use of an intermediate application, then an Application Identifier MAY not be needed to the Application Identifier section.
   5. Restructured the Mark Validation Models section to include the "<launch:codeMark> element" sub-section, the "<mark:mark> element" sub-section, and the Digital Signature sub-section.
   6. Changed "Registries may" to "Registries MAY".
   7. Changed "extensed" to "extended" in "Availability Check Form" section.
   8. Broke the mix of create forms in the "EPP <create> Command" section to a fourth "Mixed Create Form" with its own sub-section.
   9. Removed "displayed or" from "displayed or accepted" in the <launch:acceptedDate> description.
10. Replaced "given domain name is supported" with "given domain name are supported" in the "Create Response" section.
11. Changed the reference of 2303 (object does not exist) in the "Security Considerations" section to 2201 (authorization error).
12. Added arrow from "invalid" status to "pendingValidation" status and "pendingAllocation" status to "rejected" status in the State Transition Diagram.
4. Added the "C:" and "S:" example prefixes and related text in the "Conventions Used in This Document" section.

A.11. Change from 10 to 11
1. Moved the claims check response <launch:chkData> element under the <extension> element instead of the <resData> element based on the request from Francisco Obispo.

A.12. Change from 11 to 12
1. Added support for multiple validator identifiers for claims notices and marks based on a request and text provided by Mike O’Connell.
2. Removed domain:exDate element from example in section 3.3.5 based on a request from Seth Goldman on the provreg list.
3. Added clarifying text for clients not passing the launch extension on update and delete commands to servers that do not support launch applications based on a request from Sharon Wodjenski on the provreg list.

A.13. Change from 12 to WG 00
1. Changed to eppext working group draft by changing draft-tan-epp-launchphase to draft-ietf-eppext-launchphase and by changing references of draft-lozano-tmch-smd to draft-ietf-eppext-tmch-smd.

A.14. Change WG 00 to WG 01
1. Removed text associated with support for the combining of status values based on feedback from Patrick Mevzek on the provreg mailing list, discussion on the eppext mailing list, and discussion at the eppext IETF meeting on March 6, 2014.

A.15. Change WG 01 to WG 02
1. Changed the <launch:claim> element to be zero or more elements and the <launch:notice> element to be one or more elements in the
Claims Create Form. These changes were needed to be able to support more than one concurrent claims services.

A.16. Change WG 02 to WG 03

1. Added the "Implementation Status" section based on an action item from the eppext IETF-91 meeting.
2. Moved Section 7 "IANA Considerations" and Section 9 "Security Considerations" before Section 5 "Acknowledgements". Moved "Change Log" Section to end.
3. Updated the text for the Claims Check Form and the Claims Create Form to support checking for the need of the claims notice and passing the claims notice outside of the "claims" phase.
4. Added the new Trademark Check Form to support determining whether or not a trademark exists that matches the domain name independent of whether a claims notice is required on create. This was based on a request from Trung Tran and a discussion on the eppext mailing list.

A.17. Change WG 03 to WG 04

1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.

A.18. Change WG 04 to WG 05

1. Added a missing comma to the description of the <launch:phase> element, based on feedback from Keith Gaughan on the eppext mailing list.
2. Added the SIDN implementation status information.
3. Fixed a few indentation issues in the samples.

A.19. Change WG 05 to WG 06

1. Removed duplicate "TMCH Functional Specification" URIs based on feedback from Scott Hollenbeck on the eppext mailing list.
2. Changed references of example?.tld to domain?.example to be consistent with RFC 6761 based on feedback from Scott Hollenbeck on the eppext mailing list.
3. A template was added to section 5 to register the XML schema in addition to the namespace based on feedback from Scott Hollenbeck on the eppext mailing list.

A.20. Change WG 06 to WG 07

Authors’ Addresses

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US

Email: jgould@verisign.com
URI: http://www.verisigninc.com

Wil Tan
Cloud Registry
Suite 32 Seabridge House
377 Kent St
Sydney, NSW 2000
AU

Phone: +61 414 710899
Email: wil@cloudregistry.net
URI: http://www.cloudregistry.net

Gavin Brown
CentralNic Ltd
35-39 Mooregate
London, England EC2R 6AR
GB

Phone: +44 20 33 88 0600
Email: gavin.brown@centralnic.com
URI: https://www.centralnic.com
Abstract

Domain Name Registries (DNRs) may operate in special modes for certain periods of time enabling trademark holders to protect their rights during the introduction of a Top Level Domain (TLD).

One of those special modes of operation is the Sunrise Period. The Sunrise Period allows trademark holders an advance opportunity to register domain names corresponding to their trademarks before names are generally available to the public.

This document describes the format of a mark and a digitally signed mark used by trademark holders for registering domain names during the sunrise phase of generic Top Level Domains (gTLDs). Three types of mark objects are defined in this specification: registered trademarks, court-validated marks, and marks protected by statute or treaty.

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 10, 2016.
Lozano  Expires September 10, 2016

1. Introduction

Domain Name Registries (DNRs) may operate in special modes for certain periods of time enabling trademark holders to protect their rights during the introduction of a Top Level Domain (TLD).
One of those special modes of operation is the Sunrise Period. The Sunrise Period allows trademark holders an advance opportunity to register domain names corresponding to their trademarks before names are generally available to the public.

This specification was defined as part of the development of the ICANN Trademark Clearinghouse (TMCH). The ICANN TMCH is a global repository for trademark data used by DNRs, registrars and trademark holders during the registration process of domain names.

This document describes a mapping of the common elements found in trademark data. A digitally signed mark format is defined in order to support digital signatures on the mark. Finally a mapping for encoding the signed mark document is defined.

Three types of mark objects are defined in this specification: registered trademarks, court-validated marks, and marks protected by statute or treaty.

This specification is intended to be used in the gTLD space, but nothing precludes the use of this format by other entities.

The detailed policy regarding the public key infrastructure (PKI), authorized validators, and other requirements must be defined based on the local policy of the entities using this specification. In the case of gTLDs, the detailed policy regarding the use of this specification is defined in the Rights Protection Mechanism Requirements document (see [ICANN-TMCH]), and the PKI is defined in [I-D.ietf-eppext-tmch-func-spec]. Implementations will need to implement such a PKI (or an equivalent) in order for the signatures defined in this document to have any useful semantics.

The objects specified in this document can be referenced by application protocols like the Extensible Provisioning Protocol (EPP), defined in [RFC5730].

1.1. Terminology

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 RFC 2119 [RFC2119].

XML (EXtensible Markup Language) is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.
"signedMark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:signedMark-1.0". The XML namespace prefix "smd" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

"mark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:mark-1.0". The XML namespace prefix "mark" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Object Description

This section defines the Mark and Signed Mark objects. Empty complex element types and abstract elements are defined to support additional Mark and Signed Mark definitions using XML schema substitution groups. Support for replacement through the XML schema substitution groups is included in the description of the objects.

This section defines some elements as OPTIONAL. If an elements is not defined as OPTIONAL, then it MUST be included in the object.

The following elements are defined as telephone numbers: <mark:voice>, <mark:fax> and <smd:voice>. The representation of telephone numbers in this specification is derived from structures defined in [ITU.E164.2005]. Telephone numbers described in this mapping are character strings that MUST begin with a plus sign ("+", ASCII value 0x002B), followed by a country code defined in [ITU.E164.2005], followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number. An optional "x" attribute is provided to note telephone extension information.

The following elements are defined as email addresses: <mark:email> and <smd:email>. Email address syntax is defined in [RFC5322].

2.1. Holder and Contacts objects

Marks are linked to Holder objects and optionally linked to Contact objects. This section defines the <mark:holder> and <mark:contact> objects.

- The child elements of <mark:holder> include:

* A <mark:name> element that contains the name of the individual holder of the mark. At least one of <mark:name> and <mark:org>
MUST be specified, and <mark:name> is OPTIONAL if <mark:org> is specified.

* A <mark:org> element that contains the name of the organization holder of the mark. At least one of <mark:name> and <mark:org> MUST be specified, and <mark:org> is OPTIONAL if <mark:name> is specified.

* A <mark:addr> element that contains the address information of the holder of a mark. A <mark:addr> contains the following child elements:
  + One, two or three OPTIONAL <mark:street> elements that contains the holder’s street address.
  + A <mark:city> element that contains the holder’s city.
  + An OPTIONAL <mark:sp> element that contains the holder’s state or province.
  + An OPTIONAL <mark:pc> element that contains the holder’s postal code.
  + A <mark:cc> element that contains the holder’s country code. This a two-character code from [ISO3166-2].

* An OPTIONAL <mark:voice> element that contains the holder’s voice telephone number.

* An OPTIONAL <mark:fax> element that contains the holder’s facsimile telephone number.

* An OPTIONAL <mark:email> element that contains the email address of the holder.

The child elements of <mark:contact> include:

* A <mark:name> element that contains name of the responsible person.

* An OPTIONAL <mark:org> element that contains the name of the organization of the contact.

* A <mark:addr> element that contains the address information of the contact. A <mark:addr> contains the following child elements:
+ One, two or three OPTIONAL <mark:street> elements that contains the contact’s street address.
+ A <mark:city> element that contains the contact’s city.
+ An OPTIONAL <mark:sp> element that contains the contact’s state or province.
+ An OPTIONAL <mark:pc> element that contains the contact’s postal code.
+ A <mark:cc> element that contains the contact’s country code. This a two-character code from [ISO3166-2].
* A <mark:voice> element that contains the contact’s voice telephone number.
* An OPTIONAL <mark:fax> element that contains the contact’s facsimile telephone number.
* A <mark:email> element that contains the contact’s email address.

2.2. Mark

A <mark:mark> element that describes an applicant’s prior right to a given domain name.

A <mark:mark> element substitutes for the <mark:abstractMark> abstract element to define a concrete definition of a mark. The <mark:abstractMark> element can be replaced by other mark definitions using the XML schema substitution groups feature.

The child elements of the <mark:mark> element include:

One or more <mark:trademark>, <mark:treatyOrStatute> and <mark:court> elements that contains the detailed information of marks.

  o A <mark:trademark> element that contains the following child elements:

    * A <mark:id> that uniquely identifies a mark in relation to a repository of marks potentially maintained by more than one issuer. A <mark:id> value is a concatenation of the local identifier, followed by a hyphen ("-", ASCII value 0x002D), followed by the issuer identifier.

    * A <mark:markName> element that contains the mark text string.
* One or more `<mark:holder>` elements that contains the information of the holder of the mark. An "entitlement" attribute is used to identify the entitlement of the holder, possible values are: owner, assignee and licensee.

* Zero or more OPTIONAL `<mark:contact>` elements that contains the information of the representative of the mark registration. A "type" attribute is used to identify the type of contact, possible values are: owner, agent or thirdparty.

* A `<mark:jurisdiction>` element that contains the two-character code of the jurisdiction where the trademark was registered. This is a two-character code from [WIPO.ST3].

* Zero or more OPTIONAL `<mark:class>` elements that contain the WIPO Nice Classification class numbers of the mark as defined in the WIPO Nice Classification [WIPO-NICE-CLASSES].

* Zero or more OPTIONAL `<mark:label>` elements that contain the A-label form (as defined in [RFC5890]) of the label that correspond to the `<mark:markName>`.

* A `<mark:goodsAndServices>` element that contains the full description of the goods and services mentioned in the mark registration document.

* An OPTIONAL `<mark:apId>` element that contains the trademark application ID registered in the trademark office.

* An OPTIONAL `<mark:apDate>` element that contains the date the trademark was applied for.

* A `<mark:regNum>` element that contains the trademark registration number registered in the trademark office.

* A `<mark:regDate>` element that contains the date the trademark was registered.

* An OPTIONAL `<mark:exDate>` element that contains the expiration date of the trademark.

  o A `<mark:treatyOrStatute>` element that contains the following child elements:

    * A `<mark:id>`, see definition in the `<mark:trademark>` section above.
* A `<mark:markName>` element, see definition in the `<mark:trademark>` section above.

* One or more `<mark:holder>` elements, see definition in the `<mark:trademark>` section above.

* Zero or more OPTIONAL `<mark:contact>` elements, see definition in the `<mark:trademark>` section above.

* One or more `<mark:protection>` elements that contain the countries and region of the country where the mark is protected. The `<mark:protection>` element contains the following child elements:

  + A `<mark:cc>` element that contains the two-character code of the country in which the mark is protected. This is a two-character code from [ISO3166-2].

  + An OPTIONAL `<mark:region>` element that contains the name of a city, state, province or other geographic region of `<mark:country>` in which the mark is protected.

  + Zero or more OPTIONAL `<mark:ruling>` elements that contains the two-character code of the national territory in which the statute or treaty is applicable. This is a two-character code from [ISO3166-2].

  + Zero or more OPTIONAL `<mark:label>` elements, see definition in the `<mark:trademark>` section above.

* A `<mark:goodsAndServices>` element, see definition in the `<mark:trademark>` section above.

* A `<mark:refNum>` element that contains the serial number of the mark.

* A `<mark:proDate>` element that contains the date of protection of the mark.

* A `<mark:title>` element that contains the title of the treaty or statute.

* A `<mark:execDate>` element that contains the execution date of the treaty or statute.

  o A `<mark:court>` element that contains the following child elements:
* A `<mark:id>`, see definition in the `<mark:trademark>` section above.

* A `<mark:markName>`, see definition in the `<mark:trademark>` section above.

* One or more `<mark:holder>`, see definition in the `<mark:trademark>` section above.

* Zero or more OPTIONAL `<mark:contact>`, see definition in the `<mark:trademark>` section above.

* Zero or more OPTIONAL `<mark:label>`, see definition in the `<mark:trademark>` section above.

* A `<mark:goodsAndServices>`, see definition in the `<mark:trademark>` section above.

* A `<mark:refNum>` element that contains the reference number of the court’s opinion.

* A `<mark:proDate>` element that contains the date of protection of the mark.

* A `<mark:cc>` element that contains the two-character code of the country where the court is located. This a two-character code from [ISO3166-2].

* Zero or more OPTIONAL `<mark:region>` elements that contains the name of a city, state, province or other geographic region of `<mark:cc>` in which the mark is protected. In case `<mark:region>` is specified a default-deny approach MUST be assumed regarding the regions of a country.

* A `<mark:courtName>` element that contains the name of the court.

2.3. Signed Mark

The `<smd:signedMark>` is a digitally signed XML document using XML Signature [XMLDSIG]. The `<smd:signedMark>` XML document (SMD) includes a required "id" attribute of type XSD ID for use with an IDREF URI from the Signature element. The SMD might be transmitted as part of an already XML based protocol, therefore exclusive XML canonicalization as defined in [XMLC14N] MUST be used.

A `<smd:signedMark>` element substitutes for the `<smd:abstractSignedMark>` abstract element to define a concrete definition of a signed mark. The `<smd:abstractSignedMark>` element
can be replaced by other signed mark definitions using the XML schema substitution groups feature.

The child elements of the `<smd:signedMark>` element include:

- The `<smd:id>` that uniquely identifies an SMD in relation to a repository of SMDs potentially maintained by more than one issuer. The `<smd:id>` value is a concatenation of the local identifier, followed by a hyphen ("-", ASCII value 0x002D), followed by the issuer identifier.

- A `<smd:issuerInfo>` element that contains the information of the issuer of the mark registration. A "issuerID" attribute is used to specify the issuer identifier. The child elements include:
  - A `<smd:org>` element that contains the organization name of the issuer.
  - A `<smd:email>` element that contains the issuer customer support email address.
  - An OPTIONAL `<smd:url>` element that contains the HTTP or HTTPS URL of the issuer’s site.
  - An OPTIONAL `<smd:voice>` element that contains the issuer’s voice telephone number.

- A `<smd:notBefore>` element that contains the creation date and time of the SMD.

- A `<smd:notAfter>` element that contains the expiration date and time of the SMD.

- A `<mark:mark>` element that contains the mark information as defined in the Mark (Section 2.2) section.

The following is an example of an SMD:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<smd:signedMark xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
    id="smd1">
    <smd:id>0000001751376056503931-65535</smd:id>
    <smd:issuerInfo issuerID="65535">
        <smd:org>ICANN TMCH TESTING TMV</smd:org>
        <smd:email>notavailable@example.com</smd:email>
        <smd:url>https://www.example.com</smd:url>
        <smd:voice>+32.000000</smd:voice>
    </smd:issuerInfo>
</smd:signedMark>
```
<mark:mark xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
  <mark:trademark>
    <mark:id>00052013734689731373468973-65535</mark:id>
    <mark:markName>Test & Validate</mark:markName>
    <mark:holder entitlement="owner">
      <mark:org>Ag corporation</mark:org>
      <mark:addr>
        <mark:street>1305 Bright Avenue</mark:street>
        <mark:city>Arcadia</mark:city>
        <mark:sp>CA</mark:sp>
        <mark:pc>90028</mark:pc>
        <mark:cc>US</mark:cc>
      </mark:addr>
      <mark:contact type="agent">
        <mark:name>Tony Holland</mark:name>
        <mark:org>Ag corporation</mark:org>
        <mark:addr>
          <mark:street>1305 Bright Avenue</mark:street>
          <mark:city>Arcadia</mark:city>
          <mark:sp>CA</mark:sp>
          <mark:pc>90028</mark:pc>
          <mark:cc>US</mark:cc>
        </mark:addr>
        <mark:voice>+1.2025562302</mark:voice>
        <mark:fax>+1.2025562301</mark:fax>
        <mark:email>info@agcorporation.com</mark:email>
      </mark:contact>
      <mark:jurisdiction>US</mark:jurisdiction>
      <mark:class>15</mark:class>
      <mark:label>testandvalidate</mark:label>
      <mark:label>test---validate</mark:label>
      <mark:label>testand--validate</mark:label>
      <mark:label>test-et-validate</mark:label>
      <mark:label>test--validate</mark:label>
      <mark:label>test.et-validate</mark:label>
      <mark:label>testetvalidate</mark:label>
      <mark:label>testetvalidat</mark:label>
      <mark:label>testetValidate</mark:label>
      <mark:label>testet-Validate</mark:label>
      <mark:goodsAndServices>guitar</mark:goodsAndServices>
      <mark:regNum>1234</mark:regNum>
      <mark:regDate>2012-12-31T23:00:00.000Z</mark:regDate>
    </mark:holder>
  </mark:trademark>
</mark:mark>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
2.4. Encoded Signed Mark

The `<smd:encodedSignedMark>` element contains an encoded form of an SMD (described in Section 2.3), with the encoding defined by the "encoding" attribute with the default "encoding" value of "base64" [RFC4648].

The following is an example of a `<smd:encodedSignedMark>` element that uses the default "base64" for encoding a `<smd:signedMark>` element.

```xml
<smd:encodedSignedMark
    xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
    PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNtZDpzaWduZWRNYXJrIHhtbG5zOnNtZD0idXJuOmlldGY6cGFyYWlzOnhtbDpuczpzaWduZWRNYXJrLTEuMC4gPC9zbWQ6c2lnbmVkTWFyaz4=
</smd:encodedSignedMark>
```

3. Formal Syntax

Two schemas are presented here. The first schema is the schema for the signed mark. The second schema is the schema for the mark.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

3.1. Signed Mark Schema

Copyright (c) 2016 IETF Trust and the persons identified as authors of the code. All rights reserved.
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:ietf:params:xml:ns:signedMark-1.0"
  xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
  xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"
  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <annotation>
    <documentation>
      Schema for representing a Signed Trademark.
    </documentation>
  </annotation>

  <import namespace="urn:ietf:params:xml:ns:mark-1.0" />
  <import namespace="http://www.w3.org/2000/09/xmldsig#" />

  <!--
  Abstract signed mark for replacement via substitution.
  -->
  <element name="abstractSignedMark" type="smd:abstractSignedMarkType"
    abstract="true"/>

  <!--
  Empty type for use in extending for a signed mark
  -->
  <complexType name="abstractSignedMarkType"/>

  <element name="signedMark" type="smd:signedMarkType"
    substitutionGroup="smd:abstractSignedMark"/>

  <element name="encodedSignedMark" type="smd:encodedSignedMarkType"/>

  <complexType name="signedMarkType">
    <complexContent>
      <extension base="smd:abstractSignedMarkType">
        <sequence>
          <element name="id" type="mark:idType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
<element name="issuerInfo" type="smd:issuerInfoType"/>
<element name="notBefore" type="dateTime"/>
<element name="notAfter" type="dateTime"/>
<element ref="mark:abstractMark"/>
<element ref="dsig:Signature"/>
</sequence>
<attribute name="id" type="ID" use="required"/>
</complexType>
</complexContent>
</complexType>

<complexType name="encodedSignedMarkType">
  <simpleContent>
    <extension base="token">
      <attribute name="encoding" type="token" default="base64"/>
    </extension>
  </simpleContent>
</complexType>

3.2. Mark Schema

Copyright (c) 2016 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:ietf:params:xml:ns:mark-1.0"
  xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"

Lozano                Expires September 10, 2016              [Page 15]
<documentation>
Schema for representing a Trademark, also referred to as Mark.
</documentation>
</annotation>

<!-- Abstract mark for replacement via substitution. -->
<element name="abstractMark" type="mark:abstractMarkType"
abstract="true"/>

<!-- mark:mark element definition -->
<element name="mark" type="mark:markType"
substitutionGroup="mark:abstractMark"/>

<!-- Empty type for use in extending for a mark -->
<complexType name="abstractMarkType"/>

<!-- mark:mark child elements -->
<complexType name="markType">
<complexContent>
<extension base="mark:abstractMarkType">
<sequence>
<element name="trademark" type="mark:trademarkType"
minOccurs="0" maxOccurs="unbounded"/>
<element name="treatyOrStatute"
type="mark:treatyOrStatuteType" minOccurs="0"
maxOccurs="unbounded"/>
<element name="court" type="mark:courtType" minOccurs="0"
maxOccurs="unbounded"/>
</sequence>
</extension>
</complexContent>
</complexType>

<complexType name="holderType">
<sequence>
<complexType name="contactType">
    <sequence>
        <element name="name" type="token" minOccurs="0" />
        <element name="org" type="token" minOccurs="0" />
        <element name="addr" type="mark:addrType" />
        <element name="voice" type="mark:e164Type" minOccurs="0" />
        <element name="fax" type="mark:e164Type" minOccurs="0" />
        <element name="email" type="mark:minTokenType" minOccurs="0" />
    </sequence>
    <attribute name="type" type="mark:contactTypeType" />
</complexType>

<complexType name="trademarkType">
    <sequence>
        <element name="id" type="mark:idType" />
        <element name="markName" type="token" />
        <element name="holder" type="mark:holderType" maxOccurs="unbounded" />
        <element name="contact" type="mark:contactType" minOccurs="0" maxOccurs="unbounded" />
        <element name="jurisdiction" type="mark:ccType" />
        <element name="class" type="integer" minOccurs="0" maxOccurs="unbounded" />
        <element name="label" type="mark:labelType" minOccurs="0" maxOccurs="unbounded" />
        <element name="goodsAndServices" type="token" />
        <element name="apId" type="token" minOccurs="0" />
        <element name="apDate" type="dateTime" minOccurs="0" />
        <element name="regNum" type="token" />
        <element name="regDate" type="dateTime" />
        <element name="exDate" type="dateTime" minOccurs="0" />
    </sequence>
</complexType>

<complexType name="treatyOrStatuteType">
    <sequence>
        <element name="id" type="mark:idType" />
        <element name="markName" type="token" />
    </sequence>
</complexType>
<element name="holder" type="mark:holderType" maxOccurs="unbounded" />
<element name="contact" type="mark:contactType" minOccurs="0" maxOccurs="unbounded" />
<element name="protection" type="mark:protectionType" maxOccurs="unbounded" />
<element name="label" type="mark:labelType" minOccurs="0" maxOccurs="unbounded" />
<element name="goodsAndServices" type="token" />
<element name="refNum" type="token" />
<element name="proDate" type="dateTime" />
<element name="title" type="token" />
<element name="execDate" type="dateTime" />
</sequence>
</complexType>

<complexType name="courtType">
  <sequence>
    <element name="id" type="mark:idType" />
    <element name="markName" type="token" />
    <element name="holder" type="mark:holderType" maxOccurs="unbounded" />
    <element name="contact" type="mark:contactType" minOccurs="0" maxOccurs="unbounded" />
    <element name="label" type="mark:labelType" minOccurs="0" maxOccurs="unbounded" />
    <element name="goodsAndServices" type="token" />
    <element name="refNum" type="token" />
    <element name="proDate" type="dateTime" />
    <element name="cc" type="mark:ccType" />
    <element name="region" type="token" minOccurs="0" maxOccurs="unbounded" />
    <element name="courtName" type="token" />
  </sequence>
</complexType>

<!-- Address (<mark:addr>) child elements  -->
<complexType name="addrType">
  <sequence>
    <element name="street" type="token" minOccurs="1" maxOccurs="3" />
    <element name="city" type="token" />
    <element name="sp" type="token" minOccurs="0" maxOccurs="0" />
    <element name="pc" type="mark:pcType" minOccurs="0" maxOccurs="0" />
    <element name="cc" type="mark:ccType" />
  </sequence>
</complexType>
<mark:protection> child elements

<complexType name="protectionType">
  <sequence>
    <element name="cc" type="mark:ccType"/>
    <element name="region" type="token" minOccurs="0"/>
    <element name="ruling" type="mark:ccType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

Postal code definition

<simpleType name="pcType">
  <restriction base="token">
    <maxLength value="16"/>
  </restriction>
</simpleType>

Country code definition

<simpleType name="ccType">
  <restriction base="token">
    <length value="2"/>
  </restriction>
</simpleType>

Phone number with extension definition

<complexType name="e164Type">
  <simpleContent>
    <extension base="mark:e164StringType">
      <attribute name="x" type="token"/>
    </extension>
  </simpleContent>
</complexType>

Phone number with extension definition

<simpleType name="e164StringType">
  <restriction base="token">
    <pattern value="(\+[0-9]{1,3}\.[0-9]{1,14})?"/>
    <maxLength value="17"/>
  </restriction>
</simpleType>
</restriction>
</simpleType>

<!--
Id type definition  
-->
<simpleType name="idType">
  <restriction base="token">
    <pattern value="\d+-\d+"/>
  </restriction>
</simpleType>

<!--
DNS label type definition  
-->
<simpleType name="labelType">
  <restriction base="token">
    <minLength value="1"/>
    <maxLength value="63"/>
    <pattern value="[a-zA-Z0-9]([-a-zA-Z0-9]*[a-zA-Z0-9])?"/>
  </restriction>
</simpleType>

<!--
Type used for email addresses  
-->
<simpleType name="minTokenType">
  <restriction base="token">
    <minLength value="1"/>
  </restriction>
</simpleType>

<simpleType name="entitlementType">
  <restriction base="token">
    <enumeration value="owner"/>
    <enumeration value="assignee"/>
    <enumeration value="licensee"/>
  </restriction>
</simpleType>

<simpleType name="contactTypeType">
  <restriction base="token">
    <enumeration value="owner"/>
    <enumeration value="agent"/>
    <enumeration value="thirdparty"/>
  </restriction>
</simpleType>
4. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 6982 [RFC6982] before publication.

This section records the status of known implementations of the format defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982 [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982 [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

4.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-ietf-eppext-tmch-smd.

Level of maturity: Production

Coverage: All aspects of the draft-ietf-eppext-tmch-smd are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

4.2. Verisign Consolidated Top Level Domain (CTLD) SRS

Organization: Verisign Inc.

Name: Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS)

Description: The Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS) implements the server-side of draft-ietf-eppext-tmch-smd for a variety of Top Level Domains (TLD’s).

Level of maturity: Production

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark. Implements the encoding of the Mark in supporting the response of draft-ietf-eppext-launchphase.

Licensing: Proprietary

Contact: jgould@verisign.com

4.3. Verisign .COM / .NET SRS

Organization: Verisign Inc.

Name: Verisign .COM / .NET Shared Registry System (SRS)

Description: The Verisign Shared Registry System (SRS) for .COM, .NET and other IDN TLD’s implements the server-side of draft-ietf-eppext-tmch-smd.

Level of maturity: Operational Test Environment (OTE)

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark.

Licensing: Proprietary

Contact: jgould@verisign.com

4.4. REngin v3.7

Organisation: Domain Name Services (Pty) Ltd

Name: REngin v3.7
Description: Server side implementation only

Level of maturity: Production

Coverage: All aspects of draft-ietf-eppext-tmch-smd have been implemented

Licensing: Proprietary Licensing with Maintenance Contracts

Contact: info@dnservices.co.za

URL: http://domain-name.services

4.5. Uniregistry Corp. Shared Registry System (uSRS)

Organization: Uniregistry Corp.

Name: Uniregistry Corp. Shared Registry System (uSRS)

Description: Uniregistry’s Shared Registry System implements the server-side of draft-ietf-eppext-tmch-smd for its TLD registry.

Level of maturity: Production

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark. Implements the encoding of the Mark in supporting the response of draft-ietf-eppext-launchphase.

Licensing: Proprietary

Contact: fobispo@uniregistry.link

5. Acknowledgements

Special thanks to Chris Wright for creating the first prototype of a SMD; James Gould, Wil Tan and Gavin Brown for creating the mark and SMD definitions in their EPP draft launch extension on which this draft is based. Portions of the security section were shamefully copied from RFC5105. The author would like to acknowledge the following individuals for their contributions to this document: Scott Hollenbeck and Jan Jansen.

6. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. Two URI assignments have been registered by the IANA.
Registration request for the signed mark namespace:

URI: urn:ietf:params:xml:ns:signedMark-1.0

Registrant Contact: IESG

XML: None. Namespace URIs do not represent an XML specification.

Registration request for the signed mark schema:

URI: urn:ietf:params:xml:schema:signedMark-1.0

Registrant Contact: IESG

XML: See the "Formal Syntax" section of this document.

Registration request for the mark namespace:

URI: urn:ietf:params:xml:ns:mark-1.0

Registrant Contact: IESG

XML: None. Namespace URIs do not represent an XML specification.

Registration request for the mark schema:

URI: urn:ietf:params:xml:schema:mark-1.0

Registrant Contact: IESG

XML: See the "Formal Syntax" section of this document.

7. Security Considerations

The security of a Signed Mark object depends on the security of the underlying XML DSIG algorithms. As such, all the security considerations from [XMLDSIG] apply here as well.

The digital signature algorithm used in Signed Mark objects SHOULD be RSA-SHA256 [RFC4051]. The size of the RSA key SHOULD be at least 2048 bits. A valid reason for choosing something else would be if RSA-SHA256 would be deemed to not provide sufficient security.

In the case of the ICANN Trademark Clearinghouse (TMCH), Signed Mark objects use the algorithms for digesting and signing recommended in this document.
Signed Marks are used primarily for sunrise domain name registrations in gTLDs, but other third parties might be using them. A party using Signed Marks should verify that the digital signature is valid based on local policy. In the case of gTLDs, the RPM Requirements document [ICANN-TMCH] defines such policy, and the PKI is defined in [I-D.ietf-eppext-tmch-func-spec]. Implementations will need to implement such a PKI (or an equivalent) in order for the signatures defined in this document to have any useful semantics.

8. References

8.1. Normative References

[ICANN-TMCH]
ICANN, "ICANN Trademark Clearinghouse, Rights Protection Mechanism Requirements", 2013,

[ISO3166-2]
ISO, "International Standard for country codes and codes for their subdivisions", 2006,
<http://www.iso.org/iso/home/standards/country_codes.htm>.

[ITU.E164.2005]
International Telecommunication Union, "The international public telecommunication numbering plan", 2010,

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,

[RFC3688] Meallling, M., "The IETF XML Registry", BCP 81, RFC 3688,
DOI 10.17487/RFC3688, January 2004,

[RFC4051] Eastlake 3rd, D., "Additional XML Security Uniform Resource Identifiers (URIs)", RFC 4051,
DOI 10.17487/RFC4051, April 2005,

8.2. Informative References

[Refs]


Author’s Address

Lozano
Expires September 10, 2016
Gustavo Lozano
ICANN
12025 Waterfront Drive, Suite 300
Los Angeles 90292
US

Phone: +1.3103015800
Email: gustavo.lozano@icann.org
Extensible Provisioning Protocol (EPP) Domain Name Mapping Extension for Bundling Registration
draft-kong-eppext-bundling-registration-02

Abstract

This document describes an extension of Extensible Provisioning Protocol (EPP) domain name mapping for the provisioning and management of bundling registration of domain names. Specified in XML, this mapping extends the EPP domain name mapping to provide additional features required for the provisioning of bundled domain names.

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 April 15, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
Table of Contents

1. Introduction ................................................. 3
2. Terminology .................................................. 4
3. Definitions ................................................... 4
4. Overview ..................................................... 4
5. Requirement for Bundling Registration of Names .......... 5
6. Object Attributes ........................................... 6
   6.1. RDN ..................................................... 6
   6.2. BDN ..................................................... 6
7. EPP Command Mapping ......................................... 6
   7.1. EPP Query Commands ..................................... 7
      7.1.1. EPP <check> Command ............................... 7
      7.1.2. EPP <info> Command ................................. 8
      7.1.3. EPP <transfer> Query Command ..................... 9
   7.2. EPP Transform Commands ................................ 9
      7.2.1. EPP <create> Command ............................. 10
      7.2.2. EPP <delete> Command ............................. 12
      7.2.3. EPP <renew> Command .............................. 13
      7.2.4. EPP <transfer> Command ........................... 14
      7.2.5. EPP <update> Command .............................. 14
8. Formal Syntax ................................................ 14
9. Internationalization Considerations ..................... 16
10. IANA Considerations ........................................ 16
11. Security Considerations ................................... 17
12. Implementation Status ..................................... 17
13. Acknowledgements .......................................... 17
14. Change History ............................................. 18
1. Introduction

Bundled domain names are those who share the same TLD but whose second level labels are variants, or those who has identical second level labels for which certain parameters are shared in different TLDs. For example, Public Interest Registry, request to implement technical bundling of second level domains for .NGO and .ONG. So we have two kinds of bundled domain names. First one is in the form of "V-label.TLD" in which the second level labels (V-label) are variants sharing the same TLD; Second one is in the form of "LABEL.V-tld" in which the second level labels (LABEL) are same with the different TLDs (V-tld);

For the name variants, some registries adopt the policy that variant IDNs which are identified as equivalent are allocated or delegated to the same registrant. For example, the specified registration policy of Chinese Domain Name (CDN) is that a registrant can apply an original CDN in any forms: Simplified Chinese (SC) form, Traditional Chinese (TC) form, or other variant forms, then the corresponding variant CDN in SC form and that in TC form will also be delegated to the same registrant. All variant names in the same TLD contain same attributes.

The basic Extensible Provisioning Protocol (EPP) domain name mapping [RFC5731] provides the domain name registration one by one. It does not specify how to register the bundled names which share the same attributes.

In order to meet above requirements of the bundled names registration, this document describes an extension of the EPP domain name mapping [RFC5731] for the provisioning and management of bundled names. This document is specified using the Extensible Markup Language (XML) 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmleschema-1-20041028] and [W3C.REC-xmleschema-2-20041028].

The EPP core protocol specification [RFC5730] provides a complete description of EPP command and response structures. A thorough understanding of the base protocol specification is necessary to understand the extension of mapping described in this document.
This document uses lots of the concepts of the IDN, so a thorough understanding of the IDNs for Application (IDNA, described in [RFC5890], [RFC5891], and [RFC5892]) and a thorough understanding of variant approach discussed in [RFC4290] are both required.

2. Terminology

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 [RFC2119].

uLabel is defined in [RFC 5890]. uLabel is expressed in this document as a number of characters with the format of U+XXXX where XXXX is a UNICODE point.

"b-dn-1.0" in this document is used as an abbreviation for urn:ietf:params:xml:ns:b-dn-1.0.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this specification.

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented to develop a conforming implementation.

3. Definitions

The following definitions are used in this document:

- Registered Domain Name (RDN), represents the valid domain name that users submitted for registration by the first time.

- Bundled Domain Name (BDN), represents the bundled domain name produced according to the bundled domain name registration policy.

4. Overview

Domain registries have traditionally adopted a registration model whereby metadata relating to a domain name, such as its expiration date and sponsoring registrar, are stored as properties of the domain object. The domain object is then considered an atomic unit of registration, on which operations such as update, renewal and deletion may be performed.

Bundled names, brought about the need for multiple domain names to be registered and managed as a single package. In this model, the
registry typically accepts a domain registration request (i.e. EPP domain <create> command) containing the domain name to be registered. This domain name is referred to as the RDN in this document. As part of the processing of the registration request, the registry generates a set of bundled names that are related to the RDN, either programmatically or with the guidance of registration policies, and place them in the registration package together with the RDN.

The bundled names have the same properties, such as expiration date and sponsoring registrar, by sharing one domain object. So when users update any property of a domain object within a bundle package, that property of all other domain objects in the bundle package will be updated at the same time.

5. Requirement for Bundling Registration of Names

The bundled names whether they are in the form of "V-label.TLD" or in the form of "LABEL.V-tld" should share some parameter or attributes associated with domain names. Typically, Bundled names will share the following parameters or attributes:
- Registrar Ownership
- Registration and Expiry Dates
- Registrant, Admin, Billing, and Technical Contacts
- Name Server Association
- Domain Status
- Applicable grace periods (Add Grace Period, Renewal Grace Period, Auto-Renewal Grace Period, Transfer Grace Period, and Redemption Grace Period)

Because the domain names are bundled and share the same parameters or attributes, the EPP command should do some processing for these requirements:
- When performing a domain check, either BDN or RDN can be queried for the EPP command, and will return the same response.
- When performing a domain info, either BDN or RDN can be queried, the same response will include both BDN and RDN information with the same attributes.
- When performing a domain Create, either BDN or RDN will be accepted. If the domain name is available, both BDN and RDN will be registered.
- When performing a domain Delete, either BDN or RDN will be accepted. If the domain name is available, both BDN and RDN will be deleted.
- When performing a domain renew, either BDN or RDN will be accepted. Upon a successful domain renewal, both BDN and RDN will have their expiry date extended by the requested term. Upon a successful domain renewal, both BDN and RDN will conform to the same renew grace period.
When performing a domain transfer, either BDN or RDN will be accepted. Upon successful completion of a domain transfer request, both BDN and RDN will enter a pendingTransfer status. Upon approval of the transfer request, both BDN and RDN will be owned and managed by the same new registrant.

When performing a domain update, either BDN or RDN will be accepted. Any modifications to contact associations, name server associations, domain status values and authorization information will be applied to both BDN and RDN.

6. Object Attributes

This extension defines following additional elements to the EPP domain name mapping [RFC5731]. All of these additional elements can be got from <domain:info> command.

6.1. RDN

The RDN is an ASCII name or an IDN with the A-label [RFC5890] form. In this document, its corresponding element is <b-dn:rdn>. An optional attribute "uLabel" associated with <b-dn:rdn> is used to represent the U-label [RFC5890] form. An optional boolean "activated" attribute, with a default true value, is used to indicate the presence of the label in the zone file.

For example: <b-dn:rdn uLabel="U+5B9E""U+4F8B".example> xn--fsq270a.example</b-dn:rdn>

6.2. BDN

The BDN is an ASCII name or an IDN with the A-label [RFC5890] form which is converted from the corresponding BDN. In this document, its corresponding element is <b-dn:bdn>. An optional attribute "uLabel" associated with <b-dn:bdn> is used to represent the U-label [RFC5890] form.

For example: <b-dn:bdn uLabel="U+5BE6""U+4F8B".example> xn--fsqz41a.example</b-dn:bdn>

7. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in provisioning and managing bundled names via EPP.
7.1. EPP Query Commands

EPP provides three commands to retrieve domain information: <check> to determine if a domain object can be provisioned within a repository, <info> to retrieve detailed information associated with a domain object, and <transfer> to retrieve domain-object transfer status information.

7.1.1. EPP <check> Command

This extension does not add any element to the EPP <check> command or <check> response described in the EPP domain name mapping [RFC5731]. However, when either RDN or BDN is sent for check, response SHOULD contain both RDN and BDN information, which may also give some explanation in the reason field to tell the user that the associated domain name is a produced name according to some bundle domain name policy.

Example <check> Response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <domain:chkData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:cd>
S:          <domain:name avail="1">
S:            xn--fsq270a.example</domain:name>
S:        </domain:cd>
S:        <domain:cd>
S:          <domain:name avail="1">
S:            xn--fsqz41a.example</domain:name>
S:          <domain:reason>This associated domain name is a produced name based on bundle name policy.</domain:reason>
S:        </domain:cd>
S:      </domain:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```
7.1.2. EPP <info> Command

This extension does not add any element to the EPP <info> command described in the EPP domain mapping [RFC5731]. However, additional elements are defined for the <info> response.

When an <info> command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:infData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:infData> element contains the <b-dn:bundle> which has the following child elements:

- An <b-dn:rdn> element that contains the RDN, along with the attributes described below.
- An OPTIONAL <b-dn:bdn> element that contains the BDN, along with the attributes described below.

The above elements contain the following attributes:

- An optional "uLabel" attribute represents the U-label of the element.

Example <info> Response for an authorized client:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:       <domain:infData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>xn--fsq270a.example</domain:name>
S:         <domain:roid>58812678-domain</domain:roid>
S:         <domain:status s="ok"/>
S:         <domain:registrant>123</domain:registrant>
S:         <domain:contact type="admin">123</domain:contact>
S:         <domain:contact type="tech">123</domain:contact>
S:         <domain:ns>
S:             <domain:hostObj>ns1.example.cn</domain:hostObj>
S:         </domain:ns>
S:         <domain:clID>ClientX</domain:clID>
S:     </domain:infData>
S:   </resData>
S: </response>
S: </epp>
S: <domain:crID>ClientY</domain:crID>
S: <domain:crDate>2011-04-03T22:00:00.0Z</domain:crDate>
S: <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S: <domain:authInfo>
  <domain:pw>2fooBAR</domain:pw>
</domain:authInfo>
S: <domain:infData>
S: </domain:infData>
S: </resData>
S: <extension>
S: <b-dn:infData
  xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
S: <b-dn:bundle>
  <b-dn:rdn uLabel="U+5B9E"u+4F8B".example
    xn--fsq270a.example</b-dn:rdn>
  <b-dn:bdn uLabel="U+5BE6"u+4F8B".example
    xn--fsqz41a.example</b-dn:bdn>
</b-dn:bundle>
S: </b-dn:infData>
S: </extension>
S: <trID>
S: <clTRID>ABC-12345</clTRID>
S: <svTRID>54322-XYZ</svTRID>
S: </trID>
S: </response>
S:</epp>

<info> Response for the unauthorized client has not been changed, see
[RFC5731] for detail.

An EPP error response MUST be returned if an <info> command cannot be
processed for any reason.

7.1.3. EPP <transfer> Query Command

This extension does not add any element to the EPP <transfer> command
or <transfer> response described in the EPP domain mapping [RFC5731].

7.2. EPP Transform Commands

EPP provides five commands to transform domain objects: <create> to
create an instance of a domain object, <delete> to delete an instance
of a domain object, <renew> to extend the validity period of a domain
object, <transfer> to manage domain object sponsorship changes, and
<update> to change information associated with a domain object.
When these commands have been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:bundle> which has the following child elements:

- An <b-dn:rdn> element that contains the RDN, along with the attributes described below.
- An OPTIONAL <b-dn:bdn> element that contains the BDN, along with the attributes described below.

The above elements contain the following attribute:

- An optional "uLabel" attribute represents the U-label of the element.

7.2.1. EPP <create> Command

This extension defines additional elements to extend the EPP <create> command described in the EPP domain name mapping [RFC5731] for bundled names registration.

In addition to the EPP command elements described in the EPP domain mapping [RFC5731], the <create> command SHALL contain an <extension> element. The <extension> element SHOULD contain a child <b-dn:create> element that identifies the bundle namespace and the location of the bundle name schema.
Example <create> command:

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:     <command>
C:       <create>
C:         <domain:create
C:           xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:           <domain:name>xn--fsq270a.example</domain:name>
C:           <domain:period unit="y">2</domain:period>
C:           <domain:registrant>123</domain:registrant>
C:           <domain:contact type="admin">123</domain:contact>
C:           <domain:contact type="tech">123</domain:contact>
C:           <domain:authInfo>
C:             <domain:pw>2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:create>
C:     </command>
C: </epp>

When an <create> command has been processed successfully, the EPP <creData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:creData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:creData> element contains the <b-dn:bundle> element.
Example <create> Response for an authorized client:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:       <domain:creData
S:           xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>xn--fsq270a.example</domain:name>
S:         <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:         <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:creData>
S:   </resData>
S:   <extension>
S:     <b-dn:creData
S:         xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
S:       <b-dn:bundle>
S:         <b-dn:rdn uLabel="U+5B9E""U+4F8B".example
S:             >xn--fsq270a.example</b-dn:rdn>
S:         <b-dn:bdn uLabel="U+5BE6""U+4F8B".example
S:             >xn--fsqz41a.example</b-dn:bdn>
S:       </b-dn:bundle>
S:     </b-dn:creData>
S:   </extension>
S:   <trID>
S:     <clTRID>ABC-12345</clTRID>
S:     <svTRID>54322-XYZ</svTRID>
S:   </trID>
S: </response>
S:</epp>

<create> Response for the unauthorized client has not been changed, see [RFC5731] for detail.

An EPP error response MUST be returned if an <create> command cannot be processed for any reason.

7.2.2. EPP <delete> Command

This extension does not add any element to the EPP <delete> command described in the EPP domain mapping [RFC5731]. However, additional elements are defined for the <delete> response.

When a <delete> command has been processed successfully, the EPP <delData> element MUST contain child elements as described in the EPP
domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:delData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:delData> element SHOULD contain the <b-dn:bundle> element.

Example <delete> response:

```xml
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:         <b-dn:delData
S:           xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
S:          <b-dn:bundle>
S:            <b-dn:rdn uLabel="U+5B9E""U+4F8B".example>xn--fsq270a.example</b-dn:rdn>
S:            <b-dn:bdn uLabel="U+5BE6""U+4F8B".example>xn--fsqz41a.example</b-dn:bdn>
S:          </b-dn:bundle>
S:         </b-dn:delData>
S:       </extension>
S:     </response>
S: </epp>
```

An EPP error response MUST be returned if a <delete> command cannot be processed for any reason.

7.2.3. EPP <renew> Command

This extension does not add any element to the EPP <renew> command described in the EPP domain name mapping [RFC5731]. However, when either RDN or BDN is sent for renew, response SHOULD contain both RDN and BDN information. When the command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:renData> which contains <b-dn:bundle> element.
7.2.4. EPP <transfer> Command

This extension does not add any element to the EPP <transfer> command described in the EPP domain name mapping [RFC5731]. When the command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:trnData> which contains <b-dn:bundle> element.

7.2.5. EPP <update> Command

This extension does not add any element to the EPP <update> command described in the EPP domain name mapping [RFC5731]. When the command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:upData> which contains <b-dn:bundle> element.

8. Formal Syntax

An EPP object name mapping extension for bundled names is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:b-dn-1.0"
xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0"
xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<!--
Import common element types.
-->
<import namespace="urn:iana:xml:ns:eppcom-1.0"
schemaLocation="eppcom-1.0.xsd"/>
<import namespace="urn:iana:xml:ns:epp-1.0"
schemaLocation="epp-1.0.xsd"/>
<annotation>
<documentation>
Extensible Provisioning Protocol v1.0
Bundle Domain Extension Schema v1.0

Child elements found in EPP commands.

<!--
Child elements of the <b-dn:create> command
All elements must be present at time of creation
-->
<complexType name="createDataType">
  <sequence>
    <element name="rdn" type="b-dn:rdnType" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

<!--
Child elements of the <b-dn:update> command
All elements must be present at time of creation
-->
<!--
Child elements found in EPP commands.
-->
<!--
<element name="infData" type="b-dn:trnDataType"/>
<element name="delData" type="b-dn:trnDataType"/>
<element name="creData" type="b-dn:trnDataType"/>
<element name="renData" type="b-dn:trnDataType"/>
<element name="trnData" type="b-dn:trnDataType"/>
<element name="upData" type="b-dn:trnDataType"/>
<complexType name="trnDataType">
  <sequence>
    <element name="bundle" type="b-dn:bundleType" />
  </sequence>
</complexType>

<!--
<transfer> response elements.
All elements must be present at time of poll query
-->
<complexType name="bundleType">
  <sequence>
    <element name="rdn" type="b-dn:rdnType" />
  </sequence>
</complexType>
9. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 and UTF-16. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED.

As an extension of the EPP domain name mapping, the elements, element content described in this document MUST inherit the internationalization conventions used to represent higher-layer domain and core protocol structures present in an XML instance that includes this extension.

10. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. IANA is requested to assign the following two URIs.

Registration request for the IDN namespace:

- URI: urn:ietf:params:xml:ns:b-dn-1.0
o Registrant Contact: See the "Author’s Address" section of this document.

o XML: None. Namespace URI does not represent an XML specification.

Registration request for the IDN XML schema:

- URI: urn:ietf:params:xml:schema:b-dn-1.0

- Registrant Contact: See the "Author’s Address” section of this document.

- XML: See the "Formal Syntax” section of this document.

11. Security Considerations

The object mapping extension described in this document does not provide any other security services or introduce any additional considerations beyond those described by [RFC5730] or those caused by the protocol layers used by EPP.

12. Implementation Status

Note to RFC Editor: Please remove this section before publication.

- CNNIC has implemented this extension in his EPP based Chinese domain name registration system.

- Public Interest Registry, has requested to implement technical bundling of second level domains for .NGO and .ONG. This means that by registering and purchasing a domain in the .ngo TLD, for example, the NGO registrant is also registering and purchasing the corresponding name in the .ong TLD (and vice-versa for registrations in .ong).

13. Acknowledgements

The authors especially thank the authors of [RFC5730] and [RFC5731] and the following ones of CNNIC: Weiping Yang, Chao Qi. This draft extends the draft draft-kong-epp-idn-variants-mapping to support both forms of bundled names: V-label.TLD and LABEL.V-tld.

Useful comments were made by John Klensin, Scott Hollenbeck, Patrick Mevzek and Edward Lewis.
14. Change History

RFC Editor: Please remove this section.

14.1. draft-kong-epp-bundle-mapping: Version 00

- EPP extension for bundled domain name registrations.

14.2. draft-kong-epp-bundle-mapping: Version 01

- Change the proposed category from EXP to STD.
- Add the section of Implementation Status.
- Refine the text, and update the examples.

14.3. draft-kong-epp-bundle-mapping: Version 02

- Refine the texts.

15. References

15.1. Normative References


15.2. Informative References

[bundle.name]


Authors' Addresses

Ning Kong  
CNNIC  
4 South 4th Street, Zhongguancun, Haidian District  
Beijing, Beijing 100190  
China  
Phone: +86 10 5881 3147  
Email: nkong@cnnic.cn

Jiankang Yao (editor)  
CNNIC  
4 South 4th Street, Zhongguancun, Haidian District  
Beijing, Beijing 100190  
China  
Phone: +86 10 5881 3007  
Email: yaojk@cnnic.cn

Xiaodong Li  
CNNIC  
4 South 4th Street, Zhongguancun, Haidian District  
Beijing, Beijing 100190  
China  
Phone: +86 10 5881 3020  
Email: xl@cnnic.cn

Jiagui Xie  
CONAC  
Jia 31, North Guangximen, Xibahe, Chaoyang District  
Beijing, Beijing 100028  
China  
Phone: +86 10 10 5203 5025  
Email: xiejg@conac.cn
Abstract

The Canadian Internet Registration Authority (CIRA), administering the .CA country-code top-level domain, offers internationalized domain names (IDN) in French, one of Canada’s official languages. CIRA’s Extensible Provisioning Protocol (EPP) services have been augmented with an IDN EPP extension in order to support registrars desiring to register internationalized domains using French characters as bundled domains.

This document defines the extension to the Extensible Provisioning Protocol used at CIRA to support IDN operations.

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 7, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents.
1. Introduction ....................................................... 3
2. Operational Requirements .......................................... 3
  2.1. IDN Labels .................................................... 4
  2.2. Repertoires .................................................. 4
  2.3. French IDN Characters Variants ............................... 4
  2.4. Bundling and Registration Policy ............................ 5
  2.5. Domain Lists ................................................ 6
3. Object Elements .................................................. 6
  3.1. U-Labels ...................................................... 7
  3.2. Repertoires .................................................. 7
  3.3. DomainVariants .............................................. 7
  3.4. BundleDomains ............................................... 8
  3.5. Info ........................................................ 8
4. EPP Extension Object .............................................. 8
  4.1. The ciraIdnCheck Object ..................................... 8
  4.2. The ciraIdnInfo Object ...................................... 8
  4.3. The ciraIdnCreate Object .................................... 9
5. EPP Command Mapping ............................................... 9
  5.1. EPP Query Commands .......................................... 9
    5.1.1. EPP <check> Command ................................... 9
    5.1.2. EPP <info> Command ................................... 11
    5.1.3. EPP <poll> Command .................................... 14
    5.1.4. EPP <transfer> Command ................................ 14
  5.2. EPP Transform Commands ..................................... 14
    5.2.1. EPP <create> Command .................................. 14
    5.2.2. EPP <delete> Command .................................. 15
    5.2.3. EPP <renew> Command ................................... 16
    5.2.4. EPP <transfer> Command ................................ 16
    5.2.5. EPP <update> Command .................................. 16
6. Formal Syntax .................................................... 16
  6.1. Schema for cira-idn-1.0 ..................................... 16
  6.2. Schema for cira-idn-bundle-1.0 ............................... 18
7. Security Considerations .......................................... 19
8. IANA Considerations .............................................. 20
9. Acknowledgments .................................................. 20
10. References ....................................................... 20
    10.1. Normative References ...................................... 20
    10.2. Informative References ................................... 21
1. Introduction

This document describes an extension to the Extensible Provisioning Protocol (EPP) providing support for the internationalization of domain names (IDN) and other related functions. This EPP extension is used at the Canadian Internet Registration Authority (CIRA) to support IDN operations for French, one of Canada’s official languages.

The design of this EPP extension is based on a number of requirements from the CIRA registry. A first requirement is to exchange IDN labels with registrars during create operations. There is also a need to exchange information about the set of characters supported by the registry. This set is called a repertoire throughout the document, as a synonym with IDN-table.

Registry policies often prevent the registration of similar-looking IDN labels by different registrants. CIRA policies allow each IDN label to be registered independently, removing the need for a registration bundle structure as described in [RFC4290] or more recently in [ID.draft-kong-eppext-bundling-registration]. The policies also specify that all the label variants must be linked to a single registrant-registrar pair, even if some variants aren’t registered.

In order to support this, a simple list of label variants is used. No zone information or registration information is included in that list. By providing such list to a registrar, a registry has the capacity to inform a registrar of the possible label variants without relying on the accurate processing of complex Label Generation Rulesets (LGR) on the registrar side, minimizing risks of errors.

The CIRA IDN EPP extension defines three objects named createType, infDataType and checkType, respectively used in <create>, <info> and <check> EPP commands. These objects contain elements describing a "repertoire", as a set of variant Unicode code points, an IDN label in U-label form and "domainVariants", a list of variant labels accepted by the registry (also called bundle). A new type named repertoireType is also defined. This type is used to carry a set of variant Unicode code points supported by a registry.

2. Operational Requirements

This section explains the rationale and detailed requirements behind CIRA’s EPP extension for IDN.
2.1. IDN Labels

The first requirement relating to IDN support is to allow registrars to specify a U-label string in a create operation. For this purpose a new element is required, in this case called ‘u-label’.

2.2. Repertoires

French being the first IDN language supported at CIRA, it would be possible to accept EPP create operations without other information than the U-label and to consider the French character set as the default for all operations, which would include the base English character set already in place. However CIRA did not want to preclude the possibility to add support for other languages in the future or to add supplementary constrained character sets. The registrars are therefore required to specify the character set in every create operation.

The character set is being specified in a repertoire type as a string. The name ‘repertoire’ as used here represents the set of code points variants accepted by policy by the registry for a specific language. This has the same meaning as the definition of idn:table identifier defined in [ID.draft-kong-eppext-bundling-registration].

The string used to identify a repertoire may be similar in content to a language tag, but shouldn’t be confused with a language, as the character set approved by policy by a registry may represent a subset of an official language’s character set. See section 1.3 of [RFC4290] for a more detailed discussion on possible confusion in usage.

2.3. French IDN Characters Variants

The code points below are included in the implementation of French IDN by CIRA. Each French accented character is considered a variant of the base character.
<table>
<thead>
<tr>
<th>Code Point</th>
<th>French Character</th>
<th>Base Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+00E0</td>
<td>LATIN SMALL LETTER A WITH GRAVE</td>
<td>a</td>
</tr>
<tr>
<td>U+00E2</td>
<td>LATIN SMALL LETTER A WITH CIRCUMFLEX</td>
<td>a</td>
</tr>
<tr>
<td>U+00E7</td>
<td>LATIN SMALL LETTER C WITH CEDILLA</td>
<td>c</td>
</tr>
<tr>
<td>U+00E8</td>
<td>LATIN SMALL LETTER E WITH GRAVE</td>
<td>e</td>
</tr>
<tr>
<td>U+00E9</td>
<td>LATIN SMALL LETTER E WITH ACUTE</td>
<td>e</td>
</tr>
<tr>
<td>U+00EA</td>
<td>LATIN SMALL LETTER E WITH CIRCUMFLEX</td>
<td>e</td>
</tr>
<tr>
<td>U+00EB</td>
<td>LATIN SMALL LETTER E WITH DIAERESIS</td>
<td>e</td>
</tr>
<tr>
<td>U+00EE</td>
<td>LATIN SMALL LETTER I WITH CIRCUMFLEX</td>
<td>i</td>
</tr>
<tr>
<td>U+00EF</td>
<td>LATIN SMALL LETTER I WITH DIAERESIS</td>
<td>i</td>
</tr>
<tr>
<td>U+00F4</td>
<td>LATIN SMALL LETTER O WITH CIRCUMFLEX</td>
<td>o</td>
</tr>
<tr>
<td>U+00F9</td>
<td>LATIN SMALL LETTER U WITH GRAVE</td>
<td>u</td>
</tr>
<tr>
<td>U+00FB</td>
<td>LATIN SMALL LETTER U WITH CIRCUMFLEX</td>
<td>u</td>
</tr>
<tr>
<td>U+00FC</td>
<td>LATIN SMALL LETTER U WITH DIAERESIS</td>
<td>u</td>
</tr>
<tr>
<td>U+00FF</td>
<td>LATIN SMALL LETTER Y WITH DIAERESIS</td>
<td>y</td>
</tr>
<tr>
<td>U+00E6</td>
<td>LATIN SMALL LETTER AE</td>
<td>ae</td>
</tr>
<tr>
<td>U+0153</td>
<td>LATIN SMALL LIGATURE OE</td>
<td>oe</td>
</tr>
</tbody>
</table>

Figure 1: Variant code points for French IDN at CIRA

### 2.4. Bundling and Registration Policy

A group of label variants referring to the same canonical base label (without accented characters) is named a bundle in this document. This definition is similar to the definition of registration bundle provided in [RFC4290], with the difference that it isn't used for registration at CIRA but only to retrieve information through an EPP <info> operation.

Concurrent registration of different labels in a bundle is not required by CIRA policy. The registration of individual variants remains independent, with the requirement that the registrar-registrant pair remains the same for every label in the bundle. Once a single variant has been registered, registration of a variant by different registrant or registrar is prevented. This includes the registration of canonical names from a period preceding IDN support.

In a way similar to registration, transfers are handled on a per-label basis. All labels within a bundle must be transferred within 5 days otherwise the transfer is canceled.

Label variants, besides being independently registered, are also allowed to have different lifetimes, expiration times and server information. The limitations on the bundle are managed using the...
earliest registration date to the latest expiry date for all labels in a single bundle. An expired variant part of a non-expired bundle is considered ‘withheld’ and cannot be registered except by the same registrant-registrar pair.

| Variant1   | ---R++++++++++++++++++E--------- |
| Variant2   | ------R++++++++++++++++++E------ |
| Variant3   | ------------R++++++++++++++++++E-- |
| Bundle     | -----------------------------+--- |
| lifetime   | R=Registration, E=Expiry, + =valid, - =invalid |

Figure 2: Bundle lifetime

In the context of CIRA, the bundle object will provide additional information regarding the canonical base label, the registrar and the registrant. Several fields relating to important dates, such as creation date, last update, transfer date and expiration are also included. The bundle object is not used for registration and transfers.

2.5. Domain Lists

Because of the limitations on variant registration described above, it is desirable for the registry to specify the exact list of allowed label variants. A registrar could, for example, use that list to display information to prospective registrants or to explain registration errors in a registrant-facing interface.

The CIRA IDN EPP extension adds support for a domainList element returned as a result to Info operations. The domainList contains a list of allowed label variants based on the requested label.

Using a list is simpler to implement for registrars and does not require them to process complex rules (or Label Generation Rule sets) as defined in [ID.draft-davies-idntables]. This limits implementation errors in the parsing of these rules and the need to refresh and process them on a regular basis.

3. Object Elements

In order to cover the requirements described above, five elements are defined in this EPP extension. Elements u-label, repertoire and domainVariants are defined as part of the base CIRA IDN extension. The repertoire identifies a character table (idn-table) and the
domainVariants carries a list of label variants. The two other elements, bundleDomain and info, are used in the bundle manipulation part of the extension.

3.1. U-Labels

The u-label element is an optional parameter used to specify explicitly the value of a U-label. It will be validated against the A-label value found in the domain:name attribute. It is based on the type labelType.

\[\text{<element name="u-label" type="eppcom:labelType" minOccurs="0"/>}\]

3.2. Repertoires

The repertoire element is based on the repertoireType type defined in the extension and is a simple string token limited to 2 characters. The attribute value is linked to a character table describing which code points are valid for this language.

\[\text{<element name="repertoire" type="cira-idn:repertoireType"/>}\]
\[\text{<simpleType name="repertoireType">}\]
\[\text{<restriction base="token">}\]
\[\text{<length value="2"/>}\]
\[\text{</restriction>}\]
\[\text{</simpleType>}\]

3.3. DomainVariants

The optional domainVariants element is based on the domainList type, which is a sequence of elements of labelType. Each label enumerated in domainVariants represents the possible label variants for a base domain.

\[\text{<element name="domainVariants" type="cira-idn:domainList" minOccurs="0"/>}\]
\[\text{<complexType name="domainList">}\]
\[\text{<sequence>}\]
\[\text{<element name="name" type="eppcom:labelType" maxOccurs="unbounded"/>}\]
\[\text{</sequence>}\]
\[\text{</complexType>}\]
3.4. BundleDomains

The bundleDomains element uses the same domainList type as domainVariants and is used in a similar way, but in a bundle object rather than in a ciraIdnInfo object. The exact usage is described below in the extension objects section.

<element name="bundleDomains" type="cira-idn:domainList"/>

3.5. Info

The info element is based on the infoType type, built out of a label type and a repertoire id. This is similar to the u-label and repertoire types defined above, but this composed type is only used in bundle objects and may contain either a u-label or an a-label.

<element name="info" type="cira-idn-bundle:infoType" />
<complexType name="infoType">
    <sequence>
        <element name="name" type="eppcom:labelType" />
        <element name="repertoire" type="cira-idn:repertoireType" minOccurs="0"/>
    </sequence>
</complexType>

4. EPP Extension Object

4.1. The ciraIdnCheck Object

The ciraIdnCheck object is used in <check> commands requests to specify which repertoire (language) is used. Multiple repertoire elements may be specified if needed.

<element name="ciraIdnCheck" type="cira-idn:checkType" />
<complexType name="checkType">
    <sequence>
        <element name="repertoire" type="cira-idn:repertoireType" />
    </sequence>
</complexType>

4.2. The ciraIdnInfo Object

The ciraIdnInfo object is used in <info> command responses to a client when multiple IDN labels variants exist according to the registry policies.
4.3. The ciraIdnCreate Object

The ciraIdnCreate object is used in <info> command responses to a client when multiple IDN labels variants exist according to the registry policies.

5. EPP Command Mapping

5.1. EPP Query Commands

5.1.1. EPP <check> Command

The following is an example of IDN EPP Domain Check transaction in which the client includes the ciraIdnCheck object in the request:

C: <?xml version="1.0" encoding="UTF-8"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <check>
C:       <domain:check
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>abc123.ca</domain:name>
C:         <domain:name>xyz987.ca</domain:name>
C:         <domain:name>xn--r-wfan6a.ca</domain:name>
C:       </domain:check>
C:     </check>
C:   </command>
C: </epp>

C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <extension>
C:       <cira-idn:ciraIdnCheck
C:         xmlns:cira-idn="urn:ietf:params:xml:ns:cira-idn-1.0">
C:         <cira-idn:repertoire>fr</cira-idn:repertoire>
C:     </cira-idn:ciraIdnCheck>
C:   </extension>
C: </epp>
This command returns successful completions responses as specified in [RFC5730] if no error occurred, usually with a code 1000.

If the request is invalid, return code 2005 is used. The error value is set to 8309 if the specified repertoire is invalid and to 8001 if the label contains invalid characters or cannot be converted.
5.1.2. EPP <info> Command

The base CIRA IDN extension does not modify the EPP <info> command sent by the client. In the case of a bundle command, a cira-idn-bundle-info object is added to the request. The two cases are covered below.

5.1.2.1. EPP <info> command for an IDN domain

If the queried domain is an IDN domain in A-label format, the response is modified to include a ciraIdnInfo object containing all the valid label variants for the domain, including the base label.

```xml
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <info>
C:       <domain:info
xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name hosts="all">xn--r-wfan6a.ca</domain:name>
C:         <domain:authInfo>
C:           <domain:pw>password</domain:pw>
C:         </domain:authInfo>
C:       </domain:info>
C:     </info>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>

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>Command completed successfully</msg>
S:     </result>
S:   <resData>
S:     <domain:infData
xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:       <domain:name>xn--r-wfan6a.ca</domain:name>
S:       <domain:roid>CIRA-lifecycle-00122</domain:roid>
S:       <domain:status s="serverUpdateProhibited">
S:         change registrant
S:       </domain:status>
S:       <domain:status s="serverDeleteProhibited" />
S:       <domain:status s="serverRenewProhibited" />
S:       <domain:status s="serverTransferProhibited" />
S:       <domain:status s="serverHold" />
S:       <domain:registrant>rant003</domain:registrant>
S:     </domain:infData>
S:   </resData>
S: </epp>
```
This command returns successful completions responses as specified in [RFC5730] if no error occurred, usually with a code 1000. [[C2: Error codes and error values to be completed. -- JFT]]

5.1.2.2. EPP <info> command with a bundle object

A client querying bundle information will include a cira-idn-bundle:info object in the request. The server will reply with a bundle object.

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C: <command>
C:   <info>
C:       <cira-idn:domainVariants>
C:         <cira-idn:name>xn--r-wfan6a.ca</cira-idn:name>
C:         <cira-idn:name>xn--cir-cla.ca</cira-idn:name>
C:         <cira-idn:name>cira.ca</cira-idn:name>
C:       </cira-idn:domainVariants>
C:     </cira-idn:ciraIdnInfo>
C:   </info>
C: </command>
C: </epp>
C:     <cira-idn-bundle:info
C:         xmlns:cira-idnbundle=
C:             "urn:ietf:params:xml:ns:cira-idn-bundle-1.0">
C:     <cira-idn-bundle:name>
C:         xn--valuation-93a.ca
C:     </cira-idn-bundle:name>
C:     </cira-idn-bundle:info>
C: </cira-idn-bundle:info>
C:     </clTRID>
C:     ABC-12345
C: </clTRID>
C: </command>
C: </epp>

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:     </resData>
S:     <extension>
S:       <cira-idn-bundle:infData
S:           xmlns:cira-idn=
S:             "urn:ietf:params:xml:ns:cira-idn-1.0"
S:           xmlns:cira-idn-bundle=
S:             "urn:ietf:params:xml:ns:cira-idn-bundle-1.0">
S:         <cira-idn-bundle:canonicalDomainName>
S:             evaluation.ca
S:         </cira-idn-bundle:canonicalDomainName>
S:         <cira-idn-bundle:clID>rar600</cira-idn-bundle:clID>
S:         <cira-idn-bundle:registrant>
S:             rant600
S:         </cira-idn-bundle:registrant>
S:         <cira-idn-bundle:crDate>2012-12-08T16:25:01.0Z</cira-idn-bundle:crDate>
S:         <cira-idn-bundle:upID>rar600</cira-idn-bundle:upID>
S:         <cira-idn-bundle:upDate>2012-12-08T17:25:01.0Z</cira-idn-bundle:upDate>
S:         <cira-idn-bundle:bundleDomains>
S:             <cira-idn:name>evaluation.ca</cira-idn:name>
S:             <cira-idn:name>xn--evaluation-93a.ca</cira-idn:name>
S:             <cira-idn:name>xn--valution-2ya9f.ca</cira-idn:name>
S:         </cira-idn-bundle:bundleDomains>
S:     </cira-idn-bundle:infData>
S: </extension>
S: </resData>
S: </response>
S: </epp>
5.1.3. EPP <poll> Command

The EPP <poll> command is not modified by this extension.

5.1.4. EPP <transfer> Command

The transfer command is not modified by this extension. The domain:name element may contain an IDN domain in A-label format.

5.2. EPP Transform Commands

5.2.1. EPP <create> Command

The create command from the client is extended with a ciraIdnCreate object containing the repertoire information and the U-Label for the domain to create. The server answer will contain the A-label of the created domain in the domain:name element. The server answer is not modified by this extension except for return codes.

Return code 8001 is returned by the server when the domain name contains invalid characters or when the A-label cannot be converted successfully to a valid U-label. Error code 8309 is used for an invalid repertoire and 8310 is used when the A-label does not match the U-label.

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <create>
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"  
C:       <domain:name>xn--r-wfan6a.ca</domain:name>
C:       <domain:period unit="y">2</domain:period>
C:       <domain:ns>
C:         <domain:hostObj>hostname.example.net</domain:hostObj>
C:         <domain:hostObj>hostname.example.com</domain:hostObj>
C:       </domain:ns>
C:       <domain:registrant>contactid-1</domain:registrant>
C:     </create>
C:   </command>
C: </epp>
<domain:contact type="admin">contactid-1</domain:contact>
<domain:contact type="tech">nbguy</domain:contact>
<domain:contact type="tech">nbtech</domain:contact>
<domain:contact type="tech">nbadmin</domain:contact>
<domain:authInfo>
<domain:pw>password</domain:pw>
</domain:authInfo>
</domain:create>

<extension>
<cira-idn:repertoire>fr</cira-idn:repertoire>
<cira-idn:u-label>cira.ca</cira-idn:u-label>
</cira-idn:ciraIdnCreate>
</extension>
</command>
</epp>

5.2.2. EPP <delete> Command

The EPP <delete> command is not modified by this extension.
5.2.3. EPP <renew> Command

The EPP <renew> command is not modified by this extension.

5.2.4. EPP <transfer> Command

The EPP <transfer> command is not modified by this extension. The domain:name element may contain an IDN domain in A-label format. New error codes and error values may be returned based on IDN processing.

5.2.5. EPP <update> Command

The EPP <update> command is not modified by this extension. The domain:name element may contain an IDN domain in A-label format. New error codes and error values may be returned based on IDN processing.

The server returns code 8001 when the domain name contains invalid characters or when the A-label cannot be converted successfully to a valid U-label. Error code 8309 is used for an invalid repertoire and 8317 is used when the request fails due to a syntax error or a policy violation error.

6. Formal Syntax

Below are the XML schemas for cira-idn-1.0 and cira-idn-bundle-1.0. These two schemas were developed separately and are almost independent of each other, except for the latter borrowing the domainList type of the former.

6.1. Schema for cira-idn-1.0

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:cira-idn-1.0"
 xmlns:cira-idn="urn:ietf:params:xml:ns:cira-idn-1.0"
 xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
 xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">
  <!--
   Import common element types.
   -->
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
           schemaLocation="eppcom-1.0.xsd" />

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
    </documentation>
  </annotation>
</schema>
```
extension schema for IDN domain name in the .CA registry
</documentation>
</annotation>

<element name="ciraIdnCreate" type="cira-idn:createType" />
<element name="ciraIdnInfo" type="cira-idn:infDataType" />
<element name="ciraIdnCheck" type="cira-idn:checkType" />

<!--
Child elements of the <create> command.
-->
<complexType name="createType">
<sequence>
<element name="repertoire" type="cira-idn:repertoireType" />
<element name="u-label" type="eppcom:labelType" minOccurs="0"/>
</sequence>
</complexType>

<complexType name="domainList">
<sequence>
<element name="name" type="eppcom:labelType" maxOccurs="unbounded" />
</sequence>
</complexType>

<!--
Child elements of the <info> command.
-->
<complexType name="infDataType">
<sequence>
<element name="domainVariants" type="cira-idn:domainList" minOccurs="0"/>
</sequence>
</complexType>

<!--
Child elements of the <check> command.
-->
<complexType name="checkType">
<sequence>
<element name="name" type="eppcom:labelType" minOccurs="0"/>
</sequence>
</complexType>
<element name="repertoire" type="cira-idn:repertoireType" />
</complexType>

<!--
End of schema.
-->
</schema>

6.2. Schema for cira-idn-bundle-1.0

<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:ietf:params:xml:ns:cira-idn-bundle-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:cira-idn="urn:ietf:params:xml:ns:cira-idn-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <!--
  Import common element types.
  -->
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
    schemaLocation="eppcom-1.0.xsd" />
  <import namespace="urn:ietf:params:xml:ns:cira-idn-1.0"
    schemaLocation="cira-idn-1.0.xsd" />

  <annotation>
    <documentation>
    Extensible Provisioning Protocol v1.0
    bundle schema for framework for
    provisioning of cira idn bundle
    information.
    </documentation>
  </annotation>

  <!--
  Child elements found in EPP commands.
  -->
  <element name="info" type="cira-idn-bundle:infoType" />
  <!--
  Child elements of the <info> commands.
  -->
  <complexType name="infoType">
    <sequence>
7. Security Considerations

For domain labels containing a large number of IDN characters, the list of label variants can be large. For the French repertoire, the largest variant code point is from the e character and has 5 variants (including the base one). For a label size of 64 characters, the list of label variants may reach up to 320 entries. The current schema does not place a limit on the size of domainVariants element, but implementations may want limit its size for performance purpose.
8.  IANA Considerations

The CIRA IDN EPP extension is to be added to the EPP extension registry as specified in [RFC7451]. Below is the registration template.

-----BEGIN FORM-----
Name of Extension: "CIRA IDN EPP Extension"

Document Status: Informational
Reference: draft-wilcox-cira-idn-eppext
Registrant Name and Email Address: .CA Registry Operations, regops@cira.ca
TLDs: .ca
IPR Disclosure: TBD
Status: Active
Notes: None

[[C1: To be completed as needed. -- JFT]]

9.  Acknowledgments

The authors would like to thank Marc Blanchet and Audric Schiltknecht for suggestions and revisions.

10.  References

10.1.  Normative References


10.2. Informative References


[ID.draft-ietf-eppext-idnmap]

[ID.draft-kong-eppext-bundling-registration]

Authors’ Addresses
Stuart Olmstead-Wilcox
CIRA
350 Sparks Street, Suite 306
Ottawa, ON K1R 7S8
Canada

Email: stuart.olmstead-wilcox@cira.ca

Jacques Latour
CIRA
350 Sparks Street, Suite 306
Ottawa, ON K1R 7S8
Canada

Email: jacques.latour@cira.ca

Jean-Francois Tremblay
Viagenie

Email: jean-francois.tremblay@viagenie.ca
Registry Reseller Extension for the Extensible Provisioning Protocol (EPP)
draft-zhou-eppext-reseller-01

Abstract

This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], to support assigning a reseller to any existing object (domain, host, contact) as well as any future objects. Specified in Extensible Markup Language (XML), this extended mapping is applied to provide additional features required for the provisioning of registry resellers.

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 January 4, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction ..................................................... 3
2. Conventions Used in This Document ................................. 3
3. Object Attributes .................................................. 4
  3.1. Reseller Identifier ........................................... 4
  3.2. Reseller Name .................................................. 4
4. EPP Command Mapping ............................................... 4
  4.1. EPP Query Commands ............................................ 4
    4.1.1. EPP <check> Command ....................................... 4
    4.1.2. EPP <info> Command ....................................... 4
    4.1.3. EPP <transfer> Command ................................... 7
  4.2. EPP Transform Commands ....................................... 7
    4.2.1. EPP <create> Command ..................................... 7
    4.2.2. EPP <delete> Command ..................................... 8
    4.2.3. EPP <renew> Command ..................................... 8
    4.2.4. EPP <transfer> Command .................................. 9
    4.2.5. EPP <update> Command ..................................... 9
5. Formal Syntax ...................................................... 11
6. Internationalization Considerations ............................... 13
7. IANA Considerations ............................................... 13
  7.1. XML Namespace ................................................ 13
  7.2. EPP Extension Registry ....................................... 14
8. Security Considerations ........................................... 14
9. References .......................................................... 14
  9.1. Normative References ......................................... 14
  9.2. Informative References ....................................... 15
Appendix A. Change Log ............................................... 16
Authors’ Addresses .................................................. 16
1. Introduction

Domain resellers are the individuals or companies acting as agents for ICANN accredited registrars. A domain name registrar may have several resellers to help them sell domain names to end users.

Generally speaking, resellers provide domain registration information via registrar’s EPP client without reseller information. On one hand, registrars are concerned about how to identify resellers. On the other hand, end users would also be confused by the WHOIS service without corresponding reseller information. This requirement imposes a challenge for the domain registries since there is no definition of resellers in the existing EPP domain name mapping. Out of band method could solve this problem but may increase extra cost.

In order to facilitate provisioning and management of reseller information in a shared central repository, this document proposes a reseller extension of [RFC5731], [RFC5732] and [RFC5733]. The examples provided in this document are used for the domain object for illustration purposes. The host and contact object could be extended in the same way with the domain object.

A reseller mapping object defined in [ID.draft-zhou-eppext-reseller-mapping] SHOULD be created first. The reseller information specified in this document SHOULD reference the existing reseller identifier and reseller name.

This document is specified using the XML 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

2. Conventions Used in This Document

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 [RFC2119].

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this specification.

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented to develop a conforming implementation.

resellerext-1.0 in this document is used as an abbreviation for urn:ietf:params:xml:ns:resellerext-1.0.
3. Object Attributes

This extension adds additional elements to the EPP domain name mapping [RFC5731]. Only the new elements are described here.

3.1. Reseller Identifier

Reseller identifier provides the ID of the reseller of a sponsoring registrar. Its corresponding element is <resellerext:id> which refers to the <reseller:id> element defined in [ID.draft-zhou-eppext-reseller-mapping].

3.2. Reseller Name

Reseller name provides the name of the reseller of a sponsoring registrar. Its corresponding element is <resellerext:name> which refers to the <reseller:name> element defined in [ID.draft-zhou-eppext-reseller-mapping].

4. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in provisioning and managing reseller information via EPP.

4.1. EPP Query Commands

EPP provides three commands to retrieve domain information: <check> to determine if a domain object can be provisioned within a repository, <info> to retrieve detailed information associated with a domain object, and <transfer> to retrieve domain-object transfer status information.

4.1.1. EPP <check> Command

This extension does not add any elements to the EPP <check> command or <check> response described in the EPP domain name mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733].

4.1.2. EPP <info> Command

This extension does not add any element to the EPP <info> command described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733]. However, additional elements are defined for the <info> response.

Example <info> command:

```
```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <domain:info xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:authInfo>
          <domain:pw>fooBAR</domain:pw>
        </domain:authInfo>
      </domain:info>
    </info>
    <clTRID>ngcl-mIFICBNP</clTRID>
  </command>
</epp>

When an <info> command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733]. In addition, the EPP <extension> element SHOULD contain a child <reseller:infData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <reseller:infData> element contains the following child elements:

- A <resellerext:id> element that contains the identifier of the reseller of a sponsoring registrar.
- An OPTIONAL <resellerext:name> element that contains the name of the reseller of a sponsoring registrar.

Example <info> response for an authorized client:
<xml version="1.0" encoding="UTF-8" standalone="no"/>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en-US">Command completed successfully</msg>
    </result>
    <resData>
      <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:roid>EXAMPLE1-REP</domain:roid>
        <domain:status s="ok"/>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="billing">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:ns>
          <domain:hostObj>ns1.example.com</domain:hostObj>
        </domain:ns>
        <domain:clID>ClientX</domain:clID>
        <domain:crID>ClientY</domain:crID>
        <domain:crDate>2015-02-06T04:01:21.0Z</domain:crDate>
        <domain:exDate>2018-02-06T04:01:21.0Z</domain:exDate>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:infData>
    </resData>
    <extension>
      <rgp:infData xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0">
        <rgp:rgpStatus s="addPeriod"/>
      </rgp:infData>
      <resellerext:infData xmlns:resellerext="urn:ietf:params:xml:ns:resellerext-1.0">
        <resellerext:id>myreseller</resellerext:id>
        <resellerext:name>example</resellerext:name>
      </resellerext:infData>
    </extension>
    <trID>
      <clTRID>ngcl-IvJjzMZc</clTRID>
      <svTRID>test142AWQONJZ</svTRID>
    </trID>
  </response>
</epp>

<info> response for the unauthorized client has not been changed, see [RFC5731], [RFC5732] and [RFC5733] for detail.
An EPP error response MUST be returned if an <info> command cannot be processed for any reason.

4.1.3. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the EPP domain name mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733].

4.2. EPP Transform Commands

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

4.2.1. EPP <create> Command

This extension defines additional elements for the EPP <create> command described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733]. No additional elements are defined for the EPP <create> response.

The EPP <create> command provides a transform operation that allows a client to create a domain object. In addition to the EPP command elements described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733], the command MUST contain an <extension> element, and the <extension> element MUST contain a child <resellerext:create> element that identifies the extension namespace if the client wants to associate data defined in this extension to the domain object. The <resellerext:create> element contains the following child elements:

- A <resellerext:id> element that contains the identifier of the reseller of a sponsoring registrar.

Example <create> Command:
When a <create> command has been processed successfully, the EPP response is as described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733].

An EPP error response MUST be returned if a <create> command cannot be processed for any reason.

4.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733].

4.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733].
4.2.4. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733], but after a successful transfer of an object with an assigned reseller, the server SHOULD clear the assigned reseller value.

4.2.5. EPP <update> Command

This extension defines additional elements for the EPP <update> command described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733]. No additional elements are defined for the EPP <update> response.

The EPP <update> command provides a transform operation that allows a client to modify the attributes of a domain object. In addition to the EPP command elements described in the EPP domain mapping, the command MUST contain an <extension> element, and the <extension> element MUST contain a child <resellerext:update> element that identifies the extension namespace if the client wants to update the domain object with data defined in this extension. The <resellerext:update> element contains the following child elements:

- An OPTIONAL <resellerext:add> element that contains attribute values to be added to the object.
- An OPTIONAL <resellerext:rem> element that contains attribute values to be removed from the object.
- An OPTIONAL <resellerext:chg> element that contains attribute values to be changed.

At least one and only one <resellerext:add>, <resellerext:rem> or <resellerext:rem> element MUST be provided. The <resellerext:add>, <resellerext:rem> and <resellerext:rem> elements contain the following child element:

- A <resellerext:id> element that contains the identifier of the reseller of a sponsoring registrar.

Example <update> command, adding a reseller:
Example <update> command, removing a reseller:

C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <resellerext:update xmlns:resellerext="urn:ietf:params:xml:ns:resellerext-1.0">
C:        <resellerext:rem>
C:          <resellerext:id>myreseller</resellerext:id>
C:        </resellerext:rem>
C:      </resellerext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

Example <update> command, updating reseller identifier:
When an extended <update> command has been processed successfully, the EPP response is as described in the EPP domain name mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733].

5. Formal Syntax

An EPP object mapping is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

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

<schema targetNamespace="urn:ietf:params:xml:ns:resellerext-1.0"
   xmlns:resellere="urn:ietf:params:xml:ns:resellerext-1.0"
   xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
   xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

<!-- Import common element types. -->

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

<annotation>
  <documentation>
    Extensible Provisioning Protocol v1.0
    Domain Reseller Extension Schema v1.0
  </documentation>
</annotation>

<!-- Child elements found in EPP commands. -->
<element name="create" type="resellerext:createType"/>
<element name="update" type="resellerext:updateType"/>

<!-- reseller name type-->
<complexType name="createNameType">
  <restriction base="normalizedString">
    <minLength value="1"/>
    <maxLength value="255"/>
  </restriction>
</complexType>

<!-- Child elements of the <reseller:create> command
All elements must be present at time of creation
--> 
<complexType name="createType">
  <sequence>
    <!-- agent identifier that sells the domain, e.g. registrar, reseller -->
    <element name="id" type="eppcom:clIDType"/>
  </sequence>
</complexType>

<!-- Child elements of <reseller:update> command
--> 
<complexType name="updateType">
  <sequence>
    <element name="add" type="resellerext:addRemChgType" minOccurs="0"/>
    <element name="rem" type="resellerext:addRemChgType" minOccurs="0"/>
    <element name="chg" type="resellerext:addRemChgType" minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="addRemChgType">
  <sequence>
  </sequence>
</complexType>
6. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 and UTF-16. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED.

As an extension of the EPP domain name mapping, the elements, element content described in this document MUST inherit the internationalization conventions used to represent higher-layer domain and core protocol structures present in an XML instance that includes this extension.

7. IANA Considerations

7.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. IANA is requested to assignment the following URI.
Registration request for the reseller namespace:

- URI: urn:ietf:params:xml:ns:reseller-1.0
- Registrant Contact: See the "Author’s Address" section of this document.
- XML: See the "Formal Syntax" section of this document.

7.2. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

- Name of Extension: Domain Reseller Extension
- Document status: Standards Track
- Reference: (insert reference to RFC version of this document)
- Registrant Name and Email Address: See the "Author’s Address" section of this document.
- TLDs: any
- IPR Disclosure: none
- Status: active
- Notes: none

8. Security Considerations

The object mapping extension described in this document does not provide any other security services or introduce any additional considerations beyond those described by [RFC5730], [RFC5731], [RFC5732] and [RFC5733] or those caused by the protocol layers used by EPP.

9. References

9.1. Normative References

9.2. Informative References

[ID.draft-zhou-eppext-reseller-mapping]
Appendix A. Change Log

Initial -00: Individual document submitted.

-01:

* Updated abstract and introduction.
* Revised typos in info response.
* Added explanations on how to process reseller extension after successful transfer operation.
* Modified <update> explanation.
* Deleted reseller name element in <create> and <update> commands.
* Removed some inaccurate comments from xml schema.
* Modified the element name of reseller id and reseller name.

Authors’ Addresses

Linlin Zhou
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 2677
Email: zhoulinlin@cnnic.cn

Ning Kong
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3147
Email: nkong@cnnic.cn
Chao Qi
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China
Phone: +86 10 5881 3146
Email: qichao@cnnic.cn

Xiaodong Lee
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China
Phone: +86 10 5881 3020
Email: xl@cnnic.cn

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US
Email: jgould@verisign.com
Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for provisioning and management of reseller object stored in a shared central repository. Specified in Extensible Markup Language (XML), this extended mapping is applied to provide additional features required for the provisioning of resellers.

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 January 4, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction ................................................. 3
2. Conventions Used in This Document .......................... 3
3. Object Attributes .......................................... 3
   3.1. Reseller Identifier ................................. 4
   3.2. Contact and Client Identifiers .................... 4
   3.3. Reseller State .................................. 4
   3.4. Parent Identifier ................................. 4
   3.5. URL ........................................... 4
   3.6. Disclosure of Data Elements and Attributes ....... 5
4. EPP Command Mapping ......................................... 5
   4.1. EPP Query Commands ................................ 5
       4.1.1. EPP <check> Command ......................... 5
       4.1.2. EPP <info> Command ........................ 7
       4.1.3. EPP <transfer> Command ..................... 13
   4.2. EPP Transform Commands ................................ 13
       4.2.1. EPP <create> Command ......................... 13
       4.2.2. EPP <delete> Command ........................ 16
       4.2.3. EPP <renew> Command ........................ 18
       4.2.4. EPP <transfer> Command ..................... 18
       4.2.5. EPP <update> Command ......................... 18
5. Formal Syntax .............................................. 21
6. Internationalization Considerations ......................... 27
7. IANA Considerations ....................................... 28
   7.1. XML Namespace .................................... 28
   7.2. EPP Extension Registry ............................ 28
8. Security Considerations ..................................... 29
9. Normative References ....................................... 29
Appendix A. Change Log ....................................... 30
Authors’ Addresses ........................................... 30
1. Introduction

Domain resellers are the individuals or companies that act as agents for domain name registrars. A domain name registrar is a direct customer of the domain name registry, is represented as the sponsoring client to the server in [RFC5730], and may have several resellers to help them sell domain names to end users.

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping specifies the reseller object mapping.

This document is specified using the XML 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

2. Conventions Used in This Document

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 [RFC2119].

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this specification.

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented to develop a conforming implementation.

"reseller-1.0" in is used as an abbreviation for "urn:ietf:params:xml:ns:reseller-1.0". The XML namespace prefix "reseller" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

3. Object Attributes

An EPP reseller object has attributes and associated values that can be viewed and modified by the sponsoring client or the server. This section describes each attribute type in detail. The formal syntax for the attribute values described here can be found in the "Formal Syntax" section of this document and in the appropriate normative references.
3.1. Reseller Identifier

Reseller identifier provides the ID of the reseller of a sponsoring registrar. Its corresponding element is <reseller:id> defined in this document.

3.2. Contact and Client Identifiers

All EPP contacts are identified by a server-unique identifier. Contact identifiers are character strings with a specific minimum length, a specified maximum length, and a specified format. Contact identifiers use the "clIDType" client identifier syntax described in [RFC5730].

3.3. Reseller State

A reseller object MUST always have at least one associated state value. Valid values include "ok", "readonly" and "terminated".

State Value Descriptions:

- ok: the normal status value for the reseller object.
- readonly: transform commands submitted with the reseller identifier in the reseller extension would not be allowed.
- terminated: query and transform commands submitted with the reseller identifier in the reseller extension would not be allowed.

3.4. Parent Identifier

There can be more than one layer of resellers. The parent identifier, as defined with the <reseller:parentId> element, represents the parent reseller identifier in a child reseller. The parent identifier is not defined for the top level reseller, namely the registrar of the registry. An N-tier reseller has a parent reseller and at least one child reseller. A reseller customer has a parent reseller and no child resellers.

Loops SHOULD be prohibited. If reseller A has B as parent identifier, reseller B must not have reseller A as parent identifier.

3.5. URL

The URL represents the reseller web home page, as defined with the <reseller:url> element.
3.6. Disclosure of Data Elements and Attributes

This document supports the same disclosure features described in Section 2.9 of with the use of the <reseller:disclose> element. [RFC5733].

The <reseller:disclose> element MUST contain at least one of the following child elements:

- `<reseller:name type="int"/>`
- `<reseller:name type="loc"/>`
- `<reseller:addr type="int"/>`
- `<reseller:addr type="loc"/>`
- `<reseller:voice/>`
- `<reseller:fax/>`
- `<reseller:email/>`
- `<reseller:url/>`
- `<reseller:contact/>`

4. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in provisioning and managing reseller information via EPP.

4.1. EPP Query Commands

EPP provides two commands to retrieve domain information: <check> to determine if a reseller object can be provisioned within a repository, and <info> to retrieve detailed information associated with a reseller object. This document does not define a mapping for the EPP <transfer> command.

4.1.1. EPP <check> Command

The EPP <check> command is used to determine if an object can be provisioned within a repository. It provides a hint that allows a client to anticipate the success or failure of provisioning an object.
using the <create> command, as object-provisioning requirements are ultimately a matter of server policy.

In addition to the standard EPP command elements, the <check> command MUST contain a <reseller:check> element that identifies the reseller namespace. The <reseller:check> element contains the following child elements:

- One or more <reseller:id> elements that contain the server-unique identifier of the reseller objects to be queried.

Example <check> command:

```xml
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <reseller:check
C:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
C:        <reseller:id>res1523</reseller:id>
C:        <reseller:id>re1523</reseller:id>
C:        <reseller:id>1523res</reseller:id>
C:      </reseller:check>
C:    </check>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <check> command has been processed successfully, the EPP <resData> element MUST contain a child <reseller:chkData> element that identifies the reseller namespace. The <reseller:chkData> element contains one or more <reseller:cd> elements that contain the following child elements:

- A <reseller:id> element that identifies the queried object. This element MUST contain an "avail" attribute whose value indicates object availability (can it be provisioned or not) at the moment the <check> command was completed. A value of "1" or "true" means that the object can be provisioned. A value of "0" or "false" means that the object cannot be provisioned.

- An OPTIONAL <reseller:reason> element that MAY be provided when an object cannot be provisioned. If present, this element contains server-specific text to help explain why the object cannot be provisioned. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute MAY be present to identify the language if the
negotiated value is something other than the default value of "en" (English).

Example <check> response:

```xml
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <reseller:chkData
S:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
S:        <reseller:cd>
S:          <reseller:id avail="1">res1523</reseller:id>
S:        </reseller:cd>
S:        <reseller:cd>
S:          <reseller:id avail="0">re1523</reseller:id>
S:          <reseller:reason>In use</reseller:reason>
S:        </reseller:cd>
S:        <reseller:cd>
S:          <reseller:id avail="1">1523res</reseller:id>
S:        </reseller:cd>
S:      </reseller:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

An EPP error response MUST be returned if a <check> command cannot be processed for any reason.

4.1.2. EPP <info> Command

The EPP <info> command is used to retrieve information associated with a reseller object. In addition to the standard EPP command elements, the <info> command MUST contain a <reseller:info> element that identifies the reseller namespace. The <reseller:info> element contains the following child elements:

- A <reseller:id> element that contains the server-unique identifier of the reseller object to be queried.
Example <info> command:

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <info>
C:       <reseller:info
C:         xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
C:         <reseller:id>res1523</reseller:id>
C:       </reseller:info>
C:     </info>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <reseller:infData> element that identifies the reseller namespace. The <reseller:infData> element contains the following child elements:

- A <reseller:id> element that contains the server-unique identifier of the reseller object, as defined in Section 3.1.
- A <reseller:roid> element that contains the Repository Object IDentifier assigned to the reseller object when the object was created.
- A <reseller:state> element that contains the operational state of the reseller, as defined in Section 3.3.
- An OPTIONAL <reseller:parentId> element that contains the identifier of the parent object, as defined in Section 3.4.
- One or two <reseller:postalInfo> elements that contain postal-address information. Two elements are provided so that address information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of UTF-8 that can be represented in the 7-bit US-ASCII character set. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The <reseller:postalInfo> element contains the following child elements:
  * A <reseller:name> element that contains the name of the reseller, which SHOULD be the name of the organization.
* A `<reseller:addr>` element that contains address information associated with the reseller. A `<reseller:addr>` element contains the following child elements:
  
  + One, two, or three OPTIONAL `<reseller:street>` elements that contain the reseller’s street address.
  
  + A `<reseller:city>` element that contains the reseller’s city.
  
  + An OPTIONAL `<reseller:sp>` element that contains the reseller’s state or province.
  
  + An OPTIONAL `<reseller:pc>` element that contains the reseller’s postal code.
  
  + A `<reseller:cc>` element that contains the reseller’s country code.
  
  o An OPTIONAL `<reseller:voice>` element that contains the reseller’s voice telephone number.
  
  o An OPTIONAL `<reseller:fax>` element that contains the reseller’s facsimile telephone number.
  
  o A `<reseller:email>` element that contains the reseller’s email address.
  
  o A `<reseller:url>` element that contains the reseller’s web home page.
  
  o Zero or more OPTIONAL `<reseller:contact>` elements that contain identifiers for the contact objects to be associated with the reseller object. Contact object identifiers MUST be known to the server before the contact object can be associated with the reseller object. An attribute "type" associated with `<reseller:contact>` is used to represent contact types. The type values include admin, tech and billing.
  
  o A `<reseller:clID>` element that contains the identifier of the sponsoring client, who is the domain name registrar.
  
  o A `<reseller:crID>` element that contains the identifier of the client that created the reseller object.
  
  o A `<reseller:crDate>` element that contains the date and time of reseller-object creation.
- A `<reseller:upID>` element that contains the identifier of the client that last updated the reseller object. This element MUST NOT be present if the reseller has never been modified.

- A `<reseller:upDate>` element that contains the date and time of the most recent reseller-object modification. This element MUST NOT be present if the reseller object has never been modified.

- An OPTIONAL `<reseller:disclose>` element that identifies elements that require exceptional server-operator handling to allow or restrict disclosure to third parties. See Section 3.6 for a description of the child elements contained within the `<reseller:disclose>` element.

Example `<info>` response for the sponsoring client:
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <reseller:infData
S:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
S:        <reseller:id>res1523</reseller:id>
S:        <reseller:roid>res1523-REP</reseller:roid>
S:        <reseller:state>ok</reseller:state>
S:        <reseller:parentId>1523res</reseller:parentId>
S:        <reseller:postalInfo type="int">
S:          <reseller:name>Example Reseller Inc.</reseller:name>
S:          <reseller:addr>
S:            <reseller:street>123 Example Dr.</reseller:street>
S:            <reseller:street>Suite 100</reseller:street>
S:            <reseller:city>Dulles</reseller:city>
S:            <reseller:sp>VA</reseller:sp>
S:            <reseller:pc>20166-6503</reseller:pc>
S:            <reseller:cc>US</reseller:cc>
S:          </reseller:addr>
S:          <reseller:voice x="1234">+1.7035555555</reseller:voice>
S:          <reseller:fax>+1.7035555556</reseller:fax>
S:          <reseller:email>contact@reseller.example</reseller:email>
S:          <reseller:url>http://reseller.example</reseller:url>
S:          <reseller:contact type="admin">sh8013</reseller:contact>
S:          <reseller:contact type="billing">sh8013</reseller:contact>
S:        </reseller:postalInfo>
S:        <reseller:clID>ClientY</reseller:clID>
S:        <reseller:crID>ClientX</reseller:crID>
S:        <reseller:crDate>1999-04-03T22:00:00.0Z</reseller:crDate>
S:        <reseller:upID>ClientX</reseller:upID>
S:        <reseller:upDate>1999-12-03T09:00:00.0Z</reseller:upDate>
S:        <reseller:disclose flag="0">
S:          <reseller:voice/>
S:          <reseller:email/>
S:        </reseller:disclose>
S:      </reseller:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S: </epp>
Example <info> for the non-sponsoring client, according to the disclosure policy:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <reseller:infData
S:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
S:        <reseller:id>res1523</reseller:id>
S:        <reseller:roid>res1523-REP</reseller:roid>
S:        <reseller:state>ok</reseller:state>
S:        <reseller:parentId>1523res</reseller:parentId>
S:        <reseller:postalInfo type="int">
S:          <reseller:name>Example Reseller Inc.</reseller:name>
S:          <reseller:addr>
S:            <reseller:street>123 Example Dr.</reseller:street>
S:            <reseller:street>Suite 100</reseller:street>
S:            <reseller:city>Dulles</reseller:city>
S:            <reseller:sp>VA</reseller:sp>
S:            <reseller:pc>20166-6503</reseller:pc>
S:            <reseller:cc>US</reseller:cc>
S:          </reseller:addr>
S:        </reseller:postalInfo>
S:        <reseller:fax>+1.7035555556</reseller:fax>
S:        <reseller:url>http://reseller.example</reseller:url>
S:        <reseller:clID>ClientY</reseller:clID>
S:        <reseller:crID>ClientX</reseller:crID>
S:        <reseller:crDate>1999-04-03T22:00:00.0Z</reseller:crDate>
S:        <reseller:upID>ClientX</reseller:upID>
S:        <reseller:upDate>1999-12-03T09:00:00.0Z</reseller:upDate>
S:      </reseller:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

An EPP error response MUST be returned if an <info> command cannot be processed for any reason.
4.1.3. EPP <transfer> Command

The transfer semantics does not apply to reseller object. No EPP <transfer> command is defined in this document.

4.2. EPP Transform Commands

EPP provides four commands to transform reseller-object information: <create> to create an instance of a reseller object, <delete> to delete an instance of a reseller object, <transfer> to manage reseller-object sponsorship changes, and <update> to change information associated with a reseller object. This document does not define a mapping for the EPP <transfer> and <renew> command.

Transform commands are typically processed and completed in real time. Server operators MAY receive and process transform commands but defer completing the requested action if human or third-party review is required before the requested action can be completed. In such situations, the server MUST return a 1001 response code to the client to note that the command has been received and processed but that the requested action is pending. The server MUST also manage the status of the object that is the subject of the command to reflect the initiation and completion of the requested action. Once the action has been completed, all clients involved in the transaction MUST be notified using a service message that the action has been completed and that the status of the object has changed. Other notification methods MAY be used in addition to the required service message.

4.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create a reseller object. In addition to the standard EPP command elements, the <create> command MUST contain a <reseller:create> element that identifies the reseller namespace. The <reseller:create> element contains the following child elements:

- A <reseller:id> element that contains the desired server-unique identifier for the reseller to be created, as defined in Section 3.1.
- A <reseller:state> element that contains the operational status of the reseller, as defined in Section 3.3.
- An OPTIONAL <reseller:parentId> element that contains the identifier of the parent object, as defined in Section 3.4.
o One or two <reseller:postalInfo> elements that contain postal-
address information. Two elements are provided so that address
information can be provided in both internationalized and
localized forms; a "type" attribute is used to identify the two
forms. If an internationalized form (type="int") is provided,
element content MUST be represented in a subset of UTF-8 that can
be represented in the 7-bit US-ASCII character set. If a
localized form (type="loc") is provided, element content MAY be
represented in unrestricted UTF-8. The <reseller:postalInfo>
element contains the following child elements:

  * A <reseller:name> element that contains the name of the
    reseller, which SHOULD be the name of the organization.

  * A <reseller:addr> element that contains address information
    associated with the reseller. A <reseller:addr> element
    contains the following child elements:

    + One, two, or three OPTIONAL <reseller:street> elements that
      contain the reseller’s street address.

    + A <reseller:city> element that contains the reseller’s city.

    + An OPTIONAL <reseller:sp> element that contains the
      reseller’s state or province.

    + An OPTIONAL <reseller:pc> element that contains the
      reseller’s postal code.

    + A <reseller:cc> element that contains the reseller’s country
code.

  o An OPTIONAL <reseller:voice> element that contains the reseller’s
    voice telephone number.

  o An OPTIONAL <reseller:fax> element that contains the reseller’s
    facsimile telephone number.

  o A <reseller:email> element that contains the reseller’s email
    address.

  o A <reseller:url> element that contains the reseller’s web home
    page.

  o Zero or more OPTIONAL <reseller:contact> elements that contain
    identifiers for the human or organizational social information
    objects associated with the reseller object.
An OPTIONAL <reseller:disclose> element that identifies elements that require exceptional server-operator handling to allow or restrict disclosure to third parties. See Section 3.6 for a description of the child elements contained within the <reseller:disclose> element.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <reseller:create
C:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
C:        <reseller:id>res1523</reseller:id>
C:        <reseller:state>ok</reseller:state>
C:        <reseller:parentId>1523res</reseller:parentId>
C:        <reseller:postalInfo type="int">
C:          <reseller:name>Example Reseller Inc.</reseller:name>
C:          <reseller:addr>
C:            <reseller:street>123 Example Dr.</reseller:street>
C:            <reseller:street>Suite 100</reseller:street>
C:            <reseller:city>Dulles</reseller:city>
C:            <reseller:sp>VA</reseller:sp>
C:            <reseller:pc>20166-6503</reseller:pc>
C:            <reseller:cc>US</reseller:cc>
C:          </reseller:addr>
C:        </reseller:postalInfo>
C:        <reseller:voice x="1234">+1.7035555555</reseller:voice>
C:        <reseller:fax>+1.7035555556</reseller:fax>
C:        <reseller:email>contact@reseller.example</reseller:email>
C:        <reseller:url>http://reseller.example</reseller:url>
C:        <reseller:contact type="admin">sh8013</reseller:contact>
C:        <reseller:contact type="billing">sh8013</reseller:contact>
C:        <reseller:disclose flag="0">
C:          <reseller:voice/>
C:          <reseller:email/>
C:        </reseller:disclose>
C:      </reseller:create>
C:    </create>
C:  </command>
C:</epp>
```

When a <create> command has been processed successfully, the EPP <resData> element MUST contain a child <reseller:creData> element.
that identifies the reseller namespace. The `<reseller:creData>` element contains the following child elements:

- A `<reseller:id>` element that contains the server-unique identifier for the created reseller, as defined in Section 3.1.

- A `<reseller:crDate>` element that contains the date and time of reseller-object creation.

Example `<create>` response:

```xml
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <reseller:creData
S:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
S:        <reseller:id>res1523</reseller:id>
S:        <reseller:crDate>1999-04-03T22:00:00.0Z</reseller:crDate>
S:      </reseller:creData>
S:    </resData>
S:  </response>
S:</epp>
```

An EPP error response MUST be returned if a `<create>` command cannot be processed for any reason.

### 4.2.2. EPP `<delete>` Command

The EPP `<delete>` command provides a transform operation that allows a client to delete a reseller object. In addition to the standard EPP command elements, the `<delete>` command MUST contain a `<reseller:delete>` element that identifies the reseller namespace. The `<reseller:delete>` element MUST contain the following child element:

- A `<reseller:id>` element that contains the server-unique identifier of the reseller object, as defined in Section 3.1, to be deleted.
A reseller object SHOULD NOT be deleted if it is associated with other known objects. An associated reseller SHOULD NOT be deleted until associations with other known objects have been broken. A server SHOULD notify clients that object relationships exist by sending a 2305 error response code when a <delete> command is attempted and fails due to existing object relationships.

Example <delete> command:

```
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <delete>
C:      <reseller:delete xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
C:        <reseller:id>res1523</reseller:id>
C:      </reseller:delete>
C:    </delete>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <delete> command has been processed successfully, a server MUST respond with an EPP response with no <resData> element.

Example <delete> response:

```
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

An EPP error response MUST be returned if a <delete> command cannot be processed for any reason.
4.2.3. EPP <renew> Command

Renewal semantics do not apply to reseller objects, so there is no mapping defined for the EPP <renew> command.

4.2.4. EPP <transfer> Command

Transfer semantics do not apply to reseller objects, so there is no mapping defined for the EPP <transfer> command.

4.2.5. EPP <update> Command

The EPP <update> command provides a transform operation that allows a client to modify the attributes of a reseller object. In addition to the standard EPP command elements, the <update> command MUST contain a <reseller:update> element that identifies the reseller namespace. The <reseller:update> element contains the following child elements:

- A <reseller:id> element that contains the server-unique identifier of the reseller object to be updated, as defined in Section 3.1.
- An OPTIONAL <reseller:add> element that contains attribute values to be added to the object.
- An OPTIONAL <reseller:rem> element that contains attribute values to be removed from the object.
- An OPTIONAL <reseller:chg> element that contains attribute values to be changed.

At least one <reseller:add>, <reseller:rem> or <reseller:rem> element MUST be provided if the command is not being extended. All of these elements MAY be omitted if an <update> extension is present. The <reseller:add> and <reseller:rem> elements contain the following child element:

- Zero or more <reseller:contact> elements that contain the identifiers for contact objects to be associated with or removed from the reseller object. Contact object identifiers MUST be known to the server before the contact object can be associated with the reseller object.

A <reseller:chg> element contains the following OPTIONAL child elements. At least one child element MUST be present:

- A <reseller:state> element that contains the operational status of the reseller.
o A <reseller:parentId> element that contains the identifier of the parent object.

o One or two <reseller:postalInfo> elements that contain postal-address information. Two elements are provided so that address information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of UTF-8 that can be represented in the 7-bit US-ASCII character set. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The <reseller:postalInfo> element contains the following child elements:

* A <reseller:name> element that contains the name of the reseller, which SHOULD be the name of the organization.

* A <reseller:addr> element that contains address information associated with the reseller. A <reseller:addr> element contains the following child elements:

  + One, two, or three OPTIONAL <reseller:street> elements that contain the reseller’s street address.

  + A <reseller:city> element that contains the reseller’s city.

  + An OPTIONAL <reseller:sp> element that contains the reseller’s state or province.

  + An OPTIONAL <reseller:pc> element that contains the reseller’s postal code.

  + A <reseller:cc> element that contains the reseller’s country code.

o An <reseller:voice> element that contains the reseller’s voice telephone number.

o An <reseller:fax> element that contains the reseller’s facsimile telephone number.

o A <reseller:email> element that contains the reseller’s email address.

o A <reseller:url> element that contains the reseller’s web home page.
An `<reseller:disclose>` element that identifies elements that require exceptional server-operator handling to allow or restrict disclosure to third parties. See Section 2.9 in [RFC5733] for a description of the child elements contained within the `<reseller:disclose>` element.

Example `<update>` command:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <update>
      <reseller:update xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
        <reseller:id>res1523</reseller:id>
        <reseller:add>
          <reseller:contact type="tech">sh8013</reseller:contact>
        </reseller:add>
        <reseller:chg>
          <reseller:state>readonly</reseller:state>
          <reseller:postalInfo type="int">
            <reseller:org/>
            <reseller:addr>
              <reseller:street>124 Example Dr.</reseller:street>
              <reseller:street>Suite 200</reseller:street>
              <reseller:city>Dulles</reseller:city>
              <reseller:sp>VA</reseller:sp>
              <reseller:pc>20166-6503</reseller:pc>
              <reseller:cc>US</reseller:cc>
            </reseller:addr>
          </reseller:postalInfo>
          <reseller:voice>+1.7034444444</reseller:voice>
          <reseller:fax/>
          <reseller:disclose flag="1"/>
          <reseller:voice/>
          <reseller:email/>
          <reseller:disclose/>
        </reseller:chg>
      </reseller:update>
    </update>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

When an `<update>` command has been processed successfully, a server MUST respond with an EPP response with no `<resData>` element.
Example <update> response:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

An EPP error response MUST be returned if an <update> command cannot be processed for any reason.

5. Formal Syntax

An EPP object mapping is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:reseller-1.0"
   xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0"
   xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
   xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
   xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

<!-- Import common element types. -->
<import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>
<import namespace="urn:ietf:params:xml:ns:epp-1.0"/>
<import namespace="urn:ietf:params:xml:ns:contact-1.0"/>

<annotation>
<documentation>
Extensible Provisioning Protocol v1.0
reseller provisioning schema.
</documentation>
</annotation>

<!--
Child elements found in EPP commands.
-->
<element name="create" type="reseller:createType"/>
<element name="delete" type="reseller:sIDType"/>
<element name="update" type="reseller:updateType"/>
<element name="check" type="reseller:mIDType"/>
<element name="info" type="reseller:infoType"/>

<!--
Utility types.
-->
<simpleType name="stateType">
  <restriction base="token">
    <enumeration value="ok"/>
    <enumeration value="readonly"/>
    <enumeration value="terminated"/>
  </restriction>
</simpleType>

<simpleType name="contactAttrType">
  <restriction base="token">
    <enumeration value="admin"/>
    <enumeration value="billing"/>
    <enumeration value="tech"/>
  </restriction>
</simpleType>

<complexType name="contactType">
  <simpleContent>
    <extension base="eppcom:clIDType">
      <attribute name="type" type="reseller:contactAttrType"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="postalInfoType">
  <sequence>
    <element name="name" type="contact:postalLineType"/>
    <element name="addr" type="reseller:addrType"/>
  </sequence>
  <attribute name="type" type="contact:postalInfoEnumType"/>
<complexType name="addrType">
  <sequence>
    <element name="street" type="contact:optPostalLineType" minOccurs="0" maxOccurs="3"/>
    <element name="city" type="contact:postalLineType" minOccurs="0"/>
    <element name="sp" type="contact:optPostalLineType" minOccurs="0"/>
    <element name="pc" type="contact:pcType" minOccurs="0"/>
    <element name="cc" type="contact:ccType"/>
  </sequence>
</complexType>

<complexType name="discloseType">
  <sequence>
    <element name="name" type="contact:intLocType" minOccurs="0" maxOccurs="2"/>
    <element name="org" type="contact:intLocType" minOccurs="0" maxOccurs="2"/>
    <element name="addr" type="contact:intLocType" minOccurs="0" maxOccurs="2"/>
    <element name="voice" minOccurs="0"/>
    <element name="fax" minOccurs="0"/>
    <element name="email" minOccurs="0"/>
    <element name="url" minOccurs="0"/>
    <element name="contact" minOccurs="0"/>
  </sequence>
  <attribute name="flag" type="boolean" use="required"/>
</complexType>

<!--[CDATA[
Child elements of the <create> command.
-->}

<complexType name="createType">
  <sequence>
    <element name="id" type="eppcom:clIDType"/>
    <element name="state" type="reseller:stateType" minOccurs="0"/>
    <element name="parentId" type="eppcom:clIDType" minOccurs="0"/>
    <element name="postalInfo" type="reseller:postalInfoType" maxOccurs="2"/>
    <element name="voice" type="contact:e164Type" minOccurs="0"/>
    <element name="fax" type="contact:e164Type"/>
<element name="email" type="eppcom:minTokenType" minOccurs="0"/>
<element name="url" type="anyURI" minOccurs="0"/>
<element name="contact" type="reseller:contactType" minOccurs="0" maxOccurs="3"/>
<element name="disclose" type="reseller:discloseType" minOccurs="0"/>
</sequence>
</complexType>

<!-- Child element of commands that require only an identifier. -->
<complexType name="sIDType">
  <sequence>
    <element name="id" type="eppcom:clIDType"/>
  </sequence>
</complexType>

<!-- Child element of commands that accept multiple identifiers. -->
<complexType name="mIDType">
  <sequence>
    <element name="id" type="eppcom:clIDType" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!-- Child elements of the <info> commands. -->
<complexType name="infoType">
  <sequence>
    <element name="id" type="eppcom:clIDType"/>
  </sequence>
</complexType>

<!-- Child elements of the <update> command. -->
<complexType name="updateType">
  <sequence>
    <element name="id" type="eppcom:clIDType"/>
    <element name="add" type="reseller:addRemType" minOccurs="0"/>
    <element name="rem" type="reseller:addRemType"/>  
  </sequence>
</complexType>
<element name="chg" type="reseller:chgType" minOccurs="0"/>
</sequence>
</complexType>

<!-- Data elements that can be added or removed. -->
<complexType name="addRemType">
<sequence>
  <element name="contact" type="reseller:contactType" minOccurs="0"/>
</sequence>
</complexType>

<!-- Data elements that can be changed. -->
<complexType name="chgType">
<sequence>
  <element name="state" type="reseller:stateType" minOccurs="0"/>
  <element name="parentId" type="eppcom:clIDType" minOccurs="0"/>
  <element name="postalInfo" type="reseller:chgPostalInfoType" minOccurs="0" maxOccurs="2"/>
  <element name="voice" type="contact:e164Type" minOccurs="0"/>
  <element name="fax" type="contact:e164Type" minOccurs="0"/>
  <element name="email" type="eppcom:minTokenType" minOccurs="0"/>
  <element name="url" type="anyURI" minOccurs="0"/>
  <element name="disclose" type="reseller:discloseType" minOccurs="0"/>
</sequence>
</complexType>

<complexType name="chgPostalInfoType">
<sequence>
  <element name="name" type="contact:postalLineType" minOccurs="0"/>
  <element name="org" type="contact:optPostalLineType" minOccurs="0"/>
  <element name="addr" type="reseller:addrType" minOccurs="0"/>
</sequence>
</complexType>
</sequence>
  <attribute name="type" type="contact:postalInfoEnumType"
    use="required"/>
</complexType>

<!-- Child response elements. -->
<element name="chkData" type="reseller:chkDataType"/>
<element name="creData" type="reseller:creDataType"/>
<element name="infData" type="reseller:infDataType"/>

<!-- check> response elements. -->
<complexType name="chkDataType">
  <sequence>
    <element name="cd" type="reseller:checkType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="checkType">
  <sequence>
    <element name="id" type="reseller:checkIDType"/>
    <element name="reason" type="eppcom:reasonType"
      minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="checkIDType">
  <simpleContent>
    <extension base="eppcom:clIDType">
      <attribute name="avail" type="boolean"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!-- create> response elements. -->
<complexType name="creDataType">
  <sequence>
    <element name="id" type="eppcom:clIDType"/>
    <element name="crDate" type="dateTime"/>
  </sequence>
</complexType>
6. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 and UTF-16. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an `<?xml?>` declaration, use of UTF-8 is RECOMMENDED.
As an extension of the EPP reseller object mapping, the elements and element content described in this document MUST inherit the internationalization conventions used to represent higher-layer domain and core protocol structures present in an XML instance that includes this extension.

7. IANA Considerations

7.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. IANA is requested to assignment the following URI.

Registration request for the reseller namespace:

- URI: urn:ietf:params:xml:ns:reseller-1.0
- Registrant Contact: See the "Author’s Address" section of this document.
- XML: See the "Formal Syntax" section of this document.

7.2. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: Domain Reseller Object Extension

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: See the "Author’s Address" section of this document.

TLDs: any

IPR Disclosure: none

Status: active

Notes: none
8. Security Considerations

Authorization information described in [RFC5733] is not supported in this document. If the querying client is not the sponsoring registrar of the reseller, not all the object information is accessible. The disclose element defined in [RFC5733] is used to allow or restrict disclosure of object elements to third parties. Other mechanism, such as defining a registry customized authorization information list according to their local policies and regulations, is also possible.

9. Normative References


Appendix A.  Change Log

Initial -00: Individual document submitted.

-01:
  * Updated abstract text.
  * Added sentences to avoid loop of parent identifiers in section 3.4.
  * Revised typos in section 3.6.
  * Added explanation of contact type attribute in section 4.1.2.
  * Updated <info> responses.
  * Deleted description of <transfer> command in section 4.1 and 4.2.
  * Deleted whoisInfo disclose type in XML schema.
  * Deleted maxOccurs of addRemType.
  * Deleted extra "OPTIONAL" in section 4.2.5.
  * Updated typos in <update> response.

Authors' Addresses

Linlin Zhou
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing  100190
China

Phone: +86 10 5881 2677
Email: zhoulinlin@cnnic.cn

Ning Kong
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing  100190
China

Phone: +86 10 5881 3147
Email: nkong@cnnic.cn